ORCA® ORSPI4

Dual SPI4 Interface
and High-Speed SERDES FPSC

November 2003 **Preliminary Data Sheet**

Lattice Semiconductor has developed a next-generation FPSC targeted at high-speed data transmission. Built on the Series 4 reconfigurable embedded System-on-a-Chip (SoC) architecture, the ORSPI4 FPSC contains two SPI4.2 interface blocks, a high-speed Memory Controller, four channels of 0.6-3.7 Gbits/s SERDES with 8b/10b encoding and decoding and over 600K programmable system gates all on a single chip.

Embedded SPI4 Core Features

- **OIF-SPI4-02.0 compliant interfaces**
- **Dynamic timing receive interface:**
  - Full bandwidth up to 450 MHz DDR (900 Mbits/s) for all speed grades.
  - Bit de-skewing up to 16 phases of the clock
  - Capable of aligning bit-to-bit skews as large as ±1 bit periods
- **Static timing receive interface:**
  - Speeds up to 350 MHz DDR (700 Mbits/s), for all speed grades, including Quarter-Rate mode
  - Clock aligned or clock centered modes supported
- **DIP-4 and DIP-2 parity generation and checking**
- **Transmit Interface:**
  - Speeds up to 450 MHz DDR (900 Mbits/s)
  - Dedicated LVDS transmit interface for improved data eye integrity
  - Automatic idle insertion
- **256 logical ports:**
  - Embedded Calendar-based sequence port polling mechanism and bandwidth allocation. Shadow Calendar support for smooth transition to new Calendar
  - Up to 32 independent TX and 32 independent RX buffers per SPI4 interface internally. Various aggregation modes to support 1 to 32 separate embedded buffers per TX and RX
  - Up to 4 independent TX and 4 independent RX clock domain transfers to the FPGA logic
- **FIFO status support modes:**
  - 1/4 rate LVTTL or 1/4 rate LVDS
  - Automatic status handling or optionally under user control. Credit calculations based on burst size and status are also handled automatically
- **Configuration options as suggested in the OIF-SPI4-02.0 standard**
  - Configures parameters such as maximum burst size, calendar length, main and shadow calendars (1K deep each), length of training sequence etc.
- **Simple FIFO interface to the FPGA logic**
  - Provides ease of design and efficient clock domain transfers
- **Loopback modes provided for system- and chip-level debug**
- **Embedded 32-bit internal system bus plus 4-bit parity**
  - Interconnects FPGA logic, microprocessor interface (MPI), embedded RAM blocks, and embedded core blocks
  - Includes built-in system registers that act as the control and status center for the device
- **Low power operation.**
  - Full-rate SPI4.2 interfaces running at 450 MHz DDR (900 Mbits/sec) with dynamic alignment consumes 2 W of power or less. More efficient than FPGAs with soft-IP SPI4 solutions which consume in excess of 10 W.
- **Interoperability demonstrated with ORSPI4 partners.**

Embedded SERDES Core Features

- **Quad 600 Mbits/s to 3.7 Gbits/s SERDES:**
  - IEEE 802.3ae XAUI (Link State Machine & Alignment FIFOs embedded)
  - ANSI X3.230:1994 1G/2G FC-compliant (Link State Machine & Alignment FIFOs embedded)
  - Proven performance (same SERDES used in ORT82G5/ORT42G5 FPSCs)

Embedded Memory Controller Features

- **High Performance Memory Controller for interface to external buffer memory**
  - Required for Layer 2 data buffering
  - QDR II memory interface:
    - 36-bit Input and 36-bit Output bus, 18-bit address
    - 200 MHz clock rates
    - 20+ Gbits/s bandwidth
    - Supports 2- or 4-word burst mode
    - Simple FIFO interface to FPGA
    - Integrated PLL for optimized performance
    - Proven performance with multiple memory suppliers

Note: The term SPI4 refers to OIF SPI-4.2 throughout this document

High-Speed ORCA Series 4 FPGA

- **Internal performance of > 250 MHz**
- **Over 16K programmable logic elements**
- **1.5V operation (30% less power than 1.8 V operation)**
- **Comprehensive I/O selections including LVTTL, LVCMOS, GTL, GTL+, PECL, SSTL3/2, HSTL, ZBT, DDR, LVDS, bused-LVDS, and LVPECL**
- **1036-pin fpSBGA package provides enough FPGA user I/Os (498) for 4 full-duplex XGMII interfaces, 4 full-duplex PL-3 interfaces, etc; a 40% smaller 1156-pin fpBGA package is available with 356 FPGA user I/Os**

Introduction

The SPI4 blocks provide dual 10 Gbits/s physical-to-link layer interfaces in conformance to the OIF-SPI4-02.0 specification. Each block provides a full-duplex interface with an aggregate bandwidth of 13.6 Gbits/s. This is achieved by using 16 LVDS pairs each for RX and TX operating at a maximum data rate of 900 Mbits/s with a 450 MHz DDR clock. Both static and dynamic alignment are supported at the receive interface. Dynamic alignment is used to compensate for bit-to-bit skew at higher data rates, where it becomes difficult to meet tight setup/hold requirements. DIP-4 and DIP-2 parity generation and checking are supported. Data buffering of 8K bytes for both transmit and receive is provided by embedded Dual-Port RAM in each SPI4 core. Internal 1K deep main and shadow calendar supports scheduling of up to 256 ports. The Transmit and Receive Status FIFOs can also store flow control information for up to 256 ports, the maximum specified in the SPI4 specification.

An independent QDRII Memory Controller block provides data buffering between the FPGA logic and external memory and supports a throughput of greater than 20 Gbits/s. Data is transferred to and from memory through two sets of 36-bit unidirectional data lines operating at up to 200 MHz DDR. A set of 72 data signals is available to transfer data across the core-FPGA interface and allows the system to utilize the bandwidth available with second-generation Quad Data Rate (QDRII) SRAMs. Of the 72 data signals, 8 signals can be either used for parity or data. A soft IP version of this core is also available to allow a second data buffer on this device.

The High-Speed SERDES block supports four serial links, each operating at up to 3.7 Gbits/s (2.96 Gbits/s data rate with 8b/10b encoding and decoding), to provide four full-duplex synchronous interfaces with built-in RX Clock and Data Recovery (CDR) and transmitter preemphasis. The SERDES block is identical to that in the ORT82G5 FPSC, supports embedded 8b/10b encoding/decoding and implements link state machines for both 10G Ethernet, and 1G/2G/10G Fibre Channel. The state machines are IEEE P802.3ae/D4.01 XAUI based and also support FC (ANSI X3.230:1994) link synchronization.

Table 1. ORCA ORSPI4 — Available FPGA Logic

Device	PFU Rows	PFU Columns	Total PFUs	FPGA Max User I/O	LUTs	EBR Blocks	EBR Bits (K)	Usable* Gates (K)
ORSPI4	46	44	2,024	498/356	16,192	16	148	471-899

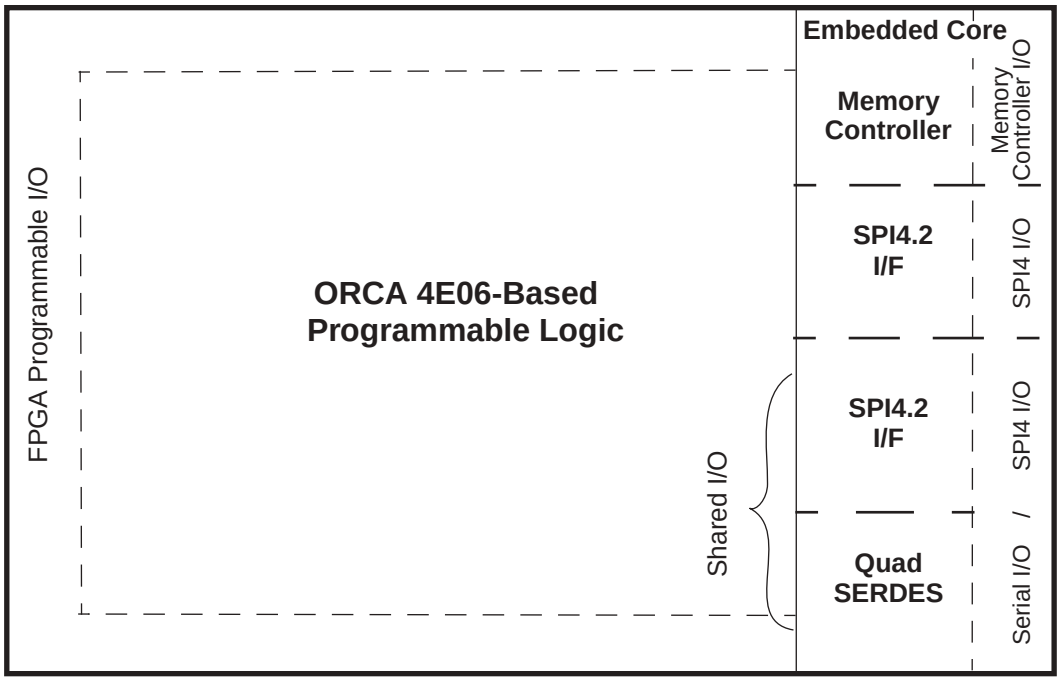
Note: The embedded core, embedded system bus, FPGA interface and MPI are not included in the above gate counts. The System Gate ranges are derived from the following: Minimum System Gates assumes 100% of the PFU's are used for logic only (No PFU RAM) with 40% EBR usage and 2 PLL's. Maximum System Gates assumes 80% of the PFU's are for logic, 20% are used for PFU RAM, with 80% EBR usage and 4 PLL's.

The ORSPI4 device is offered in two packages: 1036 fpSBGA and 1156 fpBGA. The 1036 package offers 498 FPGA User I/Os while the 1156 package offers 356 FPGA User I/Os. Additionally, the SERDES option is not available on the 1156 package.

ORSPI4 Overview

The ORSPI4 FPSC provides two SPI4.2 interface blocks, a Memory Controller and a 4-channel SERDES block, combined with FPGA logic. Based on the 1.5 V OR4E06 ORCA FPGA, it has a 46 x 44 array of Programmable Logic Cells (PLCs). The embedded core is attached to the right side of the device, as shown below, and is integrated directly into the FPGA array. A top level diagram of the basic chip configuration is shown in Figure 1.

Figure 1. ORSPI4 Basic Chip Configuration



Each of the logic blocks in the embedded core is functionally independent from the other blocks. Connections between blocks must be made through the FPGA logic. However, one of the SPI4 blocks and the SERDES block share I/Os. Hence the device may be configured to provide either two SPI4 interfaces or one SPI4 interface and one serial interface.

What Is an FPSC?

FPSCs, or Field-Programmable System Chips, are devices that combine field-programmable logic with ASIC or mask-programmed logic on a single device. FPSCs provide the time to market and the flexibility of FPGAs, the design effort savings of soft Intellectual Property (IP) cores, and the speed, design density, and economy of ASICs.

FPSC Overview

Lattice's Series 4 FPSCs are created from Series 4 *ORCA* FPGAs. To create a Series 4 FPSC, several columns of Programmable Logic Cells are integrated with an embedded logic core. Other than replacing some FPGA gates with ASIC gates, at greater than 10:1 area efficiency, none of the FPGA functionality is changed—all of the Series 4 FPGA capability is retained including the Embedded Block RAMs, MicroProcessor Interface (MPI), boundary scan, etc. Pins from the replaced columns of programmable logic are used as I/O pins for the embedded core. The remainder of the device pins retain their FPGA functionality.

FPSC Gate Counting

The total gate count for an FPSC is the sum of its embedded core (standard-cell/ASIC gates) and its FPGA gates. Because FPGA gates are generally expressed as a usable range with a nominal value, the total FPSC gate count is sometimes expressed in the same manner. Standard-cell ASIC gates are, however, 10 to 25 times more silicon-

area efficient than FPGA gates. Therefore, an FPSC with an embedded function is gate equivalent to an FPGA with a much larger gate count.

FPGA/Embedded Core Interface

The interface between the FPGA logic and the embedded core has been enhanced to allow for a greater number of interface signals than on previous FPSC architectures. Compared to bringing embedded core signals off-chip, this on-chip interface is much faster and requires less power. All of the delays for the interface are precharacterized and accounted for in the *ispLEVER* Development System.

Series 4 based FPSCs expand this interface by providing a link between the embedded block and the multi-master 32-bit system bus in the FPGA logic. This system bus allows the core easy access to many of the FPGA logic functions including the Embedded Block RAMs and the microprocessor interface.

Clock spines also can pass across the FPGA/embedded core boundary. This allows for fast, low-skew clocking between the FPGA and the embedded core. Many of the special signals from the FPGA, such as DONE and global set/reset, are also available to the embedded core, making it possible to fully integrate the embedded core with the FPGA as a system.

For even greater system flexibility, FPGA configuration RAMs are available for use by the embedded core. This allows for user-programmable options in the embedded core, in turn allowing for greater flexibility. Multiple embedded core configurations may be designed into a single device with user-programmable control over which configurations are implemented, as well as the capability to change core functionality simply by reconfiguring the device.

FPSC Design Kit

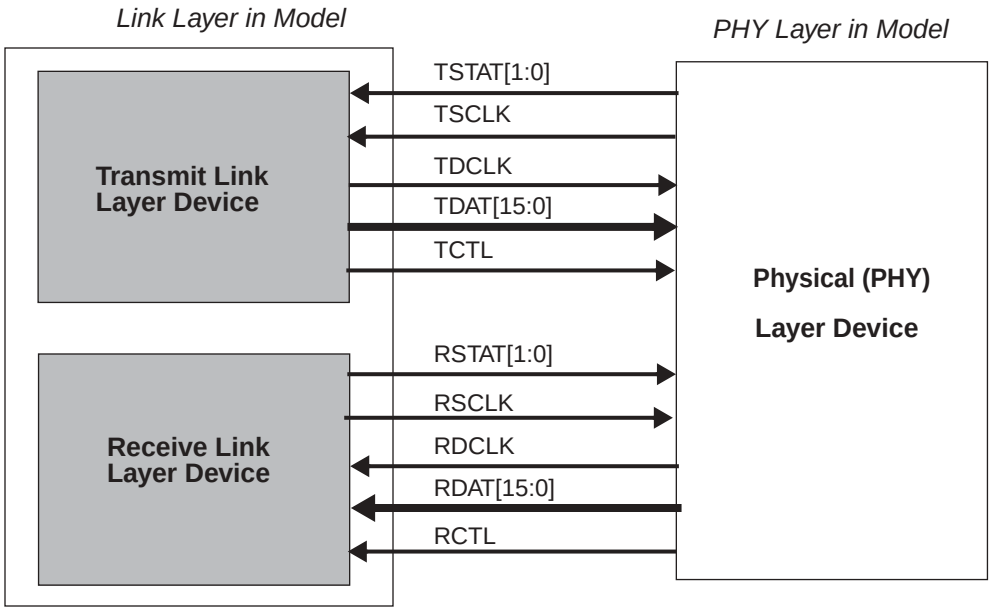
Development is facilitated by an FPSC design kit which, together with *ispLEVER* and third-party synthesis and simulation engines, provides all software and documentation required to design and verify an FPSC implementation. Included in the kit are the FPSC configuration manager, and compiled *Verilog* simulation models, *HSPICE* and/or IBIS models for I/O buffers, and complete online documentation. The kit's software coupled with *the* design environment, provides a seamless FPSC design environment. More information can be obtained by visiting the Lattice website at

http://www.latticesemi.com.

SPI4 Protocol Overview

The System Packet Interface Level 4, Phase 2 (SPI4) was defined by the Optical Internetworking Forum (OIF) as an interface for packet and cell transfers between a Physical Layer (PHY) device and a link layer device for applications requiring up to 10 Gbit/s aggregate bandwidth. The system level model for the SPI4 interface is shown in Figure 2.

Figure 2. System Model for SPI4 Interface



The details of the interface are specified in the OIF document “Implementation Agreement OIF-SPI4-02.0” (www.oiforum.com). That specification is based on the system model shown in the previous figure, which, in turn, is based on the Open System Interconnect (OSI) reference model. In the system model, a “transmit interface” sends address, start and end of packet signals and error control information from a Link Layer device to a PHY device and receives flow control (status) information from the PHY device. In the other direction, a “receive interface” at the Link Layer receives data from a PHY device and sends status information to the PHY device. While this convention provides a clear framework for defining the system level functions, a clean separation between Link Layer and Physical Layer functionality is not often seen in actual implementations.

The ORSPI4 FPSC SPI4 blocks implement the basic functions defined in the standard and also implements additional options, as suggested in the standard, to configure parameters such as maximum burst size, calendar length, length of training sequence, etc. As required by the specification, the transmit and receive interfaces operate completely independently.

Embedded Core Overview - Functions and Features

The embedded core contains four separate functional blocks, two SPI4 interface blocks, a high-speed Memory Controller block, and a quad SERDES block providing 4 channels of 0.6-3.7 Gbits/s SERDES. Features common to all blocks include:

- Improved *PowerPC*® 860 and *PowerPC* II high-speed synchronous MicroProcessor Interface that can be used for configuration, readback, device control, and device status; as well as for a general-purpose interface to the FPGA logic, RAMs, and embedded standard cell blocks. Glueless interface to synchronous *PowerPC* processors with user-configurable address space provided.
- New embedded *AMBA* ™ specification 2.0 AHB system bus (*ARM* ® processor) facilitates communication among the MicroProcessor Interface, configuration logic, and embedded core blocks.
- FPSC Design Kit available for use with *ispLEVER* development system software. Supported by industry-standard CAE tools for design entry, synthesis, simulation, and timing analysis.

SPI4 Interface Blocks - Overview

The ORSPI4 FPSC provides two independent SPI4 interface blocks in the embedded core. The two SPI4 blocks are identical and the following overview applies to both blocks. In the following sections, the SPI4 protocol conventions for “transmit” and “receive” are not followed, since in various applications the ORSPI4 FPSC could be used to perform different functions at various levels in the SPI4 protocol stack. Instead, the “transmit” functions are those used to transmit data to and receive current status information from the device at the other end of the SPI4 link. The “receive” functions are those used to receive data from and transmit current status information to the device at the other end of the SPI4 link.

Each SPI4 block supports a standard 10 Gbits/s physical-to-link layer interface in conformance to the specification. This is achieved by using 16 LVDS pairs each for RX and TX that operate at a maximum data rate of 900 Mbits/s with a 450 MHz DDR clock. Data buffering of 8 Kbytes each in the transmit and receive direction (example: 256 bytes each for up to 32 ports) is provided by embedded Dual-Port RAM (DPRAM). Aggregation of buffer space is supported for systems with less than 32 ports. The internal calendar and Transmit and Receive Status FIFOs have been sized so that applications with larger numbers of ports can be supported. The ORSPI4 has been designed to support up to 256 ports, the maximum specified in the SPI4 specification.

Despite operating independently, both the transmit path and the receive path logic perform similar functions and the partitioning of both logical blocks are quite similar as shown in Figure 3. The top level partitioning is between the logic blocks to transfer and process data and control information, and the logic blocks to generate, transfer and process status information.

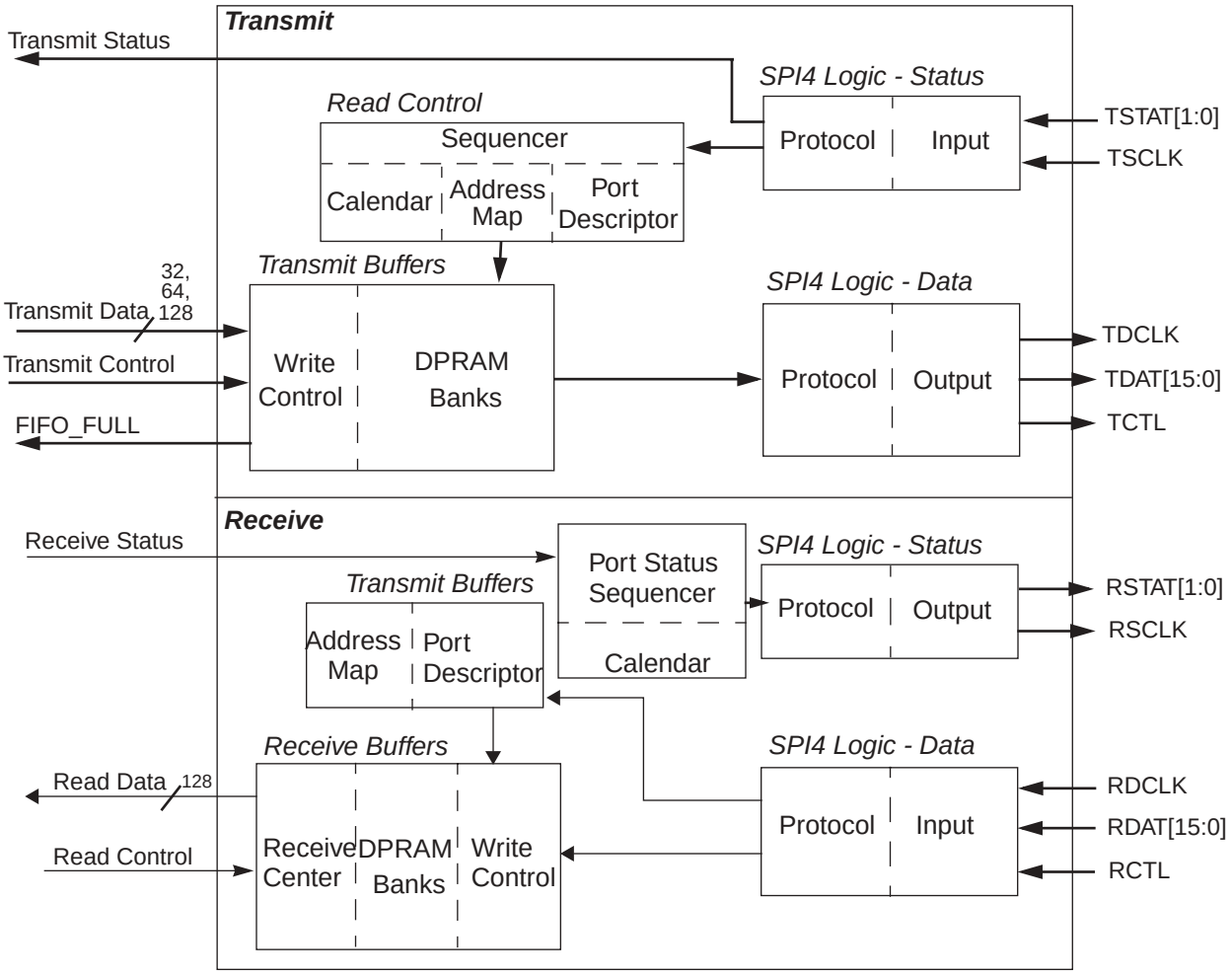
SPI4 Interface Block Features

- Each SPI4 block provides a standard 10 Gbits/s physical-to-link layer interface in conformance to the OIF-SPI4-02.0 specification. Each interface provides an aggregate bandwidth of 13.6 Gbits/s. This is achieved by using 16 LVDS pairs each for RX and TX with a maximum data rate of 900 Mbits/s using a 450 MHz DDR clock.
- The blocks can be used for applications such as interconnecting an OC-192 framer with a proprietary packetized interface, to a network processor with a SPI4 based packet interface or vice versa.
- Support for “static” or “dynamic” alignment at the receive interface. At clock rates above 350 MHz DDR, it becomes difficult to meet the tight setup/hold requirements at the receiver using static alignment. In this case, dynamic alignment is used to compensate for bit-to-bit skew.
- Dynamic alignment automatically compensates for Process, Voltage, and Temperature (PVT) changes in devices and systems.
  - Full bandwidth up to 450 MHz DDR (900 Mbits/s throughput)
  - Dynamically performs alignment based on 16 phases of the RX clock for improved accuracy
  - Alignment algorithm can be done based on excessive bit errors on the DIP-4 calculation
  - Clock skews up to +/- one clock cycle can be compensated by the dynamic alignment logic

- For low speed data, static alignment can be selected through a programmable control bit
  - Speeds up to 350 MHz DDR (700 Mbits/s throughput)
  - Dynamic alignment is bypassed and disabled to save power in static alignment mode.
  - Programmable on-edge or on-center clock/data relationship option at receiver.
  - Programmable clock delay
- Single-link and multi-link operation.
- SPI4 transmit data protocol support logic
  - Combines the data and control words from the transmit FIFO (DPRAMs) into the SPI4 format
  - Performs DIP-4 calculation over data and control words on the TX side and inserts into the payload control word
- Handles all credit calculations based on the status information automatically
- Provides optional signals to FPGA interface logic for flow control:
  - Current transmit Port ID (Calendar Port or user specified port # per calendar port)
  - Current BURST_VAL Parameter for that Port
  - Status from that Port
- Embedded Calendar-based port polling sequence mechanism and bandwidth allocation for all 256 ports
  - Programmable transmit and receive calendar tables support up to 256 ports
- Two calendars are supported in each direction
  - Main Calendar (1K deep)
  - Shadow Calendar (also 1K deep). User can reconfigure second calendar while operating off main calendar, and then switch on the next cycle to allow hitless operation
  - All calendar configuration parameters specified in the standard (CALENDAR_LEN, CALENDAR_M, etc.) are supported
- Transmit and Receive Status FIFOs provided to store flow control information for up to 256 ports.
  - Performs Status frame creation
  - DIP-2 odd parity calculated over the status frames
  - Supports either quarter-rate LVDS or LVTTL status channels
- Support for various options for flow control status creation, selectable per port:
  - Based on DPRAM FIFO fill levels
  - Based on status from FPGA interface per port
  - Both of the above
- Dual-port RAM interface to the FPGA supports flexible data widths for both the receive and transmit FPGA/core interfaces.
  - Scalable data bus enables users to configure TX interface for their respective port bandwidth requirements
  - A total of 4 DPRAM banks where each of the DPRAMs can be logically partitioned into 1, 2, 4, or 8 virtual FIFOs
  - Used for temporary storage and clock domain crossing
  - Can be configured to provide 32-, 64-, 128-bit data bus interfaces from the FPGA (plus accompanying control signals)
  - 32-bit mode: Four banks are separate and accessed independently
  - 64-bit mode: Banks 0 & 1 become a single aggregation and Banks 2 & 3 become a single aggregation
  - 128-bit mode: All four banks become a single aggregation
  - Mixed mode: One 64-bit (two banks become a single aggregation) and two banks are separate and accessed independently
- Training pattern generation
  - User controlled “alpha” repetitions of training pattern in DATA_MAX_T intervals
  - Automatic generation of training pattern during loss of synchronization

- Automatic idle generation
  - When no data for a given channel is available for transmit
  - If the receiver on other end of the link is “satisfied” for this channel
- Automatic training pattern and idle deletion in receive path
- Low-power, high performance ASIC LVDS I/Os compliant with EIA®-644
  - I/O buffers support hot insertion
  - I/O buffers proven to operate at over 900 MHz rates (Lattice ORLI10G FPSC uses same LVDS buffers)
  - On-chip center tap termination for common mode noise reduction
- Configuration options as suggested in the OIF-SPI4-02.0 standard are supported to configure parameters such as maximum burst size, calendar length, length of training sequence, etc.
- Support for three forms of loopback:
  - High-speed near end loopback which involves looping back data from the high-speed transmit block serial output to the high-speed receive block serial input. All of the logic up to the LVDS buffers is included in the loopback path. The LVDS buffers are bypassed
  - Far end loopback which involves looping back the 128-bit output data from high-speed receive block to the 128-bit input of the high-speed transmit block. Data is received at the high-speed SPI4 RX interface and transmitted at the SPI4 TX interface. The transmit protocol, receive protocol and DPRAM blocks are bypassed. This works for both static and dynamic alignment modes.
  - Low-speed near end loopback which excludes the high-speed blocks from the loopback path. This involves sourcing data from the FPGA, looping back the output of the transmit protocol block into the receive protocol block and observing data at the core-FPGA boundary
- Support for several SPI4 debug options:
  - Under software control, DIP-4 errors can be forced by inverting the DIP-4 parity bits
  - DIP-2 errors can be forced by inverting the DIP-2 parity bits
  - Eight-bit counters are provided for counting DIP-4 and DIP-2 errors
- SPI4 Status Reporting Capabilities:
  - Status information is reported through status registers.
  - Most conditions can also cause an alarm (interrupt) to be generated
  - DIP-4, DIP-2 errors
  - Deskew error from high-speed RX side
  - DPRAM Virtual FIFO overruns

Figure 3. ORSPI4 SPI4 Interface Block - Top Level Functional Partitioning



At the embedded core/FPGA interface, data buffering is provided by banks of DPRAM partitioned into FIFOs. FIFO reads and writes are completely decoupled. Data and accompanying address, packet delineation and error identification information are written into the selected FIFO as received - either from the FPGA, in the transmit case, or from the receive link. For transmit, reads are performed from the FIFOs based on pre-programmed packet format information, a pre-programmed schedule for link access as read from calendar logic, and far end status information as received from the transmit status logic. In the receive direction, the receive status logic transmits information concerning the states of the receive buffers on the receive status links, while the FPGA logic reads data from the FIFOs as needed under control of the FPGA logic.

The read/write control functions are similar if operating with external RAM. In this case, the internal DPRAM can be used as clock domain crossing FIFOs.

Formatting/deformatting, flow control processing, and error control logic forms the interface between the DPRAM banks and the SPI4 transmit and receive blocks. This logic performs the necessary conversions between the SPI4 and FPGA/core interface formats. It also performs DIP-2 (status) and DIP-4 (control) generation/checking. Finally, the SPI4 interface blocks perform the MUX/DEMUX functions for rate conversion between the internal core data paths and the SPI4 links and also provides the needed LVDS driver and receiver functions. Either static or dynamic alignment is available at the receiver interface. Dynamic alignment is used to compensate for bit-to-bit skew at higher data rates where it becomes difficult to meet tight setup and hold timing requirements.

SPI4 Transmit Path Overview

The first of the major blocks in the Transmit section contains four DPRAM banks which can be configured to provide 32-bit, 64-bit or 128-bit data bus interfaces from the FPGA to the embedded core. Providing a scalable data bus enables users to tailor the transmit interface to meet their port bandwidth requirements. For example, with a POS-PHY Level 3 (PL3) interface supporting multiple PHYs (ports), a single 32-bit interface to the Transmit DPRAM is required. For an Ethernet 10 Gbits/s interface, a single port will require a single 128-bit interface to the Transmit DPRAM.

To realize the various data bus interfaces or aggregation modes, the user must configure the mode within the embedded core via the MPI interface or the system bus. Multiple DPRAM banks can be aggregated into larger FIFOs. Division of the DPRAM banks into virtual partitions (up to eight) is also possible.

The FPGA logic initiates a write to DPRAM by providing Data, Port ID, 3-bit FIFO Address and Write Enable signals to the SPI4 block. The internal FIFO controller latches the data and port control information into a temporary hold register that stores the data until an entire 128-bit line is captured, or an EOP is asserted. The 128-bit line is then written into the selected virtual FIFO.

Associated with each FPGA data write interface, there are also control information signals and a transmit clock. The FIFO control logic transparently passes the control information to the Control memory, with the exception of the Byte Enable bits (BE[3:0]), which indicate which bytes of the associated 32-bit Word are valid.

The DPRAM read logic blocks poll port data from the DPRAM banks, based on a preconfigured calendar sequence and the current status of each active port. The SPI4 calendar is a mechanism that maintains out-of-band statistics of the current status of each port supported across the SPI4 interface. The calendar is a reverse direction flow-control mechanism used to control the dynamic bandwidth allocated for the each supported port. By periodically providing far end receive status for each port, the transmitter can modulate the amount of bandwidth allocated to a particular port dynamically.

Writes to the DPRAMs from the FPGA logic are asynchronous to the calendar polling algorithm. The SPI4 transmit logic reads data from the DPRAMs according to a strict calendar sequence algorithm and will generally not read port data from the virtual FIFOs in the sequence it was written.

Both a main and a shadow calendar are provided and are each 1K deep. This enables the user to provide finer granularity of the polling sequence based on bandwidth allocated for each port. The length of the calendar table (CALENDAR_LEN) is programmable. CALENDAR_LEN should be at least as large as the number of active ports (channels) in the system and should not exceed the upper threshold set by the parameter (MAX_CALENDAR_LEN).

There are two basic modes supported for transmitting data. Within the SPI4 core, the embedded core operates identically for all modes. At the FPGA interface, processing will be done slightly differently, depending upon the mode the user requires. Each mode is discussed below.

- Embedded memory mode - This mode is used when the ORSPI4 is interfacing to asynchronous FPGA interfaces, such as POS-PHY Level 3, 1GbE, Utopia Level 3, etc. and storing the data in the virtual DPRAM FIFOs. When operating in this mode, the SPI4 transmit logic will read port data from the FIFOs according to the calendar sequence. If there is no data, it will send idle data and advance to the next port. It is the user's responsibility to ensure the proper port data has been written to the virtual FIFO.
- External memory mode - This mode is used in conjunction with the Memory Controller or some other external memory based interface where data is available only after some fixed delay. In this mode the SPI4 transmit logic instructs the FPGA as to what port data to retrieve as well as how many bursts of data to retrieve. The FPGA is responsible to write the data read from the Memory Controller into the DPRAMs. Data is read from the DPRAM devices by the SPI4 transmit logic according to the transmit calendar.

The DPRAM read logic also includes a Port Descriptor Memory (PDM) which is a user configurable memory containing a list of read control parameters for all enabled ports to be polled. The depth of the memory is 256 locations,

which corresponds to the maximum number of ports that are supported by SPI4. The PDM data is comprised of three separate segments - a 10-bit dynamic table maintained by the SPI4 logic, a static 20-bit table, and a dynamic 3-bit register file written by the FIFO Status Update (FSU) logic. The PDM provides a mapping of the SPI4 port number to the FPGA interface device/port number, removing the burden from FPGA logic.

When port data is read from the PDM, a status update bit (the U-bit) is first examined to see whether the STAT field is new or stale. If stale, then the STAT field is not considered for the rest of processing. If the STAT field is new (U-bit=1) the STAT field is used in conjunction with other field to calculate what the new Credit field for the port should be.

A SATISFIED status indicates the corresponding port's FIFO is almost full, and only transfers using the remaining previously granted 16-byte blocks (if any) may be sent to corresponding port until the next status update. No additional transfers to that port are permitted.

When a HUNGRY status indication is received, transfers up to MAXBURST2 16-byte blocks or the remainder of what was previously granted (whichever is greater), may be sent to the corresponding port prior to the next status update. A STARVING status indication indicates that buffer underflow is imminent in the corresponding PHY port. When STARVING is received, transfers for up to MAXBURST1 16-byte blocks may be sent to the corresponding port prior to the next status update.

If the U-bit is cleared, this indicates the STAT field has already been used to update the Credit field on a previous Port servicing. Therefore, the Credit field should simply be reduced by BURST_VAL. Otherwise, the Credit field is updated to the new Credit value minus BURST_VAL. In both cases, the output of the logic is used to update the Credit field. If the Credit field is zero, and the STAT field is stale, then the port receives no service. Read accesses of the port control information need to be optimized to minimize any lost bandwidth due to the Credit field having a value of zero.

Data read from the DPRAMS is sent to the SPI4 transmit block which is responsible for the following functions:

- Combining the data and control words from the Transmit FIFO into the data format specified in the OIF SPI4 standard.
- DIP-4 calculation and insertion into the payload control word.
- Generation of idle/training control words in programmable intervals.

Training words are used to dynamically align the far end receiver. As long as a disabled status ‘11’ is received on the SPI4 status channel, the transmit interface block sends continuous training patterns (10 training control words followed by 10 training data words). When valid status is received on the status channel, user data is normally sent on the SPI4 data link. However, users can also periodically schedule training patterns in SDATA_MAX_T periods. The training patterns can be repeated “ALPHA” times. Both “ALPHA” and DATA_MAX_T are programmable control register bits.

The SPI4 transmit block contains the high-speed serializer which uses the x8 clock, synthesized by an internal PLL, to generate the high-speed data from the low-speed 128-bit FIFO data. Data is transmitted off-chip using a 16-bit LVDS data bus - TDAT[15:0], a LVDS control bit - TCTL, and a source synchronous clock - TDCLK.

The 16-bit data bus and control are DDR with respect to TDCLK. In order to support 10 Gbits/s throughput, the minimum frequency of TDCLK needs to be 622 Mbits/s (311 MHz DDR). To allow considerable margin above this minimum data rate a maximum frequency of operation of 900 Mbits/s is supported.

The Transmit Status Protocol (S4TSP) block provides the interface to the SPI4 Transmit Status interfaces. These signals can be either LVDS or LVCMOS buffers. The S4TSP block is responsible for the following functions:

- FIFO Status Decoding and Buffering.
- Framing using the status framing pattern.
- DIP-2 checking of incoming status information.

The FIFO Status Update logic block reads the Port and Status information and uses this information to update Port Descriptor Memory STAT field. Whenever a valid STAT field has been updated, the associated U-bit field is set as discussed previously. This indicates that the STAT field is new and that the Credit field for that must be re-evaluated the next time it is selected as a source for transmit data.

SPI4 Receive Path Overview

In the receive direction, data is received in SPI4 format on the LVDS I/Os at the receive interface. The data is written into DPRAM as received and read from the DPRAMs as requested by the FPGA logic. Control information is also interpreted and buffered and idles and training sequences are removed from the incoming data stream.

Receive FIFO status is transmitted from the Receive Status interface according to a pre-configured polling sequence contained within the Receive Calendar. Data is formatted into the SPI4 Receive Status format and sent to the physical links as either LVDS or LVTTL signals.

The SPI4 block contains the high-speed receive logic. Incoming LVDS signals, in SPI4 format, include the 16-bit data bus (RDAT[15:0]), a control bit (RCTL) and a source synchronous DDR clock (RDCLK). The incoming data is deserialized to a 128-bit format and the control information is converted to an 8-bit format.

The SPI4 receive block also detects training patterns and performs dynamic alignment of the incoming data. At speeds above 700 Mbit/s (350 MHz) it becomes necessary to use dynamic alignment. Skews of up to ± one clock period can be compensated by the dynamic alignment logic. For low speed incoming data, static alignment can be chosen through a programmable control bit. Various timing options of receive data vs. receive clock are also programmable.

The SPI4 block is responsible for decoding the in-band control information. It then forwards both the data and control information, such as link address, SOP, EOP and error, to the virtual FIFOs. The SPI4 block also parses the control words embedded within the incoming data. Using this control information, it performs the following functions:

- Checks DIP-4 parity
- Monitors for continuous alignment (if more than a programmable number of DIP-4 parity errors exist, there may be an alignment problem).
- Removes idle/training words.
- Extracts link address and SOP, EOP and valid packet (no error) signals.

In the receive direction there are also four Dual Port Memory (DPRAM) banks that contain a total of 8K bytes available for clock domain crossing and/or temporary buffering. As with the transmit buffers, each bank can be further partitioned up to 8 virtual memories, one for each of 8 ports. The following are the characteristics of the DPRAM virtual FIFOs:

- The DPRAM memories support asynchronous reads. Each DPRAM bank can be accessed on the FPGA side with an individual clock.
- For data buffering beyond 32 ports, the DPRAM banks can be used as clock domain crossing FIFOs before writing the data and control information into an external memory. If fewer ports are supported, the virtual memories can be aggregated, providing more buffer space for each port.
- Each DPRAM bank has a 32-bit data and 8-bit control read interface to the FPGA. When using the DPRAM memories, the data can be read as either a 32-, 64-, or 128-bit data bus with associated control signals.
- At any time, the user can poll the status of a FIFO within a DPRAM bank by providing just the read address without a valid read enable.
- A FIFO empty flag is generated by the read control logic to the FPGA. This empty flag can be programmed to indicate truly empty or < 1/4 full (1/4 full - 1).

In addition to formatting received data and sending it to the FPGA logic, the receive block also sends status information to the SPI4 status interface.

The Port Status Sequencer (PSS) block is responsible for providing port status to the SPI4 Receive Status block logic according to a pre-configured calendar sequence. Status is derived from the fill-levels of the DPRAM FIFOs and/or from the FPGA status interface.

The SPI4 Receive Status block is responsible for FIFO status encoding, calendar management, status pattern encoding (sync bits “11”), DIP-2 calculation and optional calendar selection word encoding.

The SPI4 Receive Status block contains the low speed LVTTL output buffers and LVDS output buffers necessary for the output stage of receive status logic. The option to choose between LVTTL or LVDS outputs is done by setting a control register bit.

SPI4 Debugging and Statistics Gathering Support

There are also several other features, including three loopback modes incorporated into the embedded core to assist in debugging and statistic gathering. These features involve both the transmit and receive paths.

The three forms of loopback supported directly are:

- High-speed near-end loopback
- Far-end high-speed loopback
- Low-speed near-end loopback

The SPI4 blocks support the following error insertion and status reporting options for testing:

- DIP-4 odd parity is calculated over data and control words and inserted on the TX side. DIP-4 errors can be forced by inverting the DIP-4 parity bits. DIP-4 parity is then checked at the receive interface.
- DIP-2 odd parity is calculated over the status frames and inserted on the RX side. DIP-2 errors can be forced by inverting the DIP-2 parity bits. DIP-2 parity is then checked at the transmit status interface.
- Eight-bit counters are provided for counting DIP-4 and DIP-2 errors.
- Deskew error reporting for high-speed RX side dynamic alignment. This can cause an alarm.
- DPRAM FIFO overrun reporting. These can cause an alarm.

Memory Controller - Overview

The Memory Controller block controls an interface to external Quad Data Rate (QDRII) SRAM for data buffering between the FPGA logic and external memory. The key features of the Memory Controller interface are described below:

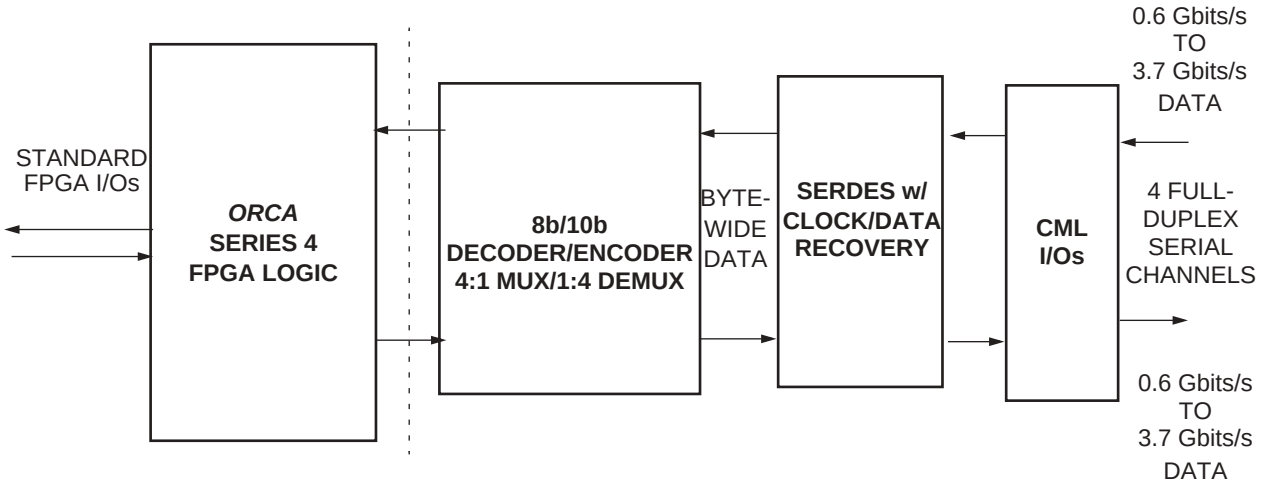
Memory Controller Features

- Independent Memory Controller interface to external Quad Data Rate (QDRII) SRAMs from multiple suppliers for data buffering.
  - Provides additional packet buffering for > 32 ports
  - Provides traffic smoothing for any number of ports
- The Controller supports a throughput of greater than 20 Gbits/s so that all the data received on the SPI4 interface at 10 Gbits/s can be buffered.
- The QDRII SRAM supports this throughput with 36 unidirectional data lines in both the read and write directions.
- The controller block provides the ability to access external QDRII SRAM through the FPGA.
  - A set of 72 data signals across the core-FPGA interface
  - Of the 72, 8 signals can be either used for parity or data.
  - Simple asynchronous FIFO interface to FPGA for ease of design. A high-speed clock signal is provided to the FPGA as an option to make the write and read synchronous, if desired.
- The core passes the data transparently to and from the QDRII SRAM in two-word or four-word bursts. Interfaces to memory are 36 bits wide and the address buses are 18 bits wide.
  - Supports the interfaces required for a 512K x 36 bit (18 Mbit) QDRII SRAM in two-word burst mode.
  - Only 17 address lines are required in four-word burst mode.
- Status/Alarm reported to user through registers
  - Data length mismatch from the write controller state machine
  - Data instruction coherency error
  - Write data, Read data FIFO overrun and underrun errors
- Additional high-speed Memory Controller can be implemented in FPGA gates if required.

SERDES Logic Block - Overview

The SERDES logic block in of the ORSPI4 contains four Clock and Data Recovery (CDR) macrocells and four Serializer/Deserializer (SERDES) macrocells to support four channels of 8b/10b (*IEEE* 802.3.2002) encoded serial links. The logic block also contains Fiber Channel and XAUI-based state machines, logic to support multi-channel alignment and MUX/DEMUX logic for the FPGA/core interface. Figure 4 shows the SERDES top level block diagram and the basic data flow. Boundary scan for the SERDES only includes programmable I/Os and does not include any of the embedded block I/Os.

Figure 4. SERDES Top Level Block Diagram.



The serial channels can each operate at up to 3.7 Gbits/s (2.96 Gbits/s data rate) with a full-duplex synchronous interface with built-in clock recovery (CDR). The 8b/10b encoding provides guaranteed ones density for the CDR, byte alignment, and error detection. The core is also capable of frame synchronization and physical link monitoring. An overview of the individual blocks in the embedded core is presented in the following paragraphs. The SERDES portion of the core contains a quad transceiver block for serial data transmission at a selectable data rate of 0.6 to 3.7 Gbits/s. Each SERDES channel features high-speed 8b/10b parallel I/O interfaces to other core blocks and high-speed CML interfaces to the serial links.

Serializer and Deserializer (SERDES)

The SERDES portion of the core contains a transceiver block for serial data transmission at a selectable data rate of 0.6-3.7 Gbits/s. Each SERDES channel features high-speed 8b/10b parallel I/O interfaces to other core blocks and high-speed CML interfaces to the serial links.

The SERDES circuitry consists of receiver, transmitter, and auxiliary functional blocks. The receiver accepts high-speed (up to 3.7 Gbits/s) serial data. Based on data transitions, the receiver locks an analog receive PLL for each channel to retime the data, then demultiplexes the data down to parallel bytes and an accompanying clock.

The transmitter operates in the reverse direction. Parallel bytes are multiplexed up to 3.7 Gbits/s serial data for off-chip communication. The transmitter generates the necessary clocks from a lower speed reference clock.

The transceiver is controlled and configured through the system bus in the FPGA logic and through the external 8-bit microprocessor interface of the FPGA. Each channel has associated dedicated registers that are readable and writable. There are also global registers for control of common circuitry and functions.

The SERDES performs 8b/10b encoding and decoding for each channel. The 8b/10b transmission code can support either Ethernet or Fibre Channel specifications for serial encoding/decoding, special characters, and error detection.

The user can disable the 8b/10b decoder to receive raw 10-bit words, which will be rate reduced by the SERDES. If this mode is chosen, the user must also bypass the multichannel alignment FIFOs.

The SERDES macrocell contains its own dedicated PLLs for both transmit and receive clock generation. The user provides a reference clock of the appropriate frequency. The receiver PLLs extract the clock from the serial input data and re-time the data with the recovered clock.

MUX/DEMUX Block

The MUX/DEMUX logic converts the data format for the high-speed serial links to a wide, low-speed format for crossing the CORE/FPGA interface. The intermediate interface to the SERDES macrocell runs at 1/10th the bit

rate of the data lane. The MUX/DEMUX converts the data rate and bus width so the interface to the FPGA core can run at 1/4th this intermediate frequency, giving a range of 25.0 to 92.5 MHz for the data rates into and out of the FPGA logic.

Multi-Channel Alignment FIFOs
In the ORSPI4 SERDES block, the four incoming data channels can be independent of each other or can be synchronized in several ways. Two channels within a SERDES block can be aligned together; channels A and B and/or channels C and D. Finally, four channels in a SERDES block can be aligned together to form a communication channel with a bandwidth of 10 Gbits/s. Individual channels within an alignment group can be disabled (i.e., powered down) without disrupting other channels.

XAUI and Fibre Channel Link State Machines
Two separate link state machines are included in the architecture. A XAUI-based link state machine is included in the embedded core to implement the IEEE 802.3ae standard. A separate state machine for Fibre Channel is also implemented.

FPGA/Embedded Core Interface
In 8b/10b mode, the FPGA logic will receive/transmit 32-bits of data (up to 92.5 MHz) and four K_CTRL bits from/to the embedded core. There are 4 data streams in each direction plus additional timing, status and control signals.

Data sent to the FPGA can be aligned using comma (/K/) characters or /A/ character as specified either by Fibre Channel or by IEEE 802.3ae for XAUI based interfaces. The alignment character is made available to the FPGA along with the data. The special characters K28.1, K28.5 and K28.7 are treated as valid comma characters by the SERDES.

If the receive channel alignment FIFOs are bypassed, then each channel will provide its own receive clock in addition to data and comma character detect signals. If the 8b/10b decoders are bypassed, then 40-bit data streams are passed to the FPGA logic. No channel alignment can be done in 8b/10b bypass mode.

SERDES Features

- Four channels of 0.6-3.7G SERDES with 8b/10b encoding/decoding are supported. The SERDES quad is IEEE P802.3ae/D4.01 XAUI based and also supports the FC (ANSI X3.230:1994) link synchronization state machine specification.
- The high-speed SERDES are programmable and support serial data rates including 622 Mbits/s, 1.0 Gbits/s, 1.25 Gbits/s, 2.5 Gbits/s, 3.125 Gbits/s, and 3.7 Gbits/s. Operation has been demonstrated on design tolerance devices at 3.7 Gbits/s across 26 in. of FR-4 backplane and at 3.2 Gbits/s across 40 in. of FR-4 backplane across temperature and voltage specifications.
- Asynchronous operation per receive channel, with the receiver frequency tolerance based on one reference clock per four channels (separate PLL per channel).
- Ability to select full-rate or half-rate operation per transmit or receive channel by setting the appropriate control registers.
- Programmable one-half amplitude transmit mode for reduced power in chip-to-chip application.
- Transmit preemphasis (programmable) for improved receive data eye opening.
- 32-bit (8b/10b) or 40-bit (raw data) parallel internal bus for data processing in FPGA logic.
- Provides a 10 Gbits/s backplane interface to a switch fabric using four 2.5 Gbit/s links. Also supports port cards at 2.5 Gbit/s.
- 3.125 Gbits/s SERDES compliant with XAUI serial data specification for 10 Gigabit Ethernet applications.
- Most XAUI features for 10 Gigabit Ethernet are embedded including the required link state machine.
- Compliant to Fibre Channel physical layer specification.

- High-Speed Interface (HSI) function for clock/data recovery serial backplane data transfer without external clocks.
- Four-channel HSI functions provide 2.96 Gbits/s serial user data interface per channel (8b/10b encoding and decoding) for a total chip bandwidth of > 10 Gbits/s (full duplex).
- SERDES have low-power CML buffers and support 1.5 V or 1.8 V I/Os. This allows use of the SERDES with optical transceiver, coaxial copper media, shielded twisted pair wiring or high-speed backplanes such as FR-4.
- Powerdown option of SERDES HSI receiver or transmitter is on a per-channel basis.
- Automatic lock to reference clock in the absence of valid receive data.
- High-speed and low-speed loopback test modes.
- No external components required for clock recovery and frequency synthesis.
- Built-in boundary scan (*IEEE* ® 1149.1 and 1149.2 JTAG) for the programmable I/Os, not including the SERDES interface.
- FIFOs can align incoming data either across groups of four channels or groups of two channels. Alignment is done either using comma characters or by using the /A/ character in XAUI mode. Optional ability to bypass the alignment FIFOs for asynchronous operation between channels (Each channel includes its own clock and frame pulse or comma detect).

FPGA Logic Overview

The *ORCA* Series 4 architecture is a new generation of SRAM-based programmable devices from Lattice. It includes enhancements and innovations geared toward today's high-speed systems on a single chip. Designed with networking applications in mind, the Series 4 family incorporates system-level features that can further reduce logic requirements and increase system speed. *ORCA* Series 4 devices contain many new patented enhancements and are offered in a variety of packages and speed grades.

The hierarchical architecture of the logic, clocks, routing, RAM, and system-level blocks create a seamless merge of FPGA and ASIC designs. Modular hardware and software technologies enable System-on-Chip integration with true plug-and-play design implementation.

The architecture consists of the following basic elements: Programmable Logic Cells (PLCs), Programmable I/O cells (PIOs), Embedded Block RAMs (EBRs), plus supporting system-level features. These elements are interconnected with a rich routing fabric of both global and local wires. An array of PLCs is surrounded by common interface blocks that provide an abundant interface to the adjacent PLCs or system blocks. Routing congestion around these critical blocks is eliminated by the use of the same routing fabric implemented within the programmable logic core.

Each PLC contains a Programmable Function Unit (PFU), Supplemental Logic Interconnect Cell (SLIC), local routing resources, and configuration RAM. Most of the FPGA logic is performed in the PFU, but decoders, *PAL*-like functions, and 3-state buffering can be performed in the SLIC. The PIOs provide device inputs and outputs and can be used to register signals and to perform input demultiplexing, output multiplexing, uplink and downlink functions, and other functions on two output signals.

Large blocks of 512 x 18 quad-port RAM complement the existing distributed PFU memory. The RAM blocks can be used to implement RAM, ROM, FIFO, multiplier, and CAM. Some of the other system-level functions include the MPI, PLLs, and the Embedded System Bus (ESB).

Detailed descriptions of the FPGA logic blocks can be found in the ***ORCA* Series 4 FPGA Datasheet**.

Programmable Logic Features

- High-performance programmable logic:
  - 0.16 µm, 7-level metal technology.
  - Internal performance of >250 MHz.
  - Over 600K usable system gates.
  - Meets multiple I/O interface standards.
  - 1.5 V operation (30% less power than 1.8 V operation), translates to greater performance.
- Traditional I/O selections:
  - LVTTL (3.3V) and LVCMOS (2.5 V and 1.8 V) I/Os.
  - Per pin-selectable I/O clamping diodes provide 3.3 V PCI compliance.
  - Individually programmable drive capability:
    24 mA sink/12 mA source, 12 mA sink/6 mA source, or 6 mA sink/3 mA source.
  - Two slew rates supported (fast and slew-limited).
  - Fast-capture input latch and input Flip-Flop (FF)/latch for reduced input setup time and zero hold time.
  - Fast open-drain drive capability.
  - Capability to register 3-state enable signal.
  - Off-chip clock drive capability.
  - Two-input function generator in output path.
- New programmable high-speed I/O:
  - Single-ended: GTL, GTL+, PECL, SSTL3/2 (Class I and II), HSTL (Class I, III, IV), ZBT, and DDR.
  - Double-ended: LVDS, bused-LVDS, and LVPECL. Programmable, (on/off) internal parallel termination (100 Ω) is also supported for these I/Os.
- New capability to (de)multiplex I/O signals:
  - New DDR on both input and output at rates up to 350 MHz (700 Mbits/s effective rate).
  - New 2x and 4x downlink and uplink capability per I/O (i.e., 50 MHz internal to 200 MHz I/O).
- Enhanced twin-quad Programmable Function Unit (PFU):
  - Eight 16-bit Look-Up Tables (LUTs) per PFU.
  - Nine user registers per PFU, one following each LUT, and organized to allow two nibbles to act independently, plus one extra for arithmetic operations.
  - New register control in each PFU has two independent programmable clocks, clock enables, local SET/RESET, and data selects.
  - New LUT structure allows flexible combinations of LUT4, LUT5, new LUT6, 4 → 1 MUX, new 8 → 1 MUX, and ripple mode arithmetic functions in the same PFU.
  - 32 x 4 RAM per PFU, configurable as single- or dual-port. Create large, fast RAM/ROM blocks (128 x 8 in only eight PFUs) using the Supplemental Logic and Interconnect Cell (SLIC) decoders as bank drivers.
  - Soft-Wired LUTs (SWL) allow fast cascading of up to three levels of LUT logic in a single PFU through fast internal routing which reduces routing congestion and improves speed.
  - Flexible fast access to PFU inputs from routing.
  - Fast-carry logic and routing to all four adjacent PFUs for nibble-wide, byte-wide, or longer arithmetic functions, with the option to register the PFU carry-out.
- Abundant high-speed buffered and non-buffered routing resources provide 2x average speed improvements over previous architectures.
- Hierarchical routing optimized for both local and global routing with dedicated routing resources. This results in faster routing times with predictable and efficient performance.
- SLIC provides eight 3-state buffers, up to a 10-bit decoder, and *PAL*™-like AND-OR-Invert (AOI) in each programmable logic cell.
- New 200 MHz embedded quad-port RAM blocks, 2 read ports, 2 write ports, and 2 sets of byte lane enables. Each embedded RAM block can be configured as:
  - 1—512 x 18 (quad-port, two read/two write) with optional built in arbitration.

  - 1—256 x 36 (dual-port, one read/one write).
  - 1—1K x 9 (dual-port, one read/one write).
  - 2—512 x 9 (dual-port, one read/one write for each).
  - 2 RAMS with an arbitrary number of words whose sum is 512 (or less) x 18 (dual-port, one read/one write).
  - Supports joining of RAM blocks.
  - Two 16 x 8-bit content addressable memory (CAM) support.
  - FIFO 512 x 18, 256 x 36, 1K x 9, or dual 512 x 9.
  - Constant multiply (8 x 16 or 16 x 8).
  - Dual variable multiply (8 x 8).
- Embedded 32-bit internal system bus plus 4-bit parity interconnects FPGA logic, MicroProcessor interface (MPI), embedded RAM blocks, and embedded standard cell blocks with 100 MHz bus performance. Included are built-in system registers that act as the control and status center for the device.
- Built-in testability:
  - Full boundary scan (*IEEE* 1149.1 and Draft 1149.2 JTAG).
  - Programming and readback through boundary scan port compliant to *IEEE* Draft 1532:D1.7.
  - TS_ALL testability function to 3-state all I/O pins.
  - New temperature-sensing diode.
- Improved built-in clock management with Programmable Phase-Locked Loops (PPLLs) provide optimum clock modification and conditioning for phase, frequency, and duty cycle from 20 MHz up to 420 MHz. Multiplication of the input frequency up to 64x and division of the input frequency down to 1/64x possible.
- New cycle stealing capability allows a typical 15% to 40% internal speed improvement after final place and route. This feature also enables compliance with many setup/hold and clock to out I/O specifications and may provide reduced ground bounce for output buses by allowing flexible delays of switching output buffers.

Programmable Logic System Features

- PCI local bus compliant for FPGA I/Os.
- Improved *PowerPC*® 860 and *PowerPC* II high-speed synchronous microprocessor interface can be used for configuration, readback, device control, and device status, as well as for a general-purpose interface to the FPGA logic, RAMs, and embedded standard cell blocks. Glueless interface to synchronous *PowerPC* processors with user-configurable address space provided.
- New embedded *AMBA* ™ specification 2.0 AHB system bus (*ARM* ® processor) facilitates communication among the microprocessor interface, configuration logic, Embedded Block RAM, FPGA logic, and embedded standard cell blocks.
- Variable size bused readback of configuration data capability with the built-in microprocessor interface and system bus.
- Internal, 3-state, and bidirectional buses with simple control provided by the SLIC.
- New clock routing structures for global and local clocking significantly increases speed and reduces skew (<200 ps for OR4E04).
- New local clock routing structures allow creation of localized clock trees.
- Two new edge clock routing structures allow up to six high-speed clocks on each edge of the device for improved setup/hold and clock to out performance.
- New Double-Data Rate (DDR) and Zero-Bus Turn-around (ZBT) memory interfaces support the latest high-speed memory interfaces.
- New 2x/4x uplink and downlink I/O capabilities interface high-speed external I/Os to reduced speed internal logic.
- Meets Universal Test and Operations PHY Interface for ATM (UTOPIA) levels 1, 2, and 3; as well as POS-PHY3.

PLC Logic
Each PFU within a PLC contains eight 4-input (16-bit) LUTs, eight latches/FFs, and one additional Flip-Flop that may be used independently or with arithmetic functions.

The PFU is organized in a twin-quad fashion; two sets of four LUTs and FFs that can be controlled independently. Each PFU has two independent programmable clocks, clock enables, local set/reset, and data selects. LUTs may also be combined for use in arithmetic functions using fast-carry chain logic in either 4-bit or 8-bit modes. The carry-out of either mode may be registered in the ninth FF for pipelining.

Each PFU may also be configured as a synchronous 32 x 4 single- or dual-port RAM or ROM. The FFs (or latches) may obtain input from LUT outputs or directly from invertible PFU inputs, or they can be tied high or tied low. The FFs also have programmable clock polarity, clock enables, and local set/reset.

The SLIC is connected from PLC routing resources and from the outputs of the PFU. It contains eight 3-state, bidirectional buffers, and logic to perform up to a 10-bit AND function for decoding, or an AND-OR with optional INVERT to perform *PAL*-like functions. The 3-state drivers in the SLIC and their direct connections from the PFU outputs make fast, true, 3-state buses possible within the FPGA, reducing required routing and allowing for real-world system performance.

Programmable I/O
The Series 4 PIO addresses the demand for the flexibility to select I/Os that meet system interface requirements. I/Os can be programmed in the same manner as in previous *ORCA* devices, with the additional new features which allow the user the flexibility to select new I/O types that support High-Speed Interfaces.

Each PIO contains four programmable I/O pads and is interfaced through a common interface block to the FPGA array. The PIO is split into two pairs of I/O pads with each pair having independent clock enables, local set/reset, and global set/reset. On the input side, each PIO contains a programmable latch/Flip-Flop, which enables very fast latching of data from any pad. The combination provides for very low setup requirements and zero hold times for signals coming on-chip. It may also be used to demultiplex an input signal, such as a multiplexed address/data signal, and register the signals without explicitly building a demultiplexer with a PFU.

On the output side of each PIO, an output from the PLC array can be routed to each output Flip-Flop, and logic can be associated with each I/O pad. The output logic associated with each pad allows for multiplexing of output signals and other functions of two output signals.

The output FF, in combination with output signal multiplexing, is particularly useful for registering address signals to be multiplexed with data, allowing a full clock cycle for the data to propagate to the output. The output buffer signal can be inverted, and the 3-state control can be made active-high, active-low, or always enabled. In addition, this 3-state signal can be registered or nonregistered.

The Series 4 I/O logic has been enhanced to include modes for speed uplink and downlink capabilities. These modes are supported through shift register logic, which divides down incoming data rates, or multiplies up outgoing data rates. This new logic block also supports high-speed DDR mode requirements where data is clocked into and out of the I/O buffers on both edges of the clock.

The new programmable I/O cell allows designers to select I/Os which meet many new communication standards permitting the device to hook up directly without any external interface translation. They support traditional FPGA standards as well as high-speed, single-ended, and differential-pair signaling. Based on a programmable, bank-oriented I/O ring architecture, designs can be implemented using 3.3 V, 2.5 V, 1.8 V, and 1.5 V referenced output levels.

Routing
The abundant routing resources of the Series 4 architecture are organized to route signals individually or as buses with related control signals. Both local and global signals utilize high-speed buffered and non-buffered routes. One PLC segmented (x1), six PLC segmented (x6), and bused half chip (xHL) routes are patterned together to provide high connectivity with fast software routing times and high-speed system performance.

Eight fully distributed primary clocks are routed on a low-skew, high-speed distribution network and may be sourced from dedicated I/O pads, PLLs, or the PLC logic. Secondary and edge-clock routing is available for fast regional clock or control signal routing for both internal regions and on device edges. Secondary clock routing can be sourced from any I/O pin, PLLs, or the PLC logic.

The improved routing resources offer great flexibility in moving signals to and from the logic core. This flexibility translates into an improved capability to route designs at the required speeds when the I/O signals have been locked to specific pins.

System-Level Features

The Series 4 also provides system-level functionality by means of its microprocessor interface, Embedded System Bus, quad-port Embedded Block RAMs, universal programmable Phase-Locked Loops, and the addition of highly tuned networking specific Phase-Locked Loops. These functional blocks allow for easy, glueless system interfacing and the capability to adjust to varying conditions in today's high-speed networking systems.

MicroProcessor Interface

The MPI provides a glueless interface between the FPGA and *PowerPC* microprocessors. Programmable in 8-bit, 16-bit, and 32-bit interfaces with optional parity to the *Motorola® PowerPC* 860 bus, it can be used for configuration and readback, as well as for FPGA control and monitoring of FPGA status. All MPI transactions utilize the Series 4 Embedded System Bus at 66 MHz performance.

A system-level microprocessor interface to the FPGA user-defined logic following configuration, through the system bus, including access to the Embedded Block RAM and general user-logic, is provided by the MPI. The MPI supports burst data read and write transfers, allowing short, uneven transmission of data through the interface by including data FIFOs. Transfer accesses can be single beat (1 x 4 bytes or less), 4-beat (4 x 4 bytes), 8-beat (8 x 2 bytes), or 16-beat (16 x 1 bytes).

System Bus

An on-chip, multimaster, 8-bit system bus with 1-bit parity facilitates communication among the MPI, configuration logic, FPGA control, and status registers, Embedded Block RAMs, as well as user logic. Utilizing the *AMBA* specification Rev 2.0 AHB protocol, the Embedded System Bus offers arbiter, decoder, master, and slave elements. Master and slave elements are also available for the user-logic and a slave interface is used for control and status of the embedded backplane transceiver portion of the ORSPI4.

The system bus control registers can provide control to the FPGA such as signaling for reprogramming, reset functions, and PLL programming. Status registers monitor INIT, DONE, and system bus errors. An interrupt controller is integrated to provide up to eight possible interrupt resources. Bus clock generation can be sourced from the microprocessor interface clock, configuration clock (for slave configuration modes), internal oscillator, user clock from routing, or from the port clock (for JTAG configuration modes).

Phase-Locked Loops

Four user PLLs are provided by the ORSPI4 FPSC. Programmable PLLs can be used to manipulate the frequency, phase, and duty cycle of a clock signal. Each PLL is capable of manipulating and conditioning clocks from 20 MHz to 420 MHz. Frequencies can be adjusted from 1/8x to 8x, the input clock frequency. Each programmable PLL provides two outputs that have different multiplication factors but can have the same phase relationships. Duty cycles and phase delays are programmable (12.5% steps of the clock period increments). An automatic input buffer delay compensation mode is available for phase delay. Each PPLL provides two outputs that can have programmable (12.5% steps) phase differences.

FPGA Configuration

The FPGA functionality is determined by internal configuration RAM. The FPGAs internal initialization/configuration circuitry loads the configuration data at power-up or under system control. The configuration data can reside externally in an EEPROM or any other storage media. Serial EEPROMs provide a simple, low pin-count method for configuring FPGAs.

The RAM is loaded by using one of several configuration modes. Supporting the traditional master/slave serial, master/slave parallel, and asynchronous peripheral modes, the Series 4 also utilizes its microprocessor interface and Embedded System Bus to perform both programming and readback. Daisy chaining of multiple devices and partial reconfiguration are also permitted.

Other configuration options include the initialization of the embedded-block RAM memories and FPSC memory, as well as system bus options and bit stream error checking. Programming and readback through the JTAG *(IEEE* 1149.2*)* port is also available, meeting In-System Programming (ISP) standards (*IEEE* 1532 Draft).

FPSC Configuration

Configuration of the ORSPI4 occurs in two stages: FPGA bit-stream configuration and embedded core setup.

Prior to becoming operational, the FPGA goes through a sequence of states, including power-up, initialization, configuration, start-up, and operation. The FPGA logic is configured by standard FPGA bit-stream configuration means as discussed in the *ORCA* Series 4 FPGA data sheet.

After the FPGA configuration is complete, the options for the embedded core are set based on the contents of registers that are accessed through the FPGA system bus. The system bus itself can be driven by an external *PowerPC* compliant microprocessor via the MPI block or via a user master interface in FPGA logic. A simple IP block that drives the system by using the user register interface and very little FPGA logic is available in the *MPI/System Bus Technical Note (TN-1017)*. This IP block sets up the embedded core via a state machine and allows the ORSPI4 to work in an independent system without an external microprocessor interface.

ORSPI4 Package Options

The ORSPI4 FPSC is available in two package options: a 1036 fpSBGA and a 1156 fpBGA. The 1036 pin package provides an OR4E06 FPGA array (16,192 LEs, 148 Kbits of Embedded Block RAM), 498 FPGA user I/Os, two SPI4 interfaces (or one SPI4 interface and a quad high-speed SERDES interface), and a high-speed QDRII SRAM Controller. The 1156 pin package provides an OR4E06 FPGA array (16,192 LEs, 148 Kbits of Embedded Block RAM), 356 FPGA user I/Os, two SPI4 interfaces (SERDES interface not available on this package), and a high-speed QDRII SRAM Controller.

Additional Information

Contact your local Lattice representative for additional information regarding the *ORCA* Series 4 FPGA devices, or visit our website at: **http://www.latticesemi.com/orca**

Links to Specs and Standards

Optical Internetworking Forum – OIF-SPI4-02.0 - www.oiforum.com

Fibre Channel Physical and Signaling Interface – FC-PH ANSI X3.230-1994 - www.t11.org

10 Gigabit Ethernet – IEEE P802.3ae - www.ieee.org

18Mb QDR-II SRAM 2-word burst – MT54W512H36B - www.cypress.com/products/micron/micron.cfm

18Mb QDR-II SRAM 4-word burst – MT54W512H36B - www.cypress.com/products/micron/micron.cfm

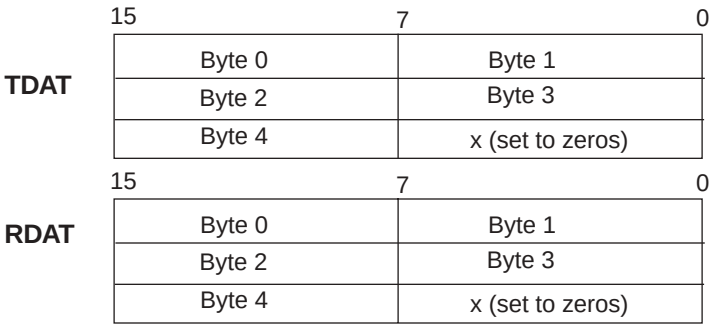
36Mb QDR-II SRAM 2-word burst – KTR323682M - www.samsung.com/Products/Semiconductor/SRAM/index.htm

36Mb QDR-II SRAM 4-word burst – KTR323684M - www.samsung.com/Products/Semiconductor/SRAM/index.htm

ORSPI4 SPI4 Data Formats

The data format across the SPI4 interface follows the OIF SPI4 convention where the lowest byte number occupies the highest bit positions within the 16-bit word as shown in Figure 5. On payload transfers that do not end on an even byte boundary, the unused byte (after the last valid byte) on bit positions 7 through 0 is set to all zeroes as shown in Figure 5.

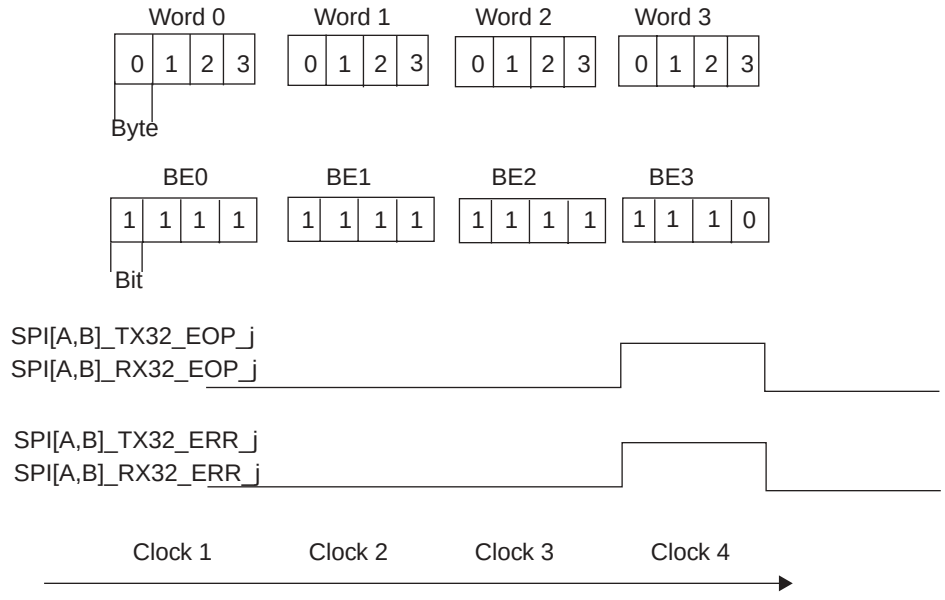
Figure 5. SPI4 Byte and Bit Ordering



At the FPGA-embedded core RX and TX interface, the same SPI4 convention is followed wherein the lowest byte number occupies the highest bit positions within a word. This is the same in all operating modes - 32-bit, 64-bit and 128-bit.

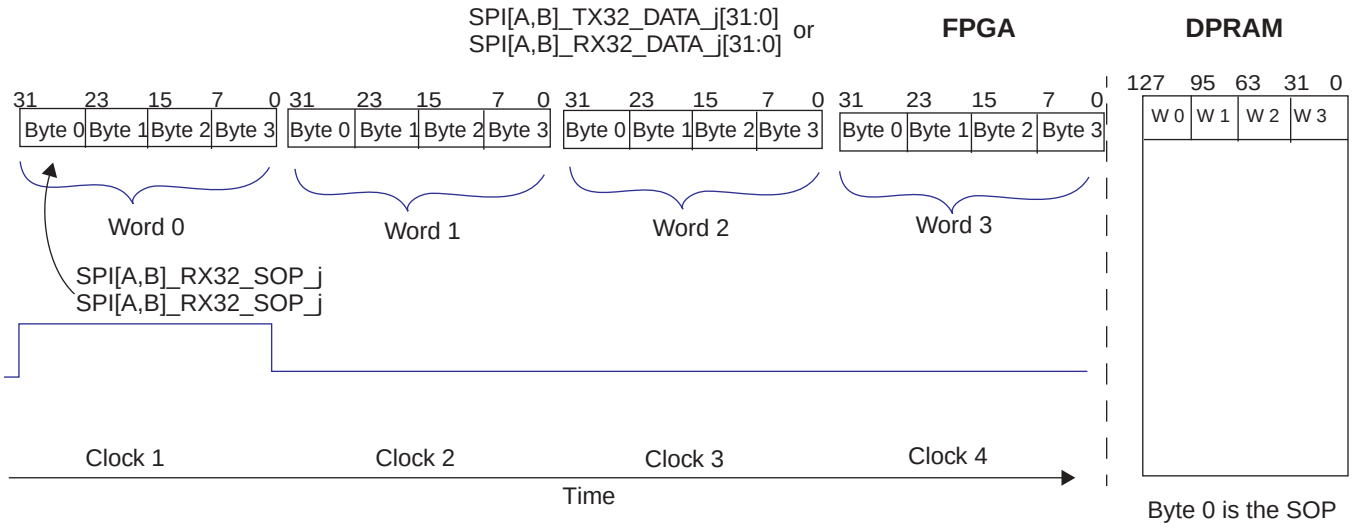
Byte enables for all data except EOP should be “1”. During an EOP, the last valid byte enable within a 32-bit, 64-bit or 128-bit word indicates the last byte for the packet. Figure 6 shows EOP signal being asserted during Word 3. In Figure 6, the last valid byte enable within “1110” in Word 3 indicates the EOP. The valid byte enables during an EOP are “1111”, “1110”, “1100” and “1000”. If the packet was errored, the ERR signal is asserted and will remain asserted until EOP as shown in Figure 6

Figure 6. Example of EOP Indication in 32-Bit Mode



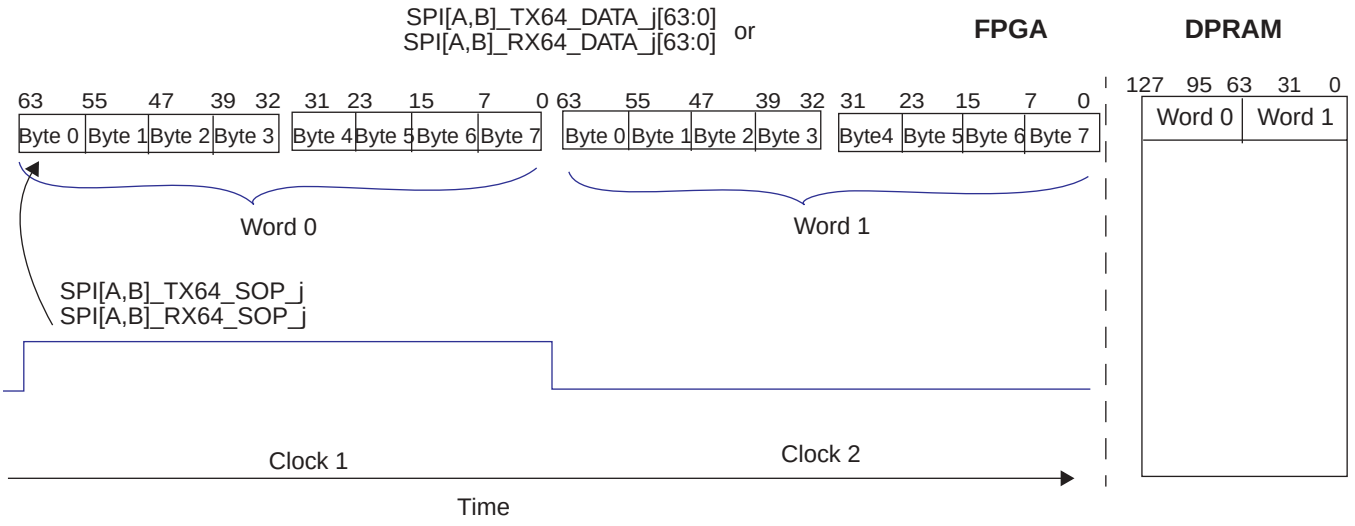
The bit ordering in the data format in the 32-bit aggregation mode is shown in Figure 7. In 32-bit mode, data transfers are done in 4-word bursts with the exception of an EOP. As shown in the figure, SOP is always aligned to the most significant 32-bit word, Word 0 of a 4-word burst. The first byte of a packet is always Word 0. The SOP indicator is high during Word 0. EOP can be any byte within the 32-bit word. The last valid byte enable within a 32-bit word indicates the end of packet. The EOP signal can be high during Word 0, Word 1, Word 2 or Word 3. It should be used in conjunction with the byte enables to determine the last byte in that data transfer. If data transfers do not end on Word 3, then the burst is terminated and Word 0 of the next burst is sent to the user. For example, if EOP occurs during Word 2, Word 3 will be discarded and Word 0 of the next word will be sent to the user.

Figure 7. Byte and Bit Ordering in 32-Bit Operating Mode



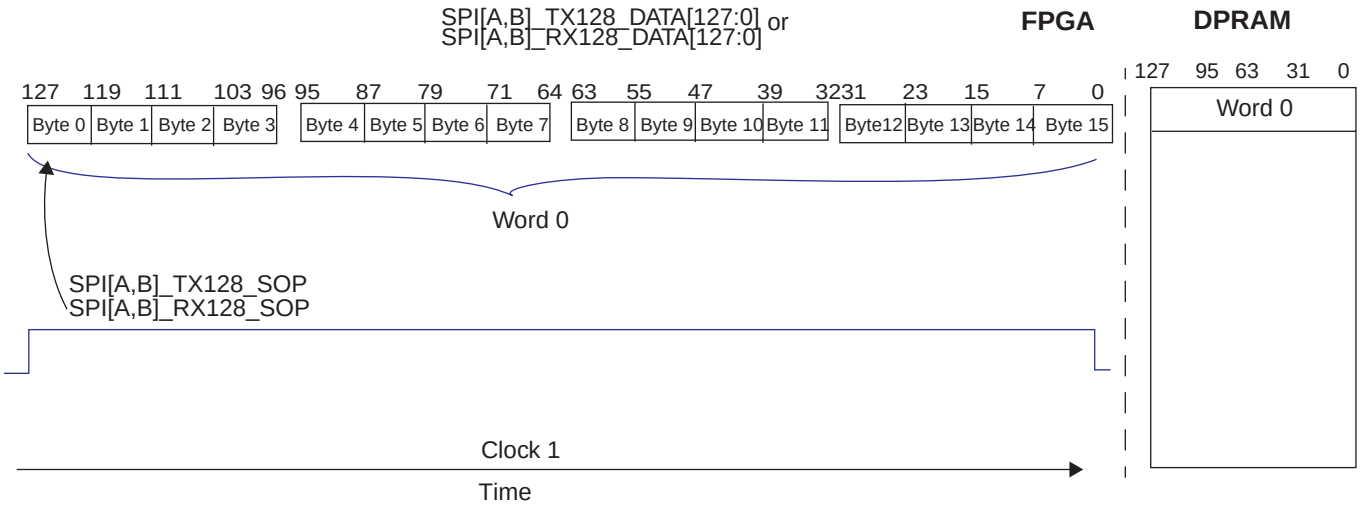
The bit ordering in the data format in the 64-bit aggregation mode is shown in Figure 8. In 64-bit mode, data transfers are done in 2-word bursts with the exception of an EOP. As shown in the figure, SOP is always aligned to the most significant 64-bit Word 0 of a 2-word burst. The first byte of a packet is always Word 0. The SOP indicator is high during word Word 0.

Figure 8. Byte and Bit Ordering in 64-Bit Mode



The bit ordering in the data format in the 128-bit aggregation mode is shown in Figure 9. The first byte of a packet is always byte 0 of the 128-bit word. As shown in the figure, SOP is always aligned to the most significant 64-bit Word 0 of a 2-word burst. SOP is always aligned to byte 0 of the 128-bit word.

Figure 9. Byte and Bit Ordering in 128-Bit Mode



SPI4 Transmit Path Functional Description

This section describes the transmit section of the SPI4 interface. The ORSPI4 device contains two identical, yet independent Transmit SPI4 compliant interfaces within the FPSC. The description provided within this section is applicable to both Transmit SPI4 interfaces.

The Transmit SPI4 interface supports the following features:

- Independent TX interface that is not tied to associated Receive SPI4 interface
- 10 Gbit/s data throughput
- 4 dual-port memories used to provide 256 Bytes of buffering for up to 32 ports. If support for more than 32 ports is required, the dual-port memories can be used, or they can be used for clock crossing purposes while data is buffered with external memory.
- Simple FIFO-like interface from FPGA to SPI4 TX embedded core block.
- Calendar Control Logic, including the Transmit Calendar, Shadow Calendar and support for up to 256 ports (the maximum allowable number of SPI4 ports).
- Mixed data width aggregation modes at the user Transmit interface. 32, 64, and 128-bit modes are supported.
- Programmable calendar table, supporting all calendar configuration parameters as specified in the SPI4 standard.
- Feedback to FPGA of currently serviced SPI4 port.

The Transmit SPI4 logic enables users to write port data from a variety of interfaces and associated clock domains using Dual Port RAMs (DPRAM) for temporary storage and clock domain crossing. Data is written into the DPRAM banks from the FPGA and read according to a pre-configured transmit calendar sequence. Data is formatted into the SPI4 format and transmitted to the physical links as LVDS signals, as specified by OIF-SPI4-02.0.

In addition to the transmit data path, out-of-band status information is received on the Transmit Status lines. This information is used, along with preconfigured Calendar sequence information, to schedule the servicing of SPI4 port for data transmission. The status information is also passed back to the FPGA logic. There are also several other features incorporated into the embedded core such as parallel loopback and SPI4 loopback to assist in debugging and statistic gathering.

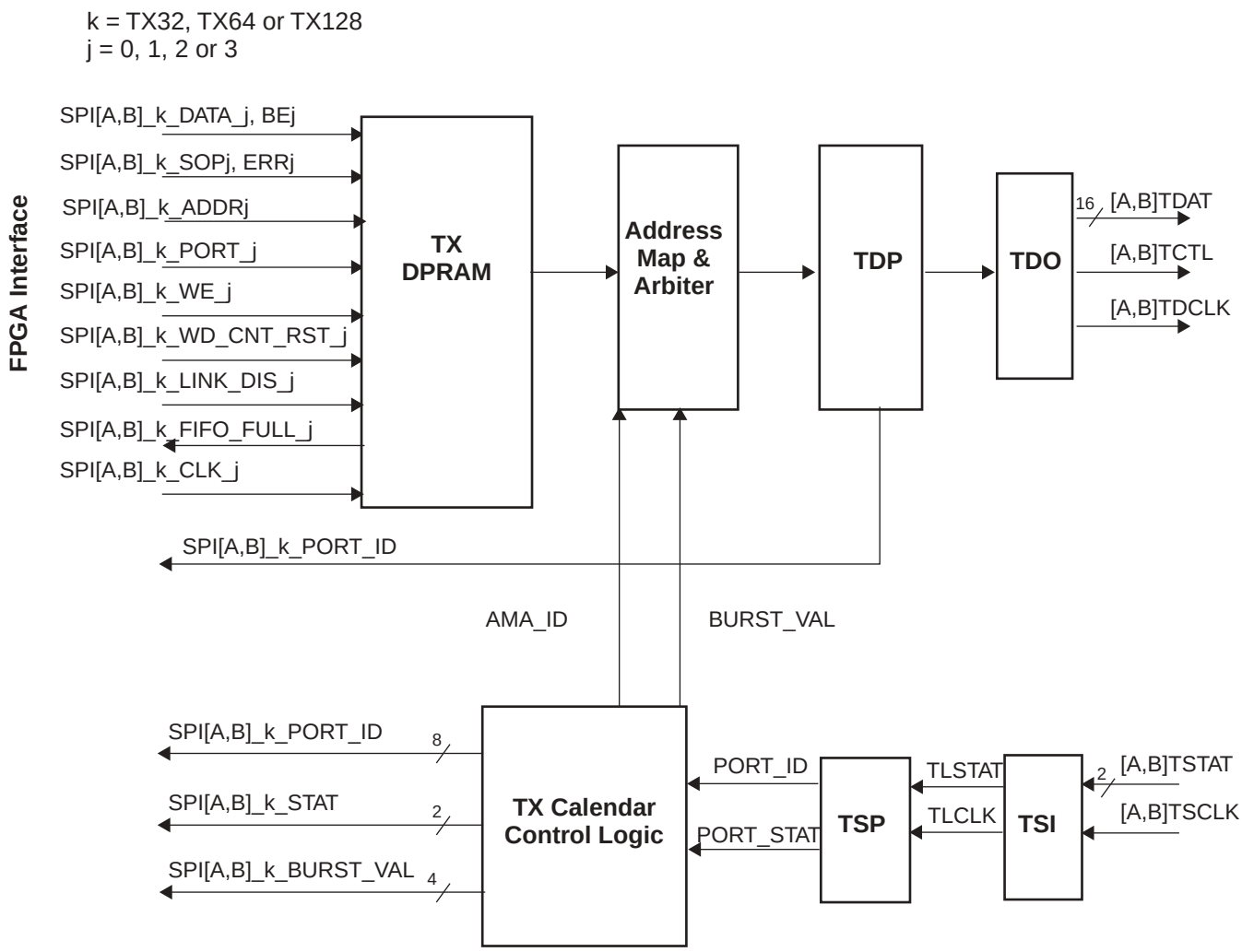
The major blocks associated with the ORSPI4 transmitter are:

- Four Transmit DPRAM Banks
- Address Map and Arbiter (AMA)
- Transmit Calendar Control Logic
  - Port Write Sequencer Logic (PWS)
- SPI4 Logic - Data
  - SPI4 Transmit Data Logic (TDP)
  - SPI4 Transmit I/O Interface (TDO)
- SPI4 Transmit Status Logic (TSP)
  - SPI4 Transmit Status Interface (TSI)
  - SPI4 Transmit Status Protocol (TSP)

These blocks will be described in detail in the following sections. The ORSPI4 Transmit functional block diagram is shown in Figure 10.

.

Figure 10. ORSPI4 Transmit Functional Block Diagram



ORSPI4 Transmit Functional Block Overview

The user can write transmit data and associated control signals to the TX DPRAM bank using either multiple 32- or 64-bit interfaces, or a single aggregated 128-bit interface. The data bus interface is software configurable. The TX DPRAM block is made up of 4 configurable DPRAMs, used for transmit data buffering and clock domain crossing. The DPRAMs can be configured via software to support up to 8 virtual partitions each, providing a maximum of 32 virtual FIFOs, each being 256 Bytes deep. Data is read from the DPRAMs according to the configured Transmit Calendar within the TX Calendar Control Logic. The control logic provides both the virtual partition address and BURST_VAL to the Address Map & Arbiter (AMA) block. The AMA services the requested partition and can optionally provide idle data if the partition is empty when serviced. The TDP block accepts transmit data and control signals from the AMA and formats the data according to the SPI4 transmit data protocol. The transmit data is then serialized within the TDO block. Data is transmitted across the SPI4 interface on a 16-bit LVDS pair bus (TDAT) which is source synchronous to the LVDS pair clock (TDCLK). The LVDS control signal pair (TCTL) is used to identify when the content of the SPI4 bus is control versus user data. The interface is an OIF-SPI-4 02.0 compliant interface that supports data rates in the range of 622-900 Mbps. Additionally, lower speed data in the range of 100-200 Mbps is also supported. To accommodate dynamic timing the TDP will insert training patterns as configured.

Far-end status is received by the TSI block and sent to the TSP block for buffering. The interface supports quarter-rate status from either LVDS or TTL inputs. The TSP block provides both the port identification and current status to the TX Calendar Control Logic, where internal tables are updated accordingly to provide proper port servicing.

ORSPI4 Transmit FPGA/Embedded Core Interface Description

The FPGA I/O interface to the ORSPI4 logic supports several interface features and varying interface characteristics, depending upon the configured mode of operation. The modes are referred to as 'aggregation modes' due to the nature of the aggregation of data busses for the different modes of operation. There are three data aggregation modes supported by the Transmit Core:

- 32-bit mode: Each of the 4 DPRAMs can be configured to use a 32-bit user data interface plus associated control signals across the FPGA interface. This interface is useful when interfacing the SPI4 data pipe to one or more sub-rate channels. This mode provides up to 4 independent 32-bit interfaces. In this mode, the interfaces do not need to be synchronous with each other.
- 64-bit mode: DPRAM pairs {[0:1], [2:3]} are configured to serve as 64-bit user data interfaces, plus associated control signals. Note that 32-bit and 64-bit modes can be combined to provide up to 3 independent interfaces.
- 128-bit mode: All four DPRAM banks are aggregated into a single FIFO with a 128-bit user-data interface from the FPGA.
- To maintain compliance with the SPI-4.2 interface protocol, the user is required to burst a minimum of 16 data bytes per write period. This is required for all modes of aggregation. For 32-bit mode, four writes are required, for 64-bit mode, two writes are required. For 128-bit mode, a single write access will provide 16 bytes of data.

Several other features are:

- Auto-increment of the DPRAM write address pointer when an EOP is detected.
- Prohibit data writes if no associated byte-enable (BE) bits are asserted within an entire 128-bit line, optimizing the FIFO memory locations by not wasting memory.
- The ability for the user to disable one or more of the interface links using the SPI[A,B]_k_LINK_DIS_j signals. This causes the AMA logic to cease polling from a DPRAM after encountering an EOP. All ports associated with a disabled DPRAM will remain unserviced until the control signal is deasserted.
- The ability for the user to reset a port's write pointer with the SPI[A,B]_k_WD_CNT_RST_j signal. This can be used to customize data applications or as a diagnostic test function.
- The ability for the user to individually poll for DPRAM FIFO fill status. Normal FIFO status is provided whenever the user attempts to write data to the DPRAMs. If an address is provided to the FIFO without a write enable, FIFO status is still provided. This feature is useful when the application requires knowing the status of individual virtual FIFOs but is not prepared to write data to it.

Table 2 lists the I/Os for the Transmit SPI4A core only. The interface signals for core B are identical with the names modified appropriately.

Table 3 and Table 4 list the I/O for the Transmit SPI4 core for 64-bit and 128-bit aggregation modes respectively.

Table 2. SPIA Core Transmit FPGA Interface in 32-Bit Mode

DPRAM	FPGA Interface I/Os	Direction From/To	Description
0	SPIA_TX32_ADDR_0[2:0]	FPGA → Core	Virtual FIFO write address. Each DPRAM can be divided up to a maximum of eight partitions. More than one port can be configured to share a virtual FIFO partition.
	SPIA_TX32_DATA_0[31:0]	FPGA → Core	User Write data. Four writes are required in order to complete an entire 128-bit line. All writes must occur in four write bursts, unless an EOP occurs.
	SPIA_TX32_BE_0[3:0]	FPGA → Core	Byte enables indicating which bytes within the 32-bit word are valid. Bit[3]-Byte enable for SPIA_TX32_DATA_0[31:24] Bit[2]-Byte enable for SPIA_TX32_DATA_0[23:16] Bit[1]-Byte enable for SPIA_TX32_DATA_0[15:8] Bit[0]-Byte enable for SPIA_TX32_DATA_0[7:0] If no BE bits are asserted for a particular location within the DPRAM, port data will be dropped and no write to the DPRAM will occur. According to the SPI-4.2 specification, the BE bits must be asserted for all but the last transfer, where an End of Packet or Error may occur. The valid combinations of BE are as follows: BE[3:0] = 1111--indicates all four bytes contain valid user data BE[3:0] = 1110--indicates three bytes are valid. EOP occurs in third Byte. BE[3:0] = 1100--indicates two bytes are valid. EOP occurs in second Byte. BE[3:0] = 1000--indicates one byte is valid. EOP occurs in first Byte.
	SPIA_TX32_SOP_0	FPGA → Core	Start of Packet indicator. A '1' indicates the start of packet for a particular port. When operating in either 32- or 64-bit mode, the SOP indicator must be asserted coincident with the first write per address line within the FIFO partition.
	SPIA_TX32_EOP_0	FPGA → Core	End of Packet indicator. A '1' indicates the end of packet for a particular port. The EOP may be asserted on any write per address line within the FIFO partition. When an EOP is detected by the DPRAM Write Pointer Logic, the address pointer will be automatically incremented to the next location within the FIFO partition range.
	SPIA_TX32_ERR_0	FPGA → Core	Packet error indication. A '1' indicates an error has occurred for the current packet being transmitted. The ERR signal must be asserted coincident with the EOP signal.
	SPIA_TX32_WE_0	FPGA → Core	Write Enable. A logic '1' causes data on the SPIA_TX32_DATA_0[31:0] bus to be captured for a write to the DPRAM.
	SPIA_TX32_PORT_0[7:0]	FPGA → Core	SPI4 port indicator, used to associate the current transmit data with a particular port number.
	SPIA_TX32_WD_CNT_RST_0	FPGA → Core	Line Write Termination indicator. This signal causes the FIFO write address pointer within the embedded core to increment to the next address location for the addressed partition. This signal can be used for custom applications as well as for diagnostic test functions.
	SPIA_TX32_LINK_DIS_0	FPGA → Core	Link disable control signal. When asserted to a logic '1', the AMA will service the particular DPRAM until it encounters an EOP. From that point on the AMA will not service the DPRAM until the LINK_DIS signal is removed. The AMA will send IDLE data across the SPI4 link. If the application continues to write data to that port FIFO, it will eventually fill and provide proper FIFO fill status to the application.
	SPIA_TX32_CLK_0	FPGA → Core	Transmit write reference clock. The clocks across the different transmit interfaces are independent from each other, and are not required to be synchronous to each other.
	SPIA_TX32_FIFO_FULL_0	Core →FPGA	FIFO full status. The status is given in response to the assertion of any valid address on the SPIA_TX32_ADDR_0[2:0] bus. This signal will be asserted to a logic '1' when the current FIFO partition has crossed the configured fill level within the partition.
Note: For SPIB replace A with B			

Table 2. SPIA Core Transmit FPGA Interface in 32-Bit Mode (Continued)

DPRAM	FPGA Interface I/Os	Direction From/To	Description
1	SPIA_TX32_ADDR_1[2:0]	FPGA → Core	Virtual FIFO Write Address. Each DPRAM can be divided up to a maximum of eight partitions. More than one port can be configured to share a virtual FIFO partition.
	SPIA_TX32_DATA_1[31:0]	FPGA → Core	User Write data. Four writes are required in order to complete an entire 128-bit line. All writes must occur in four write bursts, unless an EOP occurs.
	SPIA_TX32_BE_1[3:0]	FPGA → Core	Byte enables indicating which bytes within the 32-bit word are valid. Bit[3]-Byte enable for SPIA_TX32_DATA_0[31:24] Bit[2]-Byte enable for SPIA_TX32_DATA_0[23:16] Bit[1]-Byte enable for SPIA_TX32_DATA_0[15:8] Bit[0]-Byte enable for SPIA_TX32_DATA_0[7:0] If no BE bits are asserted for a particular location within the DPRAM, port data will be dropped and no write to the DPRAM will occur. According to the SPI-4.2 specification, the BE bits must be asserted for all but the last transfer, where an End of Packet or Error may occur. The valid combinations of BE are as follows: BE[3:0] = 1111--indicates all four bytes contain valid user data BE[3:0] = 1110--indicates three bytes are valid. EOP occurs in third Byte. BE[3:0] = 1100--indicates two bytes are valid. EOP occurs in second Byte. BE[3:0] = 1000--indicates one byte is valid. EOP occurs in first Byte.
	SPIA_TX32_SOP_1	FPGA → Core	Start of Packet indicator. A '1' indicates the start of packet for a particular port. When operating in either 32- or 64-bit mode, the SOP indicator must be asserted coincident with the first write per address line within the FIFO partition.
	SPIA_TX32_EOP_1	FPGA → Core	End of Packet indicator. A '1' indicates the end of packet for a particular port. The EOP may be asserted on any write per address line within the FIFO partition. When an EOP is detected by the DPRAM Write Pointer Logic, the address pointer will be automatically incremented to the next location within the FIFO partition range.
	SPIA_TX32_ERR_1	FPGA → Core	Packet error indication. A '1' indicates an error has occurred for the current packet being transmitted. The ERR signal must be asserted coincident with the EOP signal.
	SPIA_TX32_WE_1	FPGA → Core	Write Enable. A logic '1' causes data on the SPIA_TX32_DATA_1[31:0] bus to be captured for a write to the DPRAM.
	SPIA_TX32_PORT_1[7:0]	FPGA → Core	SPI4 port indicator, used to associate the current transmit data with a particular port number.
	SPIA_TX32_WD_CNT_RST_1	FPGA → Core	Line Write Termination indicator. This signal causes the FIFO write address pointer within the embedded core to increment to the next address location for the addressed partition. This signal can be used for custom applications as well as for diagnostic test functions.
	SPIA_TX32_LINK_DIS_1	FPGA → Core	Link disable control signal. When asserted to a logic '1', the AMA will service the particular DPRAM until it encounters an EOP. From that point on the AMA will not service the DPRAM until the LINK_DIS signal is removed. The AMA will send IDLE data across the SPI4 link. If the application continues to write data to that port FIFO, it will eventually fill
	SPIA_TX32_CLK_1	FPGA → Core	Transmit write reference clock. The clocks across the different transmit interfaces are independent from each other, and are not required to be synchronous to each other.
	SPIA_TX32_FIFO_FULL_1	Core →FPGA	FIFO full status. The status is given in response to the assertion of any valid address on the SPIA_TX32_ADDR_1[2:0] bus. This signal will be asserted to a logic '1' when the current FIFO partition has crossed the configured fill level within the partition.
Note: For SPIB replace A with B			

Table 2. SPIA Core Transmit FPGA Interface in 32-Bit Mode (Continued)

DPRAM	FPGA Interface I/Os	Direction From/To	Description
2	SPIA_TX32_ADDR_2[2:0]	FPGA → Core	Virtual FIFO Write Address. Each DPRAM can be divided up to a maximum of eight partitions. More than one port can be configured to share a virtual FIFO partition.
	SPIA_TX32_DATA_2[31:0]	FPGA → Core	User Write data. Four writes are required in order to complete an entire 128-bit line. All writes must occur in four write bursts, unless an EOP occurs.
	SPIA_TX32_BE_2[3:0]	FPGA → Core	Byte enables indicating which bytes within the 32-bit word are valid. Bit[3]-Byte enable for SPIA_TX32_DATA_0[31:24] Bit[2]-Byte enable for SPIA_TX32_DATA_0[23:16] Bit[1]-Byte enable for SPIA_TX32_DATA_0[15:8] Bit[0]-Byte enable for SPIA_TX32_DATA_0[7:0] If no BE bits are asserted for a particular location within the DPRAM, port data will be dropped and no write to the DPRAM will occur. According to the SPI-4.2 specification, the BE bits must be asserted for all but the last transfer, where an End of Packet or Error may occur. The valid combinations of BE are as follows: BE[3:0] = 1111--indicates all four bytes contain valid user data. If an EOP is present, it resides within the fourth byte. BE[3:0] = 1110--indicates three bytes are valid. EOP occurs in third Byte. BE[3:0] = 1100--indicates two bytes are valid. EOP occurs in second Byte. BE[3:0] = 1000--indicates one byte is valid. EOP occurs in first Byte.
	SPIA_TX32_SOP_2	FPGA → Core	Start of Packet indicator. A '1' indicates the start of packet for a particular port. When operating in either 32- or 64-bit mode, the SOP indicator must be asserted coincident with the first write per address line within the FIFO partition.
	SPIA_TX32_EOP_2	FPGA → Core	End of Packet indicator. A '1' indicates the end of packet for a particular port. The EOP may be asserted on any write per address line within the FIFO partition. When an EOP is detected by the DPRAM Write Pointer Logic, the address pointer will be automatically incremented to the next location within the FIFO partition range.
	SPIA_TX32_ERR_2	FPGA → Core	Packet error indication. A '1' indicates an error has occurred for the current packet being transmitted. The ERR signal must be asserted coincident with the EOP signal.
	SPIA_TX32_WE_2	FPGA → Core	Write Enable. A logic '1' causes data on the SPIA_TX32_DATA_2[31:0] bus to be captured for a write to the DPRAM.
	SPIA_TX32_PORT_2[7:0]	FPGA → Core	SPI4 port indicator, used to associate the current transmit data with a particular port number.
	SPIA_TX32_WD_CNT_RST_2	FPGA → Core	Line Write Termination indicator. This signal causes the FIFO write address pointer within the embedded core to increment to the next address location for the addressed partition. This signal can be used for custom applications as well as for diagnostic test functions.
	SPIA_TX32_LINK_DIS_2	FPGA → Core	Link disable control signal. When asserted to a logic '1', the AMA will service the particular DPRAM until it encounters an EOP. From that point on the AMA will not service the DPRAM until the LINK_DIS signal is removed. The AMA will send IDLE data across the SPI4 link. If the application continues to write data to that port FIFO, it will eventually fill and provide proper FIFO fill status to the application.
	SPIA_TX32_CLK_2	FPGA → Core	Transmit write reference clock. The clocks across the different transmit interfaces are independent from each other, and are not required to be synchronous to each other.
	SPIA_TX32_FIFO_FULL_2	Core →FPGA	FIFO full status. The status is given in response to the assertion of any valid address on the SPIA_TX32_ADDR_3[2:0] bus. This signal will be asserted to a logic '1' when the current FIFO partition has crossed the configured fill level within the partition.
Note: For SPIB replace A with B			

Table 2. SPIA Core Transmit FPGA Interface in 32-Bit Mode (Continued)

DPRAM	FPGA Interface I/Os	Direction From/To	Description
3	SPIA_TX32_ADDR_3[2:0]	FPGA → Core	Virtual FIFO Write Address. Each DPRAM can be divided up to a maximum of eight partitions. More than one port can be configured to share a virtual FIFO partition.
	SPIA_TX32_DATA_3[31:0]	FPGA → Core	User Write data. Four writes are required in order to complete an entire 128-bit line. All writes must occur in four write bursts, unless an EOP occurs.
	SPIA_TX32_BE_3[3:0]	FPGA → Core	Byte enables indicating which bytes within the 32-bit word are valid. Bit[3]-Byte enable for SPIA_TX32_DATA_0[31:24] Bit[2]-Byte enable for SPIA_TX32_DATA_0[23:16] Bit[1]-Byte enable for SPIA_TX32_DATA_0[15:8] Bit[0]-Byte enable for SPIA_TX32_DATA_0[7:0] If no BE bits are asserted for a particular location within the DPRAM, port data will be dropped and no write to the DPRAM will occur. According to the SPI-4.2 specification, the BE bits must be asserted for all but the last transfer, where an End of Packet or Error may occur. The valid combinations of BE are as follows: BE[3:0] = 1111--indicates all four bytes contain valid user data. If an EOP is present, it resides within the fourth byte. BE[3:0] = 1110--indicates three bytes are valid. EOP occurs in third Byte. BE[3:0] = 1100--indicates two bytes are valid. EOP occurs in second Byte. BE[3:0] = 1000--indicates one byte is valid. EOP occurs in first Byte.
	SPIA_TX32_SOP_3	FPGA → Core	Start of Packet indicator. A '1' indicates the start of packet for a particular port. When operating in either 32- or 64-bit mode, the SOP indicator must be asserted coincident with the first write per address line within the FIFO partition.
	SPIA_TX32_EOP_3	FPGA → Core	End of Packet indicator. A '1' indicates the end of packet for a particular port. The EOP may be asserted on any write per address line within the FIFO partition. When an EOP is detected by the DPRAM Write Pointer Logic, the address pointer will be automatically incremented to the next location within the FIFO partition range.
	SPIA_TX32_ERR_3	FPGA → Core	Packet error indication. A '1' indicates an error has occurred for the current packet being transmitted. The ERR signal must be asserted coincident with the EOP signal.
	SPIA_TX32_WE_3	FPGA → Core	Write Enable. A logic '1' causes data on the SPIA_TX32_DATA_3[31:0] bus to be captured for a write to the DPRAM.
	SPIA_TX32_PORT_3[7:0]	FPGA → Core	SPI4 port indicator, used to associate the current transmit data with a particular port number.
	SPIA_TX32_WD_CNT_RST_3	FPGA → Core	Line Write Termination indicator. This signal causes the FIFO write address pointer within the embedded core to increment to the next address location for the addressed partition. This signal can be used for custom applications as well as for diagnostic test functions.
	SPIA_TX32_LINK_DIS_3	FPGA → Core	Link disable control signal. When asserted to a logic '1', the AMA will service the particular DPRAM until it encounters an EOP. From that point on the AMA will not service the DPRAM until the LINK_DIS signal is removed. The AMA will send IDLE data across the SPI4 link. If the application continues to write data to that port FIFO, it will eventually fill and provide proper FIFO fill status to the application.
	SPIA_TX32_CLK_3	FPGA → Core	Transmit write reference clock. The clocks across the different transmit interfaces are independent from each other, and are not required to be synchronous to each other.
	SPIA_TX32_FIFO_FULL_3	Core →FPGA	FIFO full status. The status is given in response to the assertion of any valid address on the SPIA_TX32_ADDR_3[2:0] bus. This signal will be asserted to a logic '1' when the current FIFO partition has crossed the configured fill level within the partition.
Note: For SPIB replace A with B			

Table 2. SPIA Core Transmit FPGA Interface in 32-Bit Mode (Continued)

DPRAM	FPGA Interface I/Os	Direction From/To	Description
Status I/Os	SPIA_TX32_PORT_ID[7:0]	Core →FPGA	Port number of the currently serviced SPI4 data port. The value can either be the actual SPI4 value or a programmed user value. Further details are provided in the TX Calendar Control Logic description.
	SPIA_TX32_STAT[1:0]	Core →FPGA	Status of the SPI4 port currently being serviced. These signals can be used in conjunction with the SPIA_TX32_PORT_ID signal to police transmit traffic congestion for a particular SPI4 port.
	SPIA_TX32_BURST_VAL[3:0]	Core →FPGA	Indicates the number of SPI4 cycles for which the currently active port will be serviced. Each cycle indicates an attempt to read 128 bits of data from the respective FIFO partition.
	ATREFCLK_F	Core →FPGA	Transmit reference clock. This clock signal is 1/4th the SPI4 clock. This signal can be used to synchronize FPGA application logic to the FPSC logic.
	SPIA_TREFCLK_X8	FPGA → Core	Quarter-rate Transmit reference clock. This FPGA-sourced clock reference must be 2x the desired Transmit SPI4 line clock rate. When not operating the Transmit SPI4 core in Quarter-rate mode, this signal should be tied off. Ex: For a 100 MHz Transmit SPI4 line clock, SPIA_TREFCLK_X8 from the FPGA must be 200 MHz.
Misc. I/Os	SPI_SATM_A	FPGA → Core	'0' - Incoming ATSCLK is assumed to be edge-aligned with the data and centered to the data eye within the chip. '1' - Incoming ATSCLK is assumed to be skewed with respect to the data off-chip.
Note: For SPIB replace A with B			

Table 3. SPIA Core Transmit FPGA Interface in 64-Bit Mode

DPRAM	FPGA Interface I/Os	Direction From/To	Description
0	SPIA_TX64_ADDR_0[2:0]	FPGA → Core	Virtual FIFO Write Address. Each DPRAM can be divided up to a maximum of eight partitions. When operating in 64-bit mode, each partition will be 2x the depth of the corresponding partition in 32-bit mode. More than one port can be configured to share a virtual FIFO partition.
	SPIA_TX64_DATA_0[63:0]	FPGA → Core	User Write data. Two writes are required in order to complete an entire 128-bit line. All writes must occur in two write bursts, unless an EOP occurs.
	SPIA_TX64_BE_0[7:0]	FPGA → Core	Byte enables indicating which bytes within the 64-bit word are valid. Bit[7]-Byte enable for SPIA_TX64_DATA_0[63:56] Bit[6]-Byte enable for SPIA_TX64_DATA_0[55:48] Bit[5]-Byte enable for SPIA_TX64_DATA_0[47:40] Bit[4]-Byte enable for SPIA_TX64_DATA_0[39:32] Bit[3]-Byte enable for SPIA_TX64_DATA_0[31:24] Bit[2]-Byte enable for SPIA_TX64_DATA_0[23:16] Bit[1]-Byte enable for SPIA_TX64_DATA_0[15:8] Bit[0]-Byte enable for SPIA_TX64_DATA_0[7:0] If no BE bits are asserted for a particular location within the DPRAM, data will be dropped and no write to the DPRAM will occur. According to the SPI-4.2 specification, the BE bits must be asserted for all but the last transfer, where an End of Packet or Error may occur. The valid combinations of BE are as follows: BE[7:0] = 11111111--indicates all eight bytes contain valid user data. If an EOP is present, it resides within the eighth byte. BE[7:0] = 11111110--indicates seven bytes are valid. EOP occurs in seventh Byte. BE[7:0] = 11111100--indicates six bytes are valid. EOP occurs in sixth Byte. BE[7:0] = 11111000--indicates five byte is valid. EOP occurs in fifth Byte. BE[7:0] = 11110000--indicates four bytes are valid. EOP occurs in fourth Byte. BE[7:0] = 11100000--indicates three bytes are valid. EOP occurs in third Byte. BE[7:0] = 11000000--indicates two bytes are valid. EOP occurs in second Byte. BE[7:0] = 10000000--indicates one byte is valid. EOP occurs in first Byte.
	SPIA_TX64_SOP_0	FPGA → Core	Start of Packet indicator. A '1' indicates the start of packet for a particular port. When operating in 64-bit mode, the SOP indicator must be asserted coincident with the first write per address line within the FIFO partition.
	SPIA_TX64_EOP_0	FPGA → Core	End of Packet indicator. A '1' indicates the end of packet for a particular port. The EOP may be asserted on either write per address line within the FIFO partition. When an EOP is detected by the DPRAM Write Pointer Logic, the address pointer will be automatically incremented to the next location within the FIFO partition range.
	SPIA_TX64_ERR_0	FPGA → Core	Packet error indication. A '1' indicates an error has occurred for the current packet being transmitted. The ERR signal must be asserted coincident with the EOP signal.
	SPIA_TX64_WE_0	FPGA → Core	Write Enable. A logic '1' causes data on the SPIA_TX64_DATA_0[63:0] bus to be captured for a write to the DPRAM.
	SPIA_TX64_PORT_0[7:0]	FPGA → Core	SPI4 port indicator, used to associate the current transmit data with a particular port number.
	SPIA_TX64_WD_CNT_RST_0	FPGA → Core	Line Write Termination indicator. This signal causes the FIFO write address pointer within the embedded core to increment to the next address location for the addressed partition. This signal can be used for custom applications as well as for diagnostic test functions.
Note: For SPIB replace A with B			

Table 3. SPIA Core Transmit FPGA Interface in 64-Bit Mode (Continued)

DPRAM	FPGA Interface I/Os	Direction From/To	Description
0	SPIA_TX64_LINK_DIS_0	FPGA → Core	Link disable control signal. When asserted to a logic '1', the AMA will service the particular DPRAM until it encounters an EOP. From that point on the AMA will not service the DPRAM until the LINK_DIS signal is removed. The AMA will send IDLE data across the SPI4 link. If the application continues to write data to that port FIFO, it will eventually fill and provide proper FIFO fill status to the application.
	SPIA_TX64_CLK_0	FPGA → Core	Transmit write reference clock. The clocks across the different transmit interfaces are independent from each other, and are not required to be synchronous to each other.
	SPIA_TX64_FIFO_FULL_0	Core →FPGA	FIFO full status. The status is given in response to the assertion of any valid address on the SPIA_TX64_ADDR_1[2:0] bus. This signal will be asserted to a logic '1' when the current FIFO partition has crossed the configured fill level within the partition.
1	SPIA_TX64_ADDR_1[2:0]	FPGA → Core	Virtual FIFO Write Address. Each DPRAM can be divided up to a maximum of eight partitions. When operating in 64-bit mode, each partition will be 2x the depth of the corresponding partition in 32-bit mode. More than one port can be configured to share a virtual FIFO partition.
	SPIA_TX64_DATA_1[63:0]	FPGA → Core	User Write data. Two writes are required in order to complete an entire 128-bit line. All writes must occur in two write bursts, unless an EOP occurs.
	SPIA_TX64_BE_1[7:0]	FPGA → Core	Byte enables indicating which bytes within the 64-bit word are valid. Bit[7]-Byte enable for SPIA_TX64_DATA_0[63:56] Bit[6]-Byte enable for SPIA_TX64_DATA_0[55:48] Bit[5]-Byte enable for SPIA_TX64_DATA_0[47:40] Bit[4]-Byte enable for SPIA_TX64_DATA_0[39:32] Bit[3]-Byte enable for SPIA_TX64_DATA_0[31:24] Bit[2]-Byte enable for SPIA_TX64_DATA_0[23:16] Bit[1]-Byte enable for SPIA_TX64_DATA_0[15:8] Bit[0]-Byte enable for SPIA_TX64_DATA_0[7:0] If no BE bits are asserted for a particular location within the DPRAM, data will be dropped and no write to the DPRAM will occur. According to the SPI-4.2 specification, the BE bits must be asserted for all but the last transfer, where an End of Packet or Error may occur. The valid combinations of BE are as follows: BE[7:0] = 11111111--indicates all eight bytes contain valid user data. If an EOP is present, it resides within the eighth byte. BE[7:0] = 11111110--indicates seven bytes are valid. EOP occurs in seventh Byte. BE[7:0] = 11111100--indicates six bytes are valid. EOP occurs in sixth Byte. BE[7:0] = 11111000--indicates five byte is valid. EOP occurs in fifth Byte. BE[7:0] = 11110000--indicates four bytes are valid. EOP occurs in fourth Byte. BE[7:0] = 11100000--indicates three bytes are valid. EOP occurs in third Byte. BE[7:0] = 11000000--indicates two bytes are valid. EOP occurs in second Byte. BE[7:0] = 10000000--indicates one byte is valid. EOP occurs in first Byte.
	SPIA_TX64_SOP_1	FPGA → Core	Start of Packet indicator. A '1' indicates the start of packet for a particular port. When operating in 64-bit mode, the SOP indicator must be asserted coincident with the first write per address line within the FIFO partition.
	SPIA_TX64_EOP_1	FPGA → Core	End of Packet indicator. A '1' indicates the end of packet for a particular port. The EOP may be asserted on either write per address line within the FIFO partition.
Note: For SPIB replace A with B			

Table 3. SPIA Core Transmit FPGA Interface in 64-Bit Mode (Continued)

DPRAM	FPGA Interface I/Os	Direction From/To	Description
1	SPIA_TX64_ERR_1	FPGA → Core	Packet error indication. A '1' indicates an error has occurred for the current packet being transmitted. The ERR signal must be asserted coincident with the EOP signal. When an EOP is detected by the DPRAM Write Pointer Logic, the address pointer will be automatically incremented to the next location within the FIFO partition range.
	SPIA_TX64_WE_1	FPGA → Core	Write Enable. A logic '1' causes data on the SPIA_TX64_DATA_1[63:0] bus to be captured for a write to the DPRAM.
	SPIA_TX64_PORT_1[7:0]	FPGA → Core	SPI4 port indicator, used to associate the current transmit data with a particular port number.
	SPIA_TX64_WD_CNT_RST_1	FPGA → Core	Line Write Termination indicator. This signal causes the FIFO write address pointer within the embedded core to increment to the next address location for the addressed partition. This signal can be used for custom applications as well as for diagnostic test functions.
	SPIA_TX64_LINK_DIS_1	FPGA → Core	Link disable control signal. When asserted to a logic '1', the AMA will service the particular DPRAM until it encounters an EOP. From that point on the AMA will not service the DPRAM until the LINK_DIS signal is removed. The AMA will send IDLE data across the SPI4 link. If the application continues to write data to that port FIFO, it will eventually fill and provide proper FIFO fill status to the application.
	SPIA_TX64_CLK_1	FPGA → Core	Transmit write clock reference. The clocks across the different transmit interfaces are independent from each other, and are not required to be synchronous to each other.
	SPIA_TX64_FIFO_FULL_1	Core →FPGA	FIFO full status. The status is given in response to the assertion of any valid address on the SPIA_TX64_ADDR_1[2:0] bus. This signal will be asserted to a logic '1' when the current FIFO partition has crossed the configured fill level within the partition.
Status /Os	SPIA_TX64_PORT_ID[7:0]	Core →FPGA	Port number of the currently serviced SPI4 data port. The value can either be the actual SPI4 value or a programmed user value. Further details are provided in the TX Calendar Control Logic description.
	SPIA_TX64_STAT[1:0]	Core →FPGA	Status of the SPI4 port currently being serviced. These signals can be used in conjunction with the FIFO partition FIFO to police transmit traffic congestion for a particular SPI4 port.
	SPIA_TX64_BURST_VAL[3:0]	Core →FPGA	Indicates the number of SPI4 cycles the currently active port will be serviced. Each cycle indicates an attempt to read 128 bits of data from the respective FIFO partition.
	SPIA_TREFCLK_X8	FPGA → Core	Transmit reference clock. This clock signal is 1/4th the SPI4 clock. This signal can be used to synchronize FPGA application logic to the FPSC logic.
Misc. I/Os	SPI_SATM_A	FPGA → Core	'0' - Incoming ATSCLK is assumed to be edge-aligned with the data and centered to the data eye within the chip. '1' - Incoming ATSCLK is assumed to be skewed with respect to the data off-chip.
Note: For SPIB replace A with B			

Table 4. SPIA Core Transmit FPGA Interface in 128-Bit Mode

DPRAM	FPGA Interface I/Os	Direction From/To	Description
0	SPIA_TX128_ADDR[2:0]	FPGA → Core	Virtual FIFO Write Address. Each DPRAM can be divided up to a maximum of eight partitions. When operating in 128-bit mode, each partition will be 4x the depth of the corresponding partition in 32-bit mode. More than one port can be configured to share a virtual FIFO partition.
	SPIA_TX128_DATA[127:0]	FPGA → Core	User Write data. A single write will complete an entire 128-bit line.
Note: For SPIB replace A with B			

Table 4. SPIA Core Transmit FPGA Interface in 128-Bit Mode (Continued)

DPRAM	FPGA Interface I/Os	Direction From/To	Description
0	SPIA_TX128_BE[15:0]	FPGA → Core	Byte enables indicating which bytes within the 128-bit word are valid. Bit[15]-Byte enable for SPIA_TX128_DATA[127:120] Bit[14]-Byte enable for SPIA_TX128_DATA[119:112] Bit[13]-Byte enable for SPIA_TX128_DATA[111:104] Bit[12]-Byte enable for SPIA_TX128_DATA[103:96] Bit[11]-Byte enable for SPIA_TX128_DATA[95:88] Bit[10]-Byte enable for SPIA_TX128_DATA[87:80] Bit[9]-Byte enable for SPIA_TX128_DATA[79:72] Bit[8]-Byte enable for SPIA_TX128_DATA[71:64] Bit[7]-Byte enable for SPIA_TX128_DATA[63:56] Bit[6]-Byte enable for SPIA_TX128_DATA[55:48] Bit[5]-Byte enable for SPIA_TX128_DATA[47:40] Bit[4]-Byte enable for SPIA_TX128_DATA[39:32] Bit[3]-Byte enable for SPIA_TX128_DATA[31:24] Bit[2]-Byte enable for SPIA_TX128_DATA[23:16] Bit[1]-Byte enable for SPIA_TX128_DATA[15:8] Bit[0]-Byte enable for SPIA_TX128_DATA[7:0] If no BE bits are asserted for a particular location within the DPRAM, data will be dropped and no write to the DPRAM will occur. According to the SPI-4.2 specification, the BE bits must be asserted for all but the last transfer, where an End of Packet or Error may occur. The valid combinations of BE are as follows: BE[15:0] = 1111111111111111--indicates all sixteen bytes contain valid user data. If an EOP is present, it resides within the sixteenth byte. BE[15:0] = 1111111111111110--indicates fifteen bytes are valid. EOP occurs in fifteenth Byte. BE[15:0] = 1111111111111100--indicates fourteen bytes are valid. EOP occurs in fourteenth Byte. BE[15:0] = 1111111111111000-indicates thirteen byte is valid. EOP occurs in thirteenth Byte. BE[15:0] = 1111111111110000--indicates twelve bytes are valid. EOP occurs in twelfth Byte. BE[15:0] = 1111111111100000-indicates eleven bytes are valid. EOP occurs in eleventh Byte. BE[15:0] = 1111111111000000--indicates ten bytes are valid. EOP occurs in tenth Byte. BE[15:0] = 1111111110000000--indicates nine bytes are valid. EOP occurs in ninth Byte. BE[15:0] = 1111111100000000--indicates eight bytes are valid. EOP occurs in eight Byte. BE[15:0] = 1111111000000000--indicates seven bytes are valid. EOP occurs in seventh Byte. BE[15:0] = 1111110000000000--indicates six bytes are valid. EOP occurs in sixth Byte. BE[15:0] = 1111100000000000-indicates five byte is valid. EOP occurs in fifth Byte. BE[15:0] = 1111000000000000--indicates four bytes are valid. EOP occurs in fourth Byte. BE[15:0] = 1110000000000000-indicates three bytes are valid. EOP occurs in third Byte. BE[15:0] = 1100000000000000--indicates two bytes are valid. EOP occurs in second Byte. BE[15:0] = 1000000000000000--indicates one byte is valid. EOP occurs in first Byte.
	SPIA_TX128_SOP	FPGA → Core	Start of Packet indicator. A '1' indicates the start of packet for a particular port. When operating in 128-bit mode, the SOP indicator must be asserted coincident with the first line write to the FIFO partition in order to align the start of packet data with the SOP sideband signal.
Note: For SPIB replace A with B			

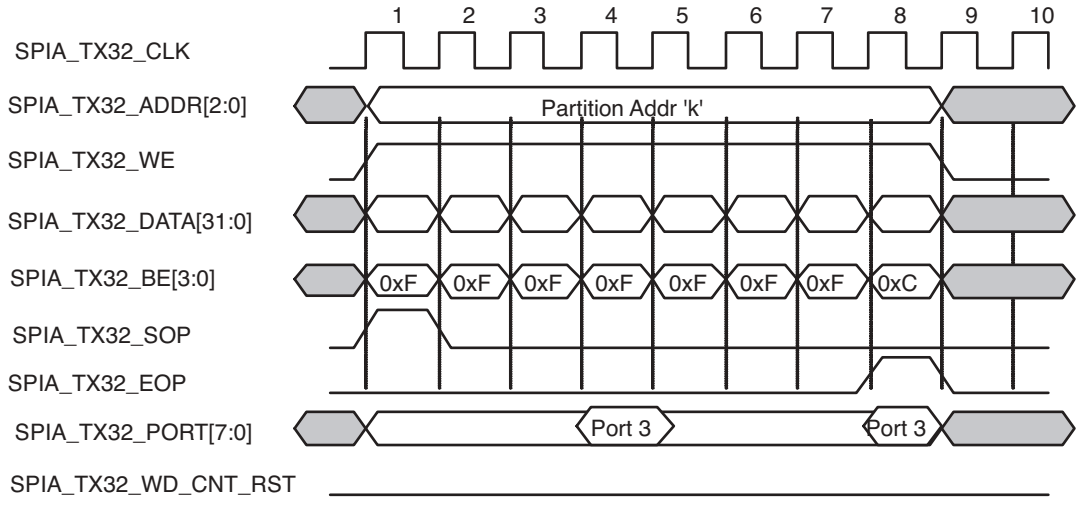
Table 4. SPIA Core Transmit FPGA Interface in 128-Bit Mode (Continued)

DPRAM	FPGA Interface I/Os	Direction From/To	Description
0	SPIA_TX128_EOP	FPGA → Core	End of Packet indicator. A '1' indicates the end of packet for a particular port. When an EOP is detected by the DPRAM Write Pointer Logic, the address pointer will be automatically incremented to the next location within the FIFO partition range.
	SPIA_TX128_ERR	FPGA → Core	Packet error indication. A '1' indicates an error has occurred for the current packet being transmitted. The ERR signal must be asserted coincident with the EOP signal.
	SPIA_TX128_WE	FPGA → Core	Write Enable. A logic '1' causes data on the SPIA_TX128_DATA_0[127:0] bus to be captured for a write to the DPRAM.
	SPIA_TX128_PORT[7:0]	FPGA → Core	SPI4 port indicator, used to associate the current transmit data with a particular port number.
	SPIA_TX128_WD_CNT_RST	FPGA → Core	Line Write Termination indicator. This signal causes the FIFO write address pointer within the embedded core to increment to the next address location for the addressed partition. This signal can be used for custom applications as well as for diagnostic test functions.
	SPIA_TX128_LINK_DIS_0	FPGA → Core	Link disable control signal. When asserted to a logic '1', the AMA will service the particular DPRAM until it encounters an EOP. From that point on the AMA will not service the DPRAM until the LINK_DIS signal is removed. The AMA will send IDLE data across the SPI4 link. If the application continues to write data to that port FIFO, it will eventually fill and provide proper FIFO fill status to the application.
	SPIA_TX128_CLK	FPGA → Core	Transmit write reference clock. There is a single clock per SPI4 interface when operating in 128-bit mode.
	SPIA_TX128_FIFO_FULL	Core →FPGA	FIFO full status. The status is given in response to the assertion of any valid address on the SPIA_TX128_ADDR[2:0] bus. This signal will be asserted to a logic '1' when the current FIFO partition has crossed the configured fill level within the partition.
Status I/Os	SPIA_TX128_PORT_ID[7:0]	Core →FPGA	Port number of the currently serviced SPI4 data port. The value can either be the actual SPI4 value or a programmed user value. Further details are provided in the TX Calendar Control Logic description.
	SPIA_TX128_STAT[1:0]	Core →FPGA	Status of the SPI4 port currently being serviced. These signals can be used in conjunction with the SPIA_TX32_PORT_ID signal to police transmit traffic congestion for a particular SPI4 port.
	SPIA_TX128_BURST_VAL[3:0]	Core →FPGA	Indicates the number of SPI4 cycles for which the currently active port will be serviced. Each cycle indicates an attempt to read 128 bits of data from the respective FIFO partition.
	ATREFCLK_F	Core →FPGA	Transmit reference clock. This clock signal is 1/4th the SPI4 clock. This signal can be used to synchronize FPGA application logic to the FPSC logic.
	SPIA_TREFCLK_X8	FPGA → Core	Quarter-rate Transmit reference clock. This FPGA-sourced clock reference must be 2x the desired Transmit SPI4 line clock rate. When not operating the Transmit SPI4 core in Quarter-rate mode, this signal should be tied off. Ex: For a 100 MHz Transmit SPI4 line clock, SPIA_TREFCLK_X8 from the FPGA must be 200 MHz.
Misc. I/Os	SPI_SATM_A	FPGA → Core	'0' - Incoming ATSCLK is assumed to be edge-aligned with the data and centered to the data eye within the chip. '1' - Incoming ATSCLK is assumed to be skewed with respect to the data off-chip.
Note: For SPIB replace A with B			

I/O Modes - SPI4 FPGA/Embedded Core Data Protocol

In order to successfully transmit data across the Transmit SPI4 link, there is a certain protocol that the data and control signals must obey. This section outlines the FPGA/embedded core interface protocol to follow for each of the operating modes of the Transmit SPI4 embedded core. As mentioned, there are three basic modes of operation when transmitting data into the embedded core from the FPGA. For all modes of operation, data is transferred synchronous to the rising edge of the respective SPI_TX_CLK signals. The following figures detail the write protocol defined for transmitting SPI4 transmit data. Figure 11 below shows basic interface signals for 32-bit mode of operation.

Figure 11. Basic 32-bit Data Write Protocol to DPRAM



Note: SPIB is identical to SPIA

During clock cycle 1, the FPGA asserts the partition Address, which indicates which partition within the DPRAM the data and control signals are to be written to. The Write Enable signal is also asserted, qualifying all data and control signals. In the figure, this is the start of a packet, so the SOP signal is also asserted. Data is presented to the interface, as well as the Byte Enable (BE) and Port ID signals. The Port ID is used to identify the PORT_ID field within the SPI4 control word. The BE vector is used to qualify which Bytes on the SPI4 transmit link are valid data. During clock cycle 4, the internal DPRAM write is automatically incremented, and the user is expected to continue write accesses to the FIFO partition. During clock cycle 8, the BE bits are set to 0xC, indicating only two of the four Bytes contain valid packet data.

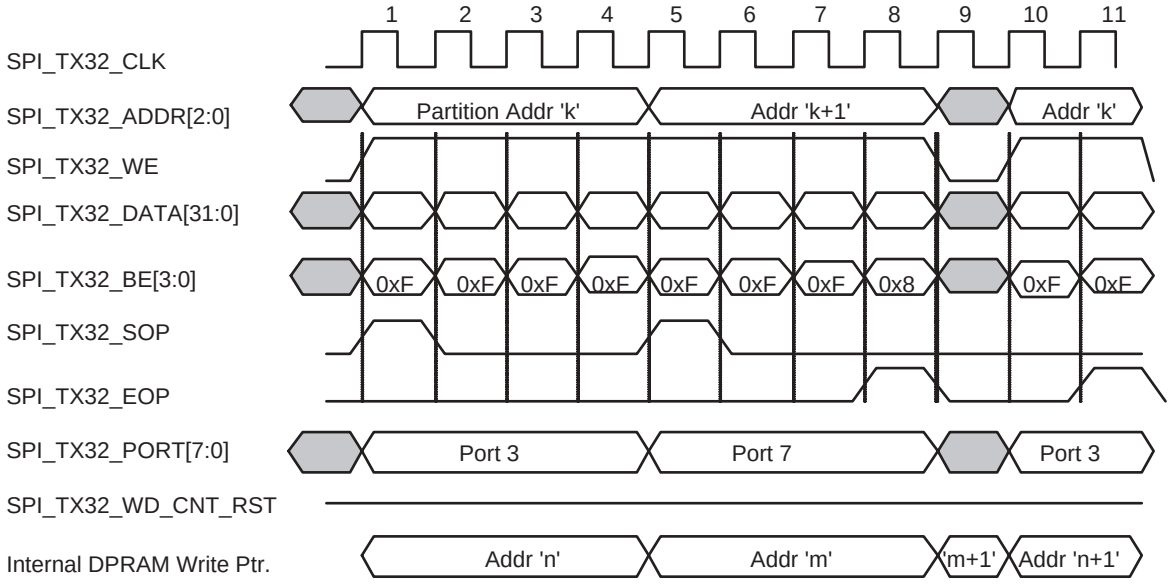
During clock cycle 8, the FPGA indicates an End of Packet condition for the current selected Port. Asserting the EOP signal causes the internal logic to terminate writes to the current Port, as well as automatically incrementing to the next address for the FIFO partition.

The Port ID signal should be asserted during an entire write sequence. At a minimum, it must be asserted at least during the last write to the current DPRAM address line (i.e. for every 128 data bits that are in a DPRAM location). In the figure above, the Port ID must be asserted during clock cycles 4 and 8.

Internal to the embedded core is logic that advances the write pointer for each DPRAM partition. Although not visible to the FPGA, the internal DPRAM pointer logic advances to the next valid address within partition range after every 4 writes when in 32-bit aggregation mode.

Figure 12 shows 32-bit write timing using fast back-to-back transfers.

Figure 12. 32-Bit Write Protocol Using Fast Back-to-Back Transfers



NOTE: SPIB is identical to SPIA

During clock cycle 1, the FPGA asserts the partition Address, which indicates which partition within the DPRAM the data and control signals are to be written to. The Write Enable signal is also asserted, qualifying all data and control signals. In the figure, this is the start of a packet, so the SOP signal is also asserted. Data is presented to the interface, as well as the Byte Enable (BE) and Port ID signals. The Port ID is used to identify the PORT_ID field within the SPI4 control word. The BE vector is used to qualify which Bytes on the SPI4 transmit link are valid data. During clock cycle 4, the FPGA has completed writing an entire 128-bit line to the addressed DPRAM partition and begins an access to another partition commencing with clock cycle 5. Since there was no EOP presented during clock cycle 4, Port 3 must be revisited later with the remaining packet data and required EOP. The internal DPRAM write is automatically incremented at the end of clock cycle 4 as well. During clock cycle 5, Port 7 is addressed as the destination partition for the next 128-bit line burst sequence. During clock cycle 8, the BE bits are set to 0x8, indicating only one of the four Bytes contain valid packet data. The EOP signal is also asserted during clock cycle 8 indicating the end of packet for the currently active Port 7.

Asserting the EOP signal causes the internal logic to terminate writes to the current Port, as well as automatically incrementing to the next address for the FIFO partition.

During clock cycle 10, the FPGA returns to Port 3, completing the packet with two 32-bit bursts, and asserting the EOP during the latter write cycle. All BE bits set to a logic ‘1’ during the second write indicates all four bytes of the write are valid.

As shown, data may be written to different Ports, in a fast back-to-back fashion, without any reduction in data throughput.

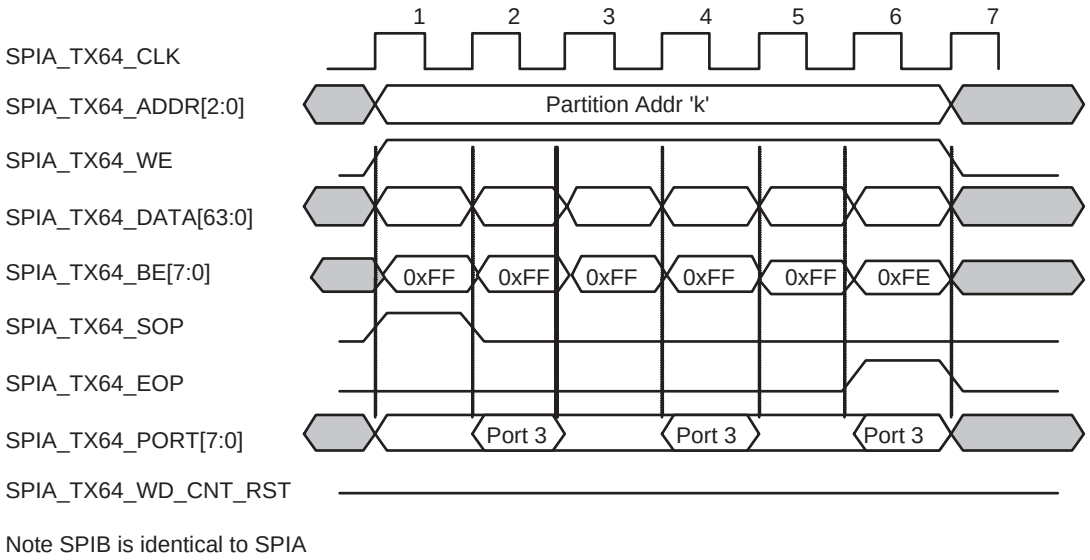
Some points to note are:

- According the SPI4 specification, the smallest packet that may be transmitted across the Transmit SPI4 link consists of a single 16-Byte transfer. This means a single occurrence of SOP and EOP may occur within any DPRAM location.
- According to the SPI4 specification, only a single Port's data may occur within a 16-Byte cycle. As a consequence, data from a single Port is permitted to be written to any single 128-bit location within the DPRAM partitions.

- The SOP must always be written into the first byte location of the 16-byte DPRAM word. (Shown in Figure 7 for the 32-bit interface mode.) With the Lattice SPI4 interface, the EOP may occur during any of the four 4-byte mode writes to fill the 16-byte DPRAM, and will be associated with the correct 128-bit line within the partition memory.

The next two figures show the 64-bit write protocol from the FPGA to the TX DPRAMs. It is functionally identical to the 32-bit mode, except two writes are required to fill an address line within a FIFO partition instead of four. Figure 13 shows the basic interface signals for 64-bit mode of operation.

Figure 13. Basic 64-Bit Data Write Protocol to DPRAM



Note SPIB is identical to SPIA

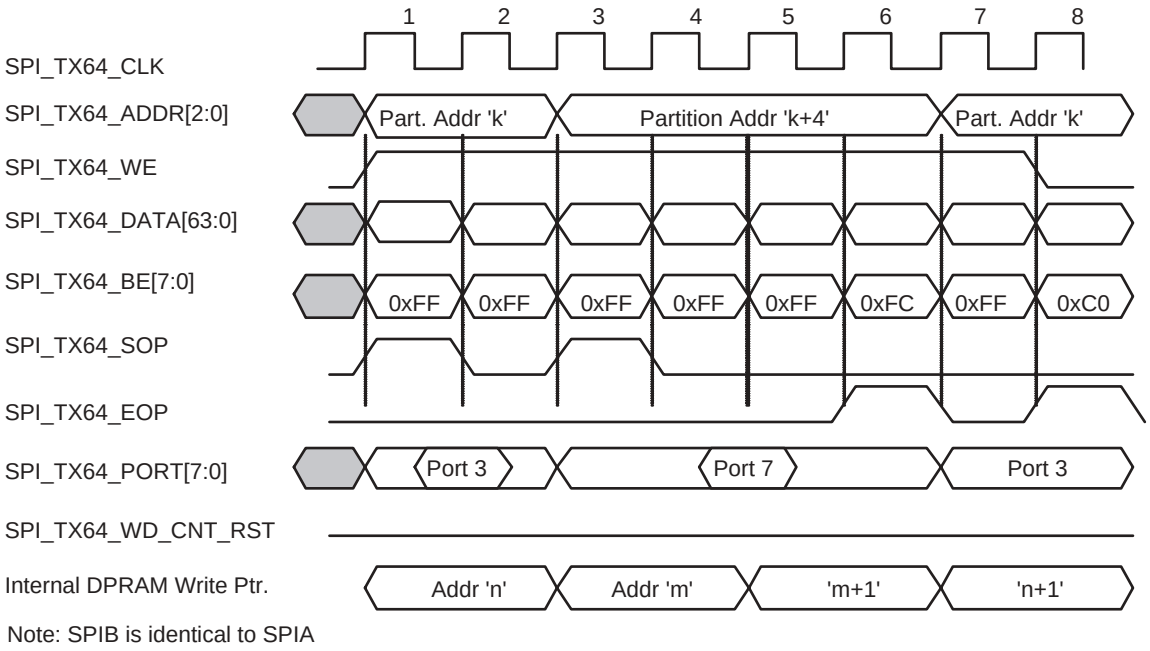
During Clock cycle 1, the FPGA asserts the partition Address, which indicates which partition within the DPRAM the data and control signals are to be written to. The Write Enable signal is also asserted, qualifying all data and control signals. From the figure, this is the start of a packet, so the SOP signal is also asserted. Data is presented to the interface, as well as the Byte Enable (BE) and Port ID signals. The BE vector is used to qualify which Bytes on the SPI4 transmit link are valid data. When operating in 64-bit mode, there are 8 BE bits. During clock cycle 6 the BE bits are set to 0xFE, indicating 7 of the eight Bytes contain valid packet data.

As with 32-bit mode, the Port ID signal should be asserted during an entire write sequence, although as a minimum it must be asserted at least during the second write to the current DPRAM address line. In the figure above, the Port ID must be asserted during clock cycles 2, 4, and 6.

During clock cycle 6, the FPGA indicates an End of Packet condition for the current selected Port. Asserting the EOP signal causes the internal logic to terminate writes to the current Port. Also, the internal DPRAM pointer logic is advanced after every second write to the DPRAM when in 64-bit aggregation mode.

As with the 32-bit aggregation mode, the “internal DPRAM Write Ptr” advances to the next valid address within partition range, but after every 2 writes when in 64-bit aggregation mode. This is because two 64-bit writes fill an entire 128-bit DPRAM line entry. The same interface logic in the FPGA can also be used as described in 32-bit mode, but the counter implemented in FPGA logic is simply a one-bit toggle FF.

Figure 14. 64-Bit Write Protocol Using Fast Back-to-Back Transfers

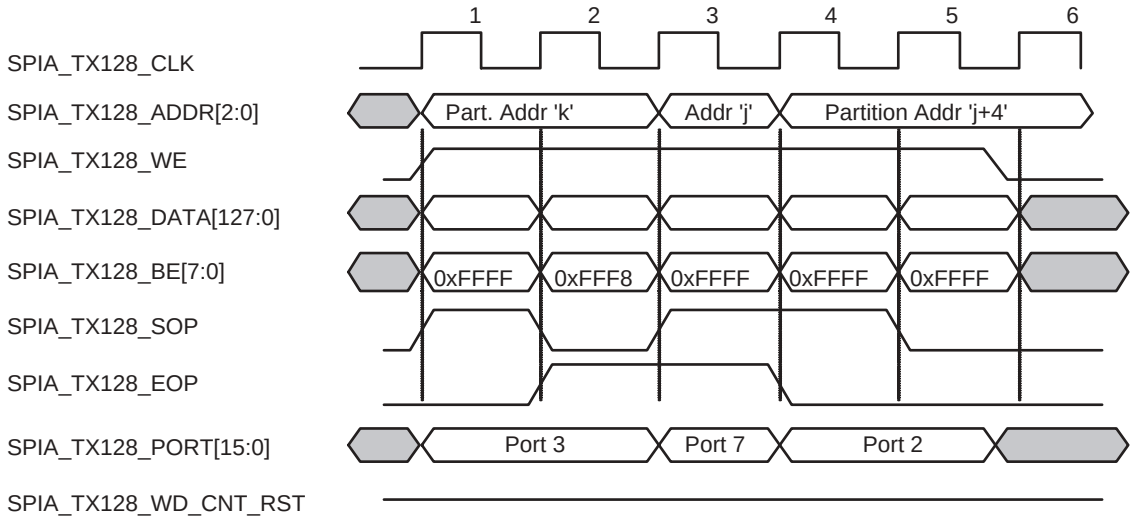


Note: SPIB is identical to SPIA

As with Figure 13, the FPGA initiates the start of a packet by asserting the FIFO partition address, WE, Data and other pertinent control signals in Figure 14. During clock cycle 2, an entire 128-bit line has been written and the internal logic increments it's write pointer to the next location within the FIFO partition. During clock cycle 3, the FPGA begins writing to a different DPRAM FIFO using a back-to-back transfer. A new Partition address and Port ID are presented to the DPRAM. Since this is the Start of the Packet, the SOP is asserted. During clock cycle 5, the internal write pointer is advanced to the next address within the FIFO partition range. On clock cycle 6, the FPGA terminates the packet by asserting the EOP signal. The internal DPRAM Write Pointer advances during clock cycle 6 to the next location and then waits for the FPGA to attempt another write access to a partition within the DPRAM. Beginning with clock cycle 7, the FPGA resumes writing to the original port. The FPGA presents the correct Partition address and asserts the required control signals. The internal write pointer decodes the Partition address and sets the internal write pointer to address 'n+1'. As with 32-bit mode, in 64-bit aggregation mode, data may be written to different Ports in a fast back-to-back fashion without any reduction in data throughput. During clock cycle 8, the FPGA terminates writing to the current packet by asserting the EOP input signals and the correct number of BE signals.

Figure 15 below shows the protocol required when operating in 128-bit aggregation mode. A single write from the FPGA fills an entire line within a FIFO partition.

Figure 15. 128-Bit Write Protocol



NOTE: SPIB is identical to SPIA

SPI4 Transmit Interface - Detailed Description

The FPGA transmit interface provides a simple FIFO-like interface, simplifying write accesses. Asserting the correct 3-bit address with the associated write enable signal causes transmit data and sideband control signals to be registered for writes to a FIFO partition within the DPRAMs. When addressing a particular FIFO partition within the DPRAM, the associated FIFO fill level of the addressed partition is immediately reflected back to the FPGA, enabling the user to monitor the current fill level of a particular partition. Based upon the configured fill level, a logic '1' on the FIFO_FULL signal will indicate either a 3/4-full condition or a completely full condition for the indexed partition.

Transmit DPRAMs

There are four DPRAMs referred to as banks 0, 1, 2 and 3. Each bank has a 32-bit data interface to the FPGA. Each DPRAM has its own individual write enable and write clock allowing it to be written by the FPGA application independently.

Each DPRAM bank is configured by software to operate in either 32-bit, 64-bit or 128-bit aggregation mode. Every DPRAM bank is also configured to contain 1, 2, 4 or 8 virtual FIFOs. The user determines the number of FIFOs depending on the number of ports and buffer requirements for the ports as required by a given application. The programming of a DPRAM bank in 32-bit, 64-bit or 128-bit mode and programming of the number of virtual FIFOs within a DPRAM bank is done through software as shown in Table 8. Note that in 64-bit mode, DPRAMs 0 and 1 must be configured identically. The combined DPRAM pair is referred to as DPRAM “0”. Similarly DPRAM pairs 2 and 3 must be configured identically. This combined DPRAM pair is referred to as DPRAM “2”. In 128-bit mode, all DPRAMs must be configured identically. The combined DPRAM banks are collectively referred to as DPRAM “0”. The aggregation modes can be used in five possible combinations as shown in Table 6. The size of the embedded data and control FIFOs for each mode is shown in Table 7. The user accesses a virtual FIFO using the 3-bit FIFO read address (refer to Table 2, Table 3, and Table 4) from the FPGA. Table 5 shows the indexed partition based upon the configured DPRAM partitioning and aggregation mode.

Table 5. FIFO Address Based on Programmed Virtual FIFOs

SPI[A,B]_TX_ADDR[2:0]	Description
XXX	Bits are ignored when the DPRAM is configured to support a single partition. (Device Select bits can be set to any value in single partition mode).
000	Selects Partition 0 when the DPRAM is configured to support 2, 4 or 8 partitions.
001	Selects Partition 1 when the DPRAM is configured to support 2, 4 or 8 partitions.
010	Selects Partition 2 when the DPRAM is configured to support 4 or 8 partitions.
011	Selects Partition 3 when the DPRAM is configured to support 4 or 8 partitions.
100	Selects Partition 4 when the DPRAM is configured to support 8 partitions.
101	Selects Partition 5 when the DPRAM is configured to support 8 partitions.
110	Selects Partition 6 when the DPRAM is configured to support 8 partitions.
111	Selects Partition 7 when the DPRAM is configured to support 8 partitions.

Table 6. Possible Combinations of Aggregation Modes

Mode	Valid
Banks 0 to 3 in 32-bit aggregation mode	Yes
Banks 0, 1 in 64-bit aggregation mode, Banks 2,3 in 64-bit mode	Yes
Banks 0,1 in 64-bit mode, Banks 2 & 3 in 32-bit mode	Yes
Banks 0,1 in 32-bit mode, Banks 2 & 3 in 64-bit mode	Yes
All banks in 128-bit	Yes
Bank 0 in 32-bit mode, Bank 1 in 64-bit mode or vice-versa	No
Bank 2 in 32-bit mode, Bank 3 in 64-bit mode or vice-versa	No

Within the embedded core, data is quad-word aligned, and written to the DPRAMs only on 128-bit boundaries. When operating in 32-bit or 64-bit mode, with the exception of EOP terminated write accesses, either 4 or 2 write accesses, respectively, must be performed before the internal FIFO address pointer can be incremented to the next location. It is required for the FPGA to perform all data writes on 128-bit boundaries. Thus, the application will write entire “lines” to the DPRAM causing the internal address pointer to increment to the next location in memory. If the FPGA attempts to write to a FIFO partition within the DPRAM and none of the 16 BE bits are set, the data will not be written into the FIFO and the internal write pointers will not be advanced. This is done to prevent erroneous data from being sent across the SPI4 link, as well as optimize FIFO memory usage.

Under certain circumstances, the application may require the SPI4 interface to disable a particular DPRAM from being serviced. To accommodate this, the user can assert the proper 'Link Disable” (LINK_DIS) signal indicating to the DPRAM logic it may service a particular DPRAM until it encounters an End of Packet (EOP). As long as the assertion of the LINK_DIS for a particular partition is asserted, all partitions within a DPRAM will not be serviced after the initial EOP is encountered. If the application attempts to perform further writes to any of the disabled FIFO partitions, data will be accepted up to the point the FIFO becomes full. After the FIFO fills, data will no longer be accepted. Data written to the disabled DPRAM will remain latent within the memory. Upon removal of the Link Disable signal, ports from the enabled DPRAM will be serviced according to the configured Calendar sequence.

Table 2, Table 3 and Table 4 above give a description of the FPGA interface signals. A detailed description of the DPRAMs is provided below.

TX DPRAM

The TX DPRAMs consist of four 256 x 160 memories, that can each be partitioned into a maximum of 8 equally sized virtual partitions, providing a maximum of 32 virtual FIFO partitions. By providing a simple FIFO-like interface to the FPGA, the FPGA can access FIFO partitions by asserting a 3-bit address field. Internal logic maintains both write and read pointers and provides a programmable FIFO_FULL flag to the FPGA logic in order to prevent overflow conditions.

Data may be written into the DPRAMs using either multiple 32-, or 64-bit interfaces, or alternatively using a single 128-bit interface. The sideband control bus is present for each individual interface. The DPRAM partition depths are a function of the interface width used, as well as the number of configured partitions. Table 7 shows the depth of virtual partitions for data and sideband control information based upon the write interface width and the number of partitions.

Table 7. Memory Size for Each Aggregation Mode and Partitioning

Write Interface Mode	# of Configured Virtual FIFO Partitions	DATA FIFO Depth (in Bytes)	Control FIFO Depth (in Bytes)
32-bit	1	2K	512
	2	1K	256
	4	512	128
	8	256	64
64-bit	1	4K	1K
	2	2K	512
	4	1K	256
	8	512	128
128-bit	1	8K	2K
	2	4K	1K
	4	2K	512
	8	1K	256

Partitioning of the Dual Port RAMs is done through customer programmable registers. Table 8 below shows Register Settings for Aggregation Mode and Partition Size.

Table 8. Register Settings for Aggregation Mode and Partition Size.

DPRAM Aggregation Mode Address 30903 (SPIA), 30A30 (SPIB)	Virtual FIFO Partition Size Address 30927 (SPIA), 30A27 (SPIB)	Configuration
DPRAM 0 (register Bits 0:1) = 00	Register Bits 0:1 = 00	32-bit data width, partition size 1
	01	32-bit data width, partition size 2
	10	32-bit data width, partition size 4
	11	32-bit data width, partition size 8
01	00	64-bit data width, partition size 1
	01	64-bit data width, partition size 2
	10	64-bit data width, partition size 4
	11	64-bit data width, partition size 8

DPRAM Aggregation Mode Address 30903 (SPIA), 30A30 (SPIB)	Virtual FIFO Partition Size Address 30927 (SPIA), 30A27 (SPIB)	Configuration
10	00	128-bit data width, partition size 1
	01	128-bit data width, partition size 2
	10	128-bit data width, partition size 4
	11	128-bit data width, partition size 8
DPRAM 1 (Register Bits 2:3) = 00	Register Bits 2:3 = 00	32-bit data width, partition size 1
	01	32-bit data width, partition size 2
	10	32-bit data width, partition size 4
	11	32-bit data width, partition size 8
01	00	64-bit data width, partition size 1
	01	64-bit data width, partition size 2
	10	64-bit data width, partition size 4
	11	64-bit data width, partition size 8
10	00	128-bit data width, partition size 1
	01	128-bit data width, partition size 2
	10	128-bit data width, partition size 4
	11	128-bit data width, partition size 8
DPRAM 2 (Register Bits 4:5) = 00	Register Bits 4:5 = 00	32-bit data width, partition size 1
	01	32-bit data width, partition size 2
	10	32-bit data width, partition size 4
	11	32-bit data width, partition size 8
01	00	64-bit data width, partition size 1
	01	64-bit data width, partition size 2
	10	64-bit data width, partition size 4
	11	64-bit data width, partition size 8
10	00	128-bit data width, partition size 1
	01	128-bit data width, partition size 2
	10	128-bit data width, partition size 4
	11	128-bit data width, partition size 8
DPRAM 3 (Register Bits 6:7) = 00	Register Bits 6:7 = 00	32-bit data width, partition size 1
	01	32-bit data width, partition size 2
	10	32-bit data width, partition size 4
	11	32-bit data width, partition size 8
01	00	64-bit data width, partition size 1
	01	64-bit data width, partition size 2
	10	64-bit data width, partition size 4
	11	64-bit data width, partition size 8
10	00	128-bit data width, partition size 1
	01	128-bit data width, partition size 2
	10	128-bit data width, partition size 4
	11	128-bit data width, partition size 8

Based upon the traffic characteristics of individual ports and number of ports supported, the user can tailor the FIFO depth to optimize performance at the FPGA/embedded core interface. When operating in 32-bit mode, four write cycles are necessary to complete an entire 128-bit line. 64-bit mode requires 2 write accesses, with 128-bit mode using a single clock per line fill.

When data is read out of the DPRAMs, via the AMA logic, it is read using a 128-bit data bus. Therefore, for each port serviced, one clock cycle is consumed to read 16 data Bytes and 4 control Bytes from each addressable location within a FIFO partition. This corresponds to the SPI4 requirement that only a single SPI4 port's data may exist on the transmit data bus within any Burst Cycle. A Burst Cycle is made up of 16 Bytes.

Note, for all modes of aggregation, if an entire 128 bits is written to any FIFO partition, with all 16 BE bits deasserted, the Quad-word will not be written to the DPRAM, nor will the internal write pointer be incremented for the selected FIFO partition.

Within the Transmit DPRAM, there are user-programmable fill level thresholds for each enabled partition. The programmable levels are either 3/4-full or absolutely full. When the fill level of an addressed partition exceeds the programmed fill level, a FIFO full condition is reported back to the FPGA. Because the fill level operates only on 128 bits, the full flag is only updated after an entire line has been written. The full condition will persist until the FIFO has been serviced and the fill level drops below the programmed fill threshold.

In order to take a DPRAM temporarily out of service without updating the Calendar, the FPGA can assert the Link Disable (SPI_TX_LINK_DIS[3:0]) input signal associated with the DPRAM to be disabled. The embedded core will continue to service the disabled ports within the DPRAM until it encounters the next End of Packet indicator. Upon detecting an EOP, it will discontinue service to the port and send Idle data on the SPI4 transmit link during the disabled port's service period. The FPGA is permitted to continue to write data to the disabled port FIFO, as it will obey all the functionality described above regarding FIFO fill level and status. When the DPRAM is to be put back into service, the user simply deasserts the LINK_DIS signal. The associated ports will then be serviced according to the configured Calendar sequence.

Address Map Arbiter (AMA) Logic Block

The Address Map Arbiter (AMA) logic block is a slave device to the TX Calendar Control logic. Upon receiving a valid AMA_ID and BURST_VAL vectors, the AMA will attempt to read out data from the indexed DPRAM FIFO partition along with the associated sideband control signals.

In order for user data to be serviced according to the Configured Calendar sequence, the TX Calendar Control logic presents the address of the SPI4 port to be serviced, along with the number of data reads that are to be performed. The AMA_ID field indicates which FIFO partition within the DPRAM is to be serviced. The BURST_VAL field indicates the number of TREFCLK cycles to be spent reading data from the indexed DPRAM. This corresponds to a data cycle as defined within the SPI4 specification. The AMA reads 160 bits of data from the DPRAM per clock cycle: 128 bits of user Transmit data and 32 bits of sideband control information. (A detailed discussion of how to relate Transmit Calendar entries to the DPRAM is provided in the TX Calendar Control Logic section).

Upon receiving a valid AMA_ID, the AMA logic will service a port by reading data from the FIFO partitions for BURST_VAL clock cycles. Both data and sideband control signals are read and sent directly to the TDP logic block. In the event a FIFO partition becomes depleted before BURST_VAL number of reads is performed, the AMA will send Idle data to the TDP and will override the associated Byte Enable (BE) control signals, indicating the current data is not user data. This will temporarily reduce the SPI4 transmit link data throughput, but will not affect the operation of the Transmit link itself.

When reading data from FIFO partitions, and the FIFOs are empty, it is possible to gain back some lost data throughput on the SPI4 Transmit link. Enabling the TX_BURST_TERMINATION control register within the embedded core causes the AMA to advance to the next port for service within four clock cycles after detecting the FIFO is empty. This feature works in conjunction with the TX Calendar Control logic. Upon detecting the currently serviced FIFO has run empty, the AMA block sends a control signal to the Calendar Control logic, which then immediately advances to the next location within the Calendar and presents a new set of AMA_ID and BURST_VAL fields to the

AMA block. Because there is a four-clock latency between detection of an empty FIFO and the assertion of the next port to service, data bandwidth gains can be achieved on ports with larger BURST_VAL values.

Transmit Data Protocol (TDP) Logic Block
The Transmit Data Protocol (TDP) block is responsible for receiving and formatting SPI4 packet data and all associated sideband control signals, and formatting the data into the correct SPI4 protocol. This block also performs DIP-4 calculation, SPI4 data word, control word and Training control word generation. Additionally, this block attempts to perform bandwidth optimization by combining ordered Bytes from 2 consecutive 128-bit words from the AMA whenever possible. This is done to maximize data throughput on the Transmit SPI4 link.

The TDP provides a small FIFO to allow for data buffering in order to concatenate valid Bytes with idles. In the event the FIFO becomes full, it sends a FIFO_FULL flag to the Transmit Calendar Control Word logic to throttle data transmission. Flow control is necessary for small periods of time while Training patterns are being sent across the Transmit SPI4 link.

Data formatting performed in this block consists of alignment, DIP-4 calculation and training pattern generation.

Aligner:
Using the SOP, EOP and Port control signals the TDP determines what type of control word is necessary to be transmitted with the corresponding data. Using the BE bits, the TDP can determine how to concatenate data into sequences of continuous 16 Byte bursts whenever possible.

DIP-4 Calculation:
After aligning the data and control words, the TDP calculates the odd parity codeword according to the OIF SPI4 specification.

Training Pattern Generation:
There are two causes of generating a Training sequence for the Transmit SPI4 link:

- Using the user-programmable DATA_MAX_T register, or
- The TDP receives four or more consecutive '11' patterns from the status block, indicating excess DIP-2 errors.

Using the programmed DATA_MAX_T register along with the user-programmable ALPHA register, the TDP internally counts down the correct number of clock cycles before transmitting training sequences across the Transmit link. The ALPHA register is used to indicate the number of Training sequences to be sent.

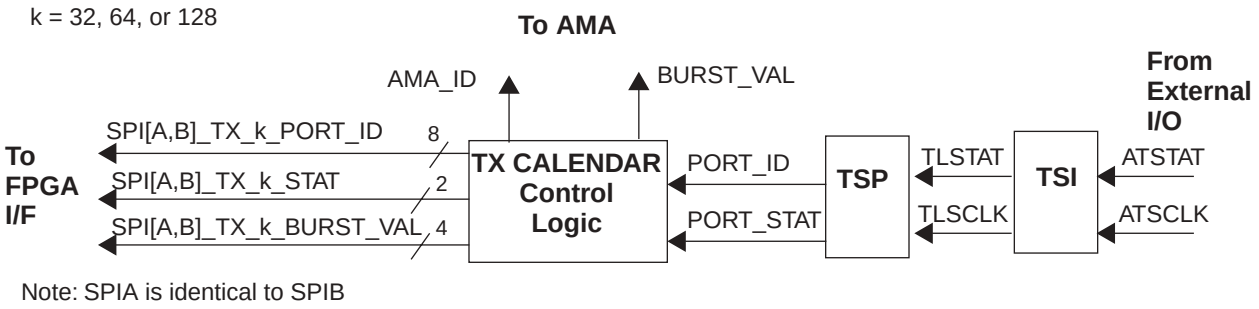
When the Transmit status receives excessive errors, the status block sends consecutive '11' patterns to the TDP. According to the SPI4 specification, the TDP will begin sending training patterns across the SPI4 link until the status error is remedied.

Transmit Data Output (TDO) Logic Block
The Transmit Data Output (TDO) block contains the high-speed serializer, which uses an x8 clock, synthesized by an internal PLL, to generate the high-speed data from the low-speed 128-bit data. Data is transmitted off-chip using a 16-bit LVDS data bus TDAT[15:0], a LVDS control bit TCTL and a source synchronous clock TDCLK. The 16-bit data bus and control are DDR signals relative to the TDCLK. The LVDS buffers contained within the TDO are fully compliant with SPI4 electrical specifications.

The TDO uses the internal embedded core transmit reference clock (TREFCLK) to generate an x4 TDCLK. The TDCLK and TREFCLK are both derived from the user-provided reference clock. The TREFCLK is used as the system clock for the entire Transmit SPI4 interface logic. The TREFCLK is also sent to the FPGA for the customer to use as a synchronizing signal source if desired.

In order to support 10 Gbits/s throughput, the minimum frequency of TDCLK needs to be 622 Mbits/s (311 MHz DDR). For applications that require less than 100 - 200 Mbits/s on TDCLK, the PLL in the TDO block can be bypassed. This allows for operation as low as 100 MHz on the Transmit SPI4 link.

Figure 16. Transmit Calendar and Status Block

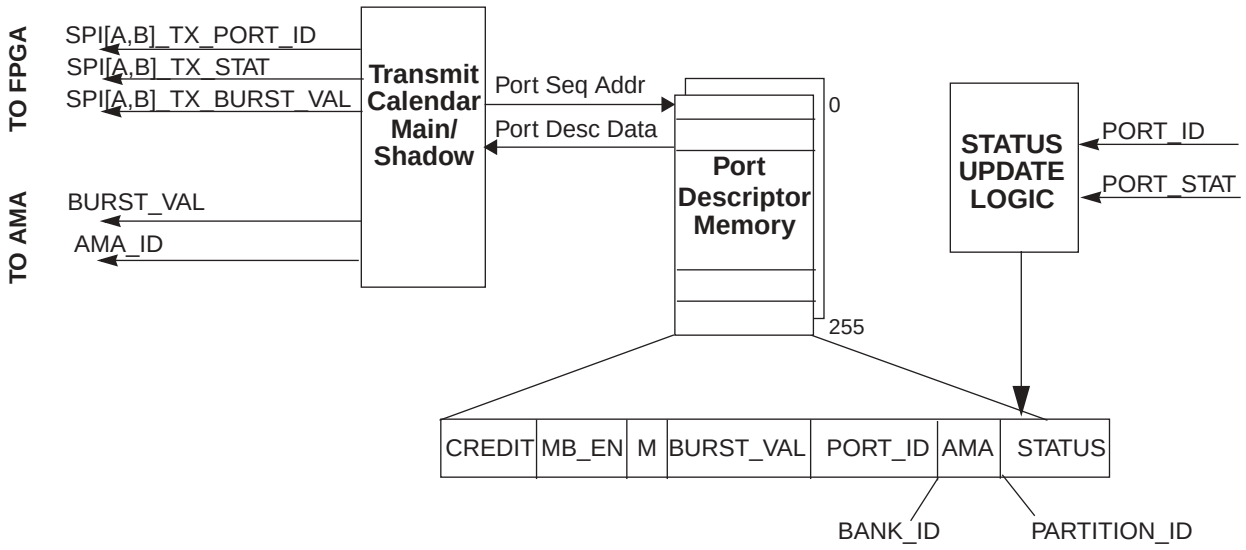


Note: SPIA is identical to SPIB

Transmit Calendar (TCC) Logic Block

The Transmit Calendar Block (TCC) is responsible for maintaining the transmit main and shadow calendars, providing port service information to the AMA, integrating transmit status update information into the servicing of ports according to the SPI4 specification, and providing port service information to the FPGA for traffic monitoring.

Figure 17. Transmit Calendar Control Logic Functional Diagram



The Transmit SPI4 interface contains both main and shadow Calendars enabling a hitless switch over between an old and new Calendar schedule. Each Calendar is 1023 locations deep, providing a high degree of flexibility and customizable port service algorithms. The user can configure the Calendar(s) through software configuration registers. (See the transmit memory map register set for identification of individual configuration register addresses).

NOTE, in order to effect a hitless switch at the logical port level, the user must ensure that port data associated with any Calendar modifications have been terminated correctly before the Calendar switches. For example, if the Shadow Calendar eliminates Port 3 from the Calendar sequence, the user must ensure there is no data associated for Port 3 in the FIFO partitions when the Calendar switches. Otherwise, service will be granted for an out-of-service port, causing higher layer protocol exceptions.

The Calendar length is also configurable to indicate the number of valid port entries. Programming the CALENDAR_LEN register indicates how many entries within the Transmit Calendar are enabled for sequencing. Since the Calendar is 1023 locations deep, a single port may be serviced more than once per Calendar sequence.

Port Service Information

The TCC sequentially runs through the Calendar sequence retrieving the port number that is to be serviced next. The field is an 8-bit vector, supporting the entire 256 possible entries of the SPI4 Calendar. The retrieved port number is used to index a Port Descriptor Memory (PDM) to retrieve all pertinent information required for maintaining proper port servicing.

There are several fields within the PDM that are required to be configured by the user before the Transmit interface can be enabled. A description of the fields within the descriptor is given below:

- **PORT_ID** - Port information sent to the FPGA when a port is being serviced. This value is configurable by the user and can contain either the SPI4 number of the port, or can contain an alias the FPGA uses to map proprietary interface channels to the corresponding SPI4 port (See the 'M'-bit description below). The FPGA logic can also use the Port_ID for status acquisition, or other link layer flow control functions. After reset this field defaults to a value of 8'h00.
- **BURST_VAL** - Port information sent to the FPGA (when a port is being serviced) and AMA. This user-configurable field indicates the maximum number of 16-Byte burst accesses the port is permitted to consecutively transmit across the SPI4 link, before segmentation must be performed. The FPGA can use this field to indicate how much data to source for the port being polled to maintain buffering within the DPRAM FIFO partition. After reset this field defaults to a value of 4'h0.
- **AMA** - This user configurable field consisting of a DPRAM BANK_ID (left-most two bits) and a DPRAM PARTITION_ID, indicates to the AMA (AMA_ID) which physical DPRAM partition address to poll for port servicing. After reset, this field defaults to a value of 5'h00.
- **M** - Indicates whether the programmed Port_ID or the SEQ_PORT_NUM contained within the Calendar Sequence Table should be broadcast to the FPGA. This is performed on a per port basis, providing a mapping function between proprietary FPGA port mapping and the SPI4 Transmit Calendar port number. EX: SPI4 Port Number 138 maps to FPGA interface Device #7, Port #17. When the 'M'- bit is set to a logic one, the programmed Port_ID (i.e. 138) is broadcast to the FPGA for the port being serviced; when set to a logic zero, the Port number from the Transmit Calendar is sent to the FPGA for that particular port. After reset, this field should default to a value of 1'b0.
- **Credits** - Indicates the number of bursts the port is permitted to transmit across the SPI4 link between status updates, for that particular port. This field is updated with calculated credit values after the port is serviced. Whenever the receive status from the TSP block presents a port status value of 2'b11, the Credit field for the indexed port is to be cleared to a value of 10'h000. This disables the port. The user may configure an initial value for each enable port. After reset, this field is to default to a value of 10'h000. This field should not be altered after the Transmit Calendar is placed in operation.
- **MB_EN** - This user-configurable register indicates whether the associated port supports MAX_BURST parameters as defined within the SPI4 specification. The Transmit Calendar supports a single MAX_BURST1 and a single MAX_BURST2 value. Logic one indicates that the configured port supports MAX_BURST. After reset, this field is to default to a value of 1'b0. This is configurable on a per-port basis.
- **STAT** - Contains Transmit FIFO status from the TSP block. Used in conjunction with the MB_EN field to update the Credit field for the indexed port. Also sent to the FPGA for status monitoring. After reset this field is set to a value of 2'h3, indicating the port is disabled.

Status Update Logic

Transmit status is received from the TSP block (described later). Both the port number and associated status of the port is sent to the Status Update Logic. This block ensures that the status for each port, as it is received, is updated correctly. The status is received according to the programmed far-end Calendar, and will generally be received asynchronously (relative to port servicing) to the local Transmit Calendar polling.

By providing the BURST_VAL, Stat, and Port_ID value to the FPGA, application logic can maintain the required data bandwidth it needs to source, in order to keep the respective FIFO partition from over/under-flowing. These

signals can also be used in conjunction with FPGA control logic to oversee the operation of the Memory Controller for accessing the Transmit SPI4 port data.

Transmit Status Input (TSI) Block
The Transmit Status Input (TSI) block provides the interface to the SPI4 Transmit Status Signals. These signals can be either LVDS or LVCMOS input buffers. This is selected using the SPI4_STATUS_IO_SEL register setting. The Status interface supports quarter-rate mode only, as defined within the SPI4 specification. The user has the option to use the status input signals either in phase with status input clock, or out of phase with the status input clock, by optionally inverting the input clock signal on-chip. Setting SPI_SATM_A and SPI_SATM_B registers to a "1" will cause them to be in phase, setting to a "0" will cause them to be out of phase.

The TSI block receives transmit status and provides both the status information and a derived clock to the TSP block.

Transmit Status Protocol (TSP) Block
The Transmit Status Protocol (TSP) block receives status from the TSI block and uses the user-configured transmit calendar to associate the current received status with a port. This block performs FIFO status decoding, buffering and DIP-2 calculations. As defined within the SPI4 specification, if DIP-2 errors occur, it sends an error flag to the TDP block for processing. The TSP provides port status information to the Transmit Calendar logic along with the associated port number. The Calendar logic then updates the appropriate Port STAT field. The TSP block also supports Hitless Bandwidth Provisioning as defined within Appendix G of the SPI4 specification.

Transmit Calendar Operation
This section gives a description of how the Transmit Calendar operates, in order to assist users in understanding how to configure the TX SPI4 programmable Calendar and configuration registers to fit their application. Figure 16 and Figure 17 will be used as reference. The Transmit Calendar logic is synchronous to TREFCLK for it's internal operations.

The Transmit Calendar logic uses an indirect addressing scheme to acquire port sequence information. This enables the user to configure the Calendar easily, while minimizing the amount of information that needs to be programmed within the Calendar table. Indirect addressing also enables users to configure the entire Port Descriptor Memory at initialization time, without having to actually enable all the entries. The user has access to modify port information within the PDM at any time, whereas the Calendar(s) can only be updated while it is not in service.

The Transmit Calendar sequentially runs through all the user-enabled port entries within the 1K memory. The Calendar rolls over to the first entry whenever the user-configured maximum Calendar length is reached. The time between polling the same port is a function of the several factors, including user-configured BURST_VAL per port, and whether the TX_BURST_TERMINATION is enabled (set to LOGIC "1").

- The user must configure the TX_CAL_LEN_MAIN to indicate the number of valid entries to sequence through in the Main Calendar. Not all applications require 1023 locations to be polled.
- If the Shadow Calendar is to be used, the user must configure the TX_CAL_LEN_SHD to indicate the number of valid entries to sequence in the Shadow Calendar. The Shadow Calendar is independent of the Main Calendar, and may have a different number of entries.
- The user must either enable TX_BURST_TERMINATION or leave it disabled. By default it is disabled.

When a port is indexed within the Calendar, the indexed vector serves as the address to the Port Descriptor Memory, to fetch all the necessary information to service the selected port. The Port Descriptor Memory (PDM) provides 256 entries; one for each possible SPI4 port. The PDM contains all the port servicing control information. When a port is indexed via the Calendar, the Calendar pointer is incremented to the next location, serving as a pseudo prefetch in case TX_BURST_TERMINATION is enabled. Every time a Port is indexed in the PDM, an internal counter loads BURST_VAL and begins decrementing. Upon nulling out, the next Calendar entry is used to index the PDM to fetch the next port to be serviced. Again, the Calendar advances to the next enabled location. As defined within the SPI4 specification, BURST_VAL indicates the number of 16-Byte cycles a port may be serviced

before segmentation must occur. Using BURST_VAL, the Transmit Calendar can maintain constant Port servicing from the Dual Port memories.

- The user must configure a BURST_VAL value for each enabled SPI4 port. The PDM contains 256 locations, i.e. one for each possible SPI4 port.

The Transmit Calendar broadcasts the AMA_ID and BURST_VAL vectors, fetched from the PDM, to the AMA block. The AMA will use the ID value to address the physical entry of the port that is to be serviced. Note the AMA_ID value is the physical address of the DPRAM and is made up of two user-configurable fields, the PARTITION_ID and BANK_ID.

- The DPRAMs consist of 4 banks, and each may be partitioned into a maximum of 8 virtual FIFOs each. The user must configure both the PARTITION_ID[2:0] and BANK_ID[1:0] fields for each enabled port within the PDM. The PARTITION_ID vector is identical to the SPIA_TX32_ADDR_0[2:0]. The user configures the same partition address for a particular port that is used as the address on transmit DPRAM write interface. The BANK_ID vector is simply derived from the physical interface used at the FPGA/embedded core interface, as shown below.
  - 32-bit mode → [00,01,10,11], providing up to 4 interfaces
  - 64-bit mode → [00, 01], providing up to 2 interfaces
  - 128-bit mode → [00], providing a single interface

For all combinations of aggregation, the values given above apply.

- More than one port can be mapped to the same partition. Simply program the same PARTITION_ID and BANK_ID values in all the PDM port entries that share the same partition.

The Transmit Calendar also broadcasts the BURST_VAL, current status and either the transmit Calendar entry vector or a user-configurable port value to the FPGA. Selection between the Calendar vector and the configured value is decided with the 'M'-bit. When logic '0', the Calendar Port vector is broadcast to the FPGA. Otherwise, the user-configurable PORT_ID field is sent. The optional PORT_ID is provided in order to assist the FPGA in mapping it's proprietary channel/port numbers to the SPI4 Calendar port value. By configuring the PORT_ID value and setting the 'M'-bit for particular ports, the FPGA will receive the programmed value instead of the SPI4 port number. This feature is very useful when the user is sharing a particular FIFO partition within the DPRAM between multiple ports. Providing these fields to the FPGA enables users to updated statistics of port servicing for external data schedulers and the external Memory Controller.

- If the user desires to have proprietary channel/port values broadcast to the FPGA, the 'M'-bit must be set to a logical “1”, and the associated PORT_ID field must be configured to the desired value.
- This is done on a per port basis, so some ports may be mapped, while others use the Transmit Calendar index vector.
- If the user maps more than one port to the same FIFO partition, the user is free to choose which Port ID field is sent to the FPGA.

As ports are serviced, their Credit fields are updated according to the requirements as defined within the SPI4 specification. Although not commonly used, the Transmit SPI4 Calendar logic allows the user to configure the Credit field with an initial value at configuration time. This is useful for some applications as well as providing diagnostic and test capabilities of the Calendar update logic. The Transmit Calendar also supports single Maxburst1 & Maxburst2 values, as defined within SPI4. The MB_EN field is enabled on a per-port basis and indicates whether the associated port differentiates between the defined Hungry and Starving states. When the MB_EN bit is set to a logical “1”, that particular port uses both the Maxburst1 and Maxburst2 fields: otherwise only the Maxburst1 field is associated with the Port’s Credit field update algorithm.

- In order to use the Maxburst1 and Maxburst2 fields, the user must program both fields to the required values. By default the value for these registers is “0”.
- The user must either configure the MB_EN field for each port to a logical “1” if the port have it’s Credit field updated using the Maxburst2 field. By default the Maxburst2 values are not used in the Credit update calculation.

- The user can optionally configure the Credit field to a particular value if desired, although it is not necessary for most applications.

During normal operation, Transmit status is received on a per port basis. The status field for each port is updated independent of where the Transmit Calendar polling exists. Internal logic prevents updates to be performed on a port while it is in use. Although uncommon, the SPI4 Transmit Calendar allows the user to configure the STAT field to some value, but only during initialization. This is intended to be used for diagnostics or for some other proprietary applications. Under normal operation, the user can ignore this function.

- The user can program the STAT[1:0] field for as many ports as desired within the PDM.

Table 9 below shows the SPI4 Status update definitions.

Table 9. SPI4 Status Field Definition

MSB	LSB	Description:
1	1	Reserved for framing or indicate a disabled status link.
1	0	SATISFIED Indicates the corresponding port's FIFO is almost full. When SATISFIED is received, only transfers using the remaining previously granted 16-byte blocks (if any) may be sent to corresponding port until the next status update. No additional transfers to that port are permitted while SATISFIED is indicated.
0	1	HUNGRY When HUNGRY is received, transfers for up to MAXBURST2 16-byte blocks or the remainder of what was previously granted (whichever is greater), may be sent to the corresponding port until the next status update.
0	0	STARVING Indicates that buffer underflow is imminent in the corresponding PHY port. When STARVING is received, transfers for up to MAXBURST1 16-byte blocks may be sent to the corresponding port until the next status update.

When transmitting across the SPI4 link, typically for clock speeds in excess of 350 MHz, dynamic alignment is required to maintain a coherent data link. In conjunction with dynamic alignment, SPI4 outlines a Training protocol that is periodically used to maintain receive data integrity. The SPI4 Transmit core supports data training through the use of a 32-bit variable named DATA_MAX_T. This 32-bit variable indicates the number of clock cycles that will occur between training sequences. SPI4 also defines an ALPHA variable indicating the number of times the training pattern is to be repeated during a training session. The TDP supports both these variables allowing the user to tailor both of these parameters to fit their particular environment and dynamic traffic requirements.

- The user must configure the 32-bit DATA_MAX_T variable to indicate the number of clock cycles that should lapse between training sequence. A value of 0x00000000 disables training. By default training is disabled.
- If training is to be used, the user must also configure the 16-bit ALPHA variable to indicate the number of times the training pattern is to be repeated. By default the value is 0x0000 which disables the training pattern.

If the number of Bytes within the training sequences do not fill entire SPI4 Burst Cycle, the unused Byte fields will be padded with Idle data in order to keep the training sequences aligned on 16 Byte boundaries.

Calendar Programming

On a SPI4 link, FIFO status information is received periodically over the status link (TSTAT bus on the SPI4 interface) from the device that sources data. Calendar is a means by which the SPI4 link conveys information to the TX data source about the availability of buffer space in the RX FIFOs that receive data from that data source. A calendar is a sequence of status messages that:

- Provides information on the buffer space for a port or traffic flow.
- Allows user to allocate bandwidth for a port or flow depending on the overall traffic characteristics.

The SPI4 status channel operates at 1/8th the SPI4 data rate, which is reasonable because data is always sent in 16-byte bursts. Thus, the fastest status update can be expected once in every 8 data clock cycles. For a given port, the frequency with which its status is reported to the far-end transmitter source depends on the allocated band-

width for the port. It is imperative that the transmitter and receiver devices are programmed with identical calendar sequences. The following examples illustrate this:

Suppose a channelized STS-48 has three STS-12 ports and four STS-3 ports (channels), each STS-12 port should have four times the bandwidth of an STS-3 port and 12 times the bandwidth of an STS-1 port within a single calendar cycle. The first step is to map each port to an 8-bit Port ID (address) as shown in Table 23

Table 10. Port ID Mapping

Port	Bandwidth	8-Bit Port ID (Port Address)
A1	STS-12	0x00
A2	STS-12	0x01
A3	STS-12	0x02
B1	STS-3	0x03
B2	STS-3	0x04
B3	STS-3	0x05
B4	STS-3	0x06

The calendar sequence is as follows:

A1, A2, A3, B1, A1, A2, A3, B2, A1, A2, A3, B3, A1, A2, A3, B4

The number of calendar entries is thus 16. This calendar sequence is programmed in the transmit calendar memory as follows:

- This example uses the main calendar. The calendar memory is selected by setting the TX_CAL_MEM_SEL bit (address 30917 in SPIA and address 30A17 in SPIB) to '1'.
- Writes are done to addresses 0x31000 - 0x3100F. This will correspond to entries 0x00 - 0x0F in the calendar memory. This is shown in Table 11. Note that the main calendar memory has 1023 locations and can be programmed through addresses 0x31000 - 0x313FE. The shadow calendar memory also has 1023 locations and can be programmed through addresses 0x31400 - 0x317FE.
- After programming the calendar memory, the TX_CAL_MEM_SET bit is set to '0'.
- The calendar length register TX_CAL_LEN_MAIN is set to 16 (address 30922 and 30923 in SPIA, 30A22 and 30A23 in SPIB).
- TX_CAL_M_MAIN (address 30921 in SPIA and 30A21 in SPIB) is set to the number of times the calendar sequence needs to be repeated between framing patterns.
- The shadow calendar can be programmed in the same identical fashion as the main calendar using the steps described above.

To enable hitless switching between main and shadow calendar memories, the TX_CAL_SW_EN bit (address 30916 in SPIA and 30A16 in SPIB) should be set to '1'. This will cause the transmit status logic to detect the calendar select word after the framing pattern. A "01" detected on the calendar select word selects the main calendar. A "10" on the calendar select word selects the shadow calendar. It is important to enable this feature in the receive device on the other end of the SPI4 link. If the receive device is a Lattice ORSPI4 device, this is done by setting RX_CAL_SW_EN bit (address 30916 in SPIA and 30A16 in SPIB) to '1'. This will cause the far-end receive status logic to insert the calendar select word after the framing pattern.

Table 11. Transmit Calendar Memory Contents

TX Calendar Memory Address	TX Calendar Memory Contents (Port ID)	Port Number
0x00	0x00	A1
0x01	0x01	A2
0x02	0x02	A3
0x03	0x03	B1
0x04	0x00	A1
0x05	0x01	A2
0x06	0x02	A3
0x07	0x04	B2
0x08	0x00	A1
0x09	0x01	A2
0x0a	0x02	A3
0x0b	0x05	B3
0x0c	0x00	A1
0x0d	0x01	A2
0x0e	0x02	A3
0x0f	0x06	B4

SPI4 FPGA/Embedded Core Timing Delays

When designing application logic within the FPGA to interface with the embedded core, it is important to consider the input setup and hold time requirements of the embedded core logic. There are two directions for signaling information between the FPGA and the embedded core. The first is where the FPGA sources transmit data and associated control signals into the embedded core. Data in this direction is always synchronous to the clock source within the FPGA. The other data path direction is used to convey Calendar port and status information from the embedded core to the FPGA. All signals in this category are synchronous to the buffered reference clock within the embedded core. All timing is handled in the *ispLEVER* software using frequency preferences on the associated clocks.

FPGA Sourced Data and Control

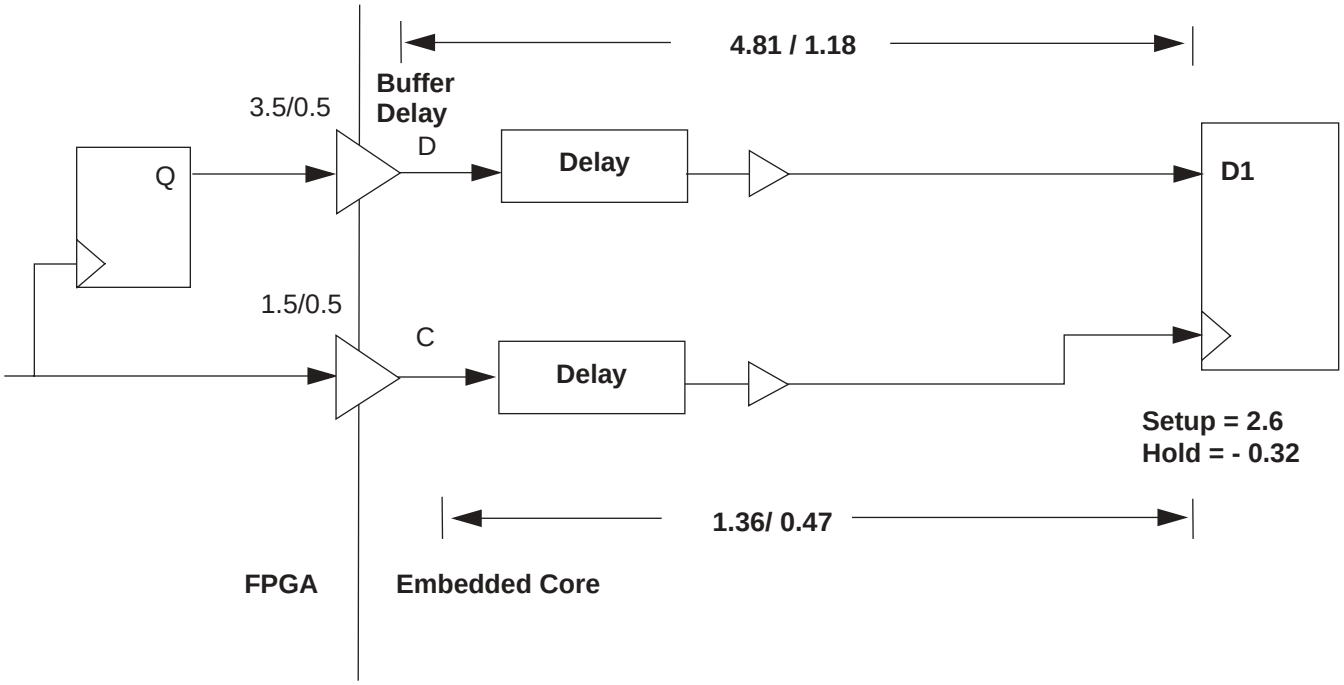
In this mode, the FPGA sends data into the embedded core relative to the rising edge of it's own clock as well as providing the clock for the embedded core. The embedded core uses the FPGA clock to register all input data and control signals. Figure 18 shows the Worst Case Slow and Worst Case Fast propagation delays for the FPGA input data and control signals as well as the clock signal.

For guaranteed performance it is important that all input signals from the FPGA be registered before being sent into the embedded core.

Clocking Schemes and Timing Diagrams - Transmit DPRAM Interface

Figure 18. Data: SPI[A,B]_DATA/SOP/EOP/BE/ERR/PORT/WE/ADDR/SPI[A,B]_TXk_WD_CNT_RST /LINK_DIS

Clock: SPI[A,B]_k_CLK_[3:0]; k=32, 64,128



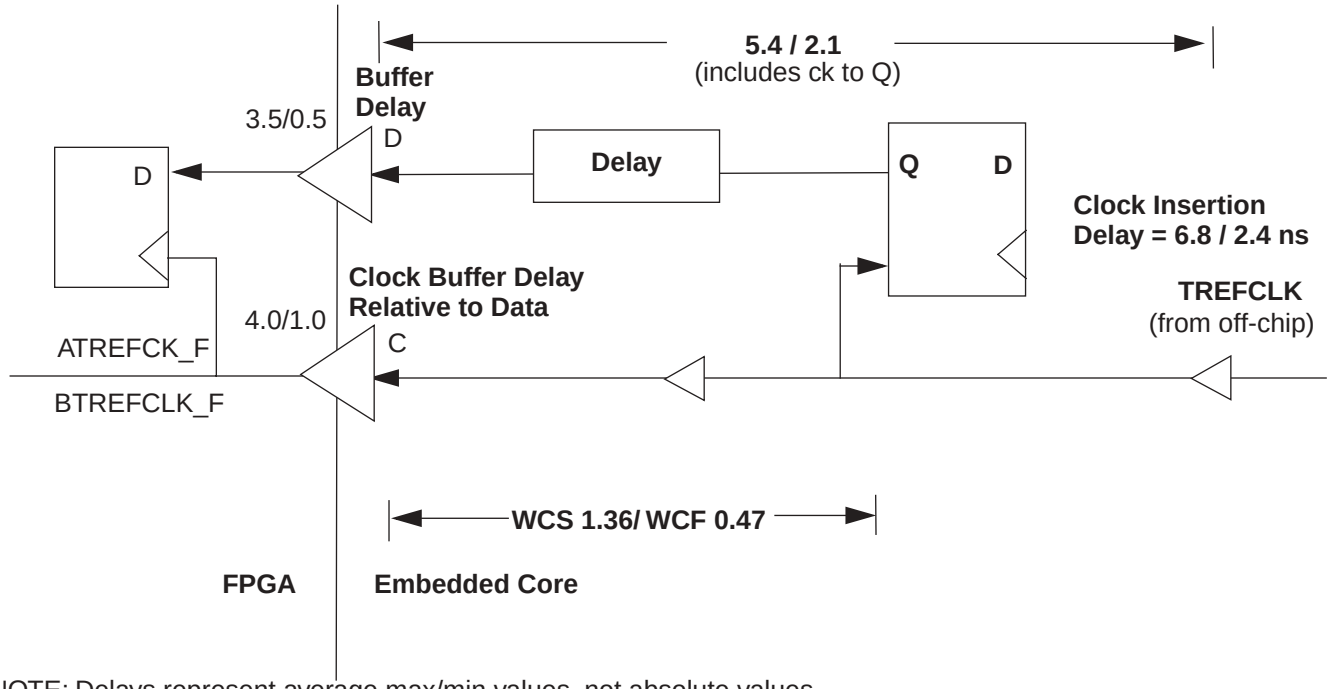
NOTE: Delays represent average max/min values, not absolute values
Actual delay values are used by *ispLEVER*

Embedded Core Sourced Status

The embedded core sources port and status information relative to a buffered version of the externally provided reference clock (TREFCLK). The FPGA receives both the status information and clock, and can use the provided clock to synchronize all status information for it's own use. Figure 19 shows the Worst Case Slow and Worst Case Fast propagation delays for the embedded core output status signals as well as the clock signal.

With the given timing delays the FPGA logic can either immediately register the provided status information or, if required, implement some combinatorial logic function before registering the information.

Figure 19. Data: ATX_PORTID[7:0], BTX_PORTID[7:0], ATX_STAT[1:0], BTX_STAT[1:0], ABURST_VAL[3:0], BBURST_VAL[3:0]
Clock: ATREFCLK, BTREFCLK



NOTE: Delays represent average max/min values, not absolute values
Actual delay values are used by *ispLEVER*

SPI4 Transmit Software Interface

The SPI4 transmit interface is configurable through a System Bus interface incorporated within the embedded core. The user can gain access to the System Bus either through the integrated MPI interface, or through FPGA resources using the System Bus Master/Slave interface. Please refer to the appropriate Lattice Semiconductor data sheets and application notes for more information regarding these interfaces.

The Transmit SPI4 interface logic incorporates many configurable control registers, as well as interrupt and status registers to monitor SPI4 performance. Table 46 provides a memory map and description of each register within the Transmit portion of the SPI4 embedded core.

Special Operating Modes

Quarter-Rate Mode

The ORSPI4 SPI4 TX interface is designed to operate at data rates much lower than 622 Mbps. Even though the OIF standard specifies a minimum data rate of 622 Mbits/s, the lower data rates are provided for users who wish to use the SPI4 link for applications supporting <10 Gbits/s aggregate bandwidth (STS-48, STS-3, Gb Ethernet, etc.). To enable this low speed mode, user should set the software register bit SPI4_QUARTER_RATE (Address 30915 in SPIA and 30A15 in SPIB) to '1'. This supports data rates in the range of 100 - 200 Mbps. Note that the normal operating modes 32-bit, 64-bit and 128-bit are independent of the quarter-rate mode. In quarter-rate mode, the transmit PLL is held in reset by setting the MRESET hardware pin to '1'. The PLL bypass mode is enabled by setting the BYPASS pin to '1' which causes the SPI[A,B]_TREFCLK_X8 clock from the FPGA to be used as the transmit reference clock. It must be two- times the desired Transmit SPI4 line clock rate. When not operating the Transmit SPI4 core in quarter-rate mode, this signal should be tied off. Ex: For a 100 MHz Transmit SPI4 line clock, SPIA_TREFCLK_X8 from the FPGA must be 200 MHz.

ORSPI4 SPI4 Receive Path Functional Description

This section describes the receive section of the SPI4 interface. Although there will be two SPI4 compliant interfaces within the FPSC, this section describes a single interface. The other interface is a duplicate and therefore needs no additional technical description.

ORSPI4 Receive Features

The Receive SPI4 interface supports the following features:

- 10 Gbits/s data throughput.
- Four dual-port memory banks supporting buffering for up to 32 ports. If support for more than 32 ports is needed, then the dual-port memories can be used for clock domain crossing purposes and data can be buffered in an external memory.
- Port Status Sequencer (PSS), including Receive Calendar (main and shadow), will support up to 256 ports, the maximum number of ports supported by the SPI4 standard.
- 32, 64 and 128-bit data width aggregation modes at the user (core/FPGA) interface.
- Programmable main and shadow calendar table. All calendar configuration parameters specified in the SPI4 standard such as CALENDAR_LEN, CALENDAR_M are supported.
- Optional dynamic alignment of receive data at the high-speed SPI4 interface. Dynamic alignment is required at rates > 700 Mbits/s (350 MHz).

The SPI4 receive logic enables users to read incoming port data using a variety of interfaces and associated clock domain options. It uses Dual Port RAMs (DPRAM) for temporary storage and clock domain crossing. Data is received in SPI4 format as LVDS signals at the receive interface. The data is written into DPRAM as received and read from the DPRAMs as requested by the FPGA logic. FIFO status is transmitted from the Receive Status interface according to a pre-configured polling sequence contained within the Receive Calendar. Data is formatted into the SPI4 Receive Status format and sent to the physical links as either LVDS or LVTTL signals.

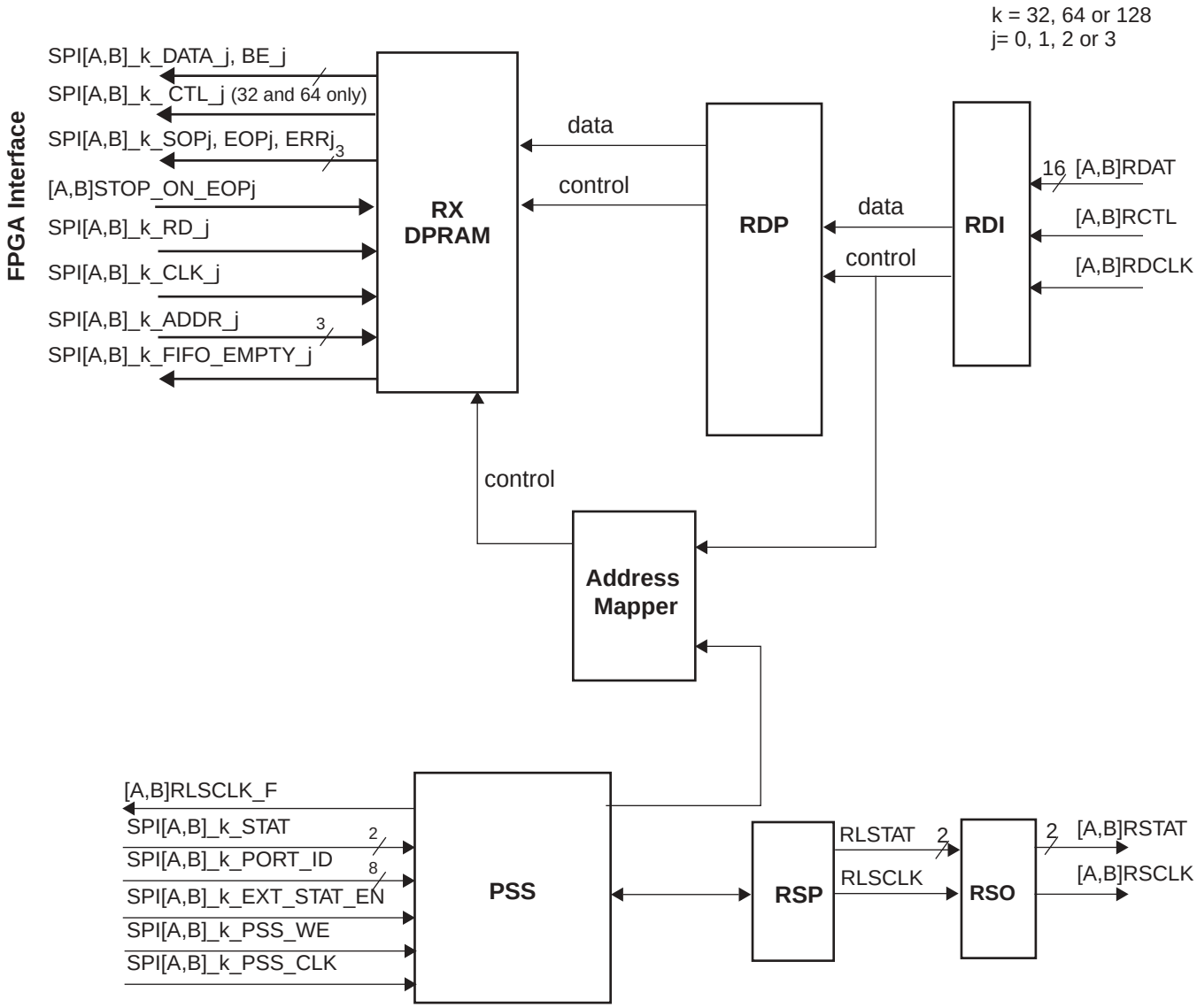
There are also several other features incorporated into the embedded core such as parallel loopback and far end loopback to assist in debugging and statistic gathering. These features, and the SPI4 data formats and initialization procedures are documented in separate sections since they involve both the transmit and receive paths.

The major blocks associated with the ORSPI4 receiver are:

- SPI4 Receive Logic - Data
  - SPI4 Receive Data Input (RDI) block
  - SPI4 Receive Data Protocol (RDP) logic
- Data Formatter
- Address Map
- DPRAM Banks
- Port Status Sequencer Logic (PSS)
- SPI4 Receive Logic - Status
  - SPI4 Receive Status Protocol (RSP) logic
  - SPI4 Receive Status Output (RSO) block

These blocks will be described in detail in the following sections. The ORSPI4 Receive functional block diagram is shown in Figure 20.

Figure 20. ORSPI4 Receive Functional Block Diagram



ORSPI4 Receive Functional Block Overview

The ORSPI4 high-speed receive logic receives high-speed data on 16 LVDS pairs (RDAT[15:0]), control on one LVDS pair (RCTL) and clock on one LVDS pair (RDCLK) at the SPI4 interface. This is an OIF-SPI-4 02.0 compliant interface that supports data rates in the range of 622-900 Mbps. Additionally lower-speed data in the range of 100-200 Mbps is also supported. The high-speed receive interface logic supports both static and dynamic alignment as specified by the OIF-SPI-4 02.0 specification. Dynamic alignment allows for +/- one bit period of skew. During dynamic alignment, training patterns are detected and deskew performed. Data is then deserialized into 128-bit data and 8-bit control bus. The SPI4 receive data protocol logic then extracts the control information such as PORT_ID, SOP, EOP and ERR into a separate sideband bus while still preserving wire-speed throughput. Data and control signals are then written into one of four dual-port RAMs (DPRAMs). Per-port buffering is supported for a maximum of 32 ports. The DPRAMs are built as multiple virtual FIFOs where each FIFO can be allocated for a port by the user through software. At the FPGA fabric, the user reads data for the desired ports from the FIFOs

based on the empty status signals presented from the FIFOs. The user can select a 32-bit, 64-bit or 128-bit wide FIFO interface depending on the FPGA design requirements.

There are two SPI4 cores in the ORSPI4 device. They are referenced in the document as SPIA and SPIB respectively.

ORSPI4 Receive Embedded Core/FPGA Interface Description

ORSPI4 Receive I/O Modes

The FPGA interface I/Os to the ORSPI4 logic block vary depending on the operating modes. There are three main user operating modes:

- 32-bit operating mode: Each of the four DPRAMs can be configured with a 32-bit user data plus controls across the FPGA interface. This mode is particularly useful if the user is aggregating multiple 32-bit interfaces into a single SPI4 interface.
- 64-bit operating mode: A DPRAM pair can be configured with 64-bit user data bus plus controls across the FPGA interface. Two such DPRAM pairs are available. More flexibility is provided by configuring one pair of DPRAM banks in 64-bit mode and the remaining two banks in 32-bit mode.
- 128-bit operating mode: All four DPRAM banks are aggregated into a single FIFO with a 128-bit user data interface.

Several other features are:

- Ability to pause for two clock cycles when an EOP is detected using the ASTOP_ON_EOP (SPIA) or BSTOP_ON_EOP (SPIB) control signal. Please refer to the timing diagrams for detailed information.

Table 12 lists I/Os for SPIA core only. They are identical in the SPIB core as well.

Table 12. SPIA Core Receive FPGA Interface in 32-Bit Mode

DPRAM	FPGA Interface I/Os	Direction From/To	Description
0	SPIA_RX32_ADDR_0[2:0]	FPGA → Core	FIFO read address.
	SPIA_RX32_DATA_0[31:0]	Core →FPGA	FIFO read data.
	SPIA_RX32_BE_0[3:0]	Core →FPGA	Byte enables Bit 3 - Byte enable for SPIA_RX32_DATA_0[31:24] Bit 2 - Byte enable for SPIA_RX32_DATA_0[23:16] Bit 1 - Byte enable for SPIA_RX32_DATA_0[15:8] Bit 0 - Byte enable for SPIA_RX32_DATA_0[7:0]
	SPIA_RX32_CTL_0	Core →FPGA	Port ID indicator. A '1' indicates that the SPIA_RX32_DATA_0 bus contains the port ID.
	SPIA_RX32_SOP_0	Core →FPGA	Start of Packet Indicator. A '1' indicates start of packet.
	SPIA_RX32_EOP_0	Core →FPGA	End of Packet Indicator. A '1' indicates end of packet.
	SPIA_RX32_ERR_0	Core →FPGA	Error. A '1' indicates an error in the current word.
	SPIA_RX32_FIFO_EMPTY_0	Core →FPGA	FIFO Empty flag. Depends on the RX_DPRAM_EMPTY_TYPE_SELA[0] software register bit in address 0x30920. When this bit is set to '0', (default) the empty flag indicates truly empty. When this bit is set to '1', the empty flag indicates that the FIFO is 1/4 full - 1.
	ASTOP_ON_EOP0	FPGA → Core	Timing control for handling end of packet.
	SPIA_RX32_CLK_0	FPGA → Core	FIFO read clock.
	SPIA_RX32_RD_0	FPGA → Core	FIFO read enable.
Note: For SPIB replace A with B			

Table 12. SPIA Core Receive FPGA Interface in 32-Bit Mode (Continued)

DPRAM	FPGA Interface I/Os	Direction From/To	Description
1	SPIA_RX32_ADDR_1[2:0]	FPGA → Core	FIFO read address.
	SPIA_RX32_DATA_1[31:0]	Core →FPGA	FIFO read data.
	SPIA_RX32_BE_1[3:0]	Core →FPGA	Byte enables Bit 3 - Byte enable for SPIA_RX32_DATA_0[31:24] Bit 2 - Byte enable for SPIA_RX32_DATA_0[23:16] Bit 1 - Byte enable for SPIA_RX32_DATA_0[15:8] Bit 0 - Byte enable for SPIA_RX32_DATA_0[7:0]
	SPIA_RX32_CTL_1	Core →FPGA	Port ID indicator. A '1' indicates that the SPIA_RX32_DATA_0 bus contains the port ID.
	SPIA_RX32_SOP_1	Core →FPGA	Start of Packet Indicator. A '1' indicates start of packet.
	SPIA_RX32_EOP_1	Core →FPGA	End of Packet Indicator. A '1' indicates end of packet.
	SPIA_RX32_ERR_1	Core →FPGA	Error. A '1' indicates an error in the current word.
	SPIA_RX32_FIFO_EMPTY_1	Core →FPGA	FIFO Empty flag. Depends on the RX_DPRAM_EMPTY_TYPE_SELA[1] software register bit in address 0x30920. When this bit is set to '0', (default) the empty flag indicates truly empty. When this bit is set to '1', the empty flag indicates that the FIFO is 1/4 full -1.
	ASTOP_ON_EOP1	FPGA → Core	Timing control for handling end of packet.
	SPIA_RX32_CLK_1	FPGA → Core	FIFO read clock.
	SPIA_RX32_RD_1	FPGA → Core	FIFO read enable.
2	SPIA_RX32_ADDR_2[2:0]	FPGA → Core	FIFO read address.
	SPIA_RX32_DATA_2[31:0]	Core →FPGA	FIFO read data.
	SPIA_RX32_BE_2[3:0]	Core →FPGA	Byte enables Bit 3 - Byte enable for SPIA_RX32_DATA_0[31:24] Bit 2 - Byte enable for SPIA_RX32_DATA_0[23:16] Bit 1 - Byte enable for SPIA_RX32_DATA_0[15:8] Bit 0 - Byte enable for SPIA_RX32_DATA_0[7:0]
	SPIA_RX32_CTL_2	Core →FPGA	Port ID indicator. A '1' indicates that the SPIA_RX32_DATA_0 bus contains the port ID.
	SPIA_RX32_SOP_2	Core →FPGA	Start of Packet Indicator. A '1' indicates start of packet.
	SPIA_RX32_EOP_2	Core →FPGA	End of Packet Indicator. A '1' indicates end of packet.
	SPIA_RX32_ERR_2	Core →FPGA	Error. A '1' indicates an error in the current word.
	SPIA_RX32_FIFO_EMPTY_2	Core →FPGA	FIFO Empty flag. Depends on the RX_DPRAM_EMPTY_TYPE_SELA[2] software register bit in address 0x30920. When this bit is set to '0', (default) the empty flag indicates truly empty. When this bit is set to '1', the empty flag indicates that the FIFO is 1/4 full -1.
	ASTOP_ON_EOP2	FPGA → Core	Timing control for handling end of packet.
	SPIA_RX32_CLK_2	FPGA → Core	FIFO read clock.
	SPIA_RX32_RD_2	FPGA → Core	FIFO read enable.
Note: For SPIB replace A with B			

Table 12. SPIA Core Receive FPGA Interface in 32-Bit Mode (Continued)

DPRAM	FPGA Interface I/Os	Direction From/To	Description
3	SPIA_RX32_ADDR_3[2:0]	FPGA → Core	FIFO read address.
	SPIA_RX32_DATA_3[31:0]	Core →FPGA	FIFO read data.
	SPIA_RX32_BE_3[3:0]	Core →FPGA	Byte enables Bit 3 - Byte enable for SPIA_RX32_DATA_0[31:24] Bit 2 - Byte enable for SPIA_RX32_DATA_0[23:16] Bit 1 - Byte enable for SPIA_RX32_DATA_0[15:8] Bit 0 - Byte enable for SPIA_RX32_DATA_0[7:0]
	SPIA_RX32_CTL_3	Core →FPGA	Port ID indicator. A '1' indicates that the SPIA_RX32_DATA_0 bus contains the port ID.
	SPIA_RX32_SOP_3	Core →FPGA	Start of Packet Indicator. A '1' indicates start of packet.
	SPIA_RX32_EOP_3	Core →FPGA	End of Packet Indicator. A '1' indicates end of packet.
	SPIA_RX32_ERR_3	Core →FPGA	Error. A '1' indicates an error in the current word.
	SPIA_RX32_FIFO_EMPTY_3	Core →FPGA	FIFO Empty flag. Depends on the RX_DPRAM_EMPTY_TYPE_SELA[3] software register bit in address 0x30920. When this bit is set to '0', (default) the empty flag indicates truly empty. When this bit is set to '1', the empty flag indicates that the FIFO is 1/4 full -1.
	ASTOP_ON_EOP3	FPGA → Core	Timing control for handling end of packet.
	SPIA_RX32_CLK_3	FPGA → Core	FIFO read clock.
	SPIA_RX32_RD_3	FPGA → Core	FIFO read enable.
Status I/Os	SPIA_RX32_PORT_ID[7:0]	FPGA → Core	Port ID used by the core as write address into the status RAM.
	SPIA_RX32_STAT[1:0]	FPGA → Core	Port Status corresponding to SPIA_RX32_PORT_ID
	SPIA_RX32_EXT_STAT_EN	FPGA → Core	When set to '1', the status from SPIA_RX32_STAT will be sent on the SPI4 status i/f.
	SPIA_RX32_PSS_WE	FPGA → Core	Write enable to the internal Status RAM.
	SPIA_RX32_PSS_CLK	FPGA → Core	Write clock to the internal Status RAM.
Misc. I/Os	SPI_DATM_A	FPGA → Core	Valid only during static data capture. Should be set to '0' in dynamic alignment mode. '0' - Bypass delay line circuit on the clock (ARDCLK) path. '1' - Enables delay line circuit on the ARDCLK path.
	SPI_DLYTAP_A[2:0]	FPGA → Core	These bits control the effective delay of the delay line circuit on the ARDCLK path. Valid values are from 0-6.
	ARLSCLK_F	Core →FPGA	Internal clock (ARDCLK/4) used to clock all receive status logic.
Note: For SPIB replace A with B			

Table 13. SPIA Core Receive FPGA Interface in 64-Bit Mode

DPRAM	FPGA Interface I/Os	Direction From/To	Description
0	SPIA_RX64_ADDR_0[2:0]	FPGA → Core	FIFO read address.
	SPIA_RX64_DATA_0[63:0]	Core →FPGA	FIFO read data.
	SPIA_RX64_BE_0[7:0]	Core →FPGA	Byte enables Bit 7 - Byte enable for SPIA_RX64_DATA_0[63:56] Bit 6 - Byte enable for SPIA_RX64_DATA_0[55:48] Bit 5 - Byte enable for SPIA_RX64_DATA_0[47:40] Bit 4 - Byte enable for SPIA_RX64_DATA_0[39:32] Bit 3 - Byte enable for SPIA_RX64_DATA_0[31:24] Bit 2 - Byte enable for SPIA_RX64_DATA_0[23:16] Bit 1 - Byte enable for SPIA_RX64_DATA_0[15:8] Bit 0 - Byte enable for SPIA_RX64_DATA_0[7:0]
	SPIA_RX64_CTL_0	Core →FPGA	Port ID indicator. A '1' indicates that the SPIA_RX64_DATA_0 bus contains the port ID.
	SPIA_RX64_SOP_0	Core →FPGA	Start of Packet Indicator. A '1' indicates start of packet.
	SPIA_RX64_EOP_0	Core →FPGA	End of Packet Indicator. A '1' indicates end of packet.
	SPIA_RX64_ERR_0	Core →FPGA	Error. A '1' indicates an error in the current word.
	SPIA_RX64_FIFO_EMPTY_0	Core →FPGA	FIFO Empty flag. Depends on the RX_DPRAM_EMPTY_TYPE_SELA[0] software register bit in address 0x30920. When this bit is set to '0', (default) the empty flag indicates truly empty. When this bit is set to '1', the empty flag indicates that the FIFO is 1/4 full -1.
	ASTOP_ON_EOP0	FPGA → Core	Timing control for handling end of packet.
	SPIA_RX64_CLK_0	FPGA → Core	FIFO read clock.
	SPIA_RX64_RD_0	FPGA → Core	FIFO read enable.
1	SPIA_RX64_ADDR_1[2:0]	FPGA → Core	FIFO read address.
	SPIA_RX64_DATA_1[63:0]	Core →FPGA	FIFO read data.
	SPIA_RX64_BE_1[7:0]	Core →FPGA	Byte enables Bit 7 - Byte enable for SPIA_RX64_DATA_1[63:56] Bit 6 - Byte enable for SPIA_RX64_DATA_1[55:48] Bit 5 - Byte enable for SPIA_RX64_DATA_1[47:40] Bit 4 - Byte enable for SPIA_RX64_DATA_1[39:32] Bit 3 - Byte enable for SPIA_RX64_DATA_1[31:24] Bit 2 - Byte enable for SPIA_RX64_DATA_1[23:16] Bit 1 - Byte enable for SPIA_RX64_DATA_1[15:8] Bit 0 - Byte enable for SPIA_RX64_DATA_1[7:0]
	SPIA_RX64_CTL_1	Core →FPGA	Port ID indicator. A '1' indicates that the SPIA_RX64_DATA_1 bus contains the port ID.
	SPIA_RX64_SOP_1	Core →FPGA	Start of Packet Indicator. A '1' indicates start of packet.
	SPIA_RX64_EOP_1	Core →FPGA	End of Packet Indicator. A '1' indicates end of packet.
	SPIA_RX64_ERR_1	Core →FPGA	Error. A '1' indicates an error in the current word.
	SPIA_RX64_FIFO_EMPTY_1	Core →FPGA	FIFO Empty flag. Depends on the RX_DPRAM_EMPTY_TYPE_SELA[0] software register bit in address 0x30920. When this bit is set to '0', (default) the empty flag indicates truly empty. When this bit is set to '1', the empty flag indicates that the FIFO is 1/4 full -1.
	ASTOP_ON_EOP1	FPGA → Core	Timing control for handling end of packet.
	SPIA_RX64_CLK_1	FPGA → Core	FIFO read clock.
	SPIA_RX64_RD_1	FPGA → Core	FIFO read enable.
Note: For SPIB replace A with B			

Table 13. SPIA Core Receive FPGA Interface in 64-Bit Mode (Continued)

DPRAM	FPGA Interface I/Os	Direction From/To	Description
Status I/Os	SPIA_RX64_PORT_ID[7:0]	FPGA → Core	Port ID used by the core as write address into the status RAM.
	SPIA_RX64_STAT[1:0]	FPGA → Core	Port Status corresponding to SPIA_RX64_PORT_ID
	SPIA_RX64_EXT_STAT_EN	FPGA → Core	When set to '1', the status from SPIA_RX64_STAT will be sent on the SPI4 status i/f.
	SPIA_RX64_PSS_WE	FPGA → Core	Write enable to the internal Status RAM.
	SPIA_RX64_PSS_CLK	FPGA → Core	Write clock to the internal Status RAM.
Misc. I/Os	SPI_DATM_A	FPGA → Core	Valid only during static data capture. Should be set to '0' in dynamic alignment mode. '0' - Bypass delay line circuit on the clock (ARDCLK) path. '1' - Enables delay line circuit on the ARDCLK path.
	SPI_DLYTAP_A[2:0]	FPGA → Core	These bits control the effective delay of the delay line circuit on the ARDCLK path. Valid values are from 0-6.
	ARLSCLK_F	Core →FPGA	Internal clock (ARDCLK/4) used to clock all receive status logic.
Note: For SPIB replace A with B			

Table 14. SPIA Core Receive FPGA Interface in 128-Bit Mode

DPRAM	FPGA Interface I/Os	Direction From/To	Description
0	SPIA_RX128_ADDR[2:0]	FPGA → Core	FIFO read address.
	SPIA_RX128_DATA[127:0]	Core →FPGA	FIFO read data.
	SPIA_RX128_BE[15:0]	Core →FPGA	Byte enables Bit 15- Byte enable for SPIA_RX128_DATA[127:120] Bit 14- Byte enable for SPIA_RX128_DATA[119:112] Bit 13- Byte enable for SPIA_RX128_DATA[111:104] Bit 12- Byte enable for SPIA_RX128_DATA[103:96] Bit 11- Byte enable for SPIA_RX128_DATA[95:88] Bit 10- Byte enable for SPIA_RX128_DATA[87:80] Bit 9- Byte enable for SPIA_RX128_DATA[79:72] Bit 8- Byte enable for SPIA_RX128_DATA[71:64] Bit 7 - Byte enable for SPIA_RX128_DATA[63:56] Bit 6 - Byte enable for SPIA_RX128_DATA[55:48] Bit 5 - Byte enable for SPIA_RX128_DATA[47:40] Bit 4 - Byte enable for SPIA_RX128_DATA[39:32] Bit 3 - Byte enable for SPIA_RX128_DATA[31:24] Bit 2 - Byte enable for SPIA_RX128_DATA[23:16] Bit 1 - Byte enable for SPIA_RX128_DATA[15:8] Bit 0 - Byte enable for SPIA_RX128_DATA[7:0]
	SPIA_RX128_PORTID[7:0]	Core →FPGA	Port ID for the current data.
	SPIA_RX128_SOP	Core →FPGA	Start of Packet Indicator. A ‘1’ indicates start of packet.
	SPIA_RX128_EOP	Core →FPGA	End of Packet Indicator. A ‘1’ indicates end of packet.
	SPIA_RX128_ERR	Core →FPGA	Error. A ‘1’ indicates an error in the current word.
	SPIA_RX128_FIFO_EMPTY	Core →FPGA	FIFO Empty flag. Depends on the RX_DPRAM_EMPTY_TYPE_SELA[0] software register bit in address 0x30920. When this bit is set to ‘0’, (default) the empty flag indicates truly empty. When this bit is set to ‘1’, the empty flag indicates that the FIFO is 1/4 full -1.
	ASTOP_ON_EOP	FPGA → Core	Timing control for handling end of packet.
	SPIA_RX128_CLK	FPGA → Core	FIFO read clock.
	SPIA_RX128_RD	FPGA → Core	FIFO read enable.
	SPIA_RX64_RD_1	FPGA → Core	FIFO read enable.
	SPIA_RX128_PORT_ID[7:0]	FPGA → Core	Port ID used by the core as write address into the status RAM.
	SPIA_RX128_STAT[1:0]	FPGA → Core	Port Status corresponding to SPIA_RX64_PORT_ID
	SPIA_RX128_EXT_STAT_EN	FPGA → Core	When set to ‘1’, the status from SPIA_RX64_STAT will be sent on the SPI4 status i/f.
	SPIA_RX128_PSS_WE	FPGA → Core	Write enable to the internal Status RAM.
	SPIA_RX128_PSS_CLK	FPGA → Core	Write clock to the internal Status RAM.
Misc. I/Os	SPI_DATM_A	FPGA → Core	Valid only during static data capture. Should be set to ‘0’ in dynamic alignment mode. ‘0’ - Bypass delay line circuit on the clock (ARDCLK) path. ‘1’ - Enables delay line circuit on the ARDCLK path.
	SPI_DLYTAP_A[2:0]	FPGA → Core	These bits control the effective delay of the delay line circuit on the ARDCLK path. Valid values are from 0-6.
	ARLSCLK_F	Core →FPGA	Internal clock (ARDCLK/4) used to clock all receive status logic.
Note: For SPIB replace A with B			

SPI4 Receive Logic Blocks - Detailed Description

RDI Block

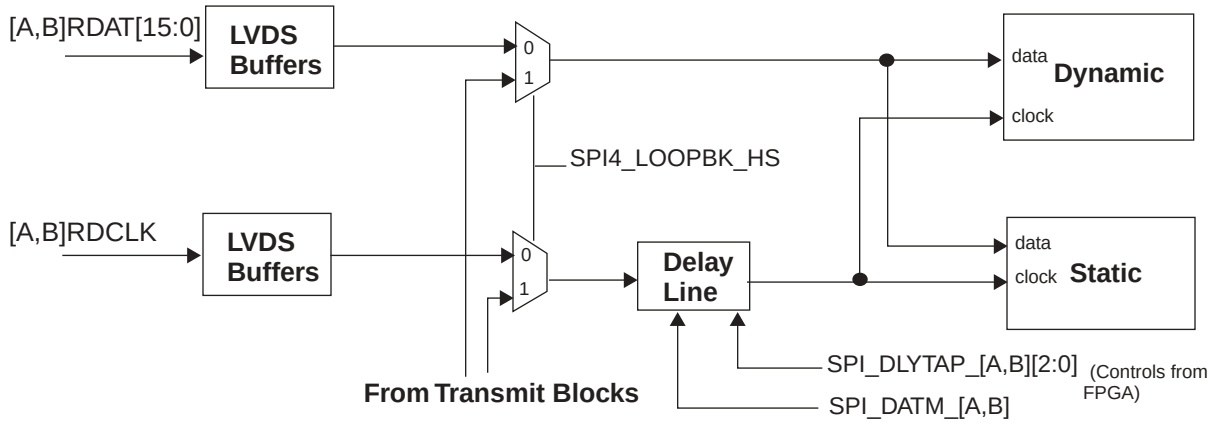
The SPI4 high-speed Receive Data Input (RDI) block contains the high-speed receive logic for the SPI4 block. Incoming LVDS signals, in SPI4 format, include the 16-bit data bus (RDAT[15:0]), a control bit (RCTL) and a source synchronous DDR clock (RDCLK). The 16-bit data bus and control signal are DDR with respect to RDCLK. The incoming data is deserialized to a 128-bit format and the control information is converted to an 8-bit format. Deserialized data, control signals and a low-speed clock (RDCLK), derived from the high-speed RDCLK, are forwarded to the protocol (RDP) block.

The RDI block alignment logic detects training patterns and perform dynamic alignment of the incoming data. For low speed incoming data at rates up to 700 Mbits/s, static alignment can be chosen through a programmable control bit. The low speed mode is described in a later section. At speeds above 700 Mb/s (350 MHz), however, it becomes necessary to use dynamic alignment. Skews of up to ± one clock period can be compensated for by the dynamic alignment logic, which chooses the best phase of the receive clock, out of 16 possibilities, to center each data bit for the best possible setup/hold margin. The receive dynamic alignment logic periodically samples the data to verify that data can be transferred reliably as temperature and voltage levels in the system vary. It then makes appropriate adjustments as needed.

The output from The RDI block is the low speed 128-bit data bus and 8-bit control bus along with a low speed clock (RDCLK) which is 1/4th the SPI4 receive clock RDCLK.

The ORSPI4 receive high-speed interface also supports static data capture as shown in Figure 21. As shown in the figure, the key difference that exists between the clock and data paths is the delay line in the clock path. The high-speed loopback works in both static and dynamic alignment modes. The delay line circuit is a series of buffers with programmable stages that are controlled through two signals from the FPGA - SPI_DATM_A (SPIA) or SPI_DATM_B (SPIB) and SPI_DLYTAP_A[2:0] (SPIA) or SPI_DLYTAP_B[2:0] (SPIB). The object of having programmable delay stages is to delay RDCLK appropriately so that both its rising and falling edges hit the data eye with sufficient setup and hold margin for data capture. Various delay settings allow the data to be centered vs. the clock for the best setup and hold margin. For details on programming the ORSPI4 receiver in static and dynamic capture modes, refer to the section “Static Capture Operating Mode”.

Figure 21. Static Data Capture



RDP Block

The SPI4 Receive Data Protocol (RDP) block receives data from RDI block and is responsible for decoding the in-band control information while preserving wire-speed throughput. It passes both data and control information, such as link address, SOP, EOP and error, to the Receive DPRAMs.

The RDP block also parses the control words embedded within the incoming data. Using this control information, it performs the following functions:

- Checks DIP-4 parity
- Monitors for continuous alignment (if more than a programmable number of DIP-4 parity errors exist, there may be an alignment problem). In the event of DIP-4 parity errors, instructs the status block to send status framing pattern “11” on the receive SPI4 status bus.
- Removes idle/training words.
- Extracts link address and SOP, EOP and valid packet (no error) signals.

Receive DPRAMs

There are four DPRAMs referred to as banks 0, 1, 2 and 3. Each bank has a 32-bit data interface to the FPGA. Each DPRAM has its own individual read enable and read clock allowing it to be read by the FPGA application independently.

Every DPRAM bank is configured by software to operate in one of 32-bit, 64-bit or 128-bit mode. Every DPRAM bank is also configured to contain 1, 2, 4 or 8 virtual FIFOs. The user determines the number of FIFOs depending on the number of ports and buffer requirements for the ports as required by a given application. The programming of a DPRAM bank in 32-bit, 64-bit or 128-bit mode and programming of the number of virtual FIFOs within a DPRAM bank is done through software as shown in Table 18. Note that in 64-bit mode, DPRAMs 0 and 1 have to be configured identically. The combined DPRAM pair is referred to as DPRAM “0”. Similarly DPRAM pairs 2 and 3 have to be configured identically. This combined DPRAM pair is referred to as DPRAM “2”. In 128-bit mode, all DPRAMs have to be configured identically. The combined DPRAM banks are collectively referred to as DPRAM “0”. The aggregation modes can be used in five possible combinations as shown in Table 16. The size of the embedded data and control FIFOs for each mode is shown in Table 17. The user accesses a virtual FIFO using the 3-bit FIFO read address (refer to Table 12, Table 13, and Table 14) from the FPGA. Table 15 shows the indexed partition based upon the configured DPRAM partitioning and aggregation mode.

In 32-bit operating mode, the user always reads data as four 32-bit word bursts (plus one additional clock cycle to read the port ID) unless an EOP is received. If an EOP occurs and the ASTOP_ON_EOPj (j=0,1,2,3) or BSTOP_ON_EOPj is ‘0’, the core will cease transmission of the remaining words in the burst and start transferring the next set of data. For example, if the EOP happens in the second word of a 4-word burst, the core will not provide the remaining two words. Instead, it will start bursting the next set of data. The exception is when the ASTOP_ON_EOPj (j=0,1,2,3) or BSTOP_ON_EOPj is asserted. When this signal is ‘1’, the core will pause for 2 clock cycles after an EOP. The byte enables (SPIA_RX32_BE_j) are set to ‘0’ indicating that the read data is not valid.

Table 15. FIFO Address Based on Programmed Virtual FIFOs

FIFO Address (from FPGA)	Description
XXX	Ignored when a DPRAM is configured to support only one virtual FIFO
000	Selects FIFO 0 when DPRAM is configured to support 2, 4 or 8 virtual FIFOs
001	Selects FIFO 1 when DPRAM is configured to support 2, 4 or 8 virtual FIFOs
010	Selects FIFO 2 when DPRAM is configured to support 4 or 8 virtual FIFOs
011	Selects FIFO 3 when DPRAM is configured to support 4 or 8 virtual FIFOs
100	Selects FIFO 4 when DPRAM is configured to support 8 virtual FIFOs
101	Selects FIFO 5 when DPRAM is configured to support 8 virtual FIFOs
110	Selects FIFO 6 when DPRAM is configured to support 8 virtual FIFOs
111	Selects FIFO 7 when DPRAM is configured to support 8 virtual FIFOs

Table 16. Possible Combinations of Aggregation Modes

Mode	Valid
Banks 0 to 3 in 32-bit aggregation mode	Yes
Banks 0, 1 in 64-bit aggregation mode, Banks 2,3 in 64-bit mode	Yes
Banks 0,1 in 64-bit mode, Banks 2 & 3 in 32-bit mode	Yes
Banks 0,1 in 32-bit mode, Banks 2 & 3 in 64-bit mode	Yes
All banks in 128-bit	Yes
Bank 0 in 32-bit mode, Bank 1 in 64-bit mode or vice-versa	No
Bank 2 in 32-bit mode, Bank 3 in 64-bit mode or vice-versa	No

Table 17. Memory Size for Each Aggregation Mode and Partitioning

Operating Mode	Number of Configured Virtual FIFO Partitions	DATA FIFO Depth (in Bytes)	Control FIFO Depth (in Bytes)
32-bit	1	2K	512
	2	1K	256
	4	512	128
	8	256	64
64-bit	1	4 K	1K
	2	2 K	512
	4	1 K	256
	8	512	128
128-bit	1	8 K	2K
	2	4 K	1K
	4	2 K	512
	8	1 K	256

Table 18. Register Bit Settings for Aggregation Mode and Partition Size

DPRAM Aggregation Mode Address 30907 (SPIA), 30A07 (SPIB)	Virtual FIFO Partition Size Address 30906 (SPIA), 30A06 (SPIB)	Configuration
DPRAM 0 (Register Bits 0:1) = 00	Register Bits 0:1 = 00	32-bit data width, partition size 1
	01	32-bit data width, partition size 2
	10	32-bit data width, partition size 4
	11	32-bit data width, partition size 8
01	00	64-bit data width, partition size 1
	01	64-bit data width, partition size 2
	10	64-bit data width, partition size 4
	11	64-bit data width, partition size 8
10	00	128-bit data width, partition size 1
	01	128-bit data width, partition size 2
	10	128-bit data width, partition size 4
	11	128-bit data width, partition size 8
DPRAM 1 (Register Bits 2:3) = 00	Register Bits 2:3 = 00	32-bit data width, partition size 1
	01	32-bit data width, partition size 2
	10	32-bit data width, partition size 4
	11	32-bit data width, partition size 8
01	00	64-bit data width, partition size 1
	01	64-bit data width, partition size 2
	10	64-bit data width, partition size 4
	11	64-bit data width, partition size 8
10	00	128-bit data width, partition size 1
	01	128-bit data width, partition size 2
	10	128-bit data width, partition size 4
	11	128-bit data width, partition size 8
DPRAM 2 (Register Bits 4:5) = 00	Register Bits 4:5 = 00	32-bit data width, partition size 1
	01	32-bit data width, partition size 2
	10	32-bit data width, partition size 4
	11	32-bit data width, partition size 8
01	00	64-bit data width, partition size 1
	01	64-bit data width, partition size 2
	10	64-bit data width, partition size 4
	11	64-bit data width, partition size 8

DPRAM Aggregation Mode Address 30907 (SPIA), 30A07 (SPIB)	Virtual FIFO Partition Size Address 30906 (SPIA), 30A06 (SPIB)	Configuration
10	00	128-bit data width, partition size 1
	01	128-bit data width, partition size 2
	10	128-bit data width, partition size 4
	11	128-bit data width, partition size 8
DPRAM 3 (Register Bits 6:7) = 00	Register Bits 6:7 = 00	32-bit data width, partition size 1
	01	32-bit data width, partition size 2
	10	32-bit data width, partition size 4
	11	32-bit data width, partition size 8
01	00	64-bit data width, partition size 1
	01	64-bit data width, partition size 2
	10	64-bit data width, partition size 4
	11	64-bit data width, partition size 8
10	00	128-bit data width, partition size 1
	01	128-bit data width, partition size 2
	10	128-bit data width, partition size 4
	11	128-bit data width, partition size 8

Receive Address Mapper

This control block decodes the port address received from the SPI4 RX interface to determine the physical memory address (to access the DPRAM memories) for that port address. Each port address must be mapped to one of the virtual FIFOs in the DPRAMs. Each virtual FIFO can be used to buffer data for a single port. Since there is a total of four DPRAMs, eight virtual FIFOs in each DPRAM provides 32 per-port buffers. The number of per-port buffers varies according to the operating mode. For example, in 32-bit mode, a total of 32 per-port buffers is possible. In 64-bit mode, a total of 16 per-port buffers is provided. In 128-bit mode, a total of eight per-port buffers is provided.

To decode the physical memory information, this block accesses a user-configured Port Descriptor Memory (PDM) which is simply a look-up-table. The PDM is an embedded memory within the ORSPI4 core. There is one each for SPIA and SPIB cores. Since 256 ports can be supported by the device, the PDM has 256 locations. The user indexes this memory using the 8-bit SPI4 Port ID as the software write address into the embedded memory space 31000 - 310FF. For example, a software write into address 31000 configures the PDM for port ID 0. The software procedure to configure the PDM is described below:

- User has to select the PDM by writing a '1' into bit 3 of address 30917 (SPIA) or 30A17 (SPIB).
- Subsequent writes to addresses 31000-310FF correspond to entries 0x00-0xFF in the PDM. These may be written in any order.
- Upon completion of the PDM configuration, the PDM select bit (bit 3 of address 30917 or 30A17) must be set to '0'.
- The contents of the PDM may also be read back. This is done by first selecting the PDM memory by writing a '1' to bit 3 of address 30917 or 30A17. Reads are then done from addresses 31000-310FF. Upon completion of the reads, the PDM select bit is set to '0'.

To allocate a buffer space to a port, the user has to choose one of the four DPRAM banks (BANK_ID field in the PDM) and a virtual FIFO (PARTITION_ID) within the chosen DPRAM bank. Each location in the PDM contains the fields described below for each port address.

Partition Address (PA): This is the least significant 3-bit field [2:0] within a PDM location. This field provides the virtual FIFO partition address for each port. Each partition is of equal size. There can be a total of 8 virtual FIFO partitions within each DPRAM bank.

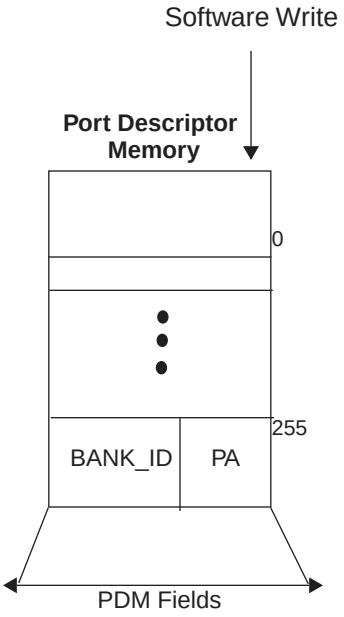
BANK_ID: This occupies bits 4:3 within a PDM location. This 2-bit field is used to select one of the four DPRAM banks to store the received data. The user initially assigns this information by programming the BANK_ID. Legal values for the BANK_ID depend on the operating mode. In 32-bit mode, the legal values are 0,1,2 and 3. In 64-bit mode, the legal values are 0 and 2. In 64-bit mode, the legal value is only 0.

While programming the PDM allocates a buffer for every port (or multiple ports), the buffer sizes themselves are configured by the user through software register settings for

- DPRAM aggregation mode (Address 30907 for SPIA and 30A07 for SPIB. Refer to Table 18)
- Number of virtual FIFOs in a DPRAM (Address 30906 for SPIA and 30A06 for SPIB. Refer to Table 17).

Figure 22 shows a block diagram of the Port Description Memory.

Figure 22. Receive Address Mapper Block Diagram



Virtual Receive Memory FIFOs (Partitions) and Port Mapping

As was the case for the transmit logic, there are up to 32 virtual receive FIFO partitioning, 256 port numbers and the receive calendar can contain up to 1023 entries. The user must set up the mapping between the FIFO partitions and the port number for each active port. The user must also set up the calendar sequence for polling the active ports.

The simplest mapping of the read DPRAM partitions, which support a maximum of 32 ports, is to have a 1:1 mapping between the FIFO partition and the port address. The 1023 depth of the calendar allows for uneven bandwidth allocation across ports where high bandwidth ports can have multiple entries within the calendar table.

The read scheduling logic, however, is designed to support up to 256 ports. Each port (up to 32 ports) can have its unique buffer. A single port or multiple ports can be assigned to a buffer by configuring the PDM. The user must, using FPGA logic, assure that data is read from the virtual FIFO using the same port sequencing that is expected

by the calendar. This ensures that the bandwidth allocated for a given port is maintained and the port's FIFO is emptied at the same rate that it is being filled up at the far-end transmitter. For example, if the calendar assumes that the buffer for port 2 is filled more frequently than the buffer for port 4, then the FPGA logic must read the buffer (virtual FIFO) for port 2 more frequently than port 4 to ensure that bandwidth requested for port 2 is maintained across the SPI4 link. The user also has a choice to not use the per-port buffers and treat the DPRAMs as pure clock-domain crossing FIFOs. This is done by simply configuring each DPRAM in 128-bit mode and setting up the number of virtual FIFO partitions to "1".

If the user application supports more than 32 ports, multiple ports need to share a buffer or external memory can be used. The ORSPI4 core provides a QDR-II SRAM Memory Controller for additional buffering. The QDR-II SRAM memory was chosen owing to the wire-speed throughput that can be realized through dedicated write and read ports. In this case, the user must implement control logic to sequence external memory reads from the selected virtual FIFOs and the FIFOs are used only for crossing between clock domains. In many cases a second QDR-II memory interface is needed, thus a soft IP version of the core is available.

Suppose there are 6 ports, and each port is mapped to a single buffer or virtual FIFO:

- The first step is to decide the aggregation mode and FIFO size depending on the bandwidth needed by the port(s). This is shown in Table 19.
- The second step is to configure the DPRAMs through software register settings shown in Table 20. Note that DPRAM banks 2 and 3 are configured identically since they are configured to be a 64-bit interface.
- The last step is to configure the PDM using the procedure described in the Address Mapper section. The contents of the PDM are shown in Table 21.

Table 19. DPRAM Configuration Example

DPRAM bank	Aggregation Mode	Number of Virtual FIFOs	Virtual FIFO Size (bytes)
0	32-bit	2	1K
1	32-bit	2	1K
2,3	64-bit	2	2K

Table 20. DPRAM Software Register Settings Example

DPRAM Bank	Aggregation Mode	Number of Virtual FIFOs
0	RX_DPRAM_0_AGGR_MODE = 00	RX_DPRAM_0_NUMFIFO = 01
1	RX_DPRAM_1_AGGR_MODE = 00	RX_DPRAM_0_NUMFIFO = 01
2	RX_DPRAM_2_AGGR_MODE = 01	RX_DPRAM_0_NUMFIFO = 01
3	RX_DPRAM_3_AGGR_MODE = 01	RX_DPRAM_0_NUMFIFO = 01

Table 21. PDM Contents Example

Port ID	Bank ID	Partition Address
0x00	00	000
0x01		001
0x02	01	000
0x03		001
0x04	10	000
0x05		001

Port Status Sequencer (PSS) Logic

In addition to formatting received data and sending it to the FPGA logic, the receive block also sends status information to the SPI4 status interface. The Port Status Sequencer (PSS) block is responsible for providing port status to the SPI4 Receive Status Protocol block (RSP) according to a pre-configured calendar sequence. Status is derived from the fill-levels of the DPRAM FIFOs and/or from the FPGA status interface.

When status is provided by FPGA logic, the FPGA application writes status information to a 256 word PSS memory in a random fashion. The PSS memory is addressed by an 8-bit port ID (SPIA_RX[32, 64 or 128]_PORT_ID[7:0]). The data written to the memory is the SPIA_RX[32, 64 or 128]_STAT and an external status enable (SPIA_RX[32, 64 or 128]_EXT_STAT_EN) for the selected port. A write is initiated by the FPGA by asserting SPIA_RX[32, 64 or 128]_PSS_WE high and providing a clock on SPIA_RX[32, 64 or 128]_PSS_CLK.

The PSS block contains a main and shadow calendar table. Only one of these calendars is in use at a given time. Once a calendar is provisioned with a certain sequence, it is not desirable to change the sequence in that calendar during active device operation. To enable hitless switching of calendars, a shadow calendar is provided. The RX_CAL_SEL control bit (Address 30916 for SPIA and 30A16 for SPIB) in the memory map controls the choice. The choice to select a shadow calendar enables hitless bandwidth reprovisioning on the SPI4 link. While one calendar is being used (e.g. main), the other calendar (shadow) can be configured independently with the desired sequence. For more details, refer to the section on calendar programming.

The Port Status Sequencer (PSS) block polls for port status. It uses the calendar address to decide which port should be polled for status. The status received from the polled port is then formatted into one of the SPI4 specified status encodings (STARVING, HUNGRY, SATISFIED). For every port, the status is polled either from the internal DPRAM FIFO flags or the user-provided status bits depending on the external status enable bit. When the external status enable bit for a given port is set to '1', it indicates that the status encoding for that port on the SPI4 bus is decided by the external user. When set to '0', the SPI4 status encoding is dictated by the status flags from the internal DPRAM banks.

The status bits sent to RSP from the PSS block is determined according to the truth table shown in Table 22. The first four columns are the bits of information looked up for the current port ID, and the last column is the status that is sent for that selected port to the RSP.

Table 22. Port Status Encoding

Virtual FIFO $fill \geq 3/4$	Virtual FIFO $fill \geq 1/2$	SPI[A,B]_k_EXT_STAT_EN	SPI[A,B]_k_STAT[1:0]	SPI4 Status Encoding
1	x	x	x	SATISFIED
0	x	1	"00"	STARVING
0	x	1	"01"	HUNGRY
0	x	1	"10"	SATISFIED
0	0	0	x	STARVING
0	1	0	x	HUNGRY
0	x	1	"11"	Disabled link

k = RX32 or RX64 or RX128

SPI4 Status Logic Blocks (RSP, RSO)

The SPI4 Receive Status Protocol (RSP) is responsible for FIFO status encoding, calendar management, status pattern encoding (sync bits "11"), DIP-2 calculation and optional calendar selection word encoding (as proposed in Appendix G of SPI4-02.0 specification).

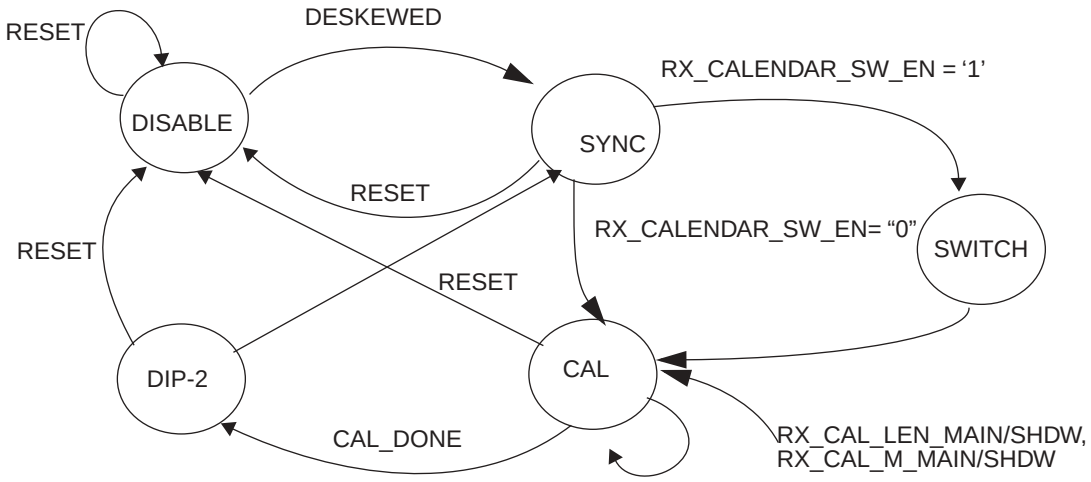
The RSP block performs the following functions:

- Status frame creation

- Decrements CALENDAR_M and CALENDAR_LEN counters
- DIP-2 calculation

The core of the RSP block is the state machine shown in Figure 23. The state machine is in the DISABLE state upon reset. Setting the register bit RX_STATUS_DIS to '1' or misalignment on the SPI4 RX data interface will cause the state machine to reset to DISABLE state and start sending "11" on the SPI4 status bus. When the received data is aligned and none of the other conditions apply, the state machine transfers to the SYNC state. The SYNC state exists for only one clock cycle. In this state the calendar management counters are initialized with the configured values from CALENDAR_LEN_RX and CALENDAR_M_RX. The next state is the CAL state if CALENDAR_SW_EN bit is set to '0' or the SWITCH state if CALENDAR_SW_EN bit is set to '1'. In the CAL state, the counter for CALENDAR_LEN_RX is decremented on each cycle and the counter for CALENDAR_M_RX is decremented at the end of each calendar sequence. If the CALENDAR_M_RX counter has not reached zero, the CALENDAR_LEN_RX counter is reset. On each clock cycle, the status is sent out on the RLSTAT bus. The DIP-2 code is calculated in each clock cycle using the method (diagonal XOR) described in the OIF-SPI4-02.0 standard. Once both calendar management counters have expired, the calendar management is complete and moves to the DIP-2 state. The current DIP-2 value coming out of the CAL state is XOR'ed with "11", the final 2-bit DIP-2 word is sent out on the RLSTAT bus and the SYNC state is entered. If the RX_CAL_SW_EN bit is enabled (set to '1') then the RSP will insert an extra word into the FIFO status frame

Figure 23. RSP State Machine



The SPI4 Receive Status Output (RSO) block contains the low speed LVTTL buffers and LVDS output buffers necessary for the output stage of receive status logic. The option to choose between LVTTL or LVDS outputs is selected by the software register bit SPI4_STATUS_IO_SEL.

Calendar Programming

On a SPI4 link, FIFO status information is sent periodically over the status link (RSTAT bus on the SPI4 interface) from the device that receives data. The calendar is a means by which the SPI4 link conveys information to a data source about the availability of buffer space in the FIFOs that receive data from that data source. A calendar is a sequence of status messages that

- Provides information on the buffer space for a port or traffic flow
- Allows user to allocate bandwidth for a port or flow depending on the overall traffic characteristics.

The SPI4 status channel operates at 1/8th the SPI4 data rate which is reasonable because data is always sent in 16-byte bursts. Thus the fastest status update can be expected once in every 8 data clock cycles. For a given port, the frequency with which its status is reported to the far-end transmitter source depends on the allocated bandwidth for the port. It is imperative that the transmitter and receiver devices are programmed with identical calendar sequences. The following examples illustrate this:

Suppose a channelized STS-48 has three STS-12 ports and four STS-3 ports (channels), each STS-12 port should have four times the bandwidth of an STS-3 port and 12 times the bandwidth of an STS-1 port within a single calendar cycle. The first step is to map each port to an 8-bit port ID (address) as shown in Table 23

Table 23. Port ID Mapping

Port	Bandwidth	8-Bit Port ID (Port Address)
A1	STS-12	0x00
A2	STS-12	0x01
A3	STS-12	0x02
B1	STS-3	0x03
B2	STS-3	0x04
B3	STS-3	0x05
B4	STS-3	0x06

The calendar sequence is as follows:

A1, A2, A3, B1, A1, A2, A3, B2, A1, A2, A3, B3, A1, A2, A3, B4

The number of calendar entries is thus 16. This calendar sequence is programmed in the receive calendar memory as follows:

- This example uses the main calendar. The calendar memory is selected by setting the RX_CAL_MEM_SEL bit (address 30917 in SPIA and address 30A17 in SPIB) to '1'.
- Writes are done to addresses 0x31000 - 0x3100F. This will correspond to entries 0x00 - 0x0F in the calendar memory. This is shown in Table 24. Note that the main calendar memory has 1023 locations and can be programmed through addresses 0x31000 - 0x313FE. The shadow calendar memory also has 1023 locations and can be programmed through addresses 0x31400 - 0x317FE.
- After programming the calendar memory, the RX_CAL_MEM_SET bit is set to '0'.
- The calendar length register RX_CAL_LEN_MAIN is set to 16 (address 30901 and 30902 in SPIA, 30A01 and 30A02 in SPIB).
- RX_CAL_M_MAIN (address 30900 in SPIA and 30A00 in SPIB) is set to the number of times the calendar sequence needs to be repeated between framing patterns.
- The shadow calendar can be programmed in the same identical fashion as the main calendar using the steps described above.

To enable hitless switching between main and shadow calendar memories, the RX_CAL_SW_EN bit (address 30916 in SPIA and 30A16 in SPIB) should be set to '1'. This will cause the receive status logic to insert the calendar select word after the framing pattern. The RX_CAL_SEL bit then selects main (bit set to '0') or shadow (bit set to '1') calendar to be sent on the outgoing status frame. It is important to enable this feature in the transmit device on the other end of the SPI4 link by setting TX_CAL_SW_EN bit (address 30916 in SPIA and 30A16 in SPIB) to '1'. This will cause the far-end transmit status logic to detect the calendar select word after the framing pattern.

Table 24. Receive Calendar Memory Contents

RX Calendar Memory Address	RX Calendar Memory Contents (Port ID)	Port Number
0x00	0x00	A1
0x01	0x01	A2
0x02	0x02	A3
0x03	0x03	B1
0x04	0x00	A1
0x05	0x01	A2
0x06	0x02	A3
0x07	0x04	B2
0x08	0x00	A1
0x09	0x01	A2
0x0a	0x02	A3
0x0b	0x05	B3
0x0c	0x00	A1
0x0d	0x01	A2
0x0e	0x02	A3
0x0f	0x06	B4

SPI4 Receive Software Interface

The SPI4 receive interface is configurable through a System Bus interface incorporated within the embedded core. The user can gain access to the System Bus either through the integrated MPI interface, or through FPGA resources using the System Bus Master/Slave interface. Please refer to the appropriate Lattice Semiconductor data sheets and application notes for more information regarding these interfaces.

The receive SPI4 interface logic incorporates many configurable control registers, as well as interrupt and status registers to monitor SPI4 performance. Table 46 provides a memory map and description of each register within the Transmit portion of the SPI4 embedded core.

Timing Diagrams

As described earlier, there are three main modes of operation - 32-bit mode, 64-bit mode and 128-bit mode. The timing diagrams described below provide a clear picture of each mode of operation.

For clarity, all timing diagrams are shown for SPIA core. They are identical for the SPIB core. The timing diagram for a 32-bit read access is shown in Figure 24. A 3-bit read address SPIA_RX32_ADDR_j where j=0,1,2,3 (which is actually the virtual FIFO partition address) is sent from the FPGA during clock T1. This FIFO address should match the virtual FIFO address that the user configured for a given port in the port descriptor memory (PDM). In other words, the virtual FIFO addressed by this 3-bit address contains the data for the port that was mapped to this buffer space in the PDM. The read control logic uses this address to generate the internal physical address.

As shown in Figure 24, the user sends SPIA_RX32_RD_j in clock cycle T1. There is a 2-clock latency from the time the read enable SPIA_RX32_RD_j is asserted to the time read data is presented to the user. Thus, in clock cycle T3 following SPIA_RX32_RD_j, the embedded core provides the 32-bit data and control. The byte enables SPIA_RX32_BE_j[3:0] indicate which bytes within the 32-bit word are valid. If the current port ID is different from the previous port ID, then the first data read from the virtual FIFO is the port ID as shown in clock cycle T3. The SPIA_RX32_CTL_j signal is set to '1' during T3 indicating that read data contains the port ID. The byte enables are set to "0001" indicating that the least significant byte of the 32-bit word contains the port ID. If data is continued to be read for the same port, then no port ID is presented to the user as this is redundant. The ASTOP_ON_EOPj is

set to ‘0’ which means that the core will terminate a 4-word burst when EOP occurs and start transmission of the next set of data. Note that a new FIFO address ‘p’ is provided during the last 4-word burst of address ‘n’. This absorbs the 2-clock latency from the time a new address is presented to the DPRAM and the time read data is presented to the user.

As shown in Figure 25, when the ASTOP_ON_EOPj is asserted, 2 idle clock cycles are inserted after an EOP occurs. During these idle cycles, the byte enable bits are set to all ‘0’s indicating that the read data is not valid. After the idle cycles, the core continues to provide read data. The ASTOP_ON_EOPj signal can be asserted anytime and held high for any length of time during a read data transfer.

At any time, the user can poll for the status of a FIFO within a DPRAM bank by providing just the read address SPIA_RX32_ADDR_j without providing the read enable. Note that the timing diagram shown in Figure 24 and Figure 25 indicate a FIFO size of 16 locations. Each location contains four 32-bit data words. Thus when SPIA_RX32_FIFO_EMPTY_j flag is ‘1’ and RX_DPRAM_EMPTY_TYPE_SELA[j] software register bit in address 30920 is set to ‘1’, it implies that the FIFO has three 4-word data left (1/4 * 16 locations - 1).

As shown in Figure 26, the ERR signal indicates that the corresponding packet is in error. The ERR signal can be asserted by the ORSPI4 core several clock cycles before EOP and remains asserted until EOP occurs. However, the user is always required to use the ERR signal in conjunction with the EOP signal. Irrespective of the ERR signal being asserted, the core always transmits the entire packet. Thus, it is relevant for the user application to sample the ERR signal simultaneously with EOP. Users should also be aware that this is in asymmetry with the transmit SPI4 core’s behavior. In the transmit direction, user always asserts ERR signal during an EOP only. Thus, if any loopback is performed from the receive SPI4 (A or B) to the transmit SPI4 (A or B), the user should ensure that the received ERR signal is not wired directly to the corresponding transmit ERR signal, but follows the transmit protocol of providing an ERR signal only in the same clock cycle as an EOP and not earlier.

Figure 24. Read Timing with ASTOP_ON_EOP Deasserted - 32-Bit Mode

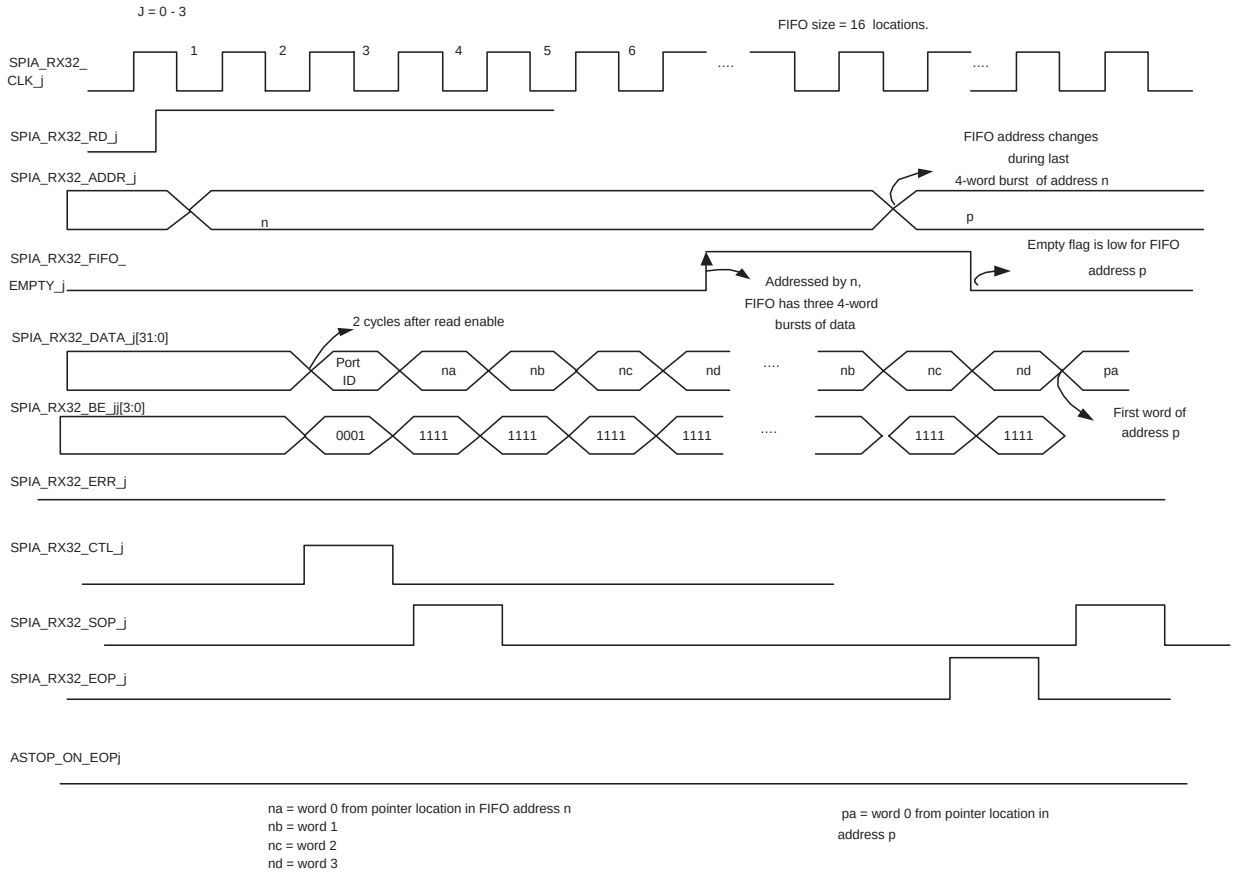
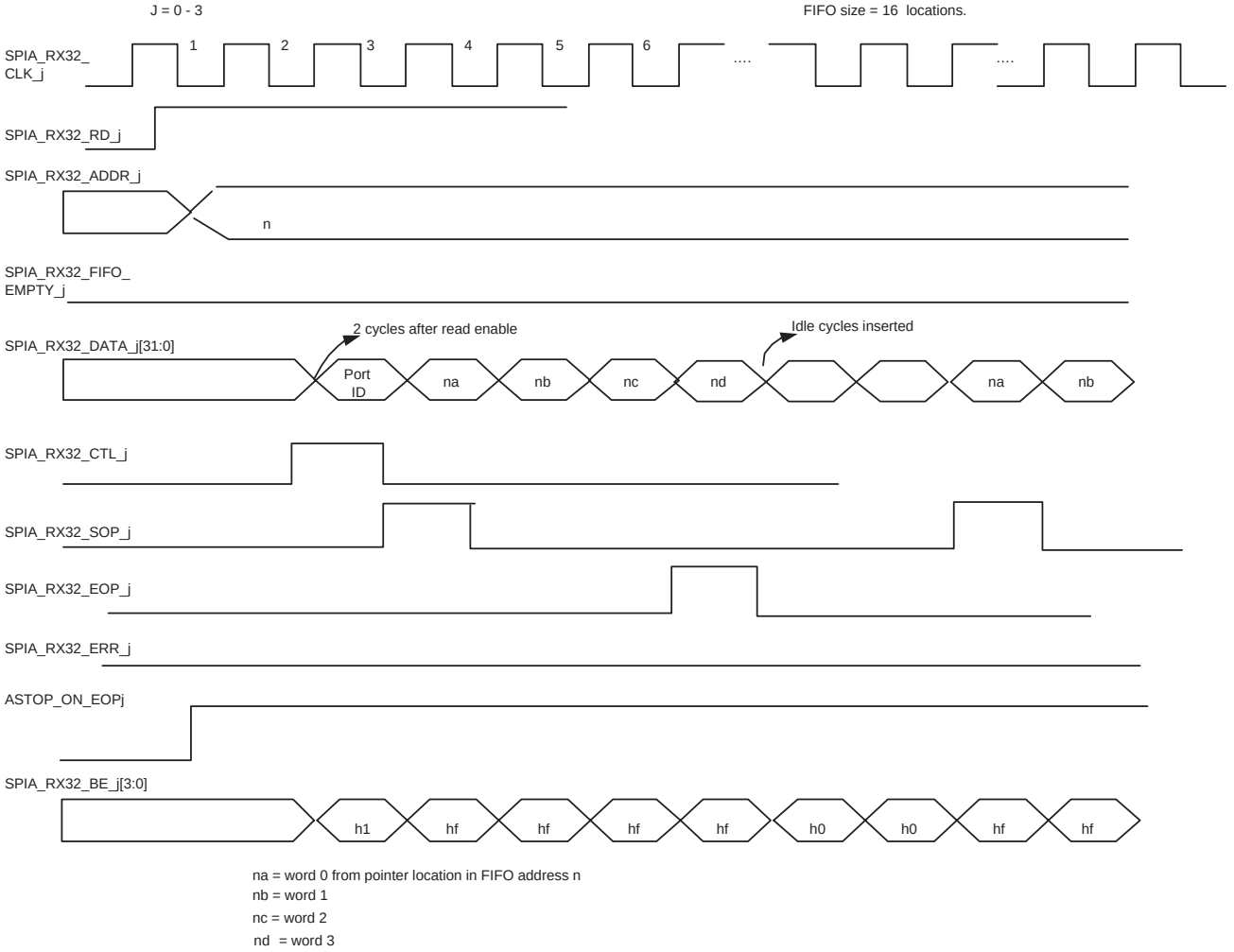
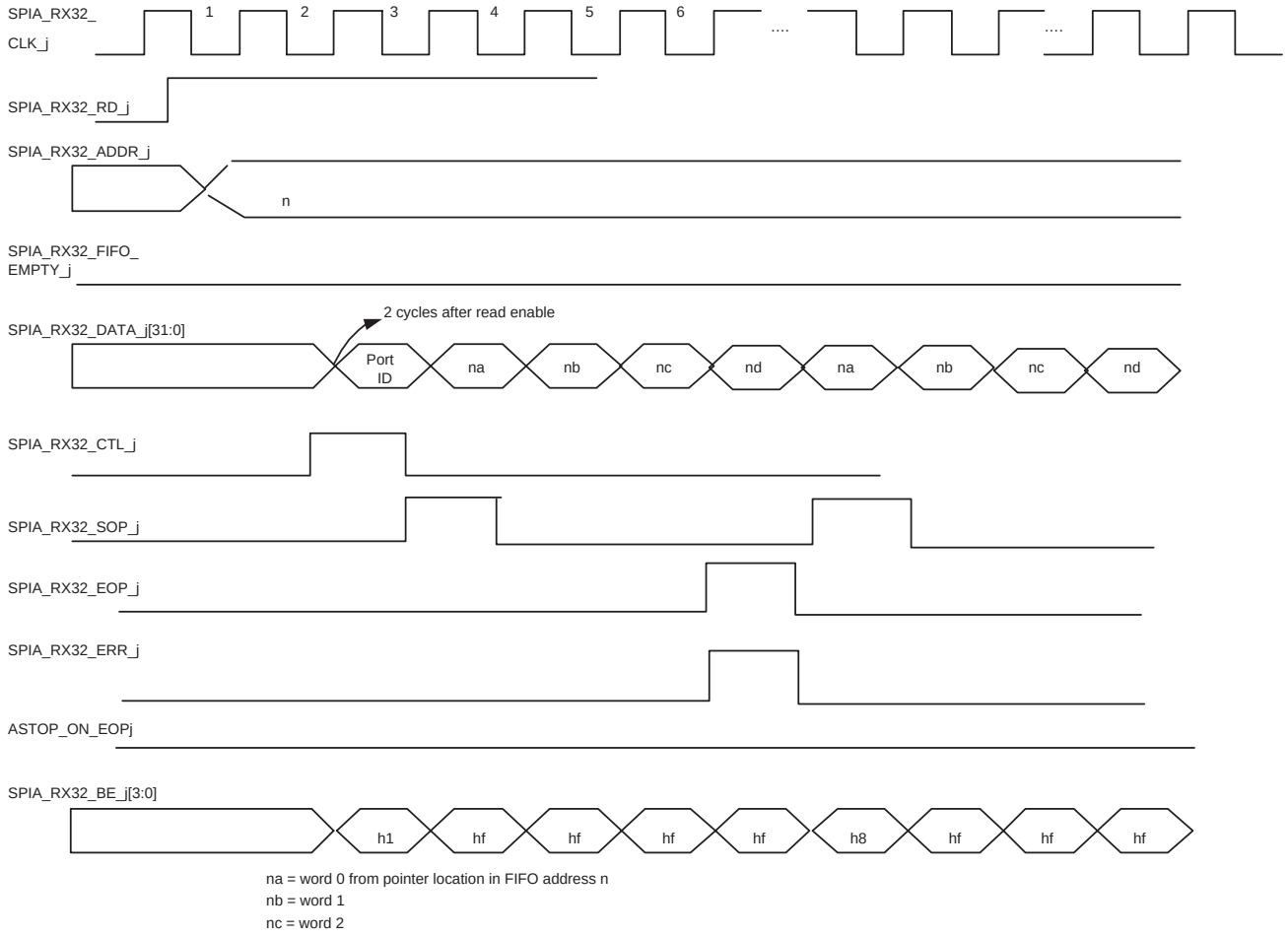


Figure 25. Read Timing with ASTOP_ON_EOP Asserted - 32-Bit Mode



na = word 0 from pointer location in FIFO address n
nb = word 1
nc = word 2
nd = word 3

Figure 26. Read Timing with ERR Signal Asserted - 32-Bit Mode



In the 64-bit data aggregation mode, DPRAMs 0 and 1 are used as a single memory. Similarly, DPRAMs 2 and 3 are combined into a single memory. The timing diagrams for the 64-bit aggregation mode are shown in Figure 27, Figure 28 and Figure 29. As shown in the timing diagrams, read data is always presented to the user in 2 clock cycles of 64-bit data each (plus one additional clock cycle for port ID). The exception is during an EOP. If EOP occurs during the first of the two-word data burst, the core terminates the burst and starts to present the next set of data. As shown in Figure 27, there is a two-clock cycle latency between the time read enable is asserted by the user and the time data is read from a virtual FIFO. In clock cycle T1, the read enable signal SPIA_RX64_RD_j is asserted along with the virtual FIFO (or buffer) address SPIA_RX64_ADDR_j. In clock cycle T3, port ID is presented followed by two data words in clock cycles T4 and T5. The port ID is always present on the least significant byte. This is indicated by the byte enable bits set to “00000001”. As shown in Figure 27, FIFO address is changed to ‘p’ during the last two-word burst from address ‘n’. This is done to absorb the two-clock cycle latency between read address and data.

As shown in Figure 28, when the ASTOP_ON_EOPj is asserted, two idle clock cycles are inserted after an EOP occurs. During these idle cycles, the byte enable bits are set to all ‘0’s indicating that the read data is not valid. After the idle cycles, the core continues to provide read data. The ASTOP_ON_EOPj signal can be asserted anytime and held high for any length of time during a read data transfer.

At any time, the user can poll for the status of a FIFO within a DPRAM bank by providing just the read address SPIA_RX64_ADDR_j without providing the read enable. Note that the timing diagram shown in Figure 27, Figure 28 and Figure 29 indicate a FIFO size of 32 locations. Each location contains two 64-bit data words. Thus when SPIA_RX64_FIFO_EMPTY_j flag is ‘1’ and RX_DPRAM_EMPTY_TYPE_SELA[j] software register bit in address 30920 is set to ‘1’, it implies that the FIFO has 7 two-word data left (1/4 * 32 locations - 1).

As shown in Figure 29, the ERR signal indicates that the corresponding packet is in error. The ERR signal can be asserted by the ORSPI4 core several clock cycles before EOP and remains asserted until the clock cycle when EOP is asserted. However, the user is always required to use the ERR signal in conjunction with the EOP signal. Irrespective of the ERR signal being asserted, the core always transmits the entire packet. Thus, it is relevant for the user application to sample the ERR signal simultaneously with EOP. Users should also be aware that this is in asymmetry with the transmit SPI4 core's behavior. In the transmit direction, user always asserts ERR signal during an EOP only. Thus, if any loopback is performed from the receive SPI4 (A or B) to the transmit SPI4 (A or B), the user should ensure that the received ERR signal is not wired directly to the corresponding transmit ERR signal, but follows the transmit protocol of providing an ERR signal only in the same clock cycle as an EOP and not earlier.

Figure 27. Read Timing with ASTOP_ON_EOP Deasserted - 64-Bit Mode

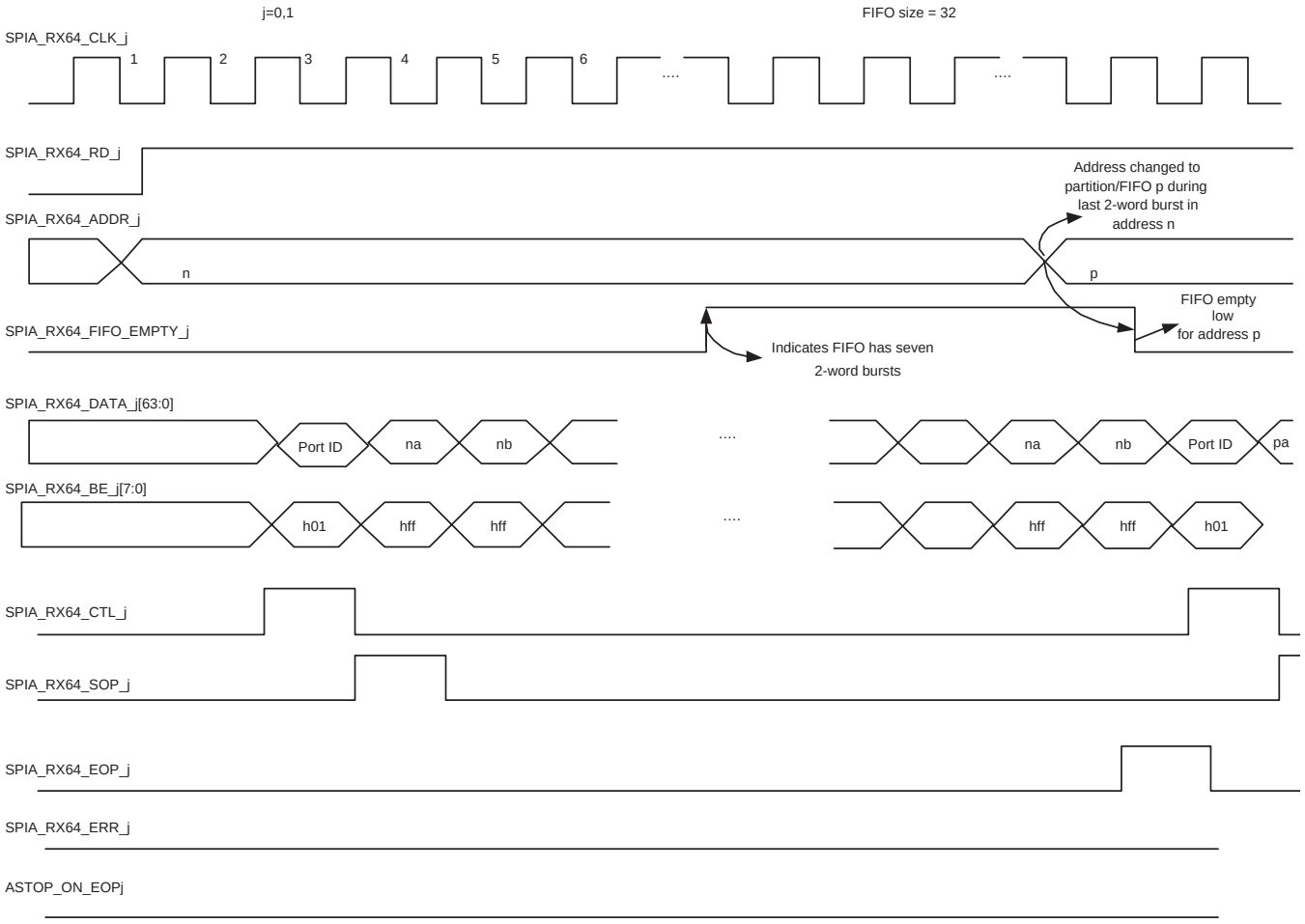
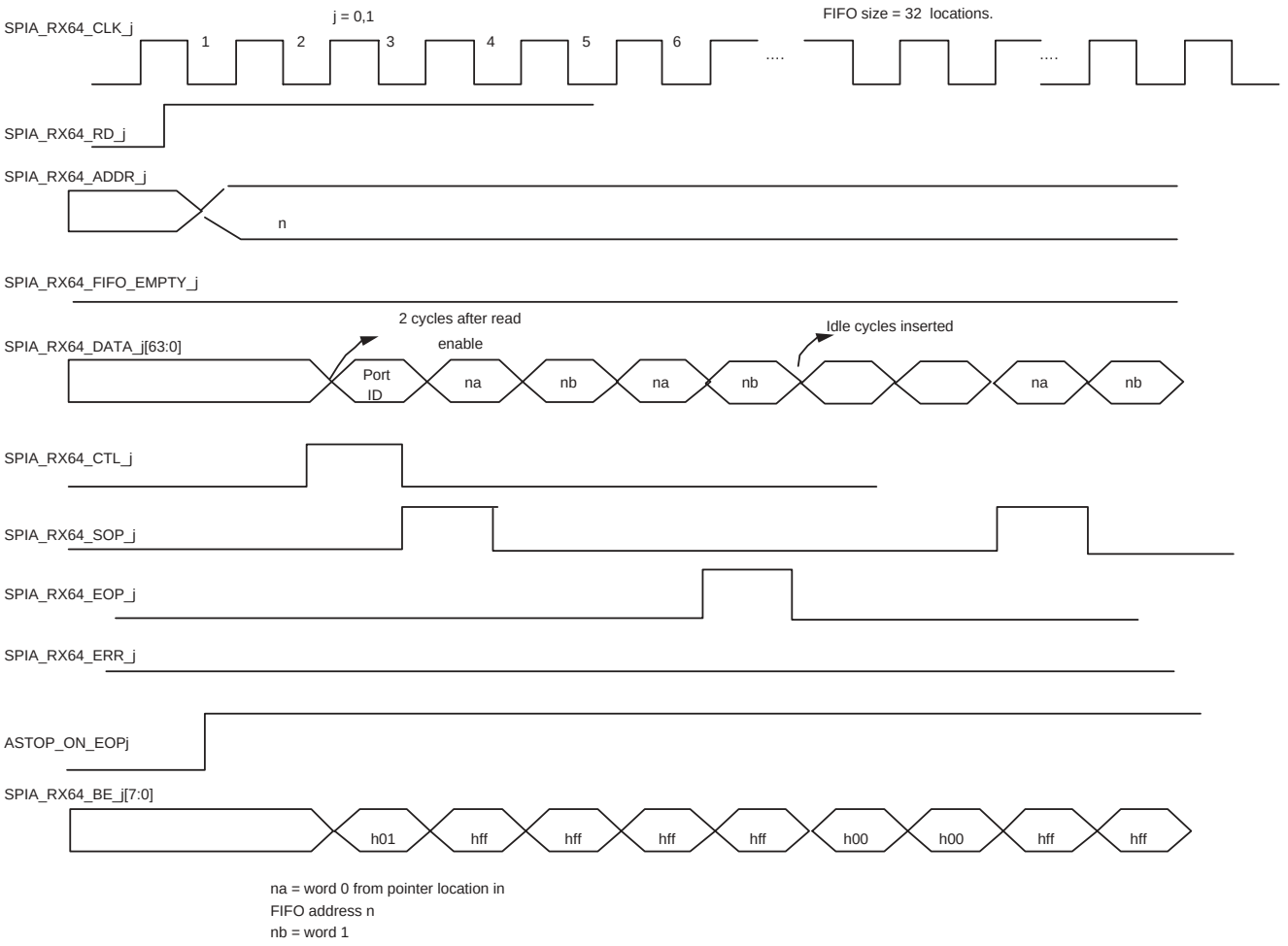
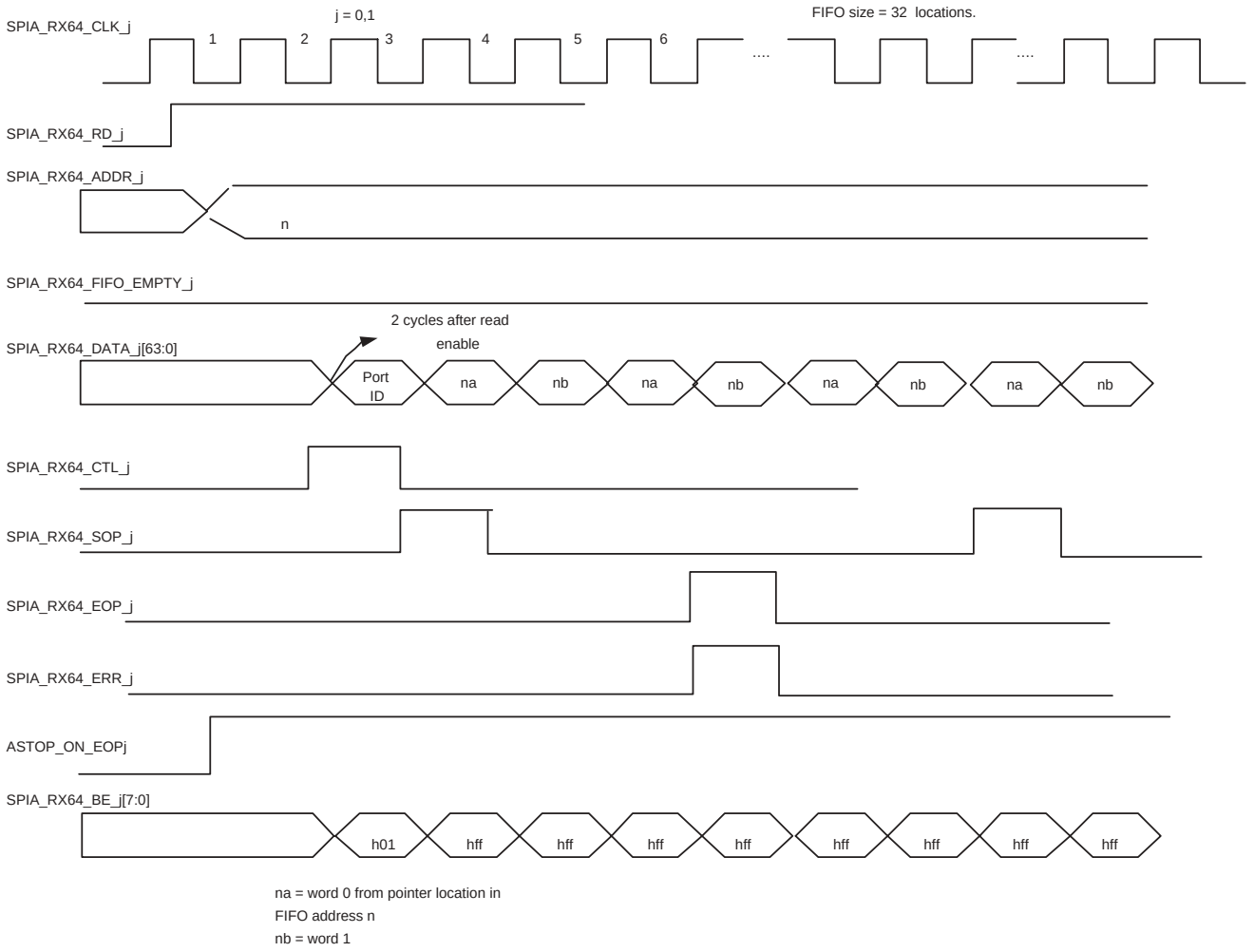


Figure 28. Read Timing with ASTOP_ON_EOP Asserted - 64-Bit Mode



na = word 0 from pointer location in FIFO address n
nb = word 1

Figure 29. Read Timing with ERR Asserted - 64-Bit Mode



In the 128-bit data aggregation mode, DPRAMs 0,1,2 and 3 are used as a single contiguous memory. All DPRAMs must be configured identically by software. In 128-bit aggregation mode, a maximum of 8 per-port buffers are possible. One significant difference between the 128-bit mode and the 32-bit and 64-bit modes is that the port ID is presented to the user on a separate bus (SPIA_RX128_PORTID[7:0]). Thus the SPIA_RX128_CTL signal has no relevance to the user and can be ignored. Read data is presented as a 128-bit word every clock cycle. As shown in Figure 30, read enable SPIA_RX128_RD and address SPIA_RX128_ADDR are presented in clock cycle T1. In T3 read data is available to the user after 2 clock cycles of latency. The FIFO address is changed to ‘p’ during the second last word of address ‘n’ such that the 2 clock cycles of latency between address and data can be absorbed.

As shown in Figure 31, when the ASTOP_ON_EOP is asserted, two idle clock cycles are inserted after an EOP occurs. During these idle cycles, the byte enable bits are set to all ‘0s indicating that the read data is not valid. After the idle cycles, the core continues to provide read data. The ASTOP_ON_EOP signal can be asserted anytime and held high for any length of time during a read data transfer.

At any time, the user can poll for the status of a FIFO within a DPRAM bank by providing just the read address SPIA_RX128_ADDR without providing the read enable. Note that the timing diagram shown in Figure 30, Figure 31 and Figure 32 indicate a FIFO size of 64 locations. Each location contains a 128-bit word. Thus when SPIA_RX128_FIFO_EMPTY flag is ‘1’ and RX_DPRAM_EMPTY_TYPE_SELA[0:3] software register bits in address 30920 are set to ‘1’, it implies that the FIFO has 15 128-bit words left (1/4 * 64 locations - 1).

As shown in Figure 32, the ERR signal indicates that the corresponding packet is in error. The ERR signal can be asserted by the ORSPI4 core several clock cycles before EOP and remains asserted until the clock cycle when EOP is asserted. However, the user is always required to use the ERR signal in conjunction with the EOP signal. Irrespective of the ERR signal being asserted, the core always transmits the entire packet. Thus, it is relevant for the user application to sample the ERR signal simultaneously with EOP. Users should also be aware that this is in asymmetry with the transmit SPI4 core's behavior. In the transmit direction, user always asserts ERR signal during an EOP only. Thus, if any loopback is performed from the receive SPI4 (A or B) to the transmit SPI4 (A or B), the user should ensure that the received ERR signal is not wired directly to the corresponding transmit ERR signal, but follows the transmit protocol of providing an ERR signal only in the same clock cycle as an EOP and not earlier.

Figure 30. Read Timing with ASTOP_ON_EOP Deasserted - 128-Bit Aggregation Mode

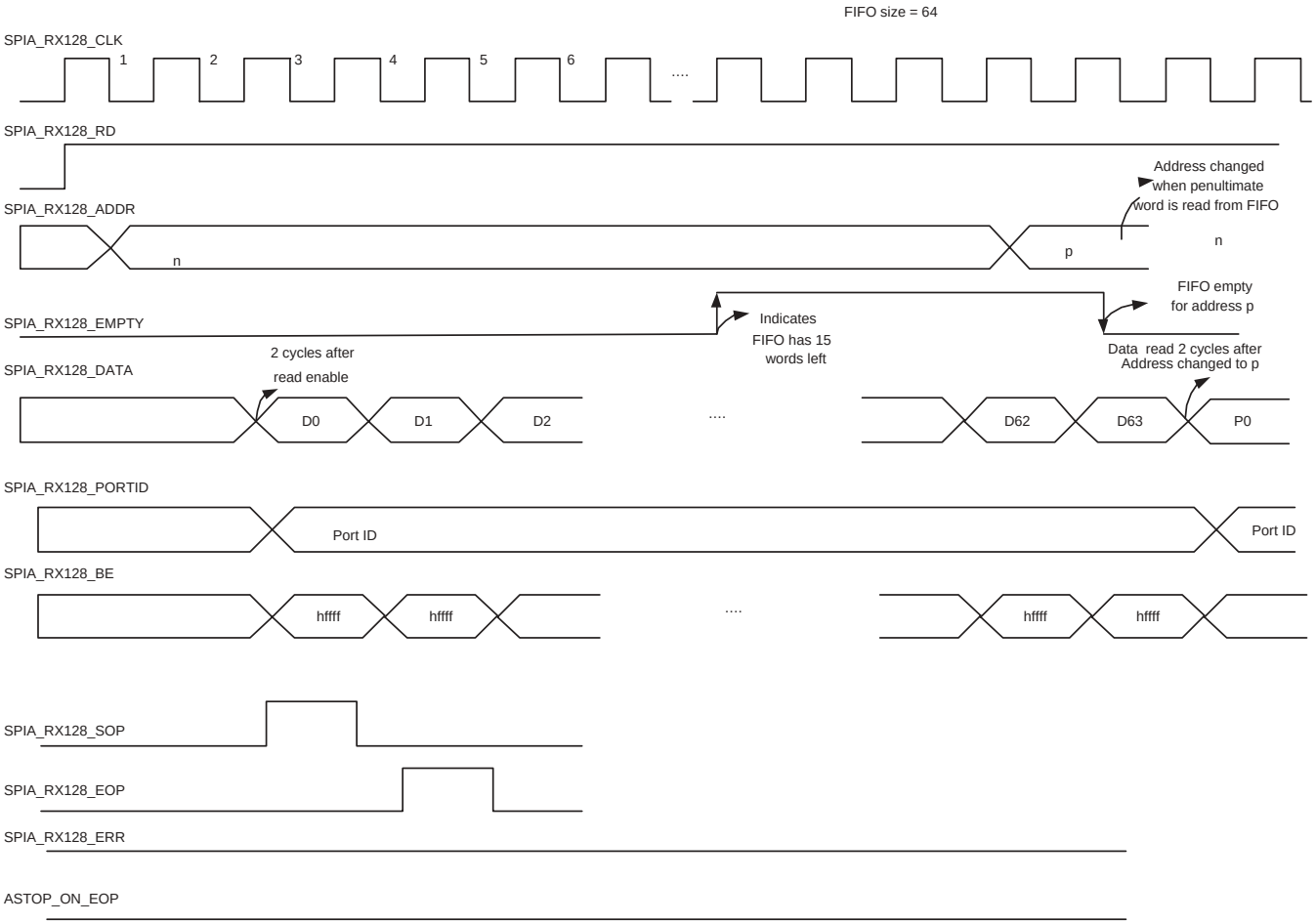


Figure 31. Read Timing with ASTOP_ON_EOP Asserted - 128-Bit Mode

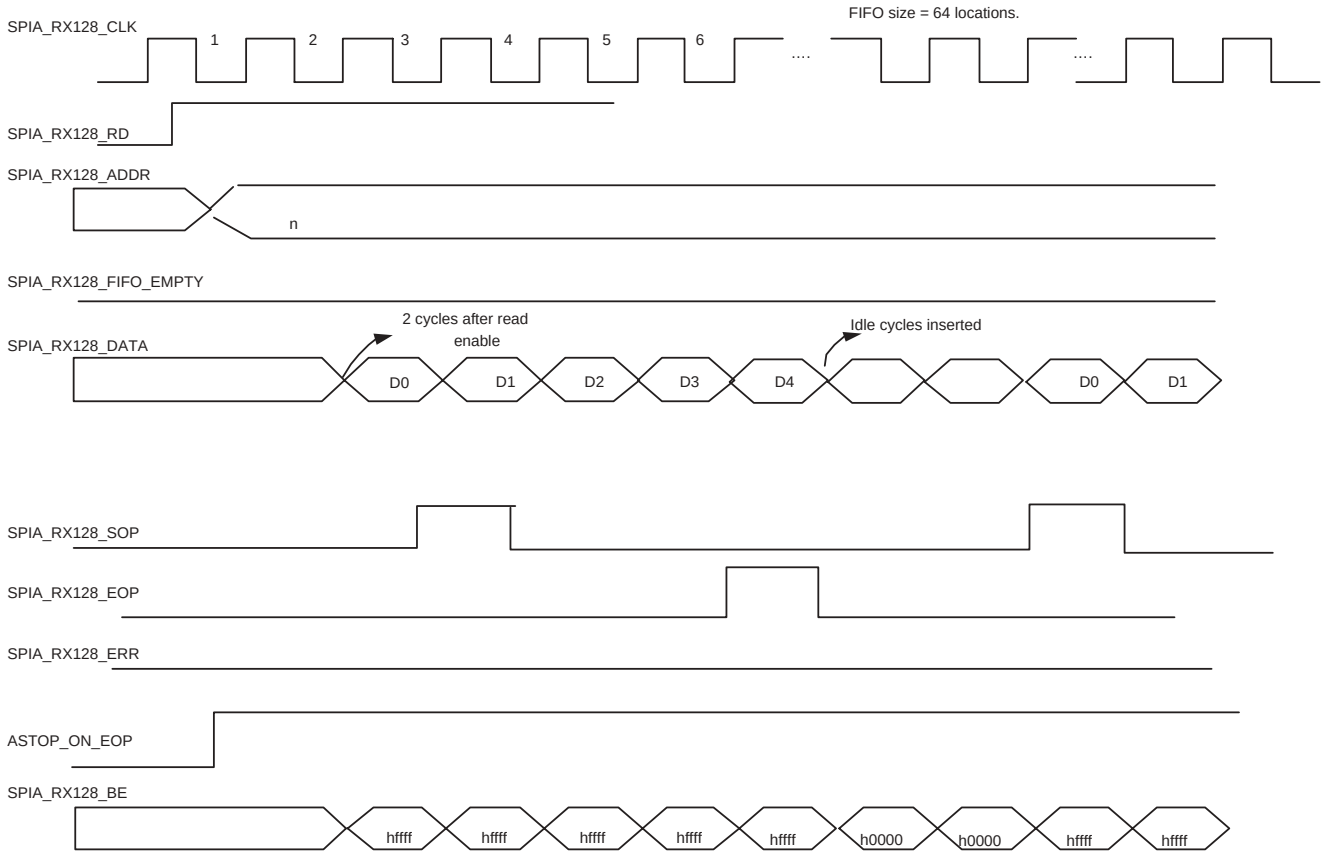
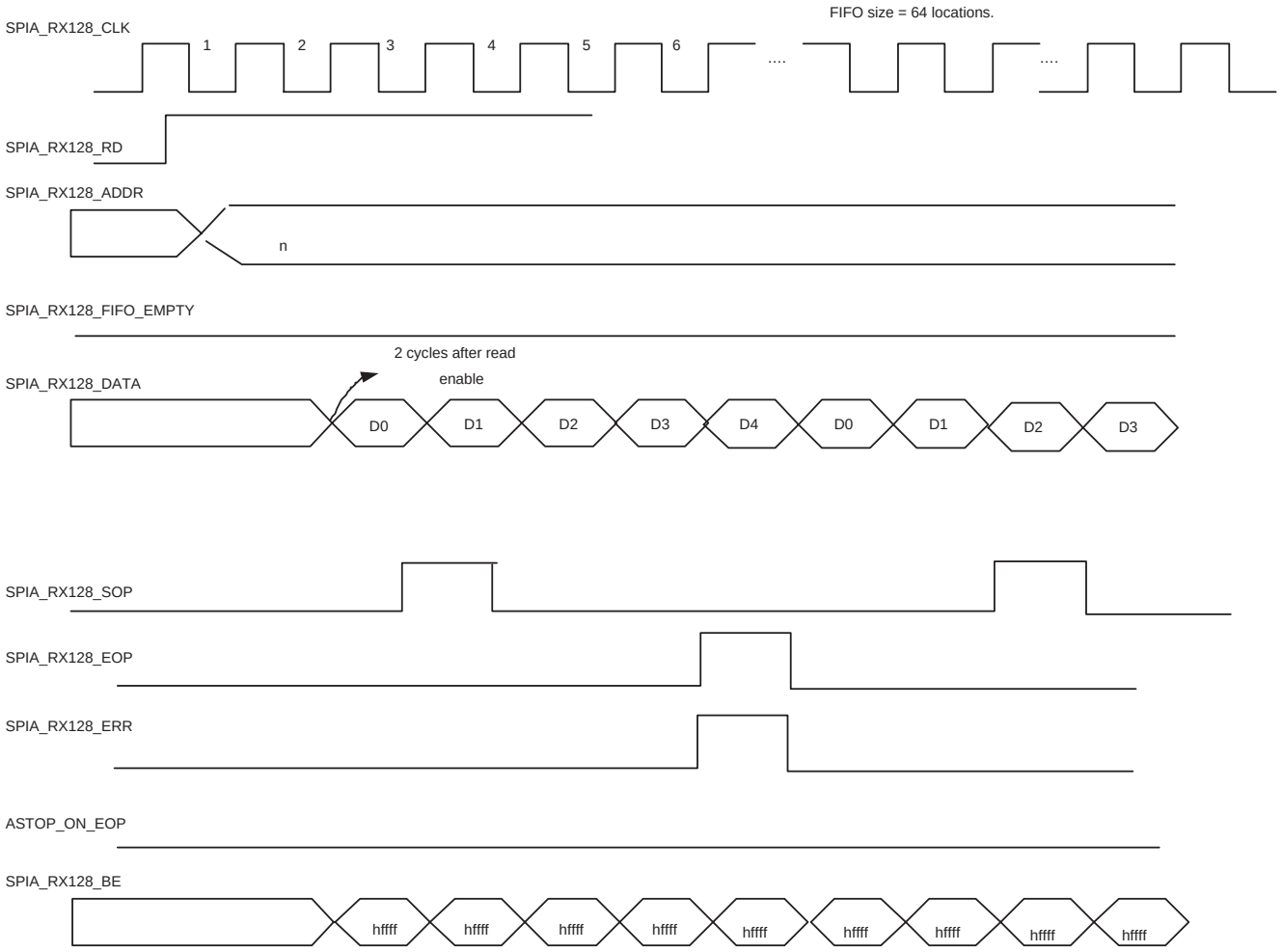


Figure 32. Read Timing with ERR Asserted - 128-Bit Mode



Special Operating Modes

Static Capture Mode

The ORSPI4 receiver supports both dynamic and static capture of data at the high-speed SPI4 interface. All the operating modes (32-bit, 64-bit and 128-bit) are independent of the type of data capture. They can operate in either static or dynamic capture mode. The choice of static vs. dynamic alignment depends on the SPI4 data rates. The static data capture is valid in the frequency range 400-700 Mbps. It also reduces the power dissipation of the device. The following control signals are relevant to enable static data capture:

SPI4_LOW_SPEED_DATA_SEL: This is a software register control bit (Address 30915 in SPIA and 30A15 in SPIB). When set to '0', the dynamic alignment circuit is held in a reset state and the static alignment mode is enabled.

SPI_DATM_A or SPI_DATM_B: This is a control signal from the FPGA. When set to '1', this gives the user the option to skew the RDCLK with respect to the SPI4 data bus RDAT on the bus by choosing delay taps through the control bits SPI_DLYTAP_A[2:0] of SPI_DLYTAP_B[2:0]. When set to '0', RDCLK is automatically centered on-chip to the input data bus RDAT.

Quarter-Rate Mode

The ORSPI4 SPI4 RX interface is also designed to operate at data rates much lower than 622 Mbps. Even though the OIF standard specifies a minimum data rate of 622 Mbits/s, the lower data rates are provided for users who

wish to use the SPI4 link for applications supporting <10 Gbits/s aggregate bandwidth (STS-48, STS-3, Gb Ethernet, etc.). To enable this low speed mode, user should set the software register bit SPI4_QUARTER_RATE (Address 30915 in SPIA and 30A15 in SPIB) to '1'. This supports data rates in the range of 100 - 200 Mbps. This bit takes precedence over SPI4_LOW_SPEED_DATA_SEL, since setting this bit to '1' will automatically reset the dynamic alignment block irrespective of SPI4_LOW_SPEED_DATA_SEL register bit setting.

SPI4 Loopback Modes

There are three forms of loopback supported directly by the ORSPI4 SPI4 blocks.

- **High-speed near end loopback:** This involves looping back data from the high-speed transmit block TDO serial output to the high-speed receive block RDI serial input (See Fig. 33). All of the logic excluding the LVDS buffers is included in the loopback path. The status path is looped from the output of the RSP block to the TSP block. This mode is enabled by setting the control register bit SPI4_LOOPBK_HS to 1.
- **Far end loopback:** This involves looping back the 128-bit output data from the RDI block to the 128-bit input of the TDO block (See Fig. 34). Data is received at the high-speed SPI4 RX interface and transmitted at the SPI4 TX interface. The status path is looped from TSTAT to RSTAT. This mode is enabled by setting control register bit SPI4_LOOPBK_FE to "1".
- **Low-speed near end loopback** which excludes the high-speed blocks from the loopback path: This involves sourcing data from the FPGA, looping back the output of TDP block into RDP block and observing data at the core-FPGA boundary (See Fig. 35). The status path is looped from the RSP block to the TSP block. This form of loopback will require an additional low-speed clock source (TSTCLK) and is enabled by setting control register bit SPI4_LOOPBK_LS to "1" and enabling SPI4_LOOPBK_HS.

Figure 33. High-Speed Near End Loopback

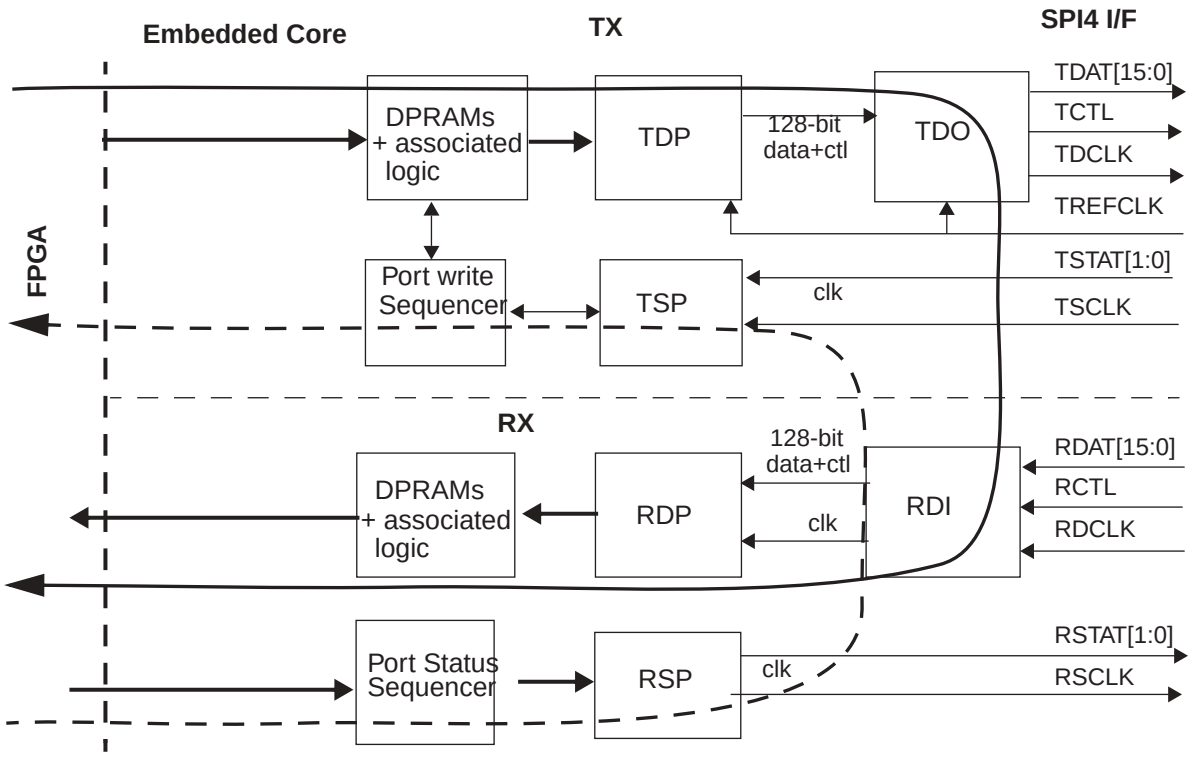


Figure 34. Far End Loopback

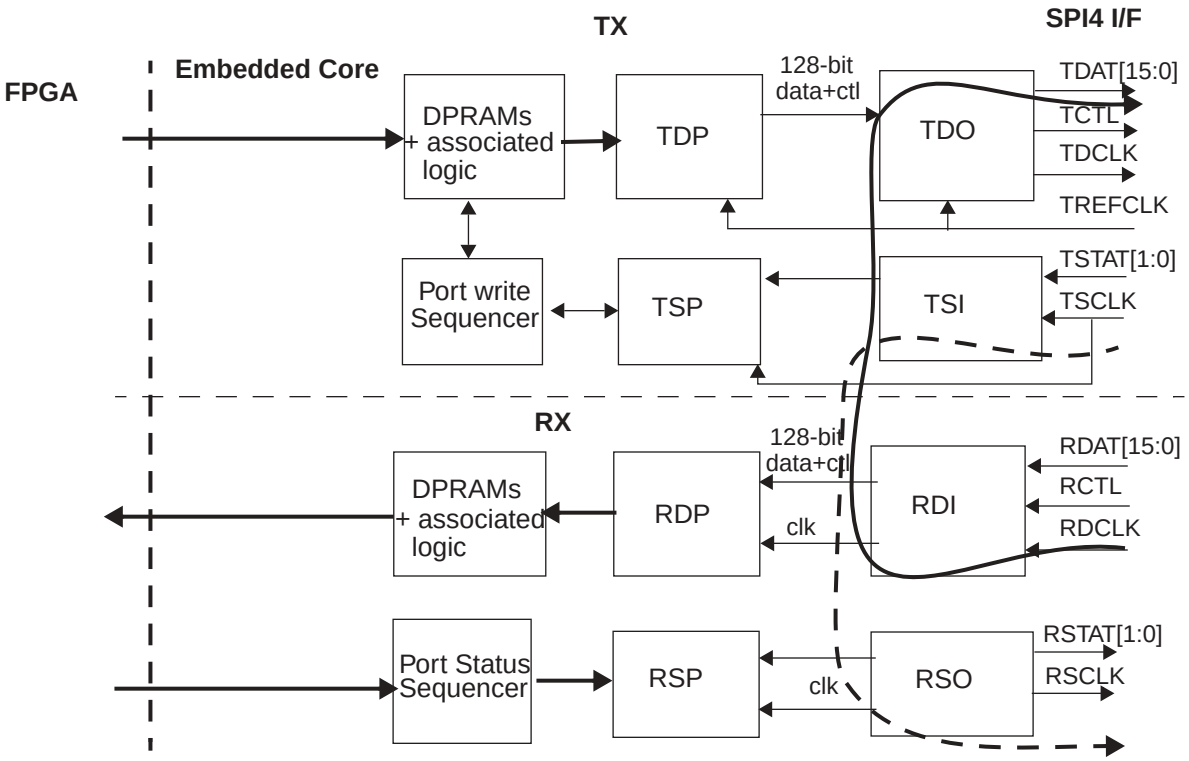
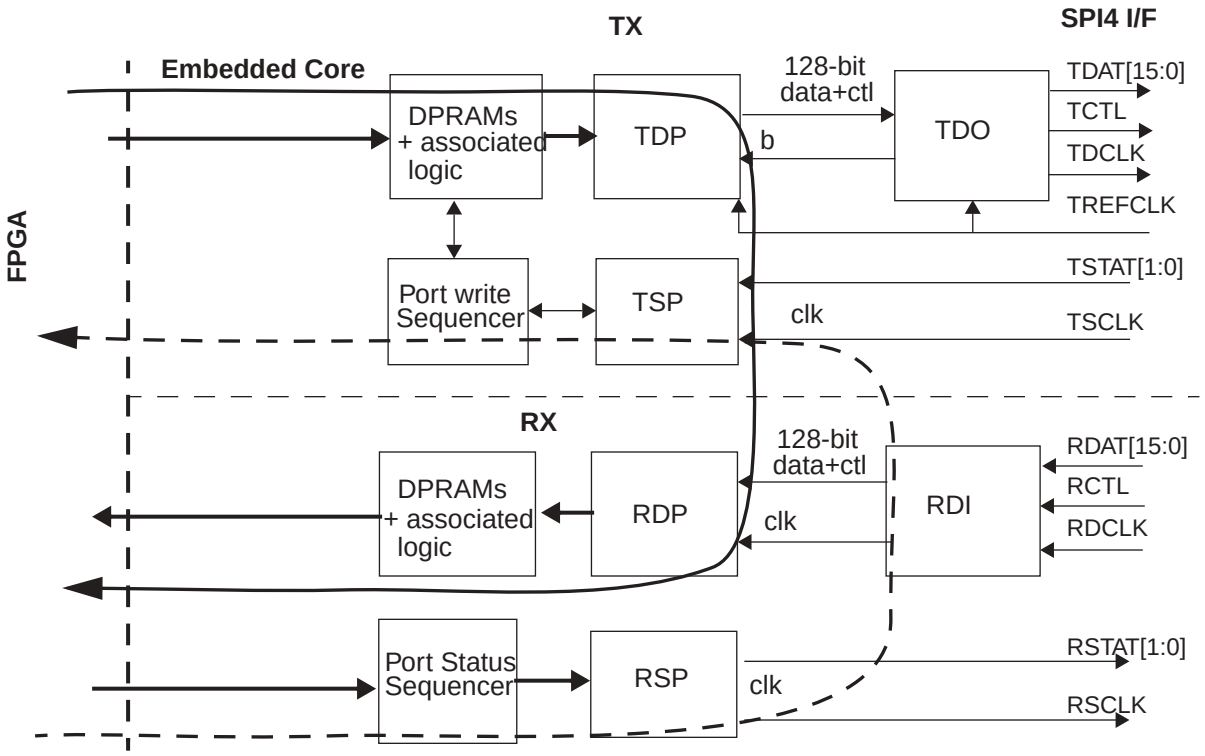


Figure 35. Low-Speed Near End Loopback (Excluding High-Speed Blocks)



SPI4 Error Insertion Capabilities

The SPI4 blocks support the following error insertion options for testing:

- DIP-4 odd parity is calculated over data and control words on the TX side. Under software control, DIP-4 errors can be forced by inverting the DIP-4 parity bits. This is done by setting TX_FORCE_DIP_ERR bit to '1' (Address 30915 in SPIA and 30A15 in SPIB).
- DIP-2 odd parity is calculated over the status frames on the RX side. Under software control, DIP-2 errors can be forced by inverting the DIP-2 parity bits. This feature can be enabled by setting RX_FORCE_DIP2_ERR to '1' (Address 30915 in SPIA and 30A15 in SPIB).

SPI4 Status Reporting Capabilities

The following status information will be reported through status registers. Most status registers also cause interrupts. The interrupts can be masked by the corresponding interrupt enables.

- DIP-4, DIP-2 errors (These cause interrupts). The associated register status bits are:
  - RX_ALGN_OFF_STS, TX_STATUS_LOF_STS, RX_DIP4_ERR_STS, TX_DIP2_ERR_STS
- DIP-4, DIP-2 error counters. These are 8-bit counters. The status registers are:
  - RX_DIP4_ERR_CNT, TX_DIP2_ERR_CNT
- Deskew status and error from high-speed RX side. These cause an interrupt. The associated register status bits are:
  - RX_DSKW_DONE_STS, RX_DSKW_ERR_STS
  - DPRAM FIFO overruns and underruns (RX and TX DPRAM memories). These can cause an interrupt.
  - SPIA or SPIB Receive PLL Loss of lock indicator. The associated register bit is
    –RX_PLL_LOL_STS

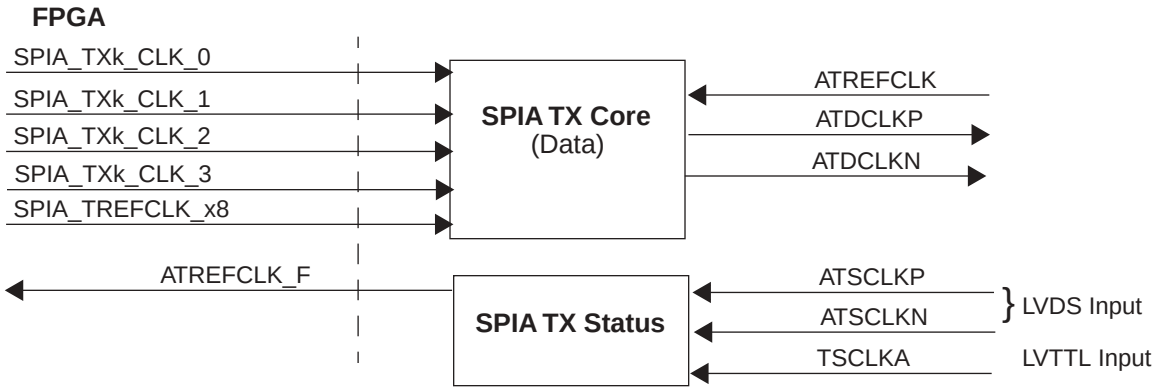
ORSPI4 SPI4 Clocking

Transmit Clocking

The following descriptions and figures show the SPIA core. The SPIB core is identical.

Figure 36. SPI4 TX Clocking

k= 32, 64, or 128



ATREFCLK: External Pad. Reference clock for the high-speed SPIA transmit block. Its frequency is 1/4th the SPI4 line clock ATDCLK. The ATREFCLK pad should be tied off during quarter-rate mode.

SPIA_TREFCLK_x8: Reference clock source from the FPGA when SPIA is operating in quarter-rate mode. Its frequency is two times the SPI4 line clock. i.e. if SPI4 line clock is 100 MHz, this clock is 200 MHz. Maximum allowable frequency is 200 MHz.

ATDCLK[P,N]: External output pads. SPI4 transmit data clock. SPI4 transmit data ATDAT and ATCTL are valid on both edges of this clock.

ATSCLK[P,N]: External input LVDS pads. SPI4 transmit status clock.

TSCLKA: External input 3.3V LVTTL transmit status input.

SPIA_TXk_CLK_j: Write clocks from FPGA to each of the four asynchronous DPRAMs.
(j = 0, 1, 2 or 3) (k = 32, 64 or 128)

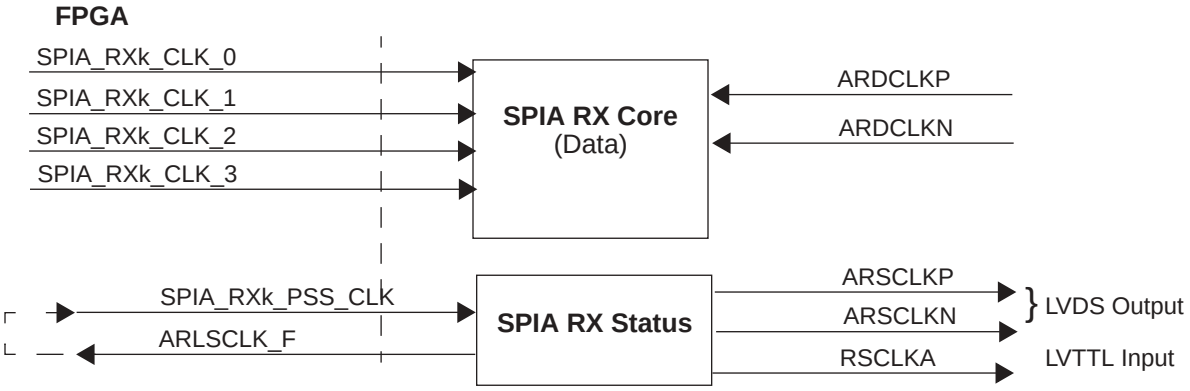
ATREFCLK_F: Clock output to the FPGA. Its frequency is the same as ATREFCLK. It is used to clock the status outputs to the FPGA, namely SPIA_TXk_PORT_ID, SPIA_TXk_STAT and SPIA_TXk_BURST_VAL.

Receive Clocking

The following descriptions and figures show the SPIA core. The SPIB core is identical.

Figure 37. SPI4 RX Clocking

k = 32, 64, or 128
j =0, 1, 2 or 3



ARDCLK[P,N]: SPI4 line clocks received with data and control. These are external LVDS input pads.

SPIA_RXk_CLK_j: Read clocks to each of the four asynchronous DPRAMs.

ARLSCLK_F: Output clock from receive status logic to FPGA.

SPIA_RXk_PSS_CLK: Write clocks from FPGA to the receive status RAM. The ARLSCLK_F clock can be used to source this clock as shown by dotted lines in the figure.

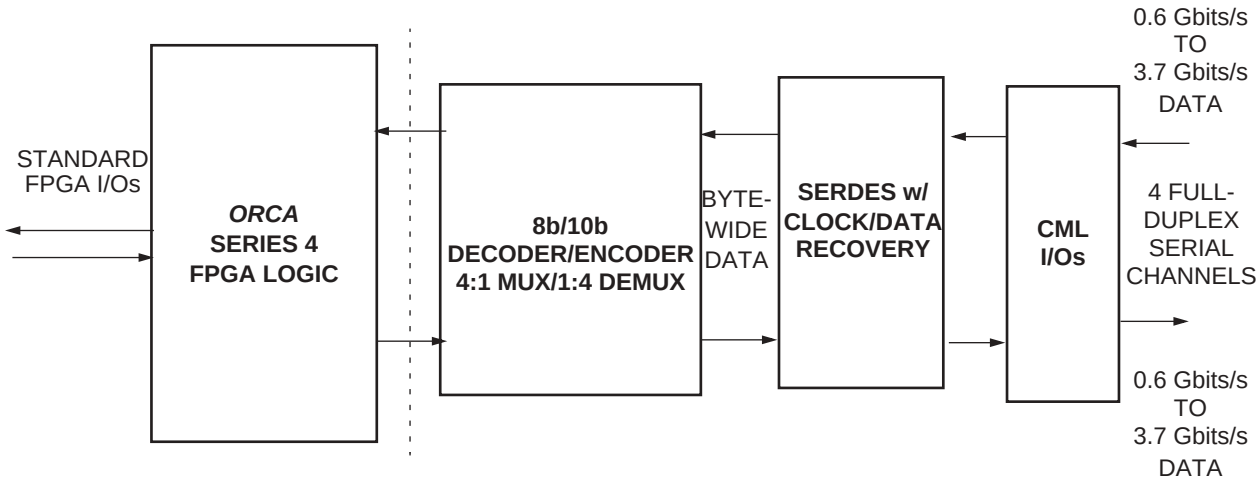
ARSCLK[P,N]: LVDS receive status outputs. External pads.

RSCLKA: 3.3 V LVTTL receive status output. External pads.

SERDES Functional Description

The SERDES portion of the ORSPI4 contains four Clock and Data Recovery (CDR) macrocells and Serialize/Deserialize (SERDES) blocks and supports 8b/10b (IEEE 802.3.2002) encoded serial links. It is intended for high-speed serial backplane data transmission. Figure 38 shows the SERDES top level block diagram and the basic data flow to and from the SERDES. Boundary scan for the SERDES only includes programmable I/Os and does not include any of the embedded block I/Os.

Figure 38. SERDES Top Level Block Diagram.



The SERDES serial channels can each operate at up to 3.7 Gbits/s (2.96 Gbits/s data rate) with a full-duplex synchronous interface with built-in clock recovery (CDR). The 8b/10b encoding provides guaranteed ones density for the CDR, byte alignment, and error detection. The core is also capable of frame synchronization and physical link monitoring. Overviews of the various blocks in the SERDES are presented in the following paragraphs

ORSPI4 SERDES Functional Block Overview

The SERDES portion of the core contains one transceiver block for serial data transmission at a selectable data rate of 0.6-3.7 Gbits/s. Each SERDES channel features high-speed 8b/10b parallel I/O interfaces to other core blocks and high-speed CML interfaces to the serial links.

The SERDES circuitry consists of receiver, transmitter, and auxiliary functional blocks. The receiver accepts high-speed (up to 3.7 Gbits/s) serial data. Based on data transitions, the receiver locks an analog receive PLL for each channel to retime the data, then de-multiplexes the data down to parallel bytes and an accompanying clock.

The transmitter operates in the reverse direction. Parallel bytes are multiplexed up to 3.7 Gbits/s serial data for off-chip communication. The transmitter generates the necessary high-speed clocks for operation from a lower speed reference clock.

The transceivers are controlled and configured through the system bus in the FPGA logic and through the external 8-bit microprocessor interface of the FPGA. Each channel has associated dedicated registers that are readable and write-able. There are also global registers for control of common circuitry and functions.

The SERDES performs 8b/10b encoding and decoding for each channel. The 8b/10b transmission code can support either Ethernet or Fibre Channel specifications for serial encoding/decoding, special characters, and error detection.

The user can disable the 8b/10b decoder to receive raw 10-bit words which will be rate reduced by the SERDES. If this mode is chosen, the user must bypass the multi-channel alignment FIFOs.

The SERDES block contains its own dedicated PLLs for both transmit and receive clock generation. The user provides a reference clock of the appropriate frequency. The receiver PLLs extract the clock from the serial input data and retime the data with the recovered clock.

MUX/DEMUX Block
The MUX/DEMUX block converts the data format for the high-speed serial links to a wide, low-speed format for crossing the CORE/FPGA interface. The intermediate interface to the SERDES macrocell runs at 1/10th the bit rate of the data lane. The MUX/DEMUX converts the data rate and bus width so the interface to the FPGA core can run at 1/4th this intermediate frequency, giving a range of 25.0-92.5 MHz for the data rates into and out of the FPGA logic.

Multi-Channel Alignment FIFOs
The four incoming data channels can be independent of each other, or can be synchronized in several ways. Two channels within a SERDES block can be aligned together; channels A and B and/or channels C and D. Alternatively, four channels in a SERDES block can be aligned together to form a communication channel with a bandwidth of 10 Gbits/s.

Individual channels within an alignment group can be disabled (i.e., powered down) without disrupting other channels.

XAUI and Fibre Channel Link State Machines
Two separate link state machines are included in the architecture. A XAUI link state machine is included in the SERDES, modeled after the IEEE 802.3ae standard. A separate state machine for Fibre Channel is also implemented.

FPGA/SERDES Interface
In 8b/10b mode, the FPGA logic will receive/transmit 32-bits of data (up to 92.5 MHz) and four K_CTRL bits from/to the SERDES. There are eight data streams in each direction plus additional timing, status and control signals.

Data sent to the FPGA can be aligned using comma (/K/) characters or the /A/ character as specified either by Fibre Channel or by IEEE 802.3ae for XAUI based interfaces. The alignment character is made available to the FPGA along with the data. The special characters K28.1, K28.5 and K28.7 are treated as valid comma characters by the SERDES.

If the receive channel alignment FIFOs are bypassed, then each channel will provide its own receive clock in addition to data and comma character detect signals. If the 8b/10b decoders are bypassed, then 40-bit data streams are passed to the FPGA logic. No channel alignment can be done in 8b/10b bypass mode.

The following table summarizes the interface signals between the FPGA logic and the core. In the table, an input refers to a signal flowing into the SERDES and an output refers to a signal flowing out of the SERDES.

Table 25. FPGA/SERDES Interface Description

FPGA/Embedded Core Interface Signal Name (x = [A, ..,D])	Input (I) to or Output (O) from Core	Signal Description
Transmit Path Signals		
TWDx[31:0]	FPGA → Core	Transmit data—channel x.
TCOMMAx[3:0]	FPGA → Core	Transmit comma character—channel x.
TBIT9x[3:0]	FPGA → Core	Transmit force negative disparity—channel x
TSYS_CLK_x	FPGA → Core	Transmit low-speed clock to the FPGA—channel x
TCK78	Core →FPGA	Transmit low-speed clock to the FPGA—SERDES Quad
Receive Path Signals		
MRWDx[39:0]	Core →FPGA	Receive data—Channel x (see Table 34).
RWCKx	Core →FPGA	Low-speed receive clock—Channel x.

RCK78	Core →FPGA	Receive low-speed clock to FPGA—SERDES Quad.
RSYS_CLK_1	FPGA → Core	Low-speed receive FIFO clock for channels A, B
RSYS_CLK_2	FPGA → Core	Low-speed receive FIFO clock for channels C, D
CV_SELx	FPGA → Core	Enable detection of code violations in the incoming data
SYS_RST_N	FPGA → Core	Synchronous reset of the channel alignment blocks.
FPGA_RESET_FC	FPGA → Core	Disables access to SERDES when high

Backplane Transceiver Core Detailed Description

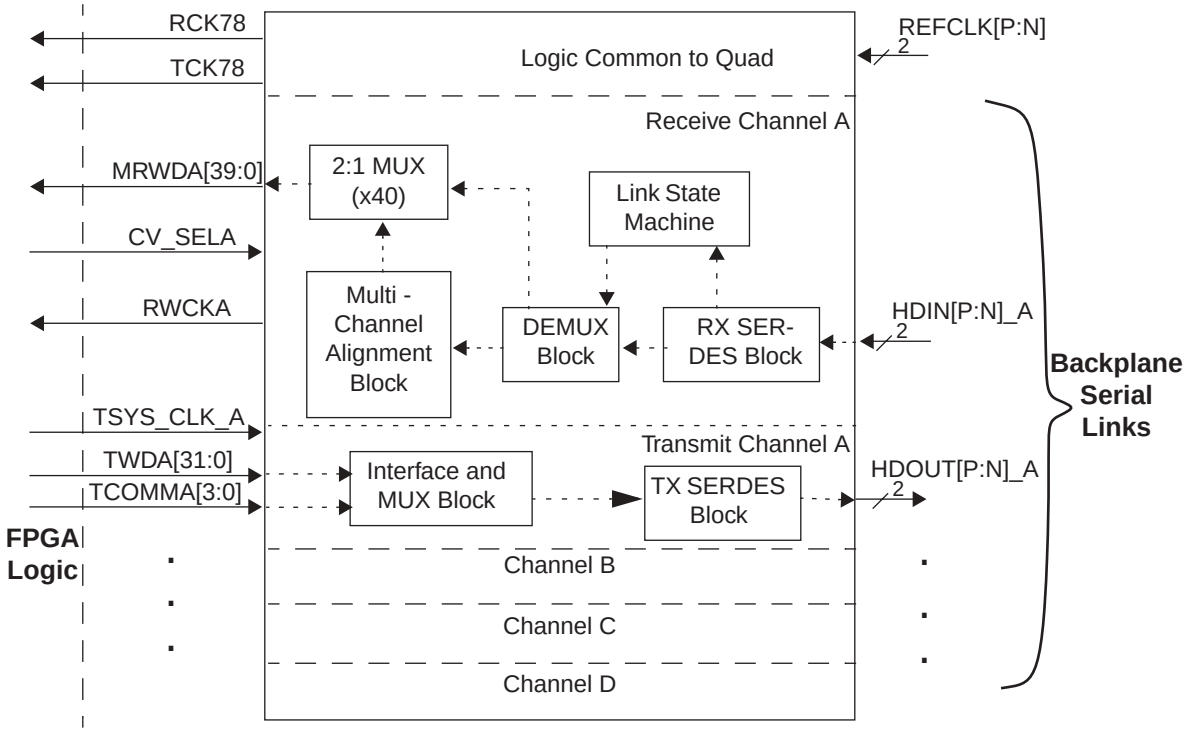
The following sections describe the various logic blocks in the SERDES portion of the FPSC. For a detailed description of the programmable logic functions, please see the ORCA Series 4 FPGA Data Sheet and related application and technical notes.

The major functional blocks in the SERDES include:

- One SERializer-DESerializer (SERDES) block and Clock and Data Recovery (CDR) circuitry
- 8b/10b encoder/decoders
- Transmit pre-emphasis circuitry
- 4-to-1 multiplexers (MUX) and 1-to-4 demultiplexers (DEMUX)
- Fibre channel synchronization state machine
- XAUI link alignment state machine
- Alignment FIFOs

A top level block diagram of the SERDES Logic is shown in Figure 39.

Figure 39. Top Level Block Diagram, SERDES Embedded Core Logic



SERDES Detailed Description

The SERDES provides transceiver functionality for four serial data channels. Each channel is identified by a channel identifier [A:D].

The data channels can operate independently or they can be combined together (aligned) to achieve higher bit rates. The mode operation of the core is defined by a set of control registers, which can be written through the system bus interface. Also, the status of the core is stored in a set of status registers, which can be read through the system bus interface.

The transmitter section for each channel accepts 40 bits (RAW MODE) of data or 32-bits of data and eight control/status bits from the FPGA logic and (optionally) encodes the data using 8b/10b encoding. It also accepts the low-speed reference clock at the REFCLK input and uses this clock to synthesize the internal high-speed serial bit clock. The data is then serialized and the serialized data are available at the differential CML output terminated in 86 Ω to drive either an optical transmitter or coaxial media or circuit board/backplane.

The receiver section receives high-speed serial data at its differential CML input port. These data are fed to the clock recovery section, which generates a recovered clock and retimes the data. The retimed data are also de-serialized and optionally 8b/10b decoded. The receiver also (optionally) recognizes the comma characters or code violations and aligns the bit stream to the proper word boundary. The resulting parallel data is (optionally) passed to the multi-channel alignment block before it is presented to the FPGA logic.

8b/10b Encoding and Decoding

In 8b/10b mode, the FPGA logic will receive/transmit 32-bits of data and four K_CTRL bits from/to the embedded core. In the transmit direction, four additional input bits can force a negative disparity present state. The SERDES logic will encode the data to, or decode the data from, a 10-bit format according to the FC-PH ANSI X3.230:1994 standard (which is also the encoding used by the IEEE 802.3ae Ethernet standard). This encoding/decoding scheme also allows for the transmission of special characters and supports error detection.

Following the definitions and conventions used in defining the 8b/10b coding rules, each valid coded character has a name corresponding to its 8-bit binary value:

- Dx.y for data characters
- Kx.y for special characters
- x = the 5-bit input value, base 10, for bits ABCDE
- y = the 3-bit input value, base 10, for bits FGH

An 8b/10b encoder is designed to maintain a neutral average disparity. Disparity is the difference between the number of “1”s and “0”s in the encoded word. Neutral disparity indicates the number of “1”s and “0”s are equal. Positive disparity indicates more “1”s than “0”s. Negative disparity indicates more “0”s than “1”s. The average disparity determines the DC component of the signals on the serial line. Running disparity is a record of the cumulative disparity of every encoded word, and is tracked by the encoder.

In order to maintain neutral disparity, two different codings are defined for each data value. The 8b/10b encoder in the transmit path selects between (+) and (-) encoded word based on calculated disparity of the present data to maintain neutral disparity.

In the receive path, the clock and data recovery blocks retime the incoming data and 8b/10b decoders generate 8-bit data based on the received 10-bit data. A sequence of valid 8b/10b coded characters has a maximum run length of 5-bits (i.e., five consecutive “1”s or five consecutive “0”s before a mandatory bit transition). This assures adequate transitions for robust clock recovery.

The recovered data is aligned on a 10-bit boundary by detecting and aligning to special characters in the incoming data stream. Data is word-aligned using the comma (/K/) character. A comma character is a special character that contains a unique pattern (0011111 or its complement 1100000) in the 10-bit space that makes it useful for delim-

iting word boundaries. The special characters K28.1, K28.5 and K28.7 contain this comma sequence and are treated as valid comma characters by the SERDES.

The following table shows all of the valid special characters. All of the special characters are made available to the FPGA logic; however only the comma characters are used by the SERDES logic. The different codings that are possible for each data value are shown as encoded word (+) and encoded word (-). The table also illustrates the 8b/10b bit labeling convention. The bit positions of the 8-bit characters are labeled as H,G,F,E,D,C,B and A and the bit positions of the 10-bit encoded characters are labeled as a, b, c, d, e, i, f, g, h, and j. The encoded words are transmitted serially with bit 'a' transmitted first and bit 'j' transmitted last.

Table 26. Valid Special Characters

K Character	HGF EDCBA 765 43210	K Control	Encoded Word (–) abcdei fghj	Encoded Word (+) abcdei fghj
K28.0	000 11100	1	001111 0100	110000 1011
K28.1 /comma/	001 11100	1	001111 1001	110000 0110
K28.2	010 11100	1	001111 0101	110000 1010
K28.3 /A/	011 11100	1	001111 0011	110000 1100
K28.4	100 11100	1	001111 0010	110000 1101
K28.5 /comma/	101 11100	1	001111 1010	110000 0101
K28.6	110 11100	1	001111 0110	110000 1001
K28.7 /comma/	111 11100	1	001111 1000	110000 0111
K23.7	111 10111	1	111010 1000	000101 0111
K27.7	111 11011	1	110110 1000	001001 0111
K29.7	111 11101	1	101110 1000	010001 0111
K30.7	111 11110	1	011110 1000	100001 0111

Transmit Path (FPGA to Backplane) Logic

The transmitter section accepts four groups of either 8-bit unencoded data or 10-bit encoded data at the parallel interface to the FPGA logic. It also uses the reference clock, REFCLK[P:N] to synthesize an internal high-speed serial bit clock. The serialized transmitted data are available at the differential CML output pins to drive either an optical transmitter, coaxial media or a circuit board backplane.

As shown in Figure 40, the basic blocks in the transmit path include:

SERDES/FPGA Interface and 4:1 Multiplexer

- Low speed parallel core/FPGA interface
- 4:1 multiplexer

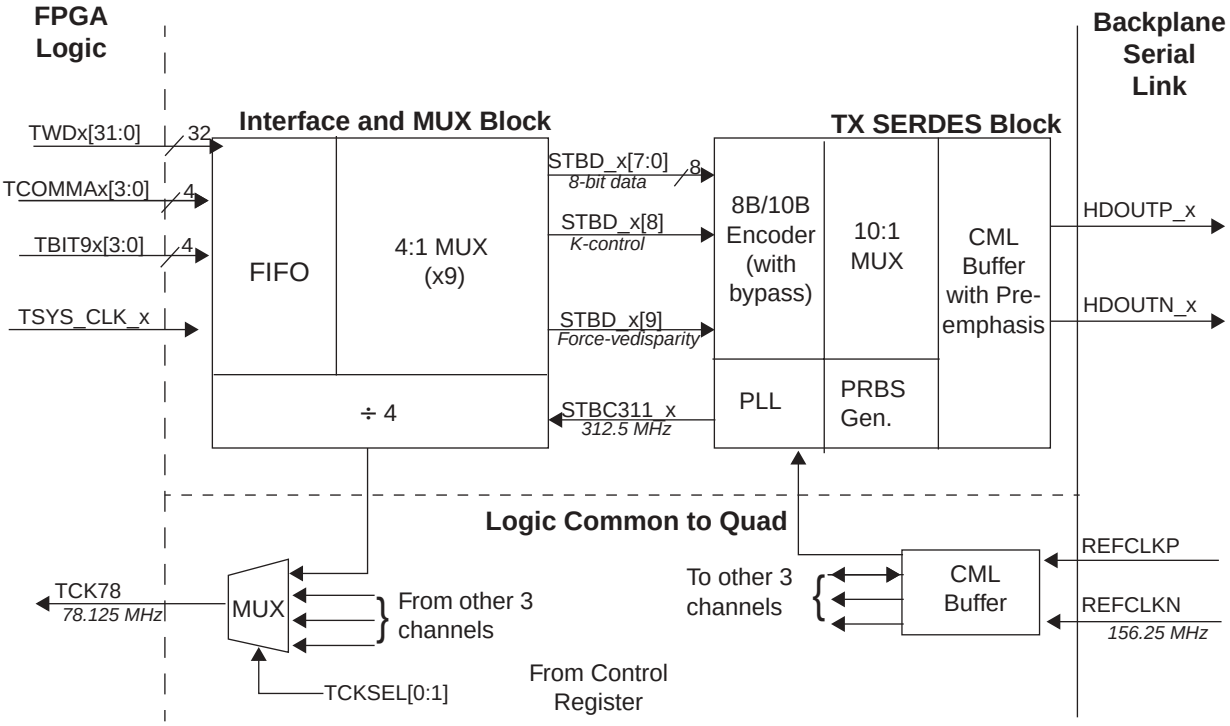
Transmit SERDES

- 8b/10b Encoder
- 10:1 Multiplexer
- CML Output Buffer

Detailed descriptions of the logic blocks are given in following sections. Detailed descriptions of transmit clock distribution, including the transmit PLL are given in later sections of this data sheet.

Figure 40. Basic Logic Blocks, Transmit Path, Single Channel (Typical Reference Clock Frequency)

Note: x= [A, B, ... D]



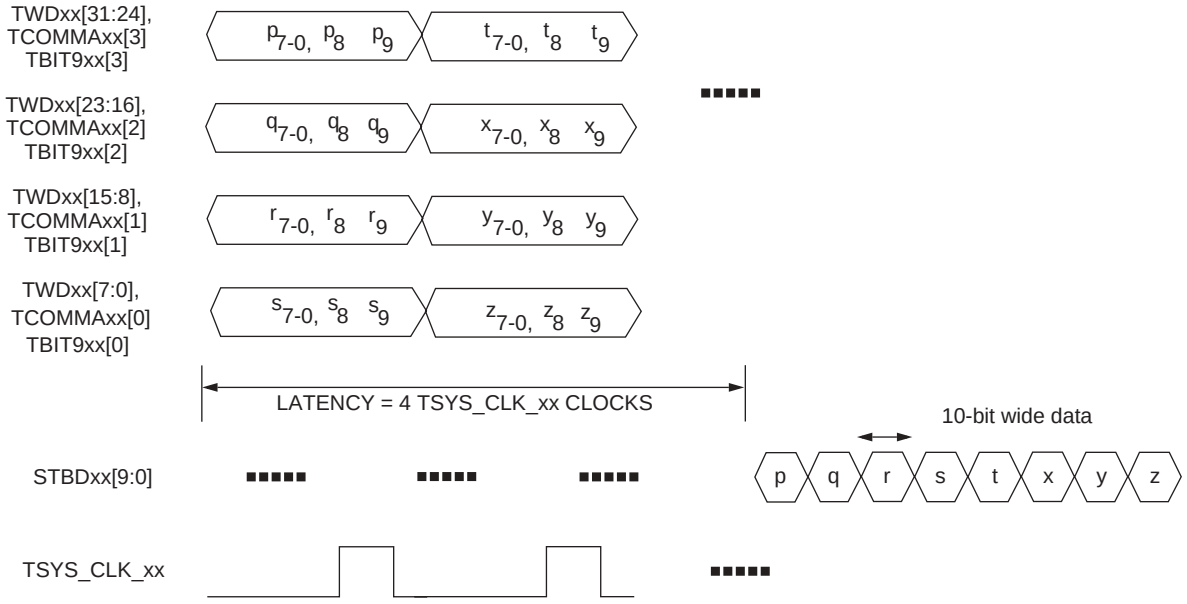
SERDES/FPGA Logic Interface and 4:1 Multiplexer

These blocks provide the data formatting and transmit data and clock signal transfers between the SERDES and the FPGA Logic. Control and status registers in the FPGA portion of the chip control the transmit logic and record status. These bits are passed to the core using the FPGA System Bus and are described in later sections of this data sheet.

The low-speed transmit interface consists of a clock and 4 data bytes, each with an accompanying control bit. The data bytes are conveyed to the MUX via the TWDx[31:0] ports (where x represents the channel label [A,B,C or D]. The control bits are TCOMMAx[3:0] which define whether the input byte is to be interpreted as data or as a special character and TBIT9x[3:0] which are used to force a negative disparity present state. The data and control signals are synchronized to the transmit clock, TSYS_CLK_x. Both the data and control are strobed into the core on the rising edge of TSYS_CLK_x. Note that each TBIT9x[3:0] controls the disparity of the encoded version of its corresponding data byte. Setting bit TBIT9C[3] to “1”, for instance, will force the 8b/10b encoder to asserts a current negative running disparity state. This will cause it to encode TWDC[31:24] positively (more “1”'s than “0”'s). Setting TBIT9x to 0 will leave the encoder free to alternate between positive and negative encoding to maintain a zero running disparity.

The MUX is responsible for taking 40 bits of data/control at the low-speed transmit interface and up-converting it to 10 bits of data/control at the SERDES transmit interface. The MUX has two clock domains - one based on the clock received from the SERDES block and a second that comes from the FPGA at 1/4 the frequency of the SERDES clock. The time sequence of interleaving data/control values is shown in Figure 41.

Figure 41. TRANSMIT MUX Block Timing - Single Channel. (NOTE: xx = A, B, C, or D)

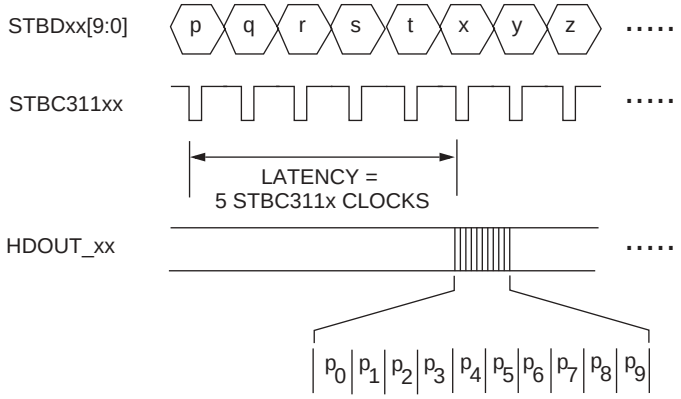


SERDES Block

The SERDES block accepts either 8-bit data to be encoded or 10-bit unencoded data at the parallel input port from the MUX/DEMUX block. It also accepts the reference clock at the REFCLK input and uses this clock to synthesize the internal high-speed serial bit clock.

The internal STBC311x clock is derived from the reference clock. The frequency of this clock depends on the setting of the half-rate/full-rate control bit setting the mode of the SERDES and the frequency of the REFCLK and/or that of the high-speed serial data. A falling edge on the STBC311x clock port will cause a new data character to be transferred into the SERDES block. The latency from the SERDES block input to the high-speed serial output is five STBC311x clock cycles, as shown in Figure 42.

Figure 42. Transmit Path Timing - Single SERDES Channel (NOTE: xx = A, B, C, or D)



Each block also sends a clock to the FPGA logic. This clock, TCK78[A,B], is sourced from one of the four MUX blocks and has the same frequency as TSYS_CLK_x, but arbitrary phase. Within each MUX block, the low frequency clock output is obtained by dividing by four, the SERDES STBC311x clock, which is used internally to synchronize the transmit data words. TCKSEL control bits select the channel to source TCK78. The internal signals STBDx[9:0] (where x is represents A, B, C or D) from the MUX block carry unencoded character data and control bits. The 10th bit (STBDx[9]) of each data lane into the SERDES is used to force a negative disparity present state.

8b/10b Encoder and 1:10 Multiplexer

The 8b/10b encoder encodes the incoming 8-bit data into a 10-bit format as described previously. The input signals to the block, STBDx[7:0] are used for the 8-bit unencoded data. STBDx[8] is used as the K_control input to indicate whether the 8 data bits need to be encoded as special characters (K_control = "1") or as data characters (K_control = 0). When STBDx[9:0] = "1", a negative disparity present state is forced. When the encoder is bypassed STBDx[9:0] serve as the data bits for the 10-bit unencoded data.

Within the 8b/10b transmission code, the bit positions of the 10-bit encoded transmission characters are labeled as a, b, c, d, e, i, f, g, h, and j in that order. Bit a corresponds to STBDx[0], bit b to STBDx[1], bit c to STBDx[2], bit d to STBDx[3], bit e to STBDx[4], bit i to STBDx[5], bit f to STBDx[6], bit g to STBDx[7], bit h to STBDx[8], and bit j to STBDx[9].

The 10-bit wide parallel data is converted to serial data by the 10:1 Multiplexer. The serial data are then sent to the CML output buffer and are transmitted serially with STBDx[0] transmitted first and STBDx[9] transmitted last.

CML Output Buffer

The transmitter's CML output buffer is terminated on-chip in 86 ohms to optimize the data eye, as well as to reduce the number of discrete components required. The differential output swing reaches a maximum of 1.2 V PP in the normal amplitude mode. A half amplitude mode can be selected via configuration register bit HAMP_x. Half amplitude mode can be used to reduce power dissipation when the transmission medium has minimal attenuation or for testing of the integrity (loss) of the physical medium.

A programmable preemphasis circuit is provided to boost the high frequencies in the transmit data signal to maximize the data eye opening at the far-end receiver. Preemphasis is particularly useful when the data are transmitted over backplanes or low-quality coax cables which have a frequency-dependent amplitude loss. For example, for FR4 material at 2.5 GHz, the attenuation compared to the 1.0 GHz value is about 3 dB. The attenuation is a result of skin effect loss of the PCB conductor and the dielectric loss of the PCB substrate. This attenuation causes intersymbol interface which results in the closing of the data eye opening at the receiver. Since this effect is predictable for a given type of PCB material, it is possible to compensate for this effect in two ways - transmitter preemphasis and receiver equalization. Each of these techniques boosts the high frequency components of the signal but transmit preemphasis is preferred due to the ease of implementation and the better power utilization. It also gives a better signal-to-noise ratio because receiver equalization amplifies both the signal and the noise at the receiver

Applying too much preemphasis when it is not required, for example when driving a short backplane path, will also degrade the data eye opening at the receiver. In the ORSPI4 the degree of transmit preemphasis can be programmed with a two-bit control from the microprocessor interface as shown in Table 27. The high-pass transfer function of the preemphasis circuit is given by the following equation, where the value of "a" is shown in Table 27.

$$H(z) = (1 - az^{-1}) \quad (1)$$

Table 27. Preemphasis Settings

PE1	PE0	Amount of Preemphasis (a)
0	0	0% (No Preemphasis)
0	1	12.5%
1	0	12.5%
1	1	25%

Receive Path (Backplane to FPGA) Logic

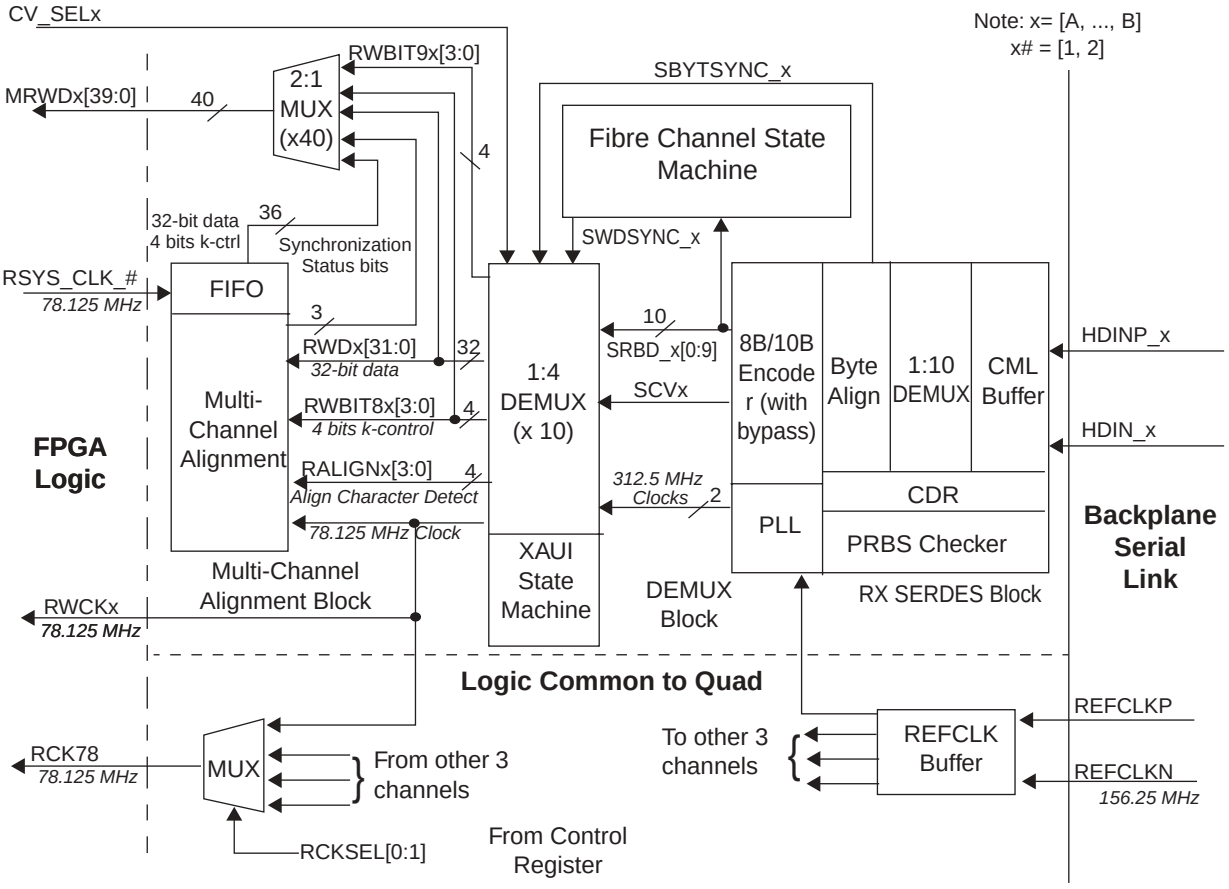
The receiver section receives high-speed serial data at the external differential CML input pins. These data are fed to the clock recovery section which generates a recovered clock and retimes the data. Therefore the receive clocks are asynchronous between channels. The retimed data are deserialized and presented as an 8-bit decoded or a 10-bit unencoded parallel data on the output port. The receiver also optionally recognizes comma characters, detects code violations and aligns the bit stream to the proper word boundary.

As shown in Figure 43, the basic blocks in the receive path include:

Receive SERDES Block

- CML input buffer
- Receive PLL
- 1:10 demultiplexer (DEMUX)
- Clock and Data Recovery (CDR) section
- 10b/8b decoder
- 1:4 demultiplexer and SERDES/FPGA interface
- 1:4 DEMUX
- Low speed parallel SERDES/FPGA logic interface
- Multi-channel alignment logic

Figure 43. Basic Logic Blocks, Receive Path, Single Channel (Typical Reference Clock Frequency)



Each channel provides its own received clock, received data and K-character detect signals to the FPGA logic. Incoming data from multiple channels can be aligned using comma (/K/) characters or /A/ character (as specified either in Fibre Channel specifications or in IEEE 802.3ae for XAUI based interfaces). If the 8b/10b decoders are bypassed, then 40-bit data streams are passed to the FPGA logic. No channel alignment can be done in this 8b/10b bypass mode.

Synchronization
The ORSPI4 SERDES RX logic performs four levels of synchronization on the incoming serial data stream. Each level builds upon the previous, providing first bit, then byte (character), then channel (32-bit word), and finally multi-channel alignment.

Bit alignment is the task of the Clock/Data Recovery (CDR) block. This block utilizes a PLL that locks to the transitions in the incoming high-speed serial data stream, and outputs the extracted clock as well as the data. If the PLL is unable to lock to the serial data stream, it instead locks to REFCLK to stabilize the Voltage-Controlled Oscillator (VCO), and periodically switches back to the serial data stream to again attempt synchronization. This process continues until a valid input data stream is detected and lock is achieved. The CDR can maintain lock on data as long as the input data stream contains an adequate data "eye" (i.e., jitter is within specification) and the maximum data stream run length is not exceeded.

Byte alignment occurs once valid bit alignment is achieved. The byte aligner looks for a particular 7-bit sequence (either 0011111 or its complement, 1100000) that, has been 8b/10b encoded per Fibre Channel or IEEE 802.3.2002 specifications, only occurs in the comma (/K/) characters K28.1, K28.5 and K28.7. Byte alignment only occurs when the ENBYSYNC_x signal for that channel is active high, and re-alignment occurs on each 7-bit sequence encountered. However, if ENBYSYNC_x is asserted active high and no comma character is encountered, and then is brought inactive low, the channel will still perform one byte alignment operation on the next comma character. Byte alignment occurs immediately when an alignment sequence is detected, so the lock time is only one clock period.

Word (32-bit) alignment can occur after the Fibre Channel (XAUI_MODE_x = "0") or XAUI (XAUI_MODE_x = "1") state machine has reached the in-synchronization state. In Fibre Channel mode, synchronization (WDSYNC_x = "1") will occur after three ordered sets of data have been received in the absence of any code violations. After this, the next ordered set will cause the output data to be aligned such that the comma character is in the most significant byte. Thus, 32-bit word alignment has been achieved when four ordered sets have been detected. The time required is directly dependent on comma-character density.

Once word alignment is accomplished, no further alignment occurs unless and until WDSYNC_x goes to zero and back to one again. Comma characters that are not located in the most significant byte position will not trigger further re-alignment while WDSYNC_x is active. This behavior is as defined by the Fibre Channel specification. However, it means that, if the channel experiences an abrupt delay change (as could occur if an external mux performs protection switch between two links) and if the delay change is close enough to a full character or characters that not enough code violations are generated to cause loss of WDSYNC_x, the channel could become misaligned and remain that way indefinitely. As mentioned above, this behavior is that defined by the Fibre Channel specification. In XAUI mode, as the state diagram later in this data sheet indicates, three error-free code-groups containing commas must be detected before synchronization is declared.

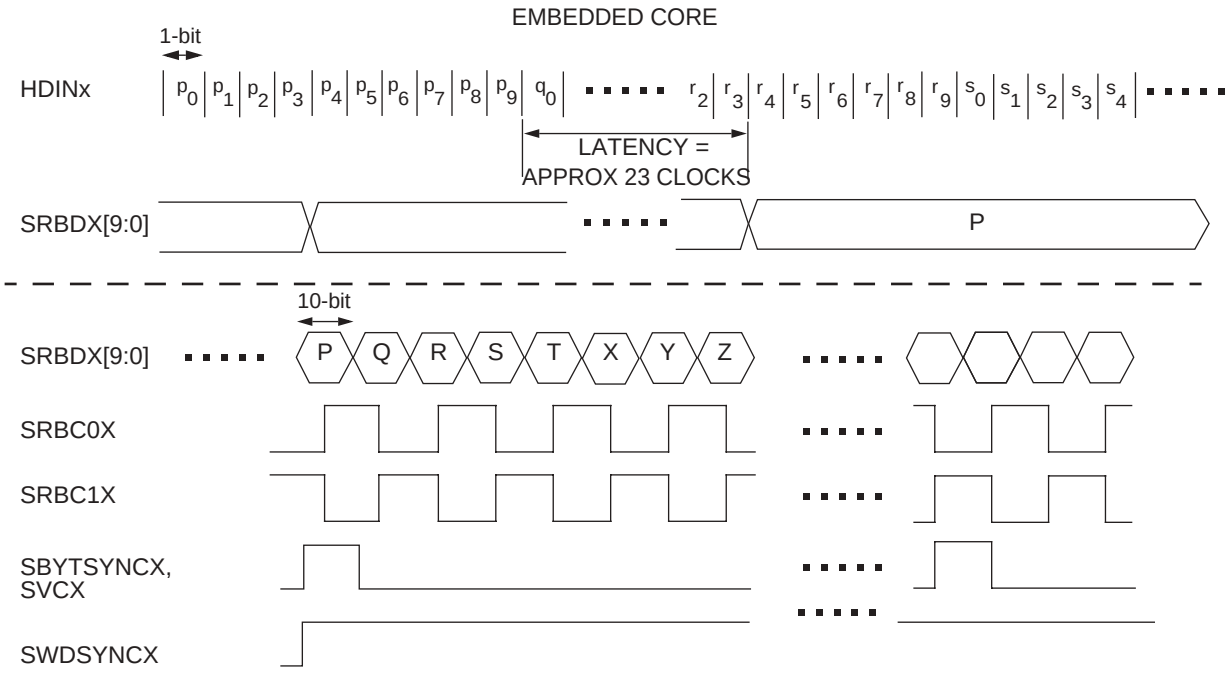
Multi (2 or 4) channel alignment (Lane alignment in XAUI mode) can be performed after 32-bit word alignment is complete. Multi-channel alignment is described in later sections of this data sheet.

Receive CML Input Buffer and SERDES
The receiver section receives high-speed serial data at its differential CML input port. The receive input is an AC coupled input. The received data is sent to the clock recovery section which generates a recovered clock and retimes the data. Valid data will be received after the receive PLL has locked to the input data frequency and phase.

The received serial data is converted to a 10-bit wide parallel data by the 1:10 demultiplexer. Clock recovery is performed by the SERDES block for each of the eight receive channels. This recovered data is then aligned to a 10-bit word boundary by detecting and aligning to a comma special character. Word alignment is done for either polarity of the comma character. The 10-bit code word is passed to the 8b/10b decoder, which provides an 8-bit byte of data, a special character indicator bit and a SBYTSYNC_x signal (where again x is a placeholder for A,...,D). Data from a SERDES channel is sent to the DEMUX block in 10-bit raw form or 8-bit decoded form. Accompanying this data are the comma-character indicator (SBYTSYNC_x), link-state indicator (SWDSYNC_x), clocks (SRBC0_x, and SRBC1_x), and code-violation indicator (SCVx). The two internal clocks operated at twice the reference clock

frequency. With the 8b10bR_x control bit of the SERDES channel set to “1”, the data presented at SRBD_x[9:0] will be decoded characters. Bit 8 will indicate whether SRBDx[7:0] represents an ordinary data character (bit 8 = 0), or whether SRBD_x[7:0] represents a special character, like a comma. Bit 9 may be either a code violation indicator or one of seven out of synchronization state indicators, as described later.

Figure 44. Receive Path Timing for a Single SERDES Channel. (NOTE: xx = A, B, C, or D)



When 8b10bR is set to “0”, the data at SRBD_x[9:0] will not be decoded. The XAUI link-state machine should not be used in this mode of operation. When in XAUI mode, the MUX/DEMUX looks for /A/ (as defined in IEEE 802.3ae v.2.1) characters for channel alignment and requires the characters to be in decoded form for this to work

1:4 Demultiplexer (DEMUX)

The 1:4 DEMUX has to accumulate four sets of characters presented to it at the SERDES receive interface and put these out at one time at the low-speed receive interface.

Another task of the 1:4 DEMUX is to recognize the synchronizing event and adjust the 4-byte boundary so that the synchronizing character leads off a new 4-byte word. In Fibre Channel mode, this synchronizing character is a comma. This feature will be referred to as DEMUX word alignment in other areas of this document. DEMUX word alignment will only occur when the communication channel is synchronized. When there is no synchronization of the link, the 1:4 DEMUX will continue to output 4-byte words at some arbitrary, but constant, boundary. There are 2 control register bits available for each channel for word alignment. They are DOWDALGN_x and NOWDALGN_x. The DOWDALGN_x bit is positive edge triggered. Writing a 0 followed by a “1” to this register bit will cause the corresponding DEMUX to look for a new comma character and align the 32-bit word such that the comma is in the most significant byte position. It is important that the comma is in the most significant byte position since the multi-channel aligner looks for comma in the most significant byte only.

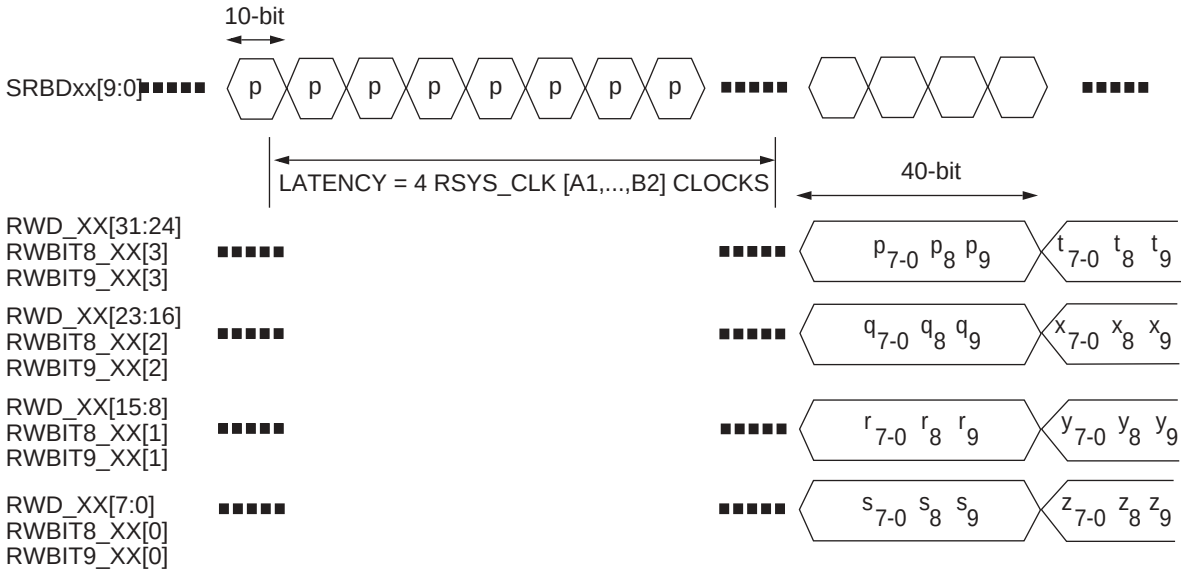
Typically, it is not necessary to set the DOWDALGN_x bit. When the link state machine loses synchronization (DEMUXWAS_x register bit is 0), the DEMUX block automatically looks for a new comma character irrespective of whether the DOWDALGN_x bit is set or not. However, as discussed earlier, the comma character may become misaligned without the Fibre Channel link state machine indicating a loss of synchronization. In such cases, the DOWDALGN_x bit must be toggled to force resynchronization.

The NOWDALGN_x bit is a level-sensitive bit. If it is a “1”, then the DEMUX does not dynamically alter the word boundary based on comma and SWDSYNC_x output of the SERDES. This might be useful if a channel were configured to bypass the multi-channel alignment FIFO and raw 40-bits of data are directed from SERDES to FPGA.

In Fibre Channel mode, the default setting (NOWDALGN_x = 0) causes the word boundary to be set as soon as the SERDES SWDSYNC_x output is a “1” and a comma character has been detected. The character that is the comma becomes the most-significant portion of the demultiplexed word. When the SERDES loses link synchronization it will drop SWDSYNC_x low. The DEMUX will begin search for word alignment as soon as SWDSYNC_x goes to “1” again.

The DEMUX passes on to the channel alignment FIFO block a set of control signals that indicate the location of the synchronizing event. RALIGN_x[3:0] are these indicators. If there is no link synchronization, all of the RALIGN_x[[3:0] bits will be zeros independent of synchronizing events that come in. When the link is synchronized, then the bit that corresponds to the time of the synchronization event will be set to a “1”. The relationship between a time sequence of values input at SRBDx[7:0] to the values output at RWD_x[39:0] is shown in Figure 45. A parallel relationship exists between SRBDx[8] and RWBIT8_x[3:0] as well as between SRBD_x[9] and RWBIT9_x[3:0].

Figure 45. Receive DEMUX Block for a Single SERDES Channel. (NOTE: xx = A, B, C, or D)l



One clock per block of two or four channels, called RCK78, is sent to the FPGA. The control bits RCKSEL[A,B] are used to select the channel that is the source for these clocks.

Link State Machines

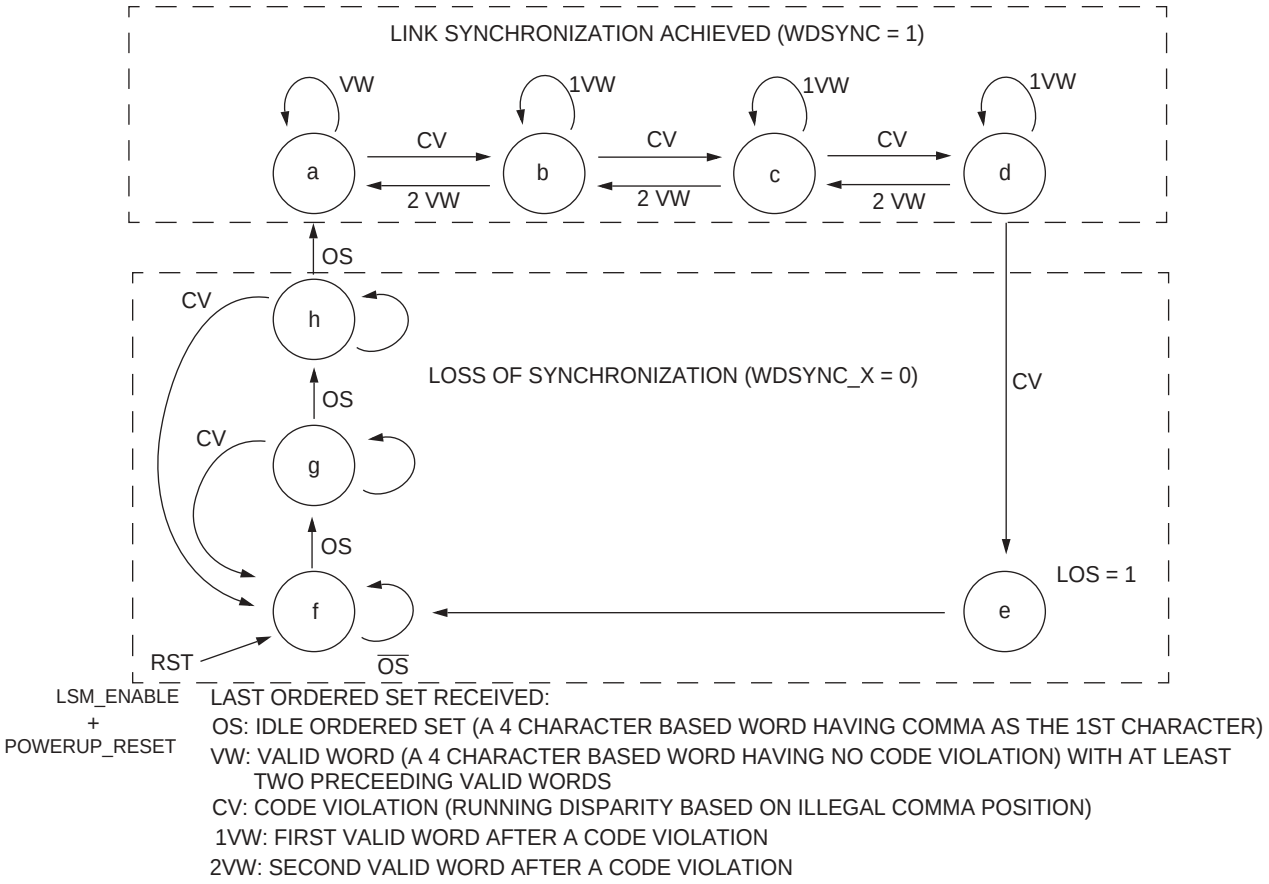
Two link state machines are included in each SERDES channel; one for XAUI applications and a second for Fibre Channel applications.

The Fibre Channel link state machine is responsible for establishing a valid link between the transmitter and the receiver and for maintaining link synchronization. The machine is initially in the loss of synchronization (LOS) state upon power-on reset. This is indicated by WDSYNC_x = 0. While in this state, the machine looks for a particular number of consecutive idle ordered sets without any invalid data transmission in between before declaring synchronization achieved. Achievement of synchronization is indicated by asserting WDSYNC_x = “1”. Specifically, the machine looks for three continuous idle ordered sets without any misaligned comma character or any running disparity based code violation in between. In the event of any such code violation, the machine would reset itself to the

ground state and start its search for the idle ordered sets again. A typical valid sequence for achieving link synchronization would be K28.5 D21.4 D21.5 D21.5 repeated three times.

In the synchronization achieved state, the machine constantly monitors the received data and looks for any kind of code violation that might result due to running disparity errors. If it were to receive four such consecutive invalid words, the link machine loses its synchronization and once again enters the loss of synchronization state (LOS). A pair of valid words received by the machine overcomes the effect of a previously encountered code violation. LOS is indicated by the status of WDSYNC_x output which now transitions from “1” to “0”. LOS is also indicated by DEMUXWAS_x status register bit. This bit is set to “0” during loss of synchronization. At this point the machine attempts to establish the link yet again. Figure 46 shows the state diagram for the Fibre Channel link state machine.

Figure 46. Fibre Channel Link State Machine State Diagram



XAUI Link Synchronization Function

For each lane, the receive section of the XAUI link state machine incorporates a synchronization state machine that monitors the status of the 10-bit alignment. A 10-bit alignment is done in the SERDES based on a comma character such as K28.5. A comma (0011111 or its complement 1100000) is a unique pattern in the 10-bit space that cannot appear across the boundary between any two valid 10-bit code-groups. This property makes the comma useful for delimiting code-groups in a serial stream. This mechanism incorporates a hysteresis to prevent false synchronization and loss of synchronization due to infrequent bit errors. For each lane, the sync_complete signal is disabled until the lane achieves synchronization. The synchronization state diagram is shown in Figure 47. The XAUI state machine does not have any control over the SERDES byte aligner. It is the user’s responsibility to control the byte aligner through software access of register map address 30800.

This state machine is modeled after draft IEEE 802.3ae, version 2.1 but will also operate properly with version 4.1 implementations. Table 28 and Table 29 describe the state variables used in Figure 47.

Note that it takes four idle ordered sets (ex: k28.5, Dx.y, Dx.y, Dx.y) to bring the state machine from a loss-of-sync to a sync_acq'd_1 state. Instead, when back-to-back commas are used, it takes a total of five commas to achieve the same result as with idle ordered sets.

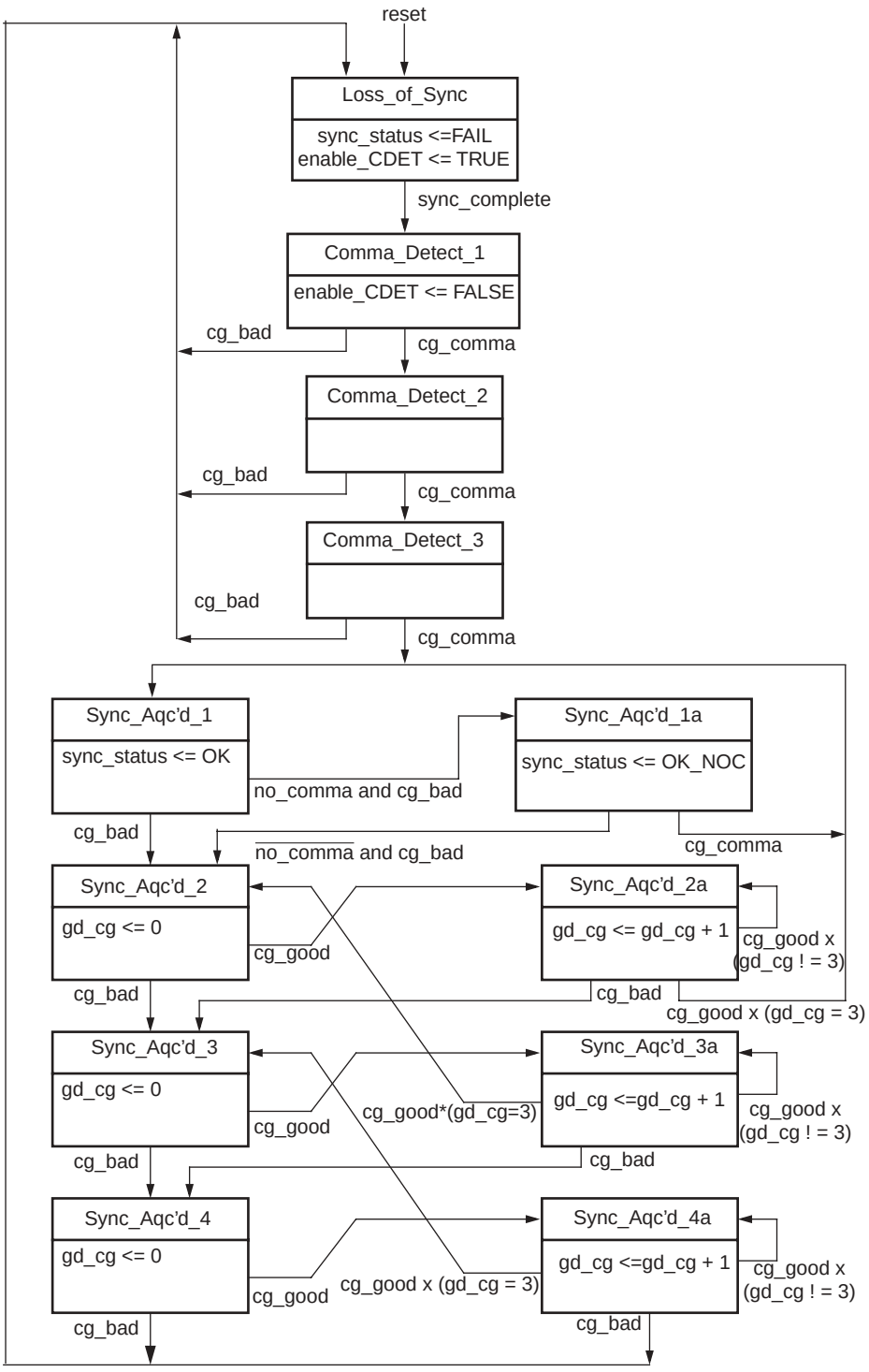
Table 28. XAUI Link Synchronization State Diagram Notation—Variables

Variable	Description
sync_status	FAIL: Lane is not synchronized (correct 10-bit alignment has not been established). OK: Lane is synchronized. OK_NOC: Lane is synchronized but a comma character has not been detected in the past 200 code-groups.
enable_CDET	TRUE: Align subsequent 10-bit words to the boundary indicated by the next received comma. FALSE: Maintain current 10-bit alignment.
gd_cg	Current number of consecutive cg_good indications.

Table 29. XAUI Link Synchronization State Diagram—Functions

Function	Description
sync_complete	Indication that alignment code-group alignment has been established at the boundary indicated by the most recently received comma.
cg_comma	Indication that a valid code-group, with correct running disparity, containing a comma has been received.
cg_good	Indication that a valid code-group with the correct running disparity has been received.
cg_bad	Indication that an invalid code-group has been received.
no_comma	Indication that comma timer has expired. The timer is initialized upon receipt of a comma.

Figure 47. XAUI Link Synchronization State Diagram



Reference Clock Requirements

There is one pair of SERDES reference clock inputs on the ORSPI4. The differential reference clock is distributed to all channels in the block. Each channel has a differential buffer to isolate the clock from the other channels. The input clock is preferably a differential signal; however, the device can operate with a single-ended input. The input reference clock directly impacts the transmit data eye, so the clock should have low jitter. In particular, jitter compo-

nents in the DC to 5 MHz range should be minimized. The required electrical characteristics for the reference clock are given in Table 68.

Note: In sections of this data sheet, the differential clocks are simply referred to as the reference clock as REFCLK.

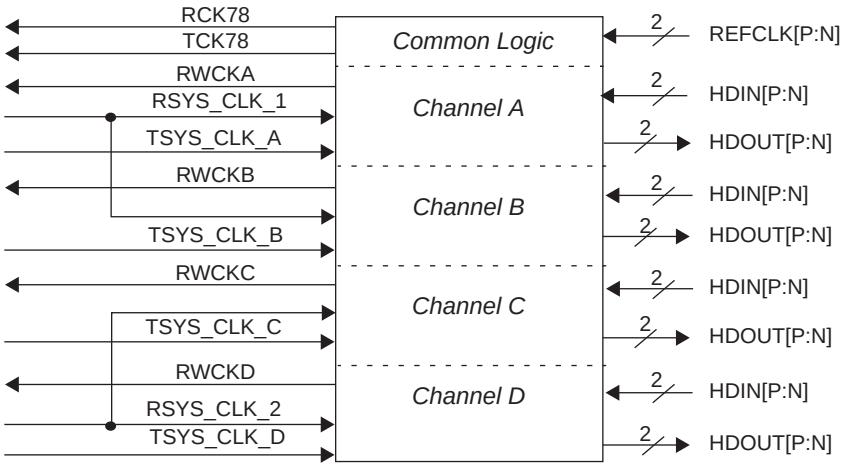
Synthesized and Recovered Clocks
The SERDES block contains its own dedicated PLLs for transmit and receive clock generation. The user provides a reference clock of the appropriate frequency, as described in the previous section. The transmitter PLL uses the REFCLK inputs to synthesize the internal high-speed serial bit clocks. The receiver PLLs extract the clock from the serial input data and retime the data with the recovered clock. The receive PLL for each channel has two modes of operation - lock to reference and lock to data with retiming.

When no data or invalid data is present on the HDINP_x and HDINN_x pins, the receive VCO will not lock to data and its frequency can drift outside of the nominal ±350 ppm range. Under this condition, the receive PLL will lock to REFCLK for a fixed time interval and then will attempt to lock to receive data. The process of attempting to lock to data, then locking to clock will repeat until valid input data exists. There is also a control register bit per channel to force the receive PLL to always lock to the reference clock.

The high-speed transmit and receive serial data links can run at 0.6 to 3.7 Gbits/s, depending on the frequency of the reference clock and the state of the control bits from the internal transmit control register. The interface to the serializer/deserializer block runs at 1/10th the bit rate of the data lane. Additionally, the MUX/DEMUX logic converts the data rate and bit-width so the FPGA core can run at 1/4th this frequency, which gives a range of 15 to 92.5 MHz for the data in and out of the FPGA.

Internal Clock Signals at the FPGA/Core SERDES Interface
There are several clock signals defined at the FPGA/SERDES interface in addition to the external reference clock. All of the SERDES clock signals are shown in Figure 48 and are described following the paragraphs.

Figure 48. SERDES Clock Signals (High-Speed Serial I/O Also Shown)



REFCLKP, REFCLKN:
These are the differential reference clocks provided to the ORSPI4 device as described earlier. They are used as the reference clock for both TX and RX paths. For operation of the serial links at 3.125 Gbits/s, the reference clocks will be at a frequency of 156.25 MHz.

RWCK[A:D]:
These are the low-speed receive clocks from the SERDES to the FPGA across the core-FPGA interface.

These are derived from the recovered low-speed complementary clocks from the SERDES blocks. RWCK_A belongs to Channel A, RWCK_B belongs to channel B and so on. With a reference clock input of 156.25 MHz, these clocks operate at 78.125 MHz.

RCK78:
These are muxed outputs of RWCKA[A:D] and RWCKB[B:D] respectively. With a reference clock input of 156.25 MHz, these clocks operate at 78.125 MHz.

RSYS_CLK_[1:2]
These clocks are inputs to the SERDES. These are used by each channel as the read clock to read received data from the alignment FIFO within the SERDES. To guarantee that there is no overflow in the alignment FIFO, it is an absolute requirement that the write and read clocks be frequency locked to 0 ppm. Examples of how to achieve this are shown in the section on recommended board-level clocking.

TCK78:
This is a muxed output from the core to the FPGA across the core-FPGA interface of one of the four transmit SERDES clocks operating at up to 92.5 MHz in the SERDES.

TSYS_CLK[A:D]:
These clocks are inputs to the SERDES from the FPGA. These are used by each channel to control the timing of the Transmit Data Path. To guarantee correct transmit operation, these clocks must be frequency locked within 0 ppm to TCK78.

List of Operating Modes SERDES

Transmit and Receive Clock Rates
Table 30 shows typical relationship between the data rates, the reference clock, the transmit TCK78 clock and the receive RCK78 clock. The selection of full-rate or half-rate for a given reference clock speed is set by bits in the transmit and receive control registers, and can be set on a per-channel basis.

Table 30. Transmit Data and Clock Rates

Data Rate	Reference Clock	TCK78[A: B] and RCK78[A:B] Clocks	Rate of Channel Selected as Clock Source
0.6 Gbits/s	60 MHz	15 MHz	Half
1.0 Gbits/s	100 MHz	25 MHz	Half
1.25 Gbits/s	125 MHz	31.25 MHz	Half
2.0 Gbits/s	100 MHz	50 MHz	Full
2.5 Gbits/s	125 MHz	62.5 MHz	Full
3.125 Gbits/s	156 MHz	78 MHz	Full
3.7 Gbits/s	185 MHz	92.5 MHz	Full

Mixing Half-rate, Full-rate Modes
When channel alignment is enabled, all receive channels within an alignment group should be configured at the same rate. For example, channels A and B, can be configured for twin alignment and full-rate mode, while channels C and D can be configured for half-rate mode. In quad alignment mode, all four channels must be configured in either half or full-rate mode. Register settings for multi-channel alignment are described in a later section.

Besides taking in a TSYS_CLK_x from the FPGA logic for each channel, the transmit path logic sends back a clock of the same frequency, but arbitrary phase. This clock, TCK78, is derived from the MUX block of one of the 2 channels in its SERDES block. The MUX blocks provide the potential source for TCK78 by a divide-by-4 of the SERDES STBC311xs clock used in synchronizing the transmit data words in the STBC311x clock domain. The STBC311x clocks are internal to the core and are not brought across the core/FPGA interface

The receiver section receives high-speed serial data at its differential CML input port and sends it in to the Clock and Data Recovery (CDR) block. The CDR block then generates a recovered clock (RWCKx) and retimes the data. Thus, the recovered receive clocks are asynchronous between channels.

Transmit Clock Source Selection

The TCKSEL[A:B] bit select the source channel of TCK78. The selection of the source for TCK78 is controlled by this bit as shown in Table 31.

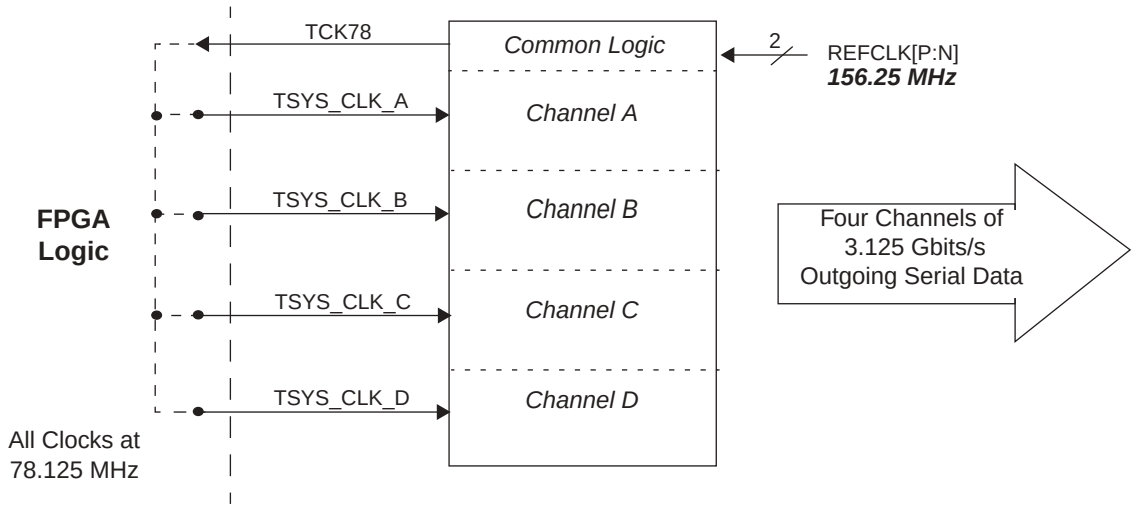
Table 31. TCK78 Source Selection

TCKSEL0	TCKSEL1	Clock Source
0	0	Channel A
1	0	Channel B
0	1	Channel C
1	1	Channel C

Recommended Transmit Clock Distribution for the ORSPI4

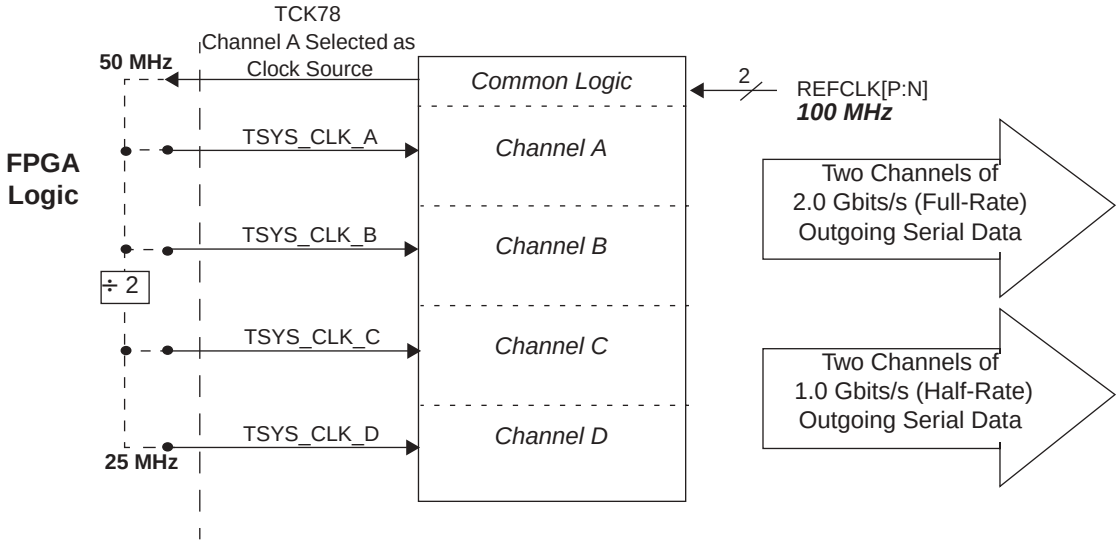
As an example of the recommended clock distribution approach, TSYS_CLK_[A:D] can be sourced by TCK78 as shown in Figure 49 if the transmit line rate are common for all four channels in a quad.

Figure 49. Transmit Clocking for a Single Block



If the transmit line rate is mixed between half and full rate among the channels, then the scheme shown in Figure 50 can be used. The figure shows TSYS_CLK_A and TSYS_CLK_B being sourced by TCK78 and TSYS_CLK_C and TSYS_CLK_D being sourced by TCK78/2 (the division is done in FPGA logic). Similar clocking would be used for Quad B.

Figure 50. Mixed Rate Transmit Clocking for a Single Block



Receive Clock Source Selection and Recommended Clock Distribution

In the receive path, one clock per bank of four channels, called RCK78, is sent to the FPGA logic. The control register bits RCKSEL[0:1] are used to select the clock source for these clocks. The selection of the source for RCK78 is controlled by these bits as shown in Table 32.

Table 32. RCK78 Source Selection

RCKSEL0	RCKSEL1	Clock Source
0	0	Channel A
1	0	Channel B
0	1	Channel C
1	1	Channel C

In the receive channel alignment bypass mode, the data and recovered clocks for the four channels are independent. The data for each channel are synchronized to the recovered clock from that channel.

Figure 51. Receive Clocking for a Single Quad

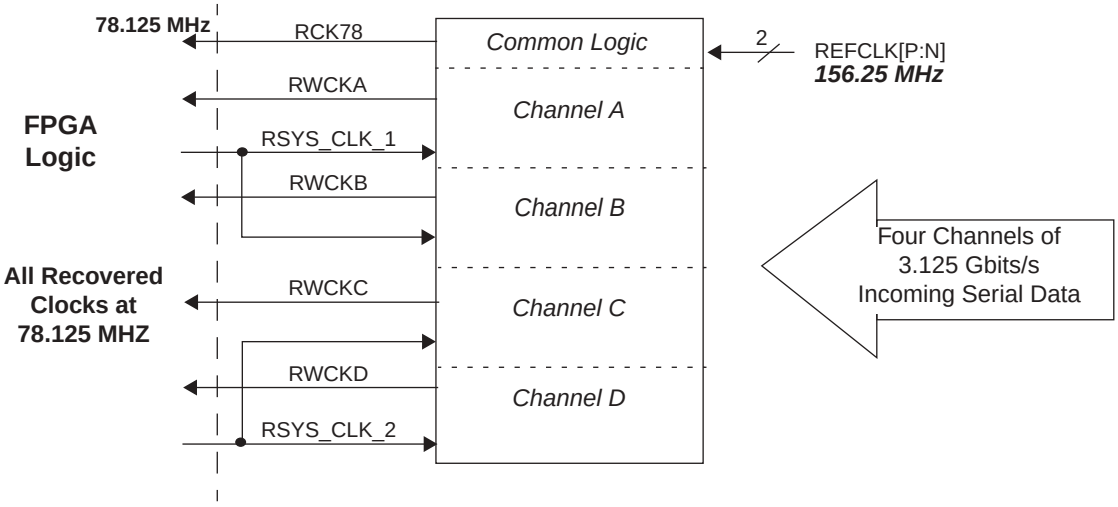
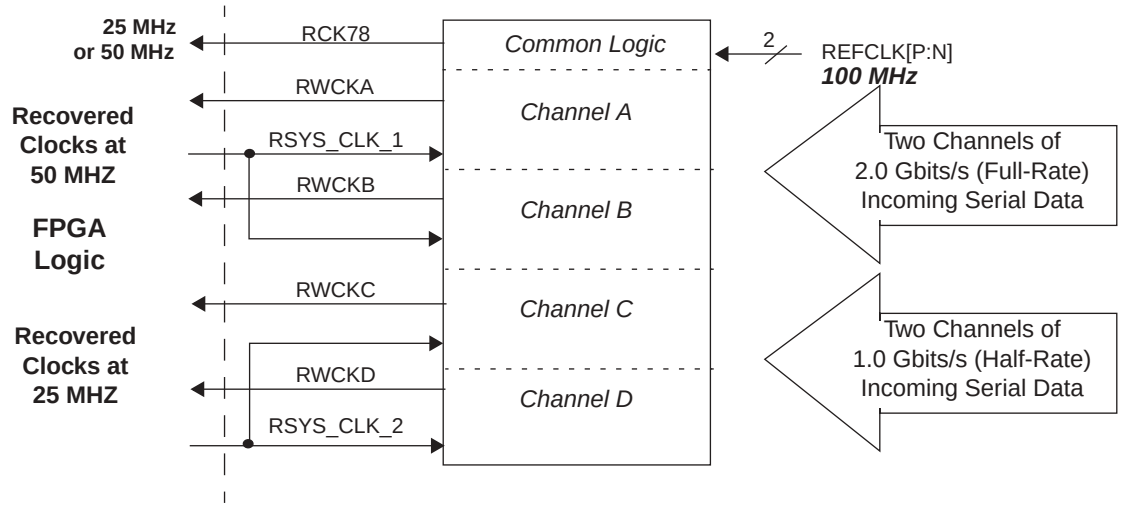


Figure 52. Receive Clocking for Mixed Line Rates



Multichannel Alignment

The alignment FIFO allows the transfer of all data to the system clock. The Multi-Channel Alignment block (Figure 43) allows the system to be configured to allow the frame alignment of multiple, slightly varying, data streams. This optional alignment ensures that matching SERDES streams will arrive at the FPGA in perfect data synchronization.

Each channel is provided with a 24 word x 36-bit FIFO. The FIFO can perform two tasks: (1) to change the clock domain from receive clock to a clock on the FPGA side, and (2) to align the receive data over 2, 4, or 8 channels. This FIFO allows a timing budget of +/- 230.4 ns that can be allocated to skew between the data lanes and for transfer to the system clock. The input to the FIFO consists of 36-bits of demultiplexed data, RALIGN_x[3:0], RWD_x[31:0], and RWBIT8_x[3:0].

The four RALIGN_x bits are control signals, and can be the alignment character detect signals, indicating the presence of a comma character in Fibre Channel mode and the /A/ character in XAUI mode. The other 32 RWD_x bits are the 8-bit data bytes from the 8b/10b decoder. The alignment character, if present, is the MSB of the data. The

RWBIT8_x indicates the presence of a Km.n control character in the receive data byte. Only RWBIT8_x and RWD_x inputs are stored in the FIFO. During alignment process, RALIGN[3]_x is used to synchronize multiple channels.

If a channel is not in any alignment group, it will set the FIFO-write-address to the beginning of the FIFO, and will set the FIFO-read-address to the middle of the FIFO, at the first assertion of RALIGN[3]_x after reset or after the resync command.

The ORSPI4 has a total of four SERDES channels. The incoming data of these channels can be synchronized in several ways or can be independent of one another. Two SERDES channels can be aligned together. Channel A and B and/or channel C and D can form a pair as shown in Figure 53. Alternately, all four SERDES channels can be aligned together to form a communication channel with a bandwidth of 10 Gbits/s as shown in Figure 54. Individual channels within an alignment group can be disabled (i.e., powered down) without disrupting other channels.

Figure 53. Twin Channel Alignment

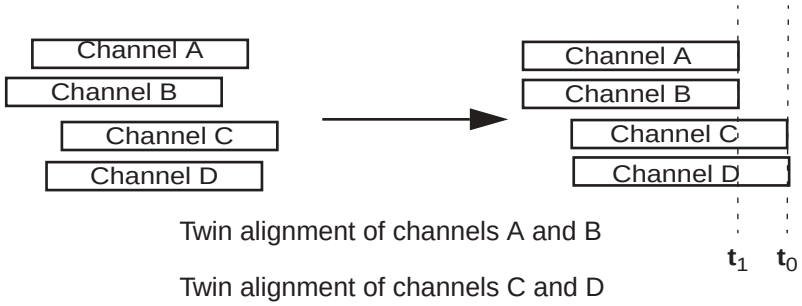
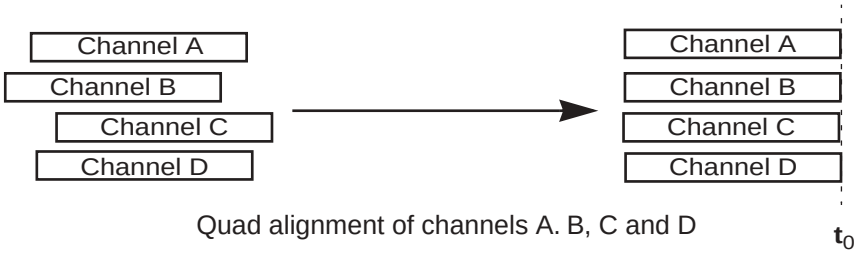


Figure 54. Alignment of SERDES Quad (4-Channels)

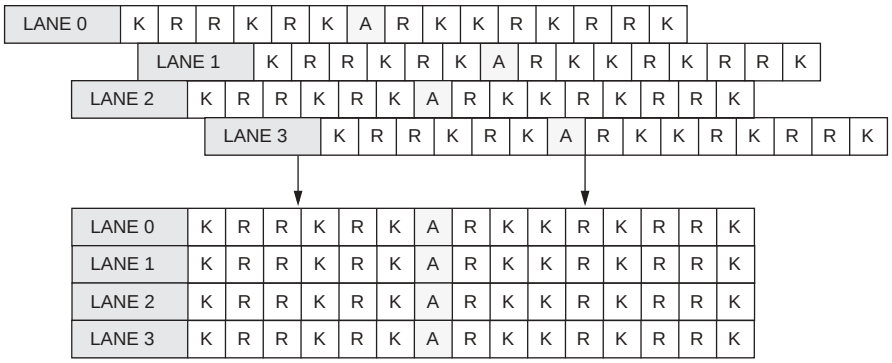


For every alignment group, there are both an OVFL and an OOS status register bit. The OVFL bit is set when alignment FIFO overflow occurs. The OOS bit is flagged when the down counter in the synchronization algorithm has reached a value of 0 and alignment characters from all channels within an alignment group have not been received.

XAUI Lane Alignment Function (Lane Deskew)

In XAUI mode, the receive section in each lane uses the /A/ code group to compensate for lane-to-lane skew. The mechanism restores the timing relationship between the four lanes by lining up the /A/ characters into a column. Figure 55 shows the alignment of four lanes based on /A/ character. A minimum spacing of 16 code-groups implies that at least ± 80 bits of skew compensation capability should be provided, which the ORSPI4 significantly exceeds.

Figure 55. Deskew Lanes by Aligning /A/ Columns



SERDES/FPGA Interface

This block provides the data formatting and receive data and clock signal transfers between the SERDES and the FPGA Logic. There are also control and status registers in the FPGA portion of the chip, which contain bits to control the receive logic and to record status. These are described in later sections of this Data Sheet and communicate with the core using the System Bus.

The demultiplexed, receive word outputs to the FPGA are shown in Figure 43. These are each 40 bits wide. There are four of these interfaces, one for each SERDES channel. Each consist of four groups of 10-bit data or four groups of decoded information depending on setting of 8b10bR_x control register bits.

Each 10-bit group of decoded information includes 8 bits of data and a one-bit K_CTRL indicator derived from the received data, and a tenth bit of status information. The function of the tenth bit varies from group to group and includes code violation, out of synchronization (OOS) indicators and the CH248_SYNC_x status bit. CH248_SYNC_x indicates the status of multi-channel alignment of channel x and is high when the count for the multi-channel alignment block reaches zero, regardless of whether or not multi-channel alignment is successful. The mapping of the 10-bit groups to the MRWD_x[39:0] bits output to the FPGA logic is summarized in Table 33 and Table 34. The various functions of the bits that vary from channel to channel, i.e., bits 29 and 19, are also described in Table 34.

Table 33. Definition of Bits of MRWDx[39:0]

Bit Index	8b10bR=0	8b10bR=1	
	NOCHALGN[A:B]=1 CV_SELx=0	NOCHALGN[A:B]=1 CV_SELx=1	NOCHALGN[A:B]=0 CV_SELx=1
39	bit 9 of 10-bit data 3	CV_A3, code violation, byte 3	CH24_SYNCx
38	bit 8 of 10-bit data 3	K_CTRL for byte 3	K_CTRL for byte 3
37	bit 7 of 10-bit data 3	bit 7 of byte3	bit 7 of byte3
36	bit 6 of 10-bit data 3	bit 6 of byte 3	bit 6 of byte 3
35	bit 5 of 10-bit data 3	bit 5 of byte 3	bit 5 of byte 3
34	bit 4 of 10-bit data 3	bit 4 of byte 3	bit 4 of byte 3
33	bit 3 of 10-bit data 3	bit 3 of byte 3	bit 3 of byte 3
32	bit 2 of 10-bit data 3	bit 2 of byte 3	bit 2 of byte 3
31	bit 1 of 10-bit data 3	bit 1 of byte 3	bit 1 of byte 3
30	bit 0 of 10-bit data 3	bit 0 of byte 3	bit 0 of byte 3
29	bit 9 of 10-bit data 2	CV_A2, code violation, byte 2	See *Table 34*
28	bit 8 of 10-bit data 2	K_CTRL for byte 2	K_CTRL for byte 2
27	bit 7 of 10-bit data 2	bit 7 of byte 2	bit 7 of byte 2
26	bit 6 of 10-bit data 2	bit 6 of byte 2	bit 6 of byte 2

Bit Index	8b10bR=0	8b10bR=1	
	NOCHALGN[A:B]=1 CV_SELx=0	NOCHALGN[A:B]=1 CV_SELx=1	NOCHALGN[A:B]=0 CV_SELx=1
25	bit 5 of 10-bit data 2	bit 5 of byte 2	bit 5 of byte 2
24	bit 4 of 10-bit data 2	bit 4 of byte 2	bit 4 of byte 2
23	bit 3 of 10-bit data 2	bit 3 of byte 2	bit 3 of byte 2
22	bit 2 of 10-bit data 2	bit 2 of byte 2	bit 2 of byte 2
21	bit 1of 10-bit data 2	bit 1 of byte 2	bit 1 of byte 2
20	bit 0 of 10-bit data 2	bit 0 of byte 2	bit 0 of byte 2
19	bit 9 of 10-bit data 1	CV_A1, code violation, byte 1	See *Table 34*
18	bit 8 of 10-bit data 1	K_CTRL for byte 1	K_CTRL for byte 1
17	bit 7 of 10-bit data 1	bit 7 of byte 1	bit 7 of byte 1
16	bit 6 of 10-bit data 1	bit 6 of byte 1	bit 6 of byte 1
15	bit 5 of 10-bit data 1	bit 5 of byte 1	bit 5 of byte 1
14	bit 4 of 10-bit data 1	bit 4 of byte 1	bit 4 of byte 1
13	bit 3 of 10-bit data 1	bit 3 of byte 1	bit 3 of byte 1
12	bit 2 of 10-bit data 1	bit 2 of byte 1	bit 2 of byte 1
11	bit 1 of 10-bit data 1	bit 1 of byte 1	bit 1 of byte 1
10	bit 0 of 10-bit data 1	bit 0 of byte 1	bit 0 of byte 1
09	bit 9 of 10-bit data 0	CV_A0, code violation, byte 1	VL (connected to ground)
08	bit 8 of 10-bit data 0	K_CTRL for byte 0	K_CTRL for byte 0
07	bit 7 of 10-bit data 0	bit 7 of byte 0	bit 7 of byte 0
06	bit 6 of 10-bit data 0	bit 6 of byte 0	bit 6 of byte 0
05	bit 5 of 10-bit data 0	bit 5 of byte 0	bit 5 of byte 0
04	bit 4 of 10-bit data 0	bit 4 of byte 0	bit 4 of byte 0
03	bit 3 of 10-bit data 0	bit 3 of byte 0	bit 3 of byte 0
02	bit 2 of 10-bit data 0	bit 2 of byte 0	bit 2 of byte 0
01	bit 1 of 10-bit data 0	bit 1 of byte 0	bit 1 of byte 0
00	bit 0 of 10-bit data 0	bit 0 of byte 0	bit 0 of byte 0

Table 34. Definition of the Status Bits of MRWDx that Vary for Different Channels

Channel Index	Bit Index	Name	Description
all	39	CH24_SYNCx	Multi-channel alignment attempt complete if 1
A	29	CV_A_OR	Code violation in one or more of the received 10-bit groups for channel A
A	19	SYNC2_1_OOS	Dual channel synchronization of channels A and B not successful if 1
B	29	CV_B_OR	Code violation in one or more of the received 10-bit groups for channel B
B	19	SYNC4_OOS	Quad channel synchronization of SERDES not successful if 1
C	29	CV_C_OR	Code violation in one or more of the received 10-bit groups for channel C
C	19	SYNC2_2_OOS	Dual channel synchronization of channels C and D not successful if 1
D	29	CV_AD_OR	Code violation in one or more of the received 10-bit groups for channel D

The code violation signals will only be valid if the corresponding CV_SELx = “1”. (If 8b10bR=0, CV_SEL should also be zero. The CV_x_OR signals are obtained by ORing four code violation signals from the 1:4 DEMUX block. These are primarily indicators of received signal quality, since a single code violation will not force a loss of sync

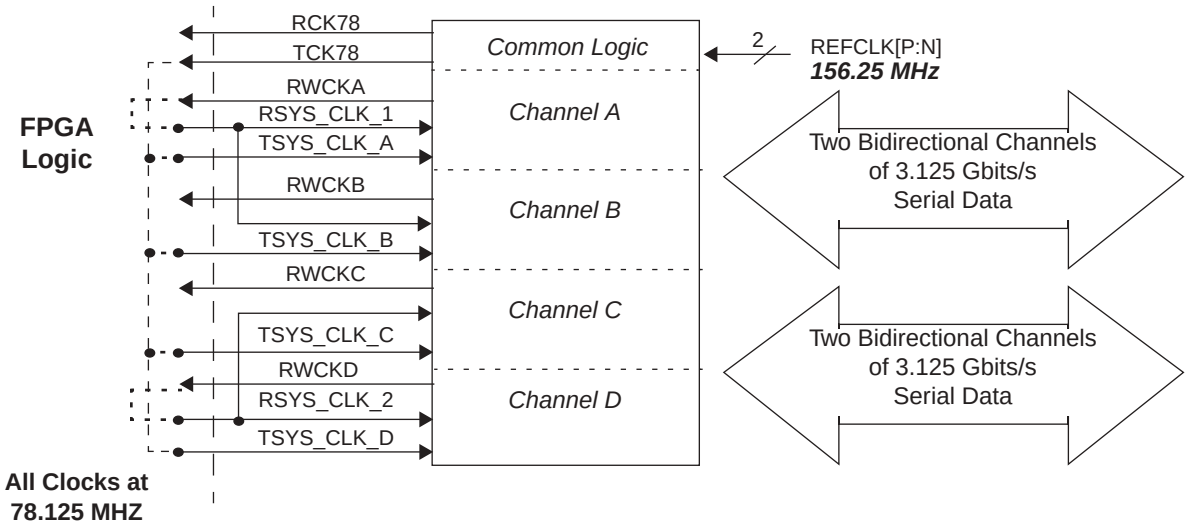
(LOS) state in the word alignment state machines. Since these signals come from the DEMUX block, if multi-channel alignment is enabled, the code violation signals correspond to data that must still be multi-channel aligned. Hence these signals provide advance notification of detected violations in data that will appear at the core/FPGA interface several clock cycles later. The exact number of clock cycles that the data is delayed depends on the skew between the incoming data for the different channels.

Multi-Channel Alignment Clocking Strategies for the ORSPI4

The data on the four channels SERDES can be independent of each other or can be synchronized in several ways. For example, two channels within the SERDES can be aligned together; channel A and B and/or channel C and D. Alternatively, all four channels in a SERDES quad can be aligned together to form a communication channel with a bandwidth of 10 Gbits/s. Individual channels within an alignment group can be disabled (i.e., powered down) without disrupting other channels. Clocking strategies for these various modes are described in the following paragraphs.

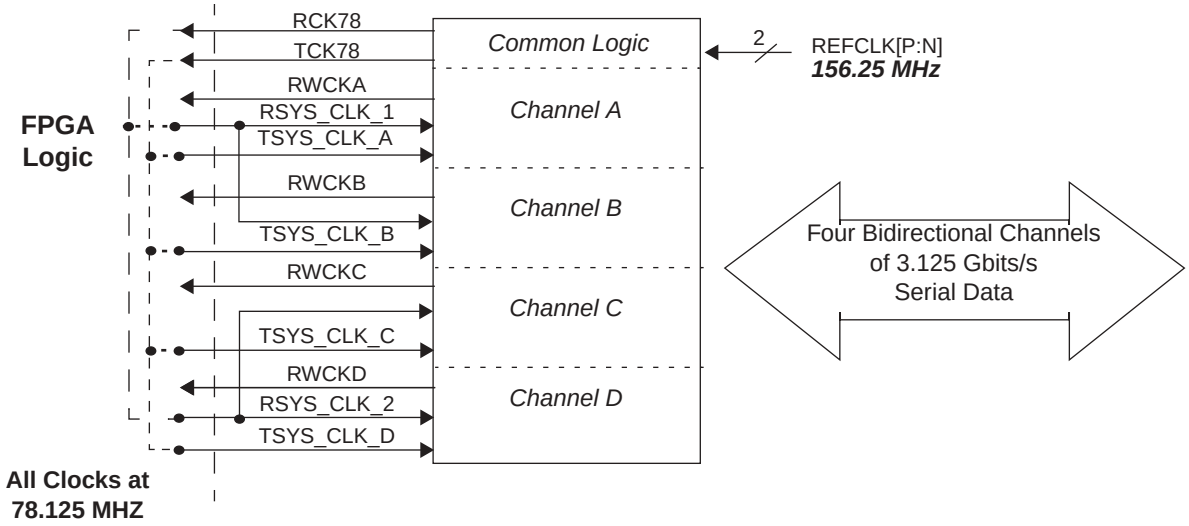
For dual alignment, both twins within a quad can be sourced by clocks that are different from the other channels, however each pair of SERDES must have the same clock. The channel pair A and B is driven on the low-speed side by RSYS_CLK_1 and the channel pair C and D are driven on the low-speed side by RSYS_CLK_2. Either RWCKA or RWCAB can be connected to RSYS_CLK_1, and either RWCKC or RWCKD can be connected to RSYS_CLK_2. A clocking example for dual alignment is shown in Figure 56.

Figure 56. Receive Clocking for a Dual Alignment in a Single Quad



For receive quad alignment, RSYS_CLK_1 and RSYS_CLK_2 can be tied together as shown in Figure 57.

Figure 57. Clocking for a Quad Receive Clocking for a Dual Alignment in a Single Quad



Multi-channel Alignment Configuration

Register settings for multi-channel alignment are shown in Table 35.

Table 35. Multi-Channel Alignment Modes

Register Bits FMPU_SYNCMODE_xx[0:1]	Mode
00	No multi-channel alignment
10	Twin channel alignment
01	Quad channel alignment

To align all 4 channels:

- FMPU_SYNMODE_[A:D] = 01
- FMPU_SYNMODE_B[A:D] = 11

To align two channels:

- FMPU_SYNMODE_[A:B] = 10 for channel A and B
- FMPU_SYNMODE_[C:D] = 10 for channel C and D

To enable/disable synchronization signal of individual channel within a multi-channel alignment group:

- FMPU_STR_EN_x = “1” enabled
- FMPU_STR_EN_x = “0” disabled where x is one of [A:D].

To resynchronize a multi-channel alignment group set the following bit to zero, and then set it to one:

- FMPU_RESYNC4 for quad [A:D]
- FMPU_RESYNC2_1 for twin channel [A:B]

- FMPU_RESYNC2_2 for twin channel [C:D]

To resynchronize an independent channel (resetting the write and the read pointer of the FIFO) set the following bit to zero, and then set it to one:

- FMPU_RESYNC1_x

SERDES Multi-Channel Alignment Sequence

1. Follow steps 1 and 2 in the start up sequence described in a later section.

2. Initiate a SERDES software reset by setting the SWRST bit to “1” and then to “0”. Note that, any changes to the SERDES configuration bits should be followed by a software reset.

3. Wait for 3 ms. REFCLK should be toggling by this time. During this time, configure the following registers.

Set the following bit in register 30820
   – XAUI_MODE_x - set to “1” for XAUI mode or keep the default value of 0 if the Fibre Channel state machine was selected.
   – Enable channel alignment by setting FMPU_SYNMODE bits in registers 30811
   – FMPU_SYNMODE_x. Set to appropriate values for 2, or 4 alignment based on Table 35.
   – Set RCLKSEL and TCKSEL bits in registers 30821.
   – RCKSEL - Choose clock source for 78 MHz RCK78 (Table 32).
   – TCKSEL - Choose clock source for 78 MHz TCK78 (Table 31). Send data on serial links.

Monitor the following status/alarm bits:
   – Monitor the following alarm bits in registers 30000, 30010, 30020, 30030,.
   – LKI-PLL_x lock indicator. A “1” indicates that PLL has achieved lock.

Monitor the following status bits in registers 30804:
   – XAUISTAT_x - In XAUI mode, they should be 10.
   – Monitor the following status bits in registers 30805
   – DEMUXWAS_x - They should be “1” indicating word alignment is achieved.
   – CH248_SYNCx - They should be “1” indicating channel alignment. This is cleared by resync.

4. Write a “1” to the appropriate resync register 30820. Note that this assumes that the previous value of the resync bits are “0”. The resync operation requires a rising edge. Two writes are required to the resync bits: write a “0” and then write a “1”. It is highly recommended to precede a resync with a word alignment, especially in situations where a disturbance in the receive SERDES path can cause misalignment of data and OOS indications without bringing the FC/XAUI state machine to a loss of sync state. A word alignment is achieved by writing a “0” and then a “1” to the appropriate DOWDALIGNx bits in registers 30810.

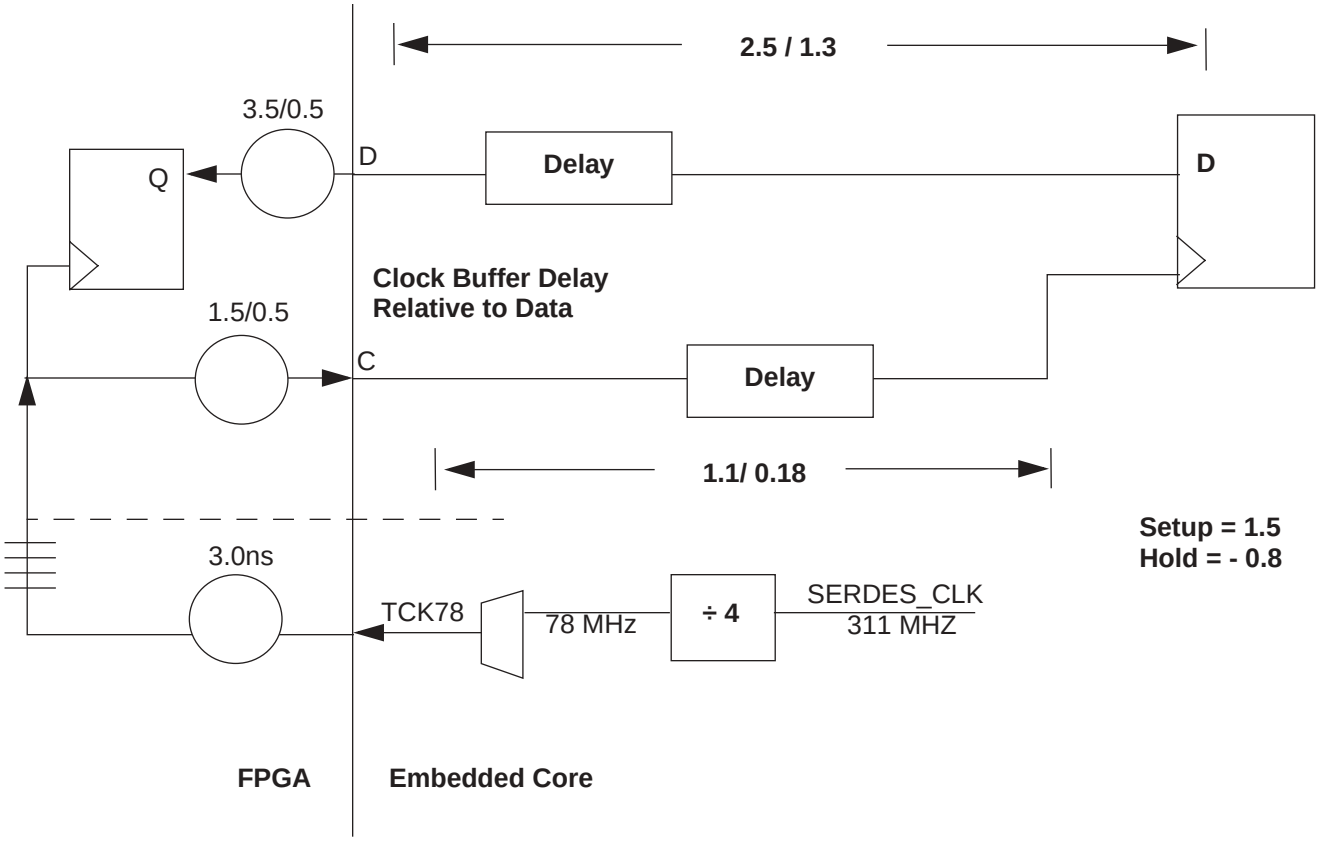
Clocking Schemes and Timing Diagrams- SERDES

Figure 58 shows a timing diagram for the transmit section of the SERDES interface. Setup and Hold numbers at the interface are also show.

Similar timing diagrams are also shown for the receive SERDES interface section in both channel alignment mode (Figure 59), and channel bypass mode (Figure 60).

SERDES Align Mode Interface

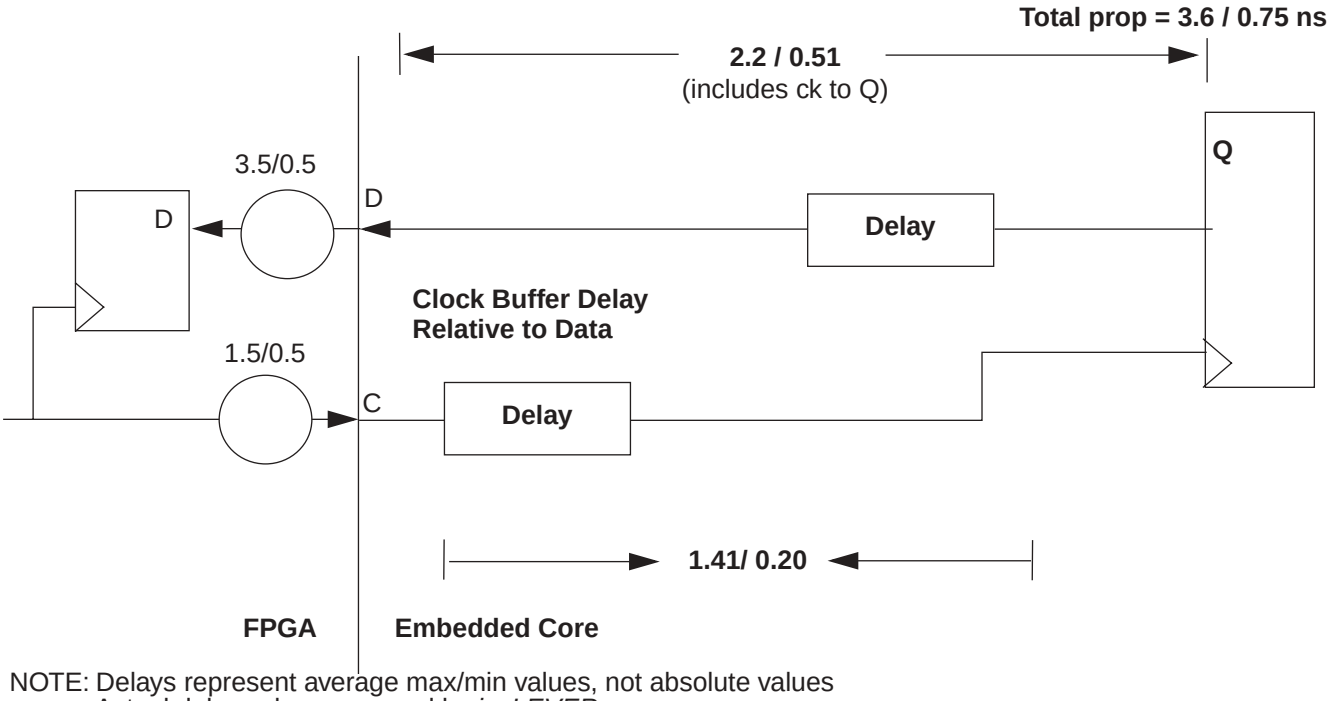
Figure 58. Data: TBIT9[A:D][3:0], TCOMMA[A:D][3:0], TWD[A:D][31:0]
Clock: TSYS_CLK[A:D]



NOTE: Delays represent average max/min values, not absolute values
Actual delay values are used by *ispLEVER*

SERDES Receive Interface in Channel Alignment Mode

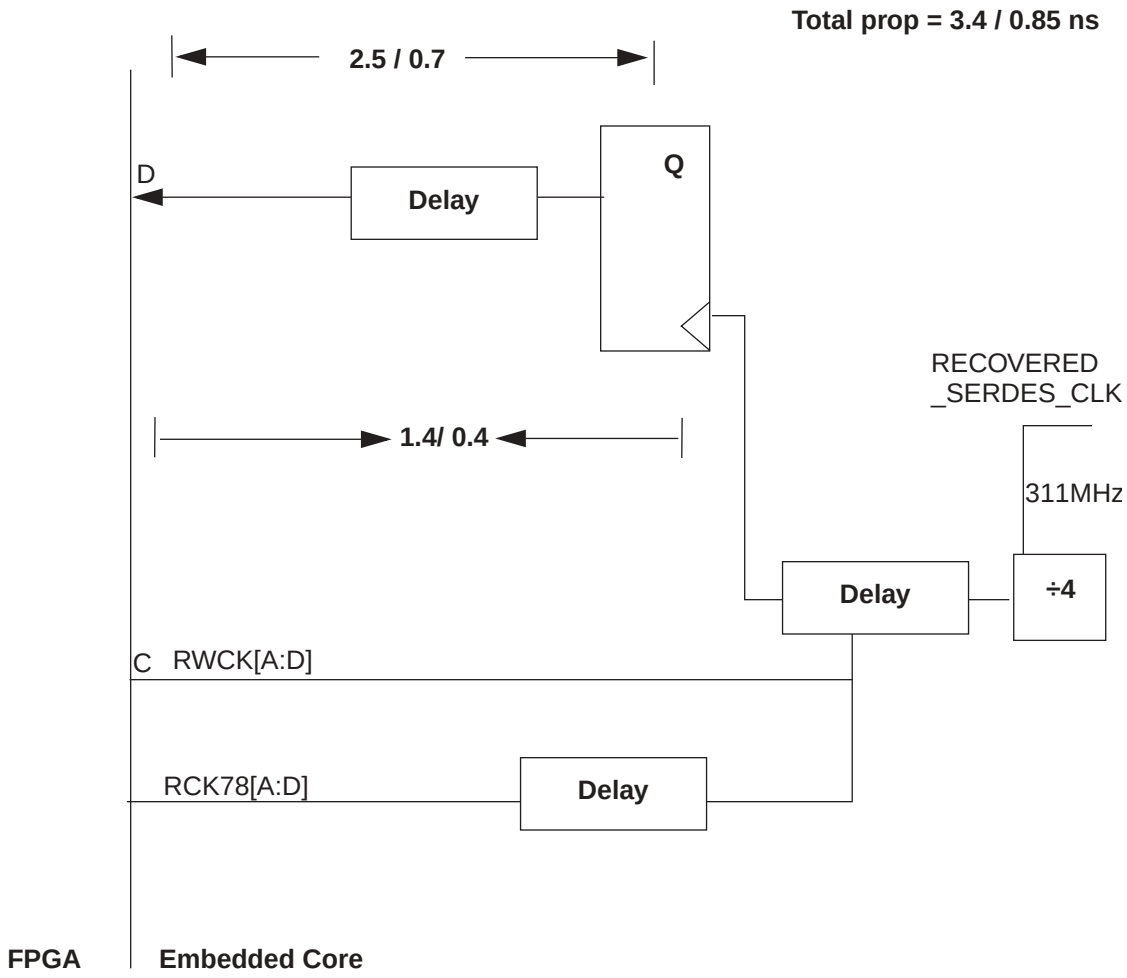
Figure 59. Data: MRWD[A:D][39:0]
Clock: RSYS_CLK_1, RSYS_CLK_2



NOTE: Delays represent average max/min values, not absolute values
Actual delay values are used by *ispLEVER*

SERDES Receive Interface in Channel Bypass Mode

Figure 60. Data: MRWD[A:D][39:0]
Clock: RWCK[A:D]



NOTE: Delays represent average max/min values, not absolute values
Actual delay values are used by *ispLEVER*

Test Modes

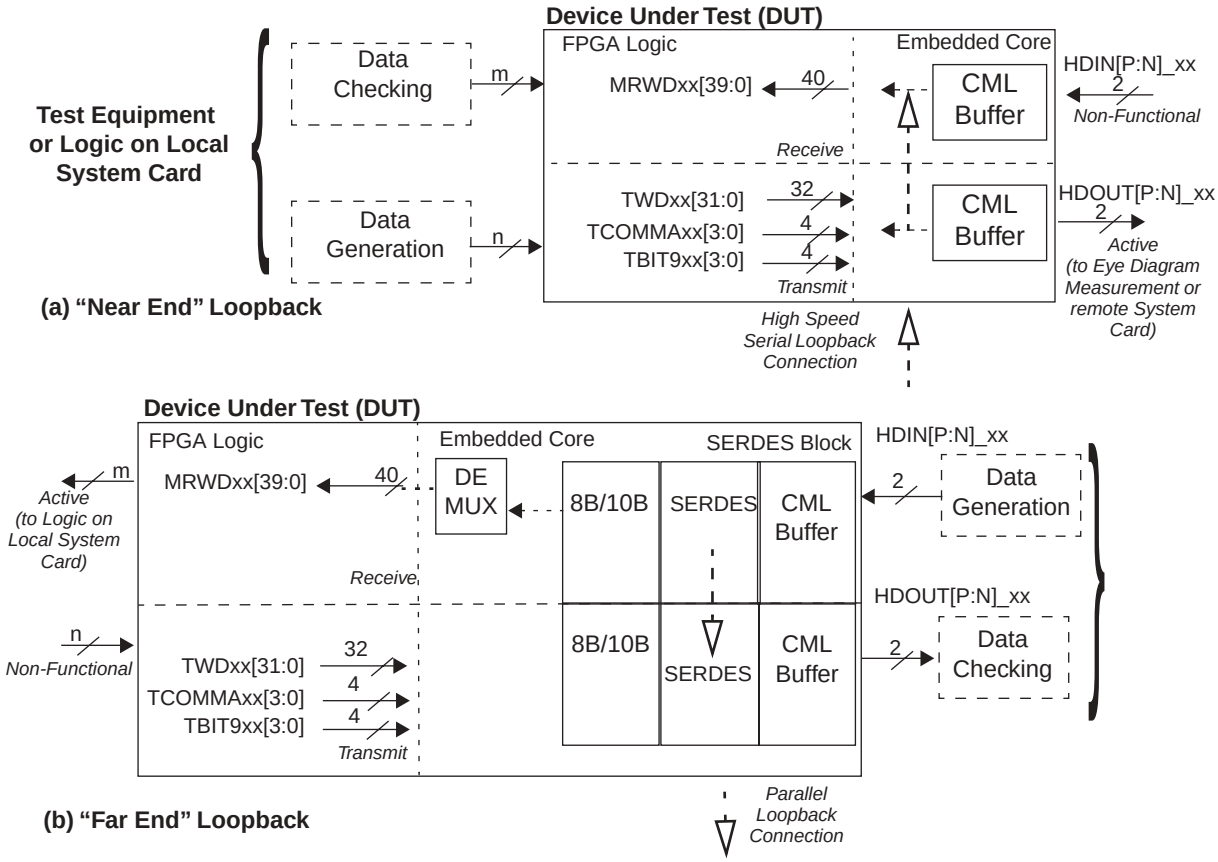
In addition to the operational logic described in the preceding sections, the SERDES contains logic to support various test modes - both for device validation and evaluation and for operating system level tests. The following sections discuss two of the test support logic blocks, supporting various loopback modes, in addition to SERDES characterization pins.

Loopback Testing

Loopback testing is performed by looping back (either internal to the SERDES, by configuring the FPGA logic or by external connections) transmitted data to the corresponding receiver inputs, or received data to the transmitter output. The loopback path may be either serial or parallel.

In general, loopback tests can be classified as “near end” or “far end”. In “near end” loopback (Figure 61(a)), data is generated and checked locally, i.e. by logic on, or connection of, test equipment to the same card as the FPSC. In “far end” loopback (Figure 61(b)), the generating and checking functions are performed remotely, either by test equipment or a remote system card.

Figure 61. "Near End" vs. "Far End" Loopback. (NOTE: xx = A, B, C, or D)



The loopback mode can also be characterized by the physical location of the loopback connection. There are two possible loopback modes supported by the SERDES logic:

- High-speed serial loopback at the CML buffer interface (near end)
- Parallel loopback at the SERDES boundary (far end)

These two loopback modes are described in more detail in the following sections. As noted earlier, other specialized loopback modes can be obtained by configuration of the FPGA logic or by connections external to the FPSC.

High-Speed Serial Loopback at the CML Buffer Interface

The high-speed serial loopback mode has the serial transmit signals looped back internally to the serial receive circuitry. The internal loopback path is from the input connection to the transmit CML buffer to the output connection from the receive CML buffer. The data are sourced on the TWDx[31:0], TCOMMAx[3:0] and TBIT9x[3:0] signal lines and received on the MRWDx[39:0] signal lines. The serial loopback path does not include the high-speed input and output buffers. If TESTEN_x is set, the HDOUTP_x and HDOUTN_x outputs are active in this mode while the CML input buffers are powered down. The device is otherwise in its normal mode of operation. This mode is normally used for tests where the data source and destination are on the same card and is the basic loopback path shown in Table 36.

The data rate selection bits, TXHR and RXHR, in the channel configuration registers must be configured to carry the same value. Table 36 summarize the settings of the control interface register configuration bits for high-speed serial loopback.

Table 36. High-Speed Serial Loopback Configuration Bit Definitions for the SERDES

Register Address	Bit Value	Bit Name	Comments
30002, 30012, 30022, 30032	Bit 0 = 0 or 1	TXHR	Set to “0” or “1”. TXHR and RXHR bits must be set to the same value.
	Bit 7 = 0 or 1	8B10BT	Set to “0” or “1”. If set to “0”, the 8b/10b encoder is excluded from the loopback path. The 8b/10b encoder and decoder selection control bits must both be set to the same value.
30801	Bit 0 = 1 (Channel A) Bit 1 = 1 (Channel B) Bit 2 = 1 (Channel C) Bit 3 = 1 (Channel D)	LOOPENB_xx	Set any of the bits 0-3 to “1” to do serial loopback on the corresponding channel.* The high-speed serial outputs will not be active.

*This test mode can also be set using TESTEN_xx in place of LOOPENB_xx. In that case, Test Mode must be set to 00000.

Parallel Loopback at the SERDES Boundary

In this parallel loopback, differential data are received at the HDINP_x and HDINN_x pins and are retransmitted at the HDOUTP_x and HDOUTN_x pins. The loopback path is at the interface between the SERDES blocks and the MUX and DEMUX blocks and uses the parallel 10-bit buses at these interfaces (see Figure 61b). The loopback connection is made such that the input signals to the TX SERDES block is the same as the output signals from the RX SERDES block. In this parallel loopback mode, the MRWDx[39:0] signal lines remain active and the TWDx[31:0], TCOMMAx[3:0] and TBIT9x[3:0] signal lines are not used. This mode is normally used for tests where serial test data is received from and transmitted to either test equipment or via a serial backplane to a remote card and is the basic loopback path shown earlier in Figure 61(b). The data rate selection bits TXHR and RXHR in the channel configuration registers must be configured to carry the same value. Also, the 8b/10b encoder and decoder are excluded from the loopback path by setting the 8b10bT and 8b10bR configuration bits to “0”. Table 37 illustrates the control interface register configuration for the parallel loopback.

Table 37. Parallel Loopback at the SERDES Boundary Configuration Bit Definitions for the SERDES

Register Address (Hex)	Bit Value	Bit Name	Comments
30002, 30012, 30022, 30032	Bit 0 = 0 or 1	TXHR	Set to “0” or “1”. TXHR and RXHR bits must be set to the same value.
	Bit 7 = 0	8b10bT	Set to “0” The 8b/10b encoder is excluded from the loopback path. The 8b/10b encoder and decoder selection control bits must both be set to “0”.
30003, 30013, 30023, 30033	Bit 0 = 0 or 1	RXHR	Set to “0” or “1”. TXHR and RXHR bits must be set to the same value.
	Bit 3 = 0	8b10bR	Set to “0”. The 8b/10b decoder is excluded from the loopback path. The 8b/10b encoder and decoder selection control bits must both be set to “0”.
30005	Bit 7 = 1	GTESTEN	SET to “1” if the loopback is done globally on all four channels.
30006, 30016, 30026, 30036	Bits[4:0]	Testmode	Set to 00001

SERDES Characterization Test Mode

The SERDES characterization mode is a test mode that allows for direct control and observation of the transmit and receive SERDES interfaces at chip ports. With these modes the SERDES logic and I/O can be tested one channel at a time in either the receive or transmit modes.

The SERDES test mode requires that the SERDES block be selected instead of the SPIB block during the Module Generation phase of the ORSPI4 in *ispLEVER*. In addition, the TESTMD[1:0]N signals need to be set to “0” during SERDES characterization.

The SERDES characterization test mode is configured by setting bits in the control registers via the system bus. The transmit characterization test mode is entered when SCHAR_ENA= “1” (address 30830, bit 4) and SCHAR_TXSEL= “1” (address 30830, bit 5). Entering this mode will cause chip port inputs to directly control the SERDES low-speed transmit ports of one of the channels as shown in Table 38.

Table 38. SERDES Transmit Characterization Mode

Required Inputs	Inputs Used During Mode
TESTMD[1:0]N=00, SCHAR_EN=1, SCHAR_TXSEL=1	TBCx (transmit) ← PMIA15 LDIN9x ← PMIA14 LDIN8x ← PMIA13 LDIN7x ← PMIA11 LDIN6x ← PMIA9 LDIN5x ← PMAI8 LDIN4x ← PMIA7 LDIN3x ← PMIA6 LDIN2x ← PMIA5 LDIN1x ← PMIA4 LDIN0x ← PMIA2

The x in the table represents a single channel in the SERDES QUAD, selected by the SCHAR_CHAN control bits (address 30830, bits 6 and 7). The decoding of SCHAR_CHAN is shown in Table 39.

Table 39. Decoding of SCHAR_CHAN

SCHAR_CHAN0	SCHAR_CHAN1	Channel
0	0	A
1	0	B
0	1	C
1	1	D

The receive characterization test mode is entered as long as TESTMD[1:0]N = 00. In this mode, one of the channels of SERDES outputs is observed at chip ports as shown in Table 40. The channel that is observed is also based on the decoding of SCHAR_CHAN as shown in Table 39. The receive characterization pins can be observed independently, whether the transmit characterization pins are enabled or not.

Table 40. SERDES Receive Characterization Mode

Required Inputs	Outputs Used During Mode
TESTMD[1:0]N=00	TBC311 → PMID20 CV → PMID21 BYTSYNC→ PMID22 WDSYNC → PMID23 RBCO → PMID24 RBC1 → PMID25 LDOUT9 → PMID26 LDOUT8 → PMID 27 LDOUT7 → PMID28 LDOUT6 → PMID29 LDOUT5 → PMID30 LDOUT4 → PMID31 LDOUT3 → PMID32 LDOUT2 → PMID33 LDOUT1 → PMID34 LDOUT0 → PMID35

Memory Controller Functional Description

The ORSPI4 device includes a Memory Controller interface to an external second generation Quad Data Rate (QDRII) memory. This is provided for additional data buffering beyond the embedded DPRAMs. In this case, the embedded DPRAMs are used as clock-crossing domain FIFOs. The key requirement for this memory interface is the support of a throughput of greater than 20 Gbits/s so that all the data received on the SPI4 interface at 10 Gbits/s can be buffered. A QDRII SRAM supports this throughput with 32 unidirectional data lines. A 36-bit interface is provided to allow for parity or control bits to be added.

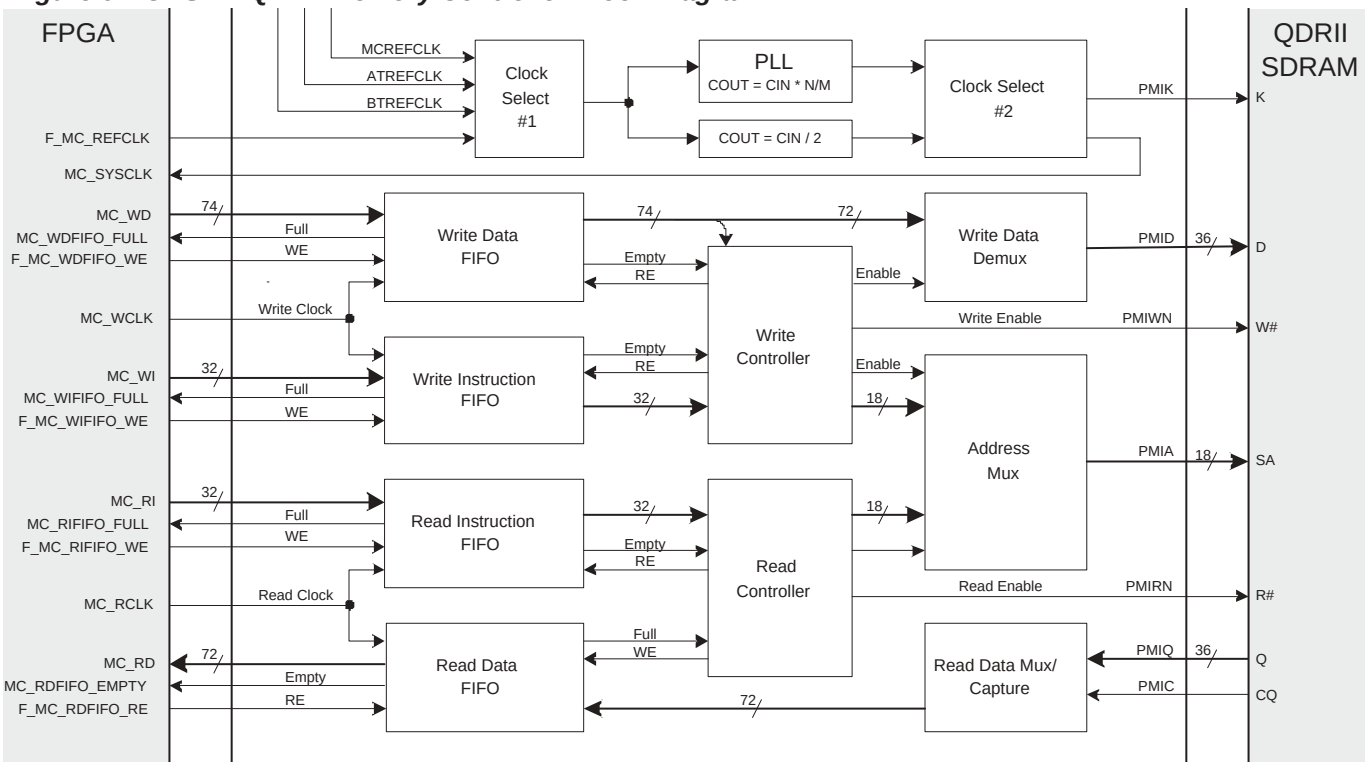
The controller block provides the ability to access an external QDRII SRAM through the FPGA. A set of 72 input data and 72 output data signals are available across the core-FPGA interface. Of the 72 signals, eight can be used either for parity or for data. The core passes the data transparently to and from the QDRII SRAM in two-word or four-word bursts. The data interfaces to memory are 36 bits wide and the address bus is 18 bits wide. This supports the interfaces required for a 512K x 36 bit (18 Mbit) QDRII SRAM.

The basic blocks of the Memory Controller are:

- FIFOs for Write Data and Instructions
- FIFOs for Read Data and Instructions
- Read and Write Controllers (including FIFO and memory interface state machines)
- Address Multiplexer (MUX)
- Write Data DEMUX and Read Data Capture/MUX Blocks

A logical block diagram of the Memory Controller is shown in Figure 62. Details of the individual blocks are outlined in the following sections. The blocks are described in sequence for a write from the FPGA and a subsequent read to the FPGA.

Figure 62. ORSPI4 QDRII Memory Controller Block Diagram.



FPGA/Memory Controller Core Interface Description

Table 41 lists the signals at the FPGA/Core interface for the Memory Controller block.

Table 41. FPGA/Embedded Core Signals

Signal	Direction From /To	Description
QDRII SRAM Read/Write Control		
MC_WD[73:0]	FPGA → Core	Write data to Memory Controller Write Data FIFO
MC_WI[31:0]	FPGA → Core	Write instruction to Memory Controller Read Instruction FIFO
MC_RD[71:0]	Core → FPGA	Read data from Memory Controller Read Data FIFO
MC_RI[31:0]	FPGA → Core	Read instruction to Memory Controller Write Instruction FIFO
MC_WDFIFO_FULL	Core → FPGA	Full flag from MC Write data FIFO. Full threshold is set by the MC_FULL_THRESHOLD register bits.
MC_WIFIFO_FULL	Core → FPGA	Full flag from MC Write instruction FIFO. MC_WIFIFO_FULL high indicates that only one more instruction can be written to the instruction FIFO.
MC_RDFIFO_EMPTY	Core → FPGA	Empty flag from MC Read data FIFO. Empty threshold is set by the MC_EMPTY_THRESHOLD register bits.
MC_RIFIFO_FULL	Core → FPGA	Full flag from MC Read instruction FIFO. MC_RDFIFO_FULL HIGH indicates that only one more instruction can be written to the instruction FIFO.
F_FMC_WDFIFO_WE	FPGA → Core	Write enable to Write Data FIFO
F_FMC_WIFIFO_WE	FPGA → Core	Write enable to Write Instruction FIFO
F_FMC_RDFIFO_RE	FPGA → Core	Read enable to Read data FIFO
F_FMC_RIFIFO_WE	FPGA → Core	Write enable to Read Instruction FIFO
MC_WCLK	FPGA → Core	Memory Controller Write clock
MC_RCLK	FPGA → Core	Memory Controller Read clock
MC_SYSCLK	Core → FPGA	Internal system clock for the Memory Controller i/f
F_MC_REFCLK	FPGA → Core	Memory Controller Reference clock

Memory Controller - Detailed Description

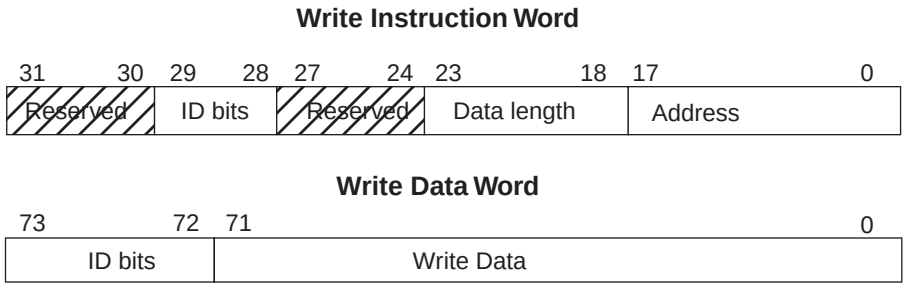
Write Data and Instruction FIFOs

The Write Data FIFO (WDFIFO) stores data written into it by logic in the FPGA. The configuration is 64 by 74 bits. Read/write pointers and fill-level logic is included. The FIFOs also function as an asynchronous clock domain boundary between the FPGA generated write clock and the Memory Controller (MC) system clock (MC_SYSCLK). Writes from the FPGA that are asynchronous to the memory clock (K, K#) are allowed. It is also possible to run the FIFO in synchronous mode by looping back the 1x clock via the FPGA to the FIFO write clock (MC_WCLK).

A set of 72 data signals is available across the core-FPGA interface. Of the 72, eight signals can be either used for parity or data. The core, however, passes the data transparently to the QDRII SRAM. Data length is specified in terms of FIFO lines. If data length is > 64 FIFO lines, then the user must segment the data.

The Write Instruction FIFO (WIFIFO) is configured 4 by 32 bits respectively, in order to store up to 4 instructions that are written to the MC by logic in the FPGA. The instruction fields are ID (data/address coherency), data length (number of cache lines) and address (for QDR memory). Write instruction and data word formats are shown in Figure 63.

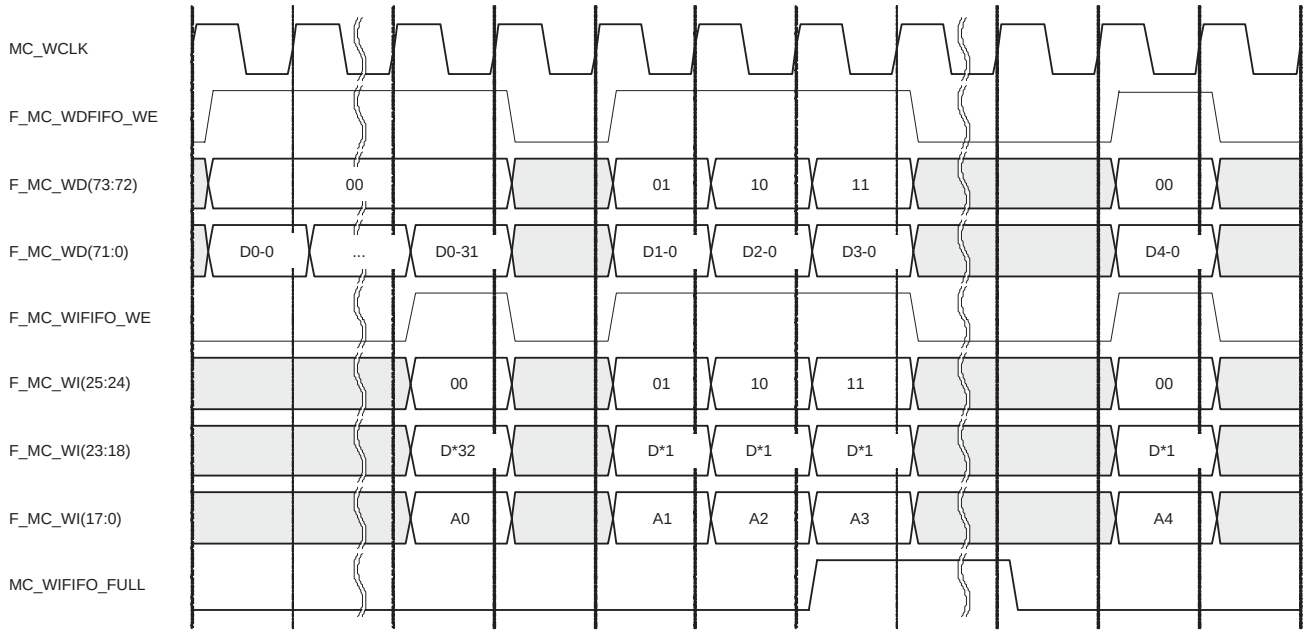
Figure 63. Write Data and Instruction Word Formats



The user starts a write cycle by writing data to the WDFIFO. The maximum data length is 63 lines deep and the minimum is one 72-bit data word (or one line). The user provides the instruction word associated with a data string during the write period for the last word of the data. This sequence can continue to repeat as long as the MC_WxFIFO_FULL flags from the embedded core remain inactive. Once the MC_WIFIFO_FULL flag is asserted, the user may send data words to the WDFIFO locations (assuming the MC_WDFIFO_FULL flag has not been asserted) and write one last instruction word to the instruction FIFO as the last data word is written into the data FIFO.

The timing diagram for the write interface for a typical write sequence is shown in Figure 64. The write sequence in this case consists of writing 32 words to location A0 followed by one word each to locations A1, A2, A3 and A4. The MC_WIFIFO_FULL is asserted after the instruction for location A2 has been written, indicating that only one more instruction may be written to the instruction FIFO. The write to location A4 cannot occur until the MC_WIFIFO_FULL flag is deasserted. The MC_WDFIFO_FULL flag remains low in this case since the full threshold for the data FIFO has not been reached

Figure 64. ORSPI4 Memory Controller Interface to FPGA — Write Timing Diagram



In the first clock cycle, the first word of the data burst is written along with the ID bit value of 00. The instruction word is updated on the last word of a data burst and contains the ID bit value, address and length of the data burst. The maximum burst length is set by the depth of the data FIFO, which is 64 FIFO lines. The data length in the

instruction word is provided in terms of FIFO lines. Data length for data associated with A0 is shown to be 32 lines. The maximum burst length is set by the depth of the data FIFO which is 64 FIFO lines.

Data is presented as 9 bytes in every MC_WCLK cycle. A new data word always begins at bit position 71 and has a unique 2-bit ID. The transition from one ID value to another indicates the start of a new data burst as shown in Figure 63. For each burst, the values of the ID fields in the data word and the instruction word are compared.

The write controller first waits for the empty flag from the instruction FIFO to be deasserted and processes the first pending instruction. The core reads the instruction word first. Based on the data length, the core starts to read the appropriate amount of data from the data FIFO. The address and data length are passed to the write controller state machine. After the last data word has been read from the data FIFO, the core increments the instruction FIFO address to the next pending instruction.

The 2-bit ID field in each data word is compared to the 2-bit ID field in the processed instruction to maintain data/instruction coherency. If the ID fields in the data and instruction words do not match, an error is sent to a software register bit. It is the responsibility of the FPGA to provide the exact amount of data specified in the data length field of the instruction word.

A programmable FIFO full flag (MC_WDFIFO_FULL) is available to the FPGA for the data FIFO. The FIFO flag threshold is set by the 4-bit software register field MC_FULL_THRESHOLD. If the MC_WDFIFO_FULL flag is programmed to indicate a truly full condition, the FPGA logic must assure that the FIFO does not overrun and that an instruction word is written in parallel with the final data word of a sequence.

Address Multiplexer (MUX)

Address information to the QDRII SRAM is multiplexed between the write port and the read port. This is possible since data is always transferred in two-word or four-word bursts (word is 36-bits). Logic in the SRAM supplies the address least significant bits for the two-word burst case and the lower two LSbs in the case of four-word bursts. It is however, necessary to make sure read and write addresses do not contend for the address bus. This function is performed by the Address MUX logic based on signals from the Memory Read Controller (MRC) and Memory Write Controller (MWC).

Figure 65 and Figure 66 show the timing of signals at the QDRII SRAM interface for two-word and four-word reads and writes. The burst size is selectable through the field MC_BURST_MODE in the configuration registers 30B03[0]. Write data is sent on the first rising edge of the positive clock signal (K), after the write address is provided. Read data is returned on the second rising edge of K, after the read address is provided. If a sequence of read and write addresses is sent, both read data and write data are available simultaneously. It is this feature that gives the QDRII SRAM its high throughput.

Note: Except for one address signal, there is no requirement that PMIA address signals be connected to a particular address input on the QDRII SRAM, since writes and reads share the one and only address bus. The only exception is PMIA17, which is only used in 2-word mode, and thus must be connected to the corresponding QDRII SRAM address input that is only present on the 2-word device (if 2-word/4-word compatibility is to be maintained). Flexibility in assigning these signals can be useful in optimizing the layout of this bus.

Figure 65. .ORSPI4 Memory Controller Interface to QDRII: 4-Word Burst Mode

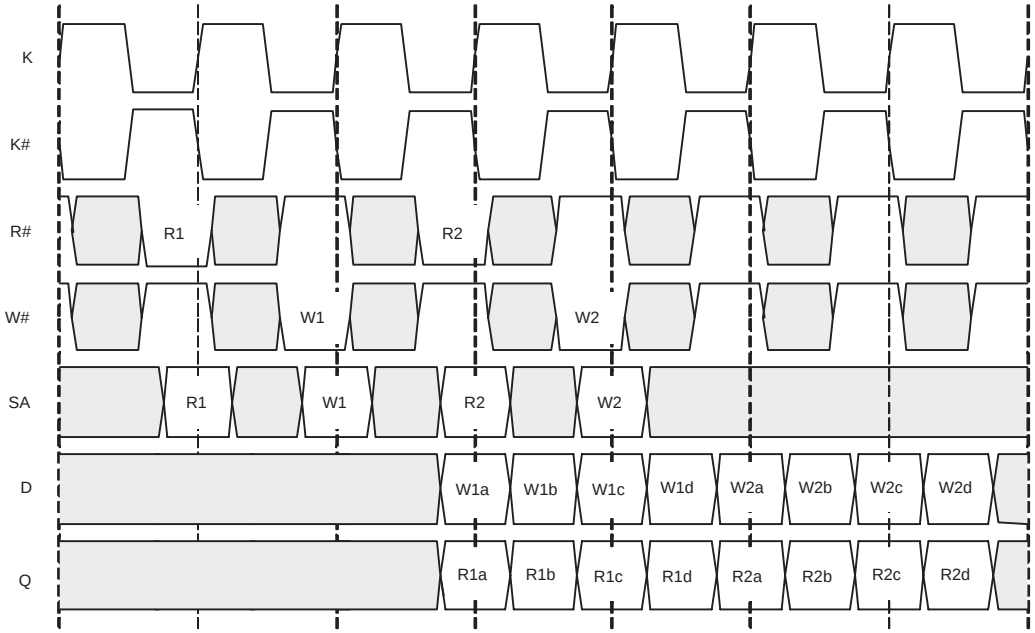
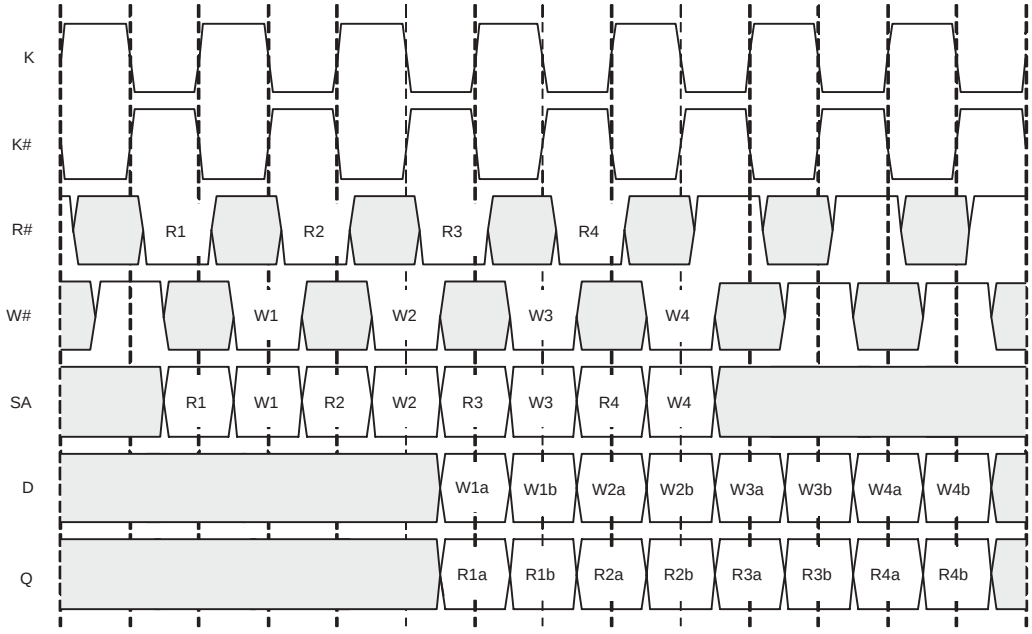


Figure 66 shows the timing of signals at the QDRII SRAM interface for two-word reads and writes. In this case, address data is sent on both the rising and falling edges of the positive clock signal. Write data is sent on the first rising edge of K before the write address and second data word are provided on the falling edge. As in the four-word case, read data is returned on the second rising edge of K after the read address is provided (on a rising edge of the positive clock).

Figure 66. ORSPI4 Memory Controller Interface to QDRII: 2-Word Burst Mode



Write Controller and Write Data DEMUX

The write timing shown in Figure 65 and Figure 66 is generated by the Memory Write Controller (MWC) logic based on instruction and status signals from the write FIFOs and synchronization information from the Memory Read Controller (MRC). The MWC has two functions. First it will read the instruction words from the instruction FIFO and separate the bit fields into address, ID and Data length. Then it will generate write cycle timing based on the type of memory configured (2 or 4 word burst).

The MWC will also do data/instruction data coherency (ID) checking and generate an error (output MC_ID_STATUS) if the test fails. The write controllers run at 2x the clock rate (MCLCKx2) in order to generate the QDRII SRAM data and address timing correctly in relation to the differential memory clock (K, K#). The MWC must also arbitrate with the MRC to ensure that there is no contention between SRAM read and write cycles, as discussed in the previous section.

The MWC generates enables to initiate burst writes. Separate read and write address counters are required and are loaded by the controllers with the content of the address field in the instruction word. In order to support 512k x 36 RAM in 2-word burst mode, 18 active address lines are required, whereas in 4-word burst mode, 17 are required. Data to the QDRII SRAM must also be demultiplexed (DEMUX) by the Write Data DEMUX block in order to convert from the MC internal format of 72 bit buses to the 36-bit DDR format for the QDR SRAM.

Read Data Capture, MUX and Read Controller

The read timing shown in Figure 65 and Figure 66 is generated by the Memory Read Controller (MRC) logic, based on information from the read instruction FIFO. The MRC has two functions. First it will read the instruction words from the instruction FIFO and separate the bit fields into address and data length information. Then it will generate read cycle timing, based on the type of memory configured (2 or 4 word burst) and provide synchronization information to the Memory Write Controller (MWC).

The read controller generates enables to initiate the read cycle. A separate read address counters is required and is loaded by the controller with the content of the address field in the instruction word. As was true with the MWR, to support 512k x 36 RAM in two word burst mode, 17 active address lines are required, whereas in 4-word burst mode, 18 are required. Data from the QDR RAM must also be multiplexed (MUX) in order to convert to the MC internal format of 72 bit buses from the 36-bit DDR format for the QDRII SRAM.

In order to capture the SRAM read data the echo clocks (CQ, CQ#) generated by the SRAM are used to latch the data. This is the most robust and flexible of schemes available to capture the data, as with this source synchronous technique, the impact of round trip delays from the data leaving the Memory Controller, to data being received back can be minimized. The outputs of the capture latches are retimed back into the MCLK2x domain in order to be written to the Read Data FIFO (RDFIFO) discussed in the next section.

Read Data and Instruction FIFOs

The Read Data FIFO (RDFIFO) is used to store the data that is returned from the QDRII SRAM. It is asynchronously read by logic in the FPGA. The RDFIFO size is 64 X 72 bits.

The Read Instruction FIFO (RIFIFO) is similar to the WIFIFO and is configured 4 by 32 bits respectively in order to store up to 4 instructions that are written to the MC by logic in the FPGA. The instruction fields are data length (number of cache lines to read) and address (for QDR memory). The read instruction and data word formats are shown in Figure 67 and Figure 68 respectively.

Figure 67. Read Data and Instruction Word Formats

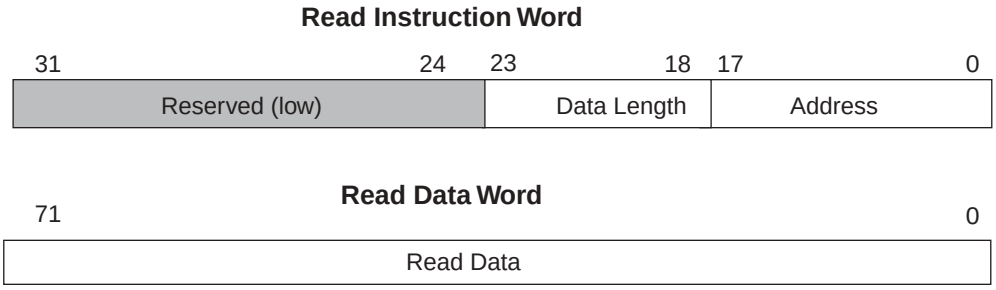
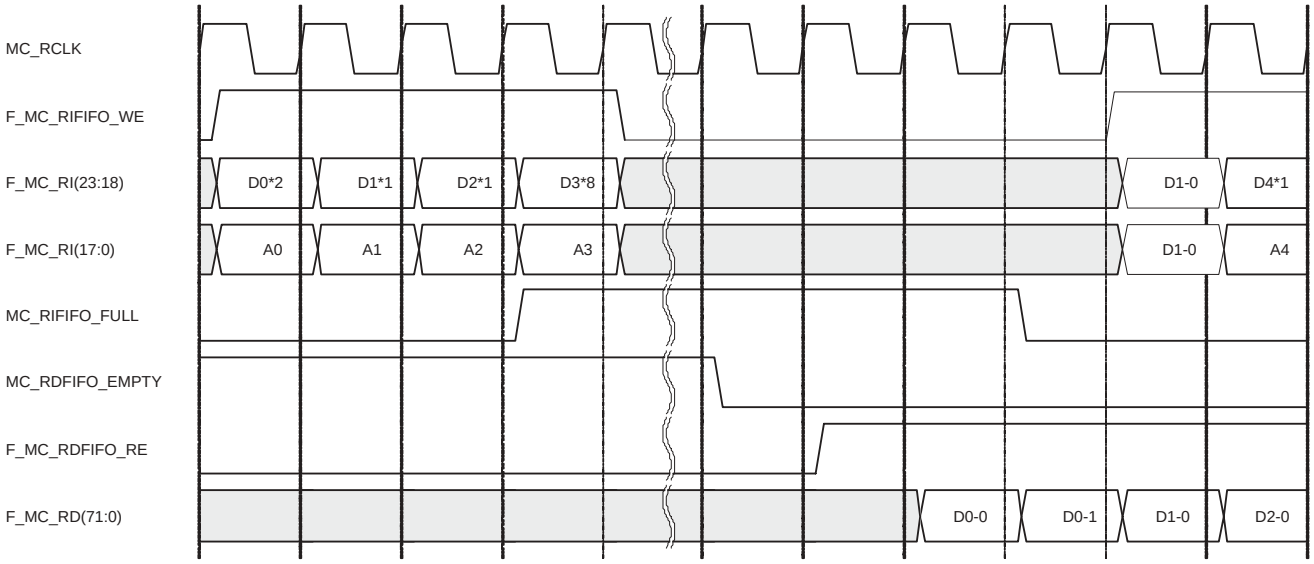


Figure 68. ORSPI4 Memory Controller Interface to FPGA — Read Timing Diagram



The user can write/read instructions into the read instruction FIFO as long as the MC_RIFIFO_FULL flag is low. If this flag is high, the FIFO can accept one more instruction. The user can start reading data from the read data FIFO once the MC_RFIFO_EMPTY flag goes low.

It is the responsibility of the user to:

- Maintain count of data from the read data FIFO and check if they match the data length provided in the corresponding instruction word
- Associate the data received with the address. Data will be received in the same order as they were requested.

The timing diagram for the read interface for a typical read sequence is shown in Figure 68. The read sequence in this case consists of reading 2 words from location A0 followed by one word each from locations A1, A2, eight words from A3 and one word from A4 and A5. The MC_RIFIFO_FULL is asserted after the instruction for location A2 has been written, indicating that only one more instruction may be written to the instruction FIFO. The read instruction write to access location A4 cannot occur until the MC_RIFIFO_FULL flag is deasserted. The MC_RDFIFO_EMPTY flag remains high until data has been read from the QDRII SRAM and is available to the FPGA logic.

A programmable FIFO empty flag (MC_RDFIFO_EMPTY) is available to the FPGA for the data FIFO. The FIFO flag threshold is set by the 4-bit software register field MC_EMPTY_THRESHOLD.

Memory Controller Status Reporting
The following status or alarms will be reported to the user through software register bits:

- Data length mismatch from the write controller state machine. This alarm bit will be set if a block of data read from the write data FIFO does not match the data length from its associated instruction.
- Data-instruction coherency error. This alarm bit will be set if the ID field in write data and its associated write instruction do not match.
- Write data, Read data FIFO overrun and underrun errors.

Clocking Schemes and Timing Diagrams- Memory Controller
The Memory Controller Unit requires three clocks:

- MC_WCLK from the FPGA, which controls the FPGA-side accesses to the Write Instruction and Write Data FIFOs (Figure 62).
- MC_RCLK from the FPGA, which controls the FPGA-side accesses to the Read Instruction and Read Data FIFOs.
- The main clock for the unit is selected by MC_ICK_SEL (configuration register 30B03[2:3]) which controls all remaining clocking, including the clocks supplied to the external QDR-II SRAM. This clock can be sourced as follows:
  - MC_ICK_SEL = 0 selects the dedicated Memory Controller external clock input MCREFCLK = Freq_In
  - MC_ICK_SEL = 1 selects the SPIA unit's external clock ATREFCLK = Freq_In;
  - MC_ICK_SEL = 2 selects the SPIB unit's external clock BTREFCLK = Freq_In, or
  - MC_ICK_SEL = 3 selects the FPGA-supplied signal F_MC_REFCLK= Freq_In.

  The above-selected clock, can then be modified in frequency, by MC_OCK_SEL (configuration register 30B03[7]) as follows:

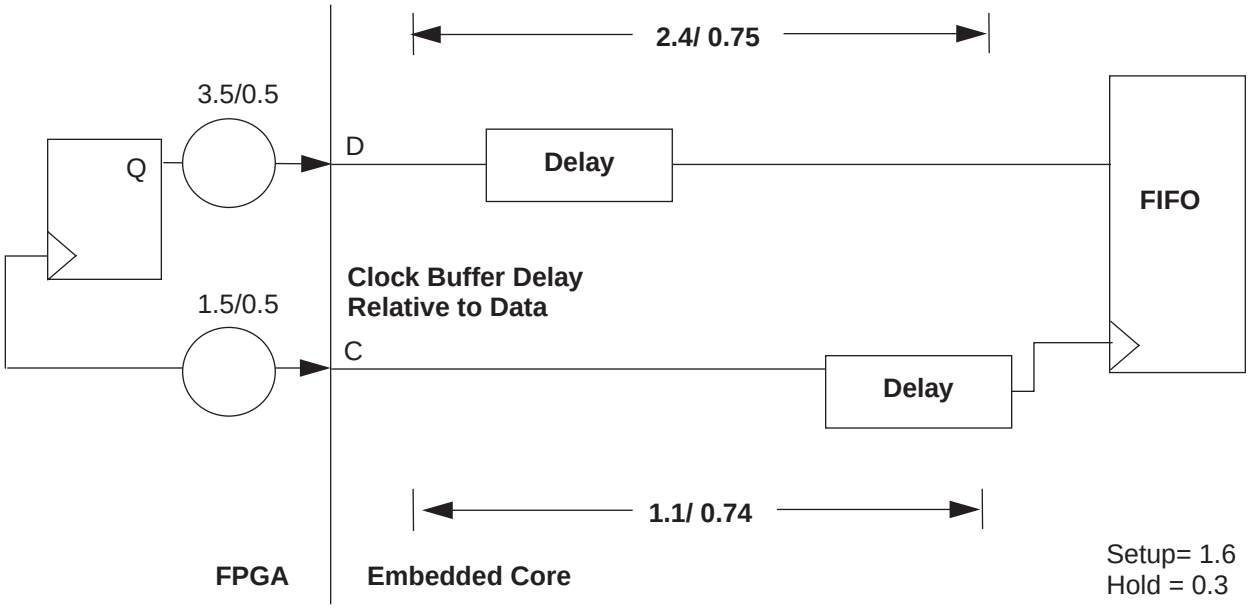
  - MC_OCK_SEL = 0 selects a PLL, such that Freq_Out = (Freq_In * (PLL_N + 1)/(2 * (PLL_M + 1))); or
  - MC_OCK_SEL = 1 selects a divide-by-2, such that Freq_Out = (Freq_In / 2).

  The PLL_N value is obtained from the configuration register 30B04[1:3]. PLL_M value is obtained from configuration register 30B04[5:7].

At the right side of the four internal FIFOs in Figure 62, the data is transferred as two 36-bit words per Freq_Out clock cycle, but is then time-multiplexed into a single 36-bit DDR bus operating at frequency (Freq_Out * 2) for transfer to the external QDR-II SRAM. The clock supplied to the QDR-II SRAM is at frequency Freq_Out.

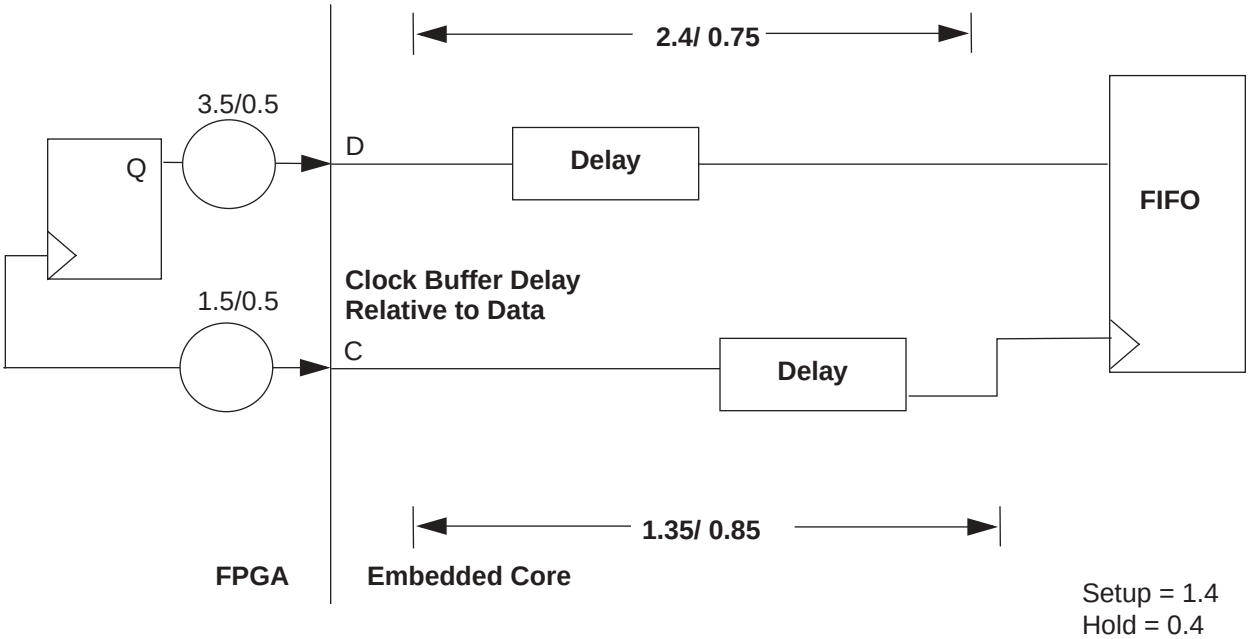
Figure 69 through Figure 75 show the timing diagrams for the embedded controller/FPGA interface.

Figure 69. Data: F_MC_WD[73:0], F_MC_WI[31:0], F_MC_RI[31:0]
Clock: F_MC_WCLK



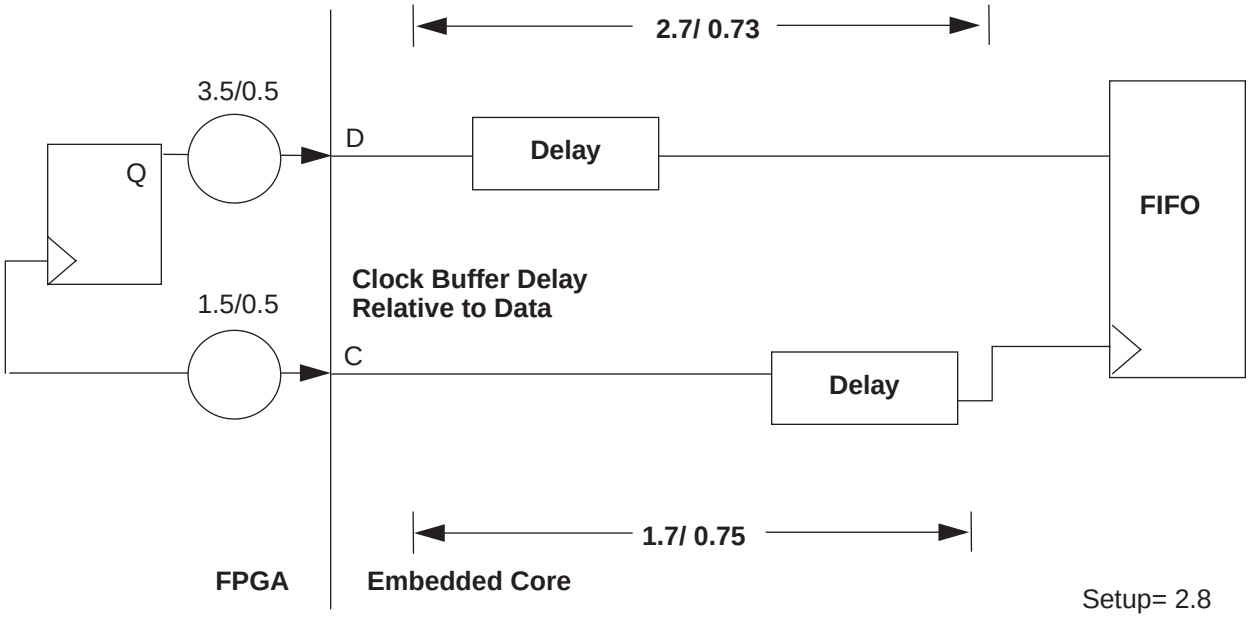
NOTE: Delays represent average max/min values, not absolute values
Actual delay values are used by *ispLEVER*

Figure 70. F_MC_RI[31:0]
Clock: F_MC_RCLK



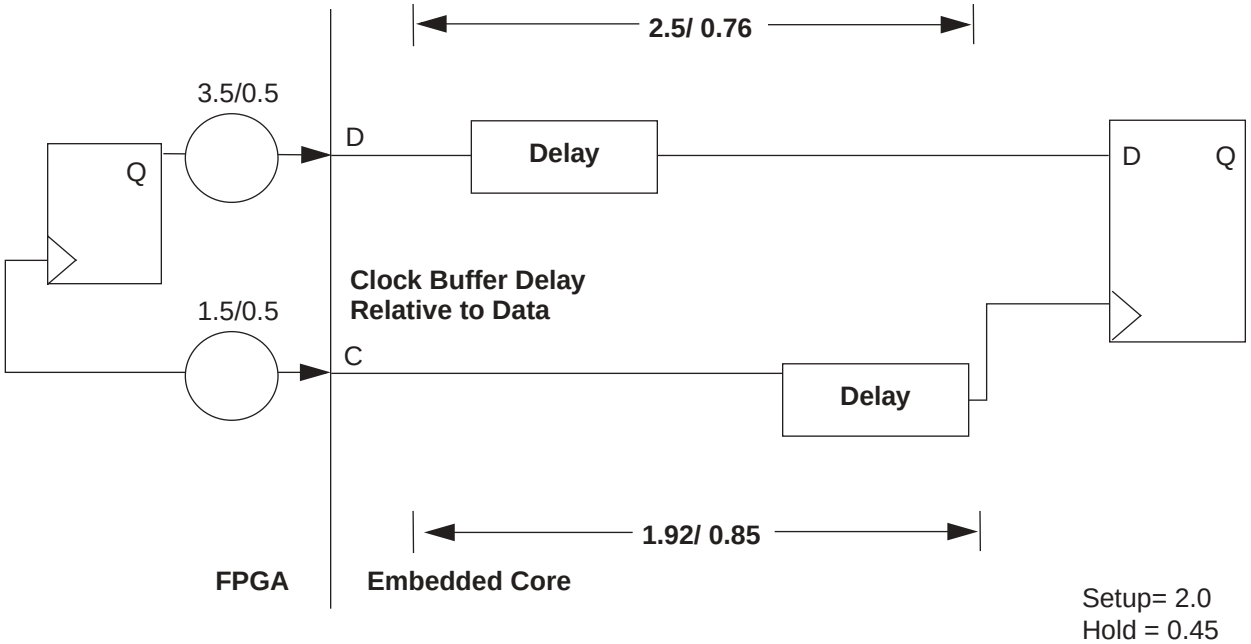
NOTE: Delays represent average max/min values, not absolute values
Actual delay values are used by *ispLEVER*

Figure 71. F_MC_WIFIFO_WE, F_MC_WDFIFO_WE Clock: F_MC_WCLK



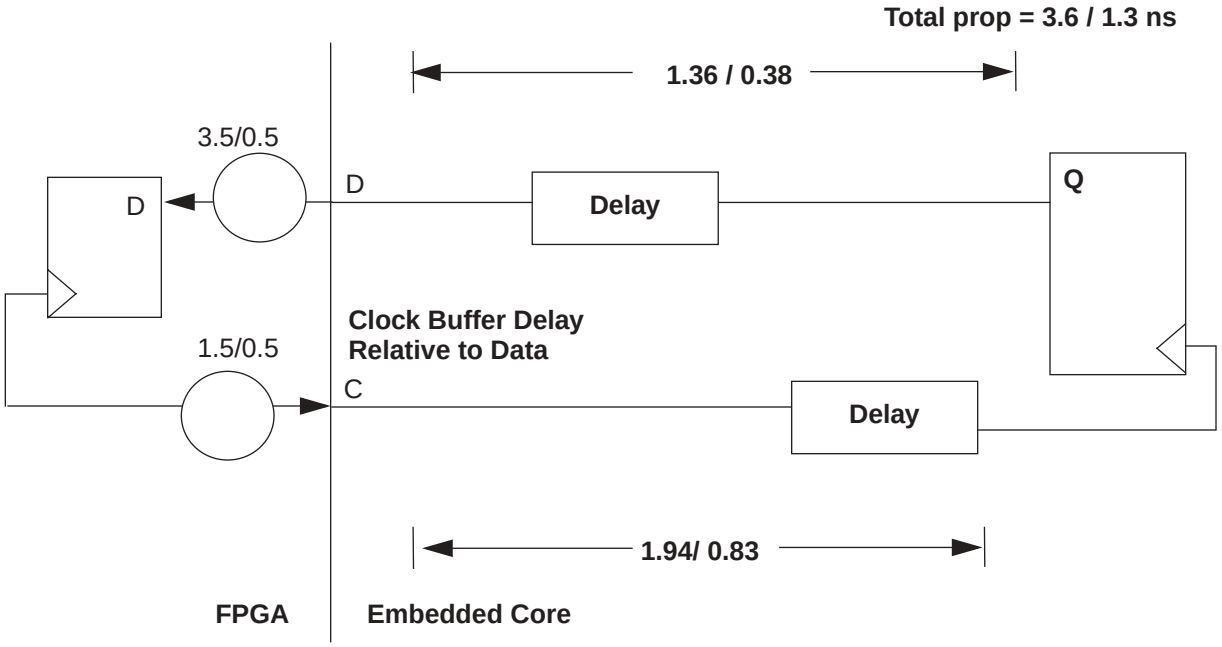
NOTE: Delays represent average max/min values, not absolute values
Actual delay values are used by *ispLEVER*

Figure 72. F_MC_RDFIFO_RE, F_MC_RIFIFO_WE Clock: F_MC_RCLK



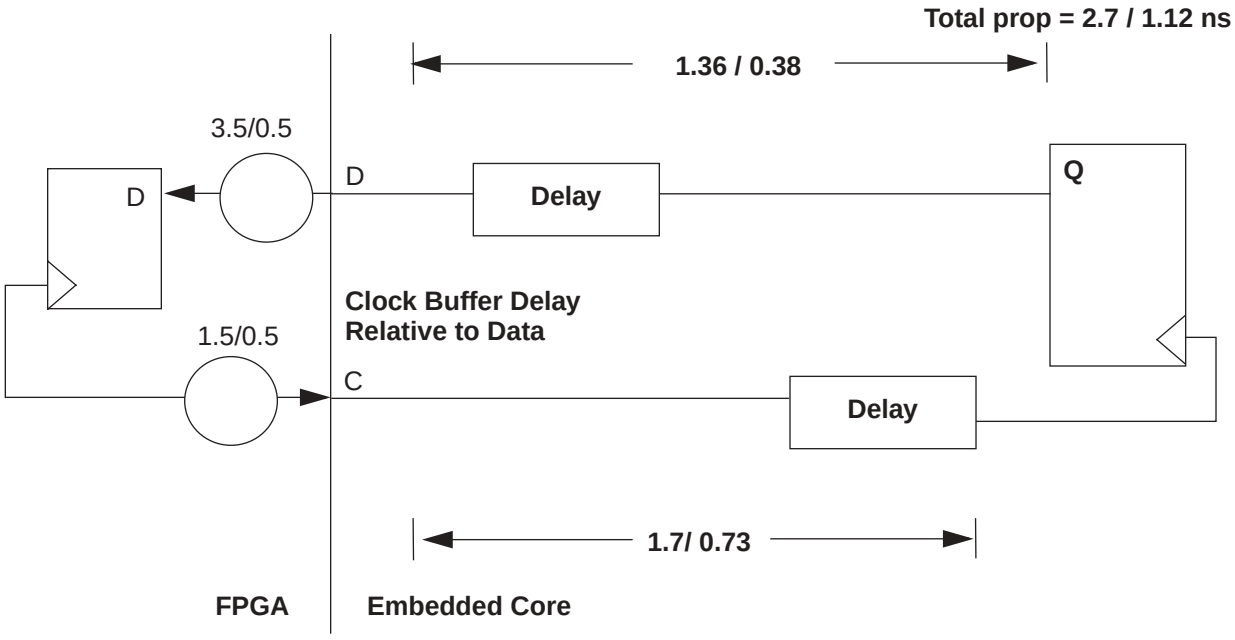
NOTE: Delays represent average max/min values, not absolute values
Actual delay values are used by *ispLEVER*

Figure 73. Data: F_MC_RD[71:0]
Clock: F_MC_RCLK



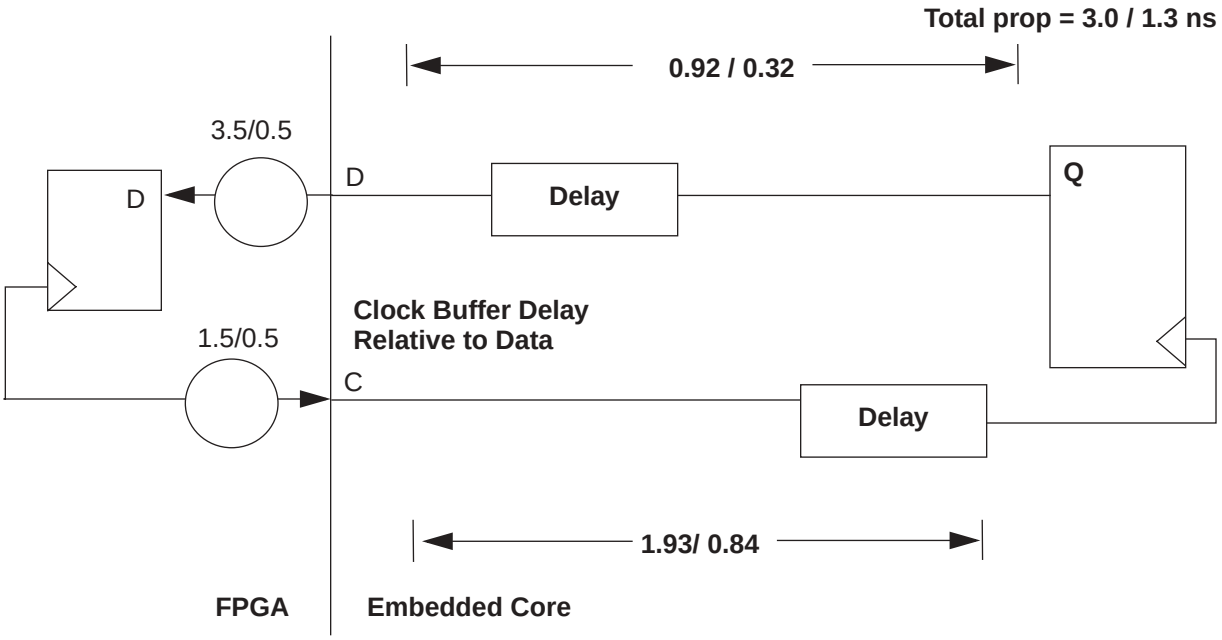
NOTE: Delays represent average max/min values, not absolute values
Actual delay values are used by *ispLEVER*

Figure 74. Data: F_MC_WIFIFO_FULL, MC_WDFIFO_FULL
Clock: F_MC_WCLK



NOTE: Delays represent average max/min values, not absolute values
Actual delay values are used by *ispLEVER*

Figure 75. Data: F_MC_RIFIFO_FULL, MC_RDFIFO_EMPTY
Clock: F_MC_RCLK



NOTE: Delays represent average max/min values, not absolute values
Actual delay values are used by *ispLEVER*

Software

Software for Configuration

There are two ways to write to the ORSPI4 memory map either via MPI (Microprocessor Interface) or via UMI (User Master Interface) to the FPGA logic. Both the interfaces use the system bus to perform the transactions with the registers. The MPI is provided to talk to any Power PC microprocessor whereas UMI is used for any customer-defined interface and interfaces extremely well with ORCASTRA™ to graphically change and monitor the register map. The registers are divided into different domains, the details of which can be found in the technical note TN1017. ORSPI4 device core registers can be accessed by addressing the 30000 onwards domain. Since there are four different macros inside the ORSPI4 core, all these macros have been assigned their individual sub domains.

Table 42. Address Allocation - Register Elements

Address (0x)	Description
309xx	SPIA addresses
30Axx	SPIB addresses
308xx	SERDES addresses
30Bxx	QDR MEM controller addresses and some shared by SPIA and SPIB

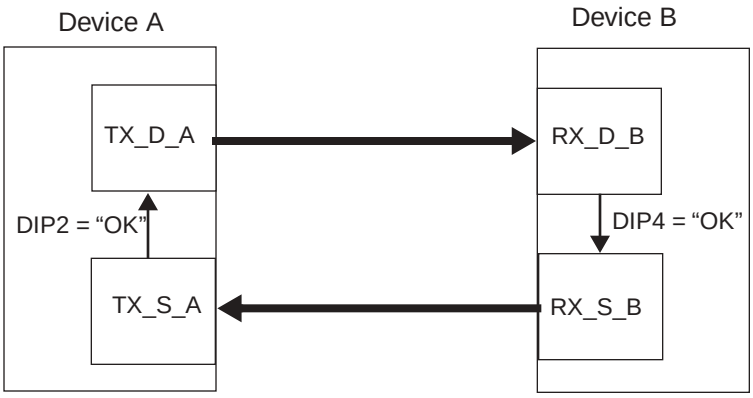
ORSPI4 Start-up Sequence

SPI4 Link Start-up Protocol

Immediately after reset, and before the link synchronization, the SPI4 link must initialize to a known interface protocol. This section describes the initialization protocol for a generic SPI4 link. The actual start-up procedure with the appropriate software register settings is described in the next section. Figure 76 shows the SPI4 link reference diagram that is referenced in the descriptions below. Where appropriate, the corresponding ORSPI4 software register settings are also described as relevant to this start-up protocol.

A reset can be either hard or soft, but is assumed that the entire SPI4 link is being reset. Immediately after the reset signal is deasserted and before the link synchronization, the following actions/conditions must occur/exist.

Figure 76. SPI4 Link Functional Block Diagram



1. The Receive Data FIFOs are emptied (RX_D_B)

2. Any outstanding credits are cleared in the Transmit Data Path (TX_D_A)

   – This is done by presetting all the Credit values in the TX credit memory to 10'h000. This is done by first selecting the TX credit memory by writing a '1' to TX_CRED_MEM_SEL (Address 30917 in SPIA and 30A17 in SPIB) and then writing to address range 31000-310FF that correspond to locations 0 - 255 in the TX credit memory.
   – Additionally, the STAT fields for all ports must be disabled to 2'b11. This can be done by first selecting the TX status memory by writing a '1' to TX_STAT_MEM_SEL (Address 30917 in SPIA and 30A17 in SPIB) and then writing to address range 31000 - 310FF that correspond to locations 0 - 255 in the TX status memory.

3. The Data Transmitter (TX_D_A as shown in Figure 76) sends continuous Training Patterns until it receives valid FIFO status on TX_S_A. A signal from the status block needs to be sent to the data block to indicate this. The Transmit PDM polling is disabled at this point.

4. The Receive Data interface RX_D_B ignores all incoming data until it has observed the training pattern and acquired synchronization. This can be observed by polling the RX_DSKW_DONE_STS and RX_DSKW_ERR_STS interruptible status bits (Address 3091C in SPIA and 30A1C in SPIB). This is described in detail in the next section. As long as the receiver is not deskewed (or synchronization is not complete), RX_S_B will continue to send framing pattern 2'b11 on the SPI4 status link.

5. RX_D_B asserts DIP-4 = "OK" which is indicated by a '0' on the RX_ALGN_OFF_STS interruptible status bit (Address 3091C in SPIA and 30A1C in SPIB). This bit is set to '1' if the number of consecutive DIP-4 errors exceeds the programmed threshold. This threshold is programmed by writing to RX_DIP4_ERR_TH control register bits (Address 30910 in SPIA and 30A10 in SPIB).

6. After the receiver has declared link synchronization and a calendar has been provisioned, the Receive Status Channel RX_S_B begins sending valid FIFO status for each enabled port contained in the Calendar. Valid Status is Frame-based with DIP-2 code words as part of the protocol. Calendar is declared as “provisioned” when RX_CAL_LEN_MAIN (or RX_CAL_LEN_SHD if shadow calendar is being used) has a value > 0. It is recommended to configure the calendar table and write the RX_CAL_LEN_MAIN last.

7. After the Transmitter Status Channel TX_S_A receives a configurable number of consecutive valid DIP-2 code words, it begins transmitting data bursts of the enabled ports within the configured Calendar. The DIP-2 error threshold can be programmed by writing to TX_DIP2_ERR_TH (Address 30944 in SPIA and 30A44 in SPIB).

At this point the end-to-end link is established and valid data/status is being transferred across the SPI4 link.

There may be cases where the Transmitter in Device “A” (TX_D_A) is active, but the Receiver in Device “B” (RX_D_B) is in reset condition. In this situation, after the Receive Reset signal has been deasserted, the link follows the procedure defined above from Step 4. When the receiver is in a reset condition, the RX_S_B block will be sending the “1 1" framing pattern to the Transmit Status Channel interface (TX_S_A). The transmitter will follow the procedure prescribed above as well.

There are also cases where the Data Path transmitter in Device “A” (TX_D_A) is in a reset condition, and the Receiver in Device “B” (RX_D_B) is active. In this situation, after the Transmit Reset signal is deasserted, the link follows the procedure defined above from Step 2. When the transmitter is in a reset condition, the RX_D_B block will not be receiving valid DIP-4 values. The RX_D_B block removes the DIP-4 = “OK” signal and the RX_S_B block begins to send continuous “1 1" framing pattern to the Transmit Status Channel interface (TX_S_A).

SPIA or SPIB Start-up sequence

After the device is powered up, it resets itself with a Power-up reset, designed to trigger itself at power up. The first thing required by the SPI4 cores, is to perform baseline process followed by training if dynamic alignment is chosen (SPI4_LOW_SPEED_DATA_SEL = ‘0’). This is required by the SPIA and SPIB high-speed receivers for internal alignment. Dynamic alignment may also be required during normal transmission and reception of SPI4 data because Process, Voltage and Temperature variations can alter the alignment.

The procedure is as follows (only shown for SPIA)

- Enable the interrupts related to the high-speed receive interface block by writing FF to 30913. This would enable the status update on the register 3091C. It is also recommended to enable other interrupts as well. For example registers 30912 and 30914 so that the corresponding statuses can be monitored.
- Set register 30910 to 0x71 this would set the DIP4 Error threshold to 7 and will enable baseline for SPIA.
- Soft reset the SPIA core by writing 0x02 to 30B20 and then clearing this bit. This can also be achieved by pulsing the s4a_fpga_reset FPGA/CORE interface pin.
- Read the 3091C register - it should read 0x51
  - Bit 0 = 0 indicates that the RDI PLL has locked to the REFCLKA
  - Bit 1 = 1 indicates that the Baseline has been done
  - Bit 2 = 0 indicates that there are no Baseline errors
  - Bit 3 = 1 indicates that receive de-skew has been achieved. This bit will stay high if the enable training is not set in the register 30910, it will also be set once the training has been done.
  - Bit 4 = 0 indicates that there are no receive de-skew errors, these errors will only show up if the training has been enabled and de-skewing is not successful.
  - Bit 5 = 0 indicates that training is not detected
  - Bit 6 = 0 indicates that byte-lanes' alignment is not off, this bit is also relevant if the training is enabled and Bytes are not being aligned properly.
  - Bit 7 = 1 indicates that the TX status is showing LOF, because the TX_CAL_M_MAIN and TX_CAL_LEN_MAIN are not programmed to non-zero values.

- Enable internal loopback by writing 0x08 to 30915. This is only required if there is no other SPI4 device talking to ORSPI4. External loopback can also be performed in which case this register bit should not be set.
- Program the TX_DATA_MAX_T and TX_ALPHA values in the registers 30935, 30936, 30937, 30940, 30941 and 30942. Setting a non-zero value to these registers will automatically cause the transmit block to initiate the SPI4 training sequence.
- Program the RX_FIFO_THRESHOLD_H (30911) and TX_FIFO_THRESHOLD_H (30945) values to 0x78 and 0x17 respectively. These are watermarks for internal asynchronous FIFOs. These are the recommended values.
- Disable baseline and enable training by writing 0x72 to 30910.
- Read the 3091C register and it should read 0x55
  - Bit 0 = 0 indicates that the receive PLL has locked to REFCLKA
  - Bit 1 = 1 indicates that the Baseline has been done
  - Bit 2 = 0 indicates that there are no Baseline errors
  - Bit 3 = 1 indicates that RDI de-skew has been achieved
  - Bit 4 = 0 indicates that there are no RDI de-skew errors
  - Bit 5 = 1 indicates that training is detected.
  - Bit 6 = 0 indicates that byte-lanes' alignment is not off.
  - Bit 7 = 1 indicates that the TX status is showing LOF, because TX_CAL_M_MAIN and TX_CAL_LEN_MAIN are not programmed to non-zero values.
- Disable the training by writing 0x70 to the register 30910. The DIP4 error threshold is still kept at 7 which is a reasonable number. This means that if the receiver sees 7 consecutive DIP4 errors then it will flag DIP4 error status (Bit 6 in the register 3091D). It is recommended that the training be kept enabled at all times so the dynamic alignment is performed to compensate for process, voltage and temperature variations.
- Now the register 3091C should read 51 again. After the TX_CAL_M_MAIN is programmed this register should read 50.
- Read register 3091D twice, the first value read could indicate DIP4 and DIP2 errors, if clears to 00 after the second read then there are no errors. The errors during the first read indicate that the errors existed before the training was performed.
- Read the register 3090A to read the DIP4 error count. This register will keep counting if the bit 5 of 3091D is set, otherwise this register will show 00.
- Read the register 3090B to read the DIP2 error count. This register will keep counting if the bit 6 of 3091D is set, otherwise this register will show 00.

After the above configuration is completed, the device is ready to transmit and receive data. The user can provision the Calendars and the Port Descriptor Memories for appropriate bandwidth. This is an indirect addressing mechanism where a single bit in register 30917 selects which memory space needs to be addressed. Once one of the bit is selected then the user can write to 31000 to 31FFF address space. The following sequence shows how to do that. The following example shows how to program TX PDM, RX PDM, TX calendar and RX calendar.

- Four ports (0, 1, 2 and 3) of SPI4 data with equal bandwidth.
- No DPRAM Partitions (Virtual FIFOs).

- Write 20 HEX to 30917 to select the RX calendar indirect addressing
- Next program the RX calendar memory through the following writes:
  - Write 00 to the 31000 location.
  - Write 01 to the 31001 location.
  - Write 02 to the 31002 location.
  - Write 03 to the 31003 location.
- Write 10 HEX to register 30917 to select the RX PDM indirect addressing. Since there are no partitions for the DPRAMS, only the bits which correspond to the BANK_ID of the DPRAMs will be written. Bits that correspond to

the PARTITION_ID of the DPRAMs are written to “000”.
  – Write 00 HEX to the 31000 location which means that data will read from the DPRAM0 for port 0.
  – Write 08 HEX to the 31001 location which means that data will be read from the DPRAM1 for port 1.
  – Write 10 HEX to the 31002 location which means that data will be read from the DPRAM2 for port 2.
  – Write 18 HEX to the 31003 location which means that data will be read from the DPRAM3 for port 3.

- Write 08 HEX to 30917 to select the TX calendar indirect addressing
  – Write 00 to the 31000 location.
  – Write 01 to the 31001 location.
  – Write 02 to the 31002 location.
  – Write 03 to the 31003 location.

- Write 04 HEX to 30917 to select the TX PDM indirect addressing. TX PDM has a total of 6 fields to be written and they are divided into 3 bytes. These three bytes correspond to a single memory location for TX PDM, which has a total of 256 locations.
  – Write the port ID 00 HEX to 31000 corresponding to port 0.
  – Write 0F HEX to 31001 (000 to partition ID field and 1111 to BURST_VAL field).
  – Write 0C HEX to 31002 which implies:
    - MB_EN bit is set to ‘1’ which means that MAX-BURST1 and MAX-BURST2 both will be supported, otherwise only MAX-BURST1 will be used.
    - M bit is set to ‘1’ which means the port ID field programmed in the TX PDM is propagated to the FPGA on the SPIA_TX*k*_PORT_ID (*k*=32, 64, 128) signal. If M bit is not set then the port ID programmed in the TX calendar will be broadcast to the FPGA.
    - DPRAM BANK_ID field is set to “00”.
  – Address location 31003 is not used.
  – Write 01 HEX to 31004 corresponding to port 1.
  – Write 0F HEX to 31005 (000 to partition ID field and 1111 to BURST_VAL field).
  – Write 0D HEX to 31006 (MB_EN bit is set to ‘1’, M bit is set to ‘1’ and DPRAM BANK_ID field is set to “01”).
  – Address location 31007 is not used.
  – Write 02 HEX to 31008 corresponding to port 2.
  – Write 0F HEX to 31009 (000 to partition ID field and 1111 to BURST_VAL field).
  – Write 0E HEX to 3100A (MB_EN bit is set to ‘1’, M bit is set to ‘1’ and DPRAM BANK_ID field is set to “10”).
  – Address location 3100B is not used.
  – Write 03 HEX to 3100C corresponding to port 3.
  – Write 0F HEX to 3100D (000 to partition ID field and 1111 to BURST_VAL field).
  – Write 0F HEX to 3100E (MB_EN bit is set to ‘1’, M bit is set to ‘1’ and DPRAM BANK_ID field is set to “10”).
  – Address location 3100F is not used.

SERDES Start-Up Sequence

The following sequence is required by the SERDES. For information required for simulation that may be different than this sequence, see the ORSPI4 Design Kit.

1. Initiate a hardware reset by making RESETN low. Keep this low during FPGA configuration of the device. The device will be ready for operation 3 ms after the low to high transition of RESETN.

2. Configure the following SERDES internal and external registers. Note that after device initialization, all alarm and status bits should be read once to clear them. A subsequent read will provide the valid state. Set the following bits in register 30800:

- Bits LCKREFN_[A:D] to “1”, which implies lock to data.
- Bits ENBYSYNC_[A:D] to “1” which enables dynamic alignment to comma.

Set the following bits in register 30801:

- Bits LOOPENB_[A:D] to “1” if high-speed serial loopback is desired.

Set the following bits in registers 30002, 30012, 30022, 30032:

- TXHR set to “1” if TX half-rate is desired.
- 8B10BT set to “1”

Set the following bits in registers 30003, 30013, 30023, 30033:

- RXHR Set to “1” if RX half-rate is desired.
- 8B10BR set to “1”.
- LINKSM set to “1” if the Fibre Channel state machine is desired.

Assert GSWRST bit by writing two “1”'s. Deassert GSWRST bit by writing two “0”s. Wait 3ms. If higher speed serial loopback has been selected, the receive PLLs will use this time to lock to the new serial data.

Monitor the following alarm bits in registers 30000, 30010, 30020, 30030:

- LKI, PLL lock indicator. “1” indicates that PLL has achieved lock.

3. If 8b/10b mode is enabled, enable link synchronization by periodically sending the following sequence three times:

- K28.5 D21.4 D21.5 D21.5 or any other idle ordered set (starting with a /comma/) in FC mode.
- /comma/ characters for the XAUI state machine and /A/ characters for word and channel alignment in XAUI

Resets

Global Resets

Global resets affect all blocks in the ORSPI4 Embedded ASIC Core (EAC) section (SPIA, SPIB, MC, MPI, and SERDES. A global reset can be caused by one the following: power-up reset, bitstream download without a partial reconfiguration enabled, hardware reset, or FPGA Global Set Reset (GSR).

Power-Up Reset

The power-up reset process begins when the power supply voltage ramps up to approximately 80% of the nominal value of 1.5 V. Following this event, the device will be ready for normal operation after 3 ms. For more information on power-up reset, please refer to the ORCA SERIES 4 FPGA data sheet at www.latticesemi.com.

Bitstream Download (DONE=0)

During bitstream download, the FPGA DONE signal remains low until the part is fully configured During this time, all ORSPI4 EAC blocks remain in RESET. During partial reconfiguration, an FPGA register bit can be set to prevent a RESET of the EAC while DONE=0. For more information on bitstream configuration, please refer to the ORCA SERIES 4 FPGA data sheet at www.latticesemi.com.

Hardware Reset (RESETN)

A hardware reset is initiated by making the RESETN low for at least two microprocessor clock cycles. The device will be ready for operation 3 ms after the low to high transition of the RESETN. This reset function affects all EAC blocks.

FPGA Global Set Reset (GSR)

The FPGA Global Set Reset signal (GSR) can be made to reset the SPIA, SPIB and SERDES blocks of the ORSPI4 core. This can be done during the ORSPI4 Module/IP Generation phase of the ORSPI4 core in *ispLEVER*. During this phase, the “Disable GSR from resetting data path in FPSC Core” check button is left unchecked to

enable GSR to reset these blocks. For more information on GSR, please refer to the ORCA SERIES 4 FPGA data sheet at www.latticesemi.com.

SPIA-only Resets
The SPIA block can be individually reset though Software Reset-via the microprocessor interface, or using an FPGA interface signal.

Software Reset (SOFT_RESET_S4A)
A register bit used to reset the SPIA is SOFT_RESET_ S4A (30B20, bit 6). This bit, when set to “1”, disables the SPIA block.

FPGA Interface Reset (FPGA_RESET_S4A)
FPGA_RESET_S4A performs the same function as SOFT_RESET_ S4A. The difference is that it is an FPGA interface signal instead of a configuration register bit.

SPIB-only Resets
The SPIB block can be individually reset though Software Reset-via the microprocessor interface, or using an FPGA interface signal.

Software Reset (SOFT_RESET_S4B)
A register bit used to reset the SPIB is SOFT_RESET_ S4B (30B20, bit 7). This bit, when set to “1”, disables the SPIB block.

FPGA Interface Reset (FPGA_RESET_S4B)
FPGA_RESET_S4B performs the same function as SOFT_RESET_ S4B. The difference is that it is an FPGA interface signal instead of a configuration register bit.

MC-only Resets
The Memory Controller (MC) block can be individually reset though Software Reset-via the microprocessor interface, or using an FPGA interface signal.

Software Reset (SOFT_RESET_MC)
A register bit used to reset the MC is SOFT_RESET_ MC (30B20, bit 5). This bit, when set to “1”, disables the MC block.

FPGA Interface Reset (FPGA_RESET_MC)
FPGA_RESET_MC performs the same function as SOFT_RESET_ MC. The difference is that it is an FPGA interface signal instead of a configuration register bit.

SERDES-only Resets
The SERDES block can be individually reset though Software Reset-via the microprocessor interface, or using FPGA interface signals.

Software Reset (SWRST and HARD_RESET_FC)
Using the software reset option via the microprocessor interface, each channel can be individually reset by setting SWRSTx (bit 2) to a logic “1” in the channel configuration register (30004,30014,30024,30034)). The device will be ready 3 ms after the SWRSTx bit is de-asserted. Similarly, all four channels per quad SERDES can be reset by setting the global reset bit GSWRST (30005, bit 2). The device will be ready for normal operation 3 ms after the GSWRST bit is de-asserted. Note that the software reset option resets only SERDES internal registers and counters. The microprocessor registers are not affected. It should also be noted that the embedded block couldn't be accessed until after FPGA configuration is complete. Also note that the SWRSTx and GSWRST are active when the corresponding memory map register bit is set high.

Another register bit used to reset the SERDES is HARD_RESET_FC (30B20, bit 4). This bit, when active, disables the SERDES and prevents any access to its internal microprocessor registers (300xx range).

FPGA Interface Reset signals (SYS_RST_N and FPGA_RESET_FC)
FPGA_RESET_FC performs the same function as HARD_RESET_FC. The difference is that it is an FPGA interface signal instead of a configuration register bit.

SYS_RST_N is a synchronous active low reset FPGA interface signal. This signal, when active, resets the read side of the multi-channel alignment FIFOs.

The following table describes the different methods of resetting various parts of the EAC. It lists the logic values set on the FPGA interface and the primary ASB pins.

Table 43. ORSPI4 RESET

Signal Name	RESET Type	ORSPI4 Core Affected				
		MPI	SPIA	SPIB	M_CTRL	SERDES
PWRUPRES = 1	FPGA signal[1]	•	•	•	•	•
DONE = 0	FPGA I/O[1]	•	•	•	•	•
RESETN = 0	EAC IO	•	•	•	•	•
GSRN = 0 and GSRN_DISABLE = 0	FPGA signal		•	•	•	•
FPGA_RESET_S4A = 1	Interface Signal		•			
SOFT_RESET_S4A = 1	Software Register Bit		•			
FPGA_RESET_S4B = 1	Interface Signal			•		
SOFT_RESET_S4B = 1	Software Register Bit			•		
FPGA_RESET_MC = 1	Interface Signal				•	
SOFT_RESET_MC = 1	Software Register Bit				•	
FPGA_RESET_FC = 1	Interface Signal					•
HARD_RESET_FC = 1	Software Register Bit					•
SYS_RST_N (Channel Alignment FIFO RESET)	Interface Signal					•

1. Please refer to the ORCA SERIES 4 FPGA data sheet at www.latticesemi.com.

I/O TRI-STATE Functions

The EAC I/Os can be tri-stated using one of the three signals below.

TS_ALL
This is an FPGA signal that globally tri-states all EAC IOs in addition to the FPGA IOs. For more information on TS_ALL, please refer to the ORCA SERIES 4 FPGA data sheet at http://www.latticesemi.com.

TRISTN
This is an ORSPI4 EAC active low IO that globally tri-states all EAC IOs.

Power-Up
In addition to its RESET function, power-up also puts the ORSPI4 FPSC IO in tri-state until the chip is fully powered up.

Power Down

When set low, the ORSPI4 FPSC “PDN” powers down the following:

- all PLLs in the SPIA, SPIB, Memory Controller and SERDES
- LVDS and HSTL buffers on the EAC section of the ORSPI4
- SPIA, SPIB, and SERDES logic

ORSPI4 Memory Map

The ORSPI4 device features a variety of programming options which include all the OIF-SPI-4.2.0 specified configurable parameters. The base addresses for each functional block in the ORSPI4 device is shown in Figure 77 and Table 44.

The SPIA and SPIB receive cores contain RAM blocks that are configured through writes to the address range 0x31000 - 0x317FF. Prior to configuring any of these memories, the user has to select one of the memories by writing into the appropriate memory select bit in address 0x30917 (SPIA) and 0x30A17 (SPIB).

All interrupts on ORSPI4 are maskable and edge generated. Each interrupt source has a corresponding status latch bit that can be viewed as software status register bits. At the end of a status register read, both the status and edge-triggered interrupt register will be cleared. If the active interrupt condition still persists after the status register read, the status register will continue to show this condition but the edge-triggered interrupt register will remain clear not causing another interrupt.

Every interrupt or status register bit has an interrupt enable bit that must be set to '1' to generate the associated interrupt or read the status. Each interrupt enable for the DPRAMs in SPIA and SPI4 blocks controls or enables eight interrupt sources. These enable registers are at register address 30912, 30943, (SPIA) 30A12, and 30A43 (SPIB). These 32 bits enable/disable 256 interrupts. All other interrupt enables are unique to an interrupt source.

To make the interrupt structure user-friendly, two special top-level interrupt registers are provided. Each of the bits in these top-level interrupt registers point to specific functional blocks that caused an interrupt. The user can poll these top-level interrupt registers to check which block has caused an interrupt and then poll the relevant lower-level interrupts corresponding to a block. Each of the bits in the top-level interrupt status register 30B29 shown in Table 45 point to a specific functional block in the ORSPI4 device. Each of the bits in the SPIA or SPIB DPRAM top-level interrupt status register 30B2A is the collective OR of its associated lower-level interrupts.

Figure 77. SPI4 Core Programming Addresses in ORSPI4

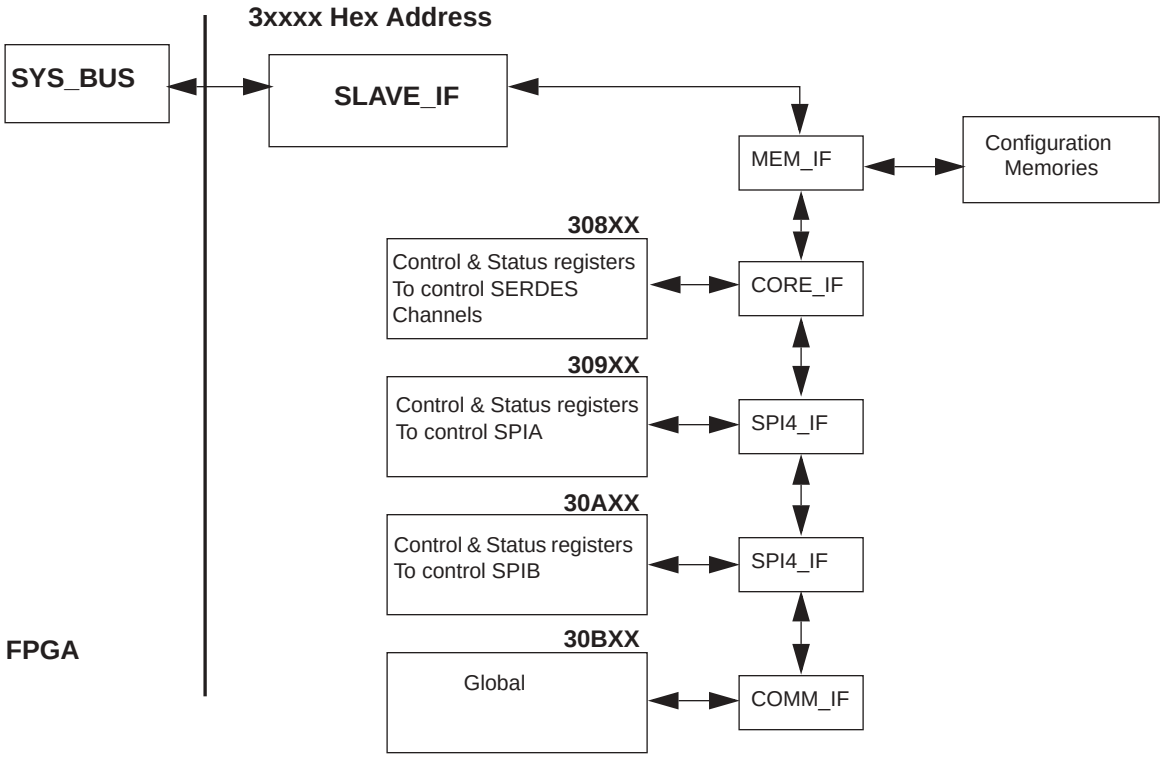


Table 44. ORSPI4 Memory Space

Address (Hex)	Description
3000x	Channel A in SERDES, internal registers
3001x	Channel B in SERDES, internal registers
3002x	Channel C in SERDES, internal registers
3003x	Channel D in SERDES, internal registers
308xx	Channel registers outside the SERDES
3090x - 3092x	SPI4 Core A Control registers
30930 - 3094x	SPI4 Core A Status registers
30950 - 3097x	SPI4 Core B Control registers
30980 - 3099x	SPI4 Core B Status registers
31000 - 317FF	SPI4 Core A/B configurable RAM address space
30100 - 3010F	Memory Controller registers

Table 45. Top-level Interrupt Status Register 30B29

BIT	Interrupt Source	Associated Registers
0	SERDES	3000, 3010, 3020, 3030
1	Memory Controller	30B09
2	Memory Controller	30B08
3	DPRAM	SPIA (3090C-3090F, 30918-3091B, 30928-3092F) SPIB (30A0C-3090F, 30A18-30A1B, 30A28-30A2F)
4	SPI_B_2	30A1C
5	SPI_B_1	30A1D
6	SPI_A_2	3091C
7	SPI_A_1	3091D

Table 46 details the memory map for the FPSC portion of the ORSPI4 device.

Addresses for the control registers for the FPGA portion of the device are detailed in the ORCA Series 4 datasheet. This table shows the databus oriented for the PPC interface. DB0 is the MSB, while DB7 is the LSB. If the user master interface is used to perform operations to the ASIC core then the databus must be used in the opposite notation, where DB7 is the MSB and DB0 is the LSB.

Table 46. Memory Map

(0x) Absolute Address	Bit	Name	Reset Value (0x)	Description
SERDES Alarm Registers (Read Only), x = [A, ...,D]				
30000 30010 30020 30030	[0]	Reserved		
	[1]	LKI_x	00	Receive PLL Lock Indication, Channel x. LKI_x = “1” indicates the receive PLL is locked.
	[2:7]	Reserved		

Table 46. Memory Map (Continued)

(0x) Absolute Address	Bit	Name	Reset Value (0x)	Description
SERDES Alarm Mask Registers (Read/Write), x = [A, ...,D]				
30001 30011 30021 30031	[0]	Reserved	FF	Reserved. Must be set to “1”. Set to “1” on device reset.
	[1]	MLKI_x		Mask Receive PLL Lock Indication, Channel x.
	[2:7]	Reserved		
SERDES Common Transmit and Receive Channel Configuration Registers (Read/Write), x = [A, ...,D]				
30002 30012 30022 30032	[0]	TXHR_x	00	Transmit Half Rate Selection Bit, Channel x. When TXHR_x = “1”, HDOUT_x's baud rate = (REFCLK*10) and TCK78 =(REFCLK/4); when TXHR_x=0, HDOUT_x's baud rate = (REFCLK*20) and TCK78=(REFCLK[A:B]/2). TXHR_x = 0 on device reset.
	[1]	PWRDNT_x		Transmit Powerdown Control Bit, Channel x. When PWRDNT_x = “1”, sections of the transmit hardware are powered down to conserve power. PWRDNT_x = 0 on device reset.
	[2]	PE0_x		Transmit Preemphasis Selection Bit 0, Channel x. PE0_x and PE1_x select one of three preemphasis settings for the transmit section. PEO_x=PE1_x = 0, Preemphasis is 0% PEO_x=1, PE1_x = 0 or PEO_x=0, PE1_x = 1, Preemphasis is 12.5% PEO_x=PE1_x = 1, Preemphasis is 25%. PEO_x=PE1_x = 0 on device reset.
	[3]	PE1_x		
	[4]	HAMP_x		Transmit Half Amplitude Selection Bit, Channel x. When HAMP_x = “1”, the transmit output buffer voltage swing is limited to half its normal amplitude. Otherwise, the transmit output buffer maintains its full voltage swing. HAMP_x = 0 on device reset.
	[5]	Reserved		Reserved. Must be set to 0 Set to “0” on device reset.
	[6]	Reserved		
	[7]	8b10bT_x		Transmit 8b/10b Encoder Enable Bit, Channel x. When 8b10bT_x = “1”, the 8b/10b encoder in the transmit path is enabled. Otherwise, the data is passed unencoded. 8b10bT_x = “0” on device reset.
30003 30013 30023 30033	[0]	RXHR_x	20	Receive Half Rate Selection Bit, Channel x. When RXHR_x =”1”, HDIN_x's baud rate = (REFCLK*10) and RCK78=(REFCLK/4); when RXHR_x= “0”, HDIN_x's baud rate = (REFCLK*20) and RCK78=(REFCLK/2). RXHR_x = “0” on device reset.
	[1]	PWRDNR_x		Receiver Power Down Control Bit, Channel x. When PWRDNR_x = 1, sections of the receive hardware are powered down to conserve power. PWRDNR_x = “0” on device reset.
	[2]	Reserved		Reserved. Must be set to “1”. Set to “1” on device reset.
	[3]	8b10bR_x		Receive 8b/10b Decoder Enable Bit, Channel x. When 8b10bR = “1”, the 8b/10b decoder in the receive path is enabled. Otherwise, the data is passed undeocded. 8b10bR_x = “0” on device reset.
	[4]	LINKSM_x		Link State Machine Enable Bit, Channel x. When LINKSM_x = “1”, the receiver Fiber Channel link state machine is enabled. Otherwise, the Fibre Channel link state machine is disabled. NOTE: LINKSM_x is ignored when XAUI_MODE_x= “1”. LINKSM_x = “0” on device reset.
	[5:7]	Reserved		

Table 46. Memory Map (Continued)

(0x) Absolute Address	Bit	Name	Reset Value (0x)	Description
SERDES Common Transmit and Receive Channel Configuration Registers (Read/Write), x = [A, ...,D]				
30004 30014 30024 30034	[0]	Reserved	40	Reserved. Must be set to “0” Set to “0” on device reset.
	[1]	MASK_x		Transmit and Receive Alarm Mask Bit, Channel x. When MASK_x = “1”, the transmit and receive alarms of a channel are prevented from generating an interrupt (i.e., they are masked or disabled). The MASK_x bit overrides the individual alarm mask bits in the Alarm Mask Registers. MASK_x = “1” on device reset.
	[2]	SWRST_x		Transmit and Receive Software Reset Bit, Channel x. When SWRST_x = “1”, this bit provides the same function as the hardware reset, except that all configuration register settings are unaltered. This is not a self-clearing bit. Once set, this bit must be manually set and cleared. SWRST_x = “0” on device reset.
	[3:6]	Reserved		
	[7]	TESTEN_x		Transmit and Receive Test Enable Bit, Channel x. When TESTEN_x = “1”, the transmit and receive sections are placed in test mode. The TestMode[4:0] bits in the Global Control Registers specify the particular test, and must also be set. NOTE: When the global test enable bit GTESTEN = “0”, the individual channel test enable bits are used to selectively place a channel in test or normal mode. When GTESTEN = “1”, all channels are set to test mode regardless of their TESTEN setting TESTEN_x = “0” on device reset.
SERDES Global Control Registers (Read/Write) Acts on all four Channels in SERDES Quad A or SERDES Quad B.				
30005	[0]	Reserved	44	Reserved Set to “0” on device reset.
	[1]	GMASK		Global Mask. When GMASK = “1”, the transmit and receive alarms of all channel in the SERDES quad are prevented from generating an interrupt (i.e., they are masked or disabled). The GMASK bit overrides the individual MASK_x bits. GMASK = “1” on device reset.
	[2]	GSWRST		Software reset bit. The GSWRST bit provides the same function as the hardware reset for the transmit and receive sections of all four channels, except that the device configuration settings are not affected when GSWRST is asserted. This is not a self-clearing bit. Once set, this bit must be manually set and cleared. The GSWRST bit overrides the individual SWRST_x bits. GSWRST = “0” on device reset.
	[3]	GPWRDNT		Powerdown Transmit Function. When GPWRDNT = “1”, sections of the transmit hardware for all four channels of are powered down to conserve power. The GPWRDNT bit overrides the individual PWRDNT_x bits. GPWRDNT= “0” on device reset.
	[4]	GPWRDNR		Powerdown Receive Function. When GPWRDNR = “1”, sections of the receive hardware for all four channels are powered down to conserve power. The GPWRDNR bit overrides the individual PWRDNR_x bits. GPWRDNR = “0” on device reset.
	[5]	Reserved		Reserved, “1” on device reset.
	[6]	Reserved		
	[7]	GTESTEN		Test Enable Control. When GTESTEN = “1”, the transmit and receive sections of all four channels are placed in test mode. The GTESTEN bit overrides the individual TESTEN_x bits. GTESTEN = “0” on device reset.
30006	[0:4]	TestMode	00	TestMode - See Test Mode section for settings
	[5:7]	Reserved		

Table 46. Memory Map (Continued)

(0x) Absolute Address	Bit	Name	Reset Value (0x)	Description
Control Registers (Read/Write), x = [A, ...,D]				
30800	[0]A [1]B [2]C [3]D	ENBYSYNC[A:D]	00	ENBYSYNC[A:D] = “1” Enables Receiver Byte Synchronization for Channel x. ENBYSYNC[A:D] = “0” on device reset.
	[4]A [5]B [6]C [7][D]	LCKREFN[A:D]		LCKREFN[A:D]= “0” Locks the receiver PLL to ref reference clock for Channel x. LCKREFN[A:D]= “1” = Locks the receiver to data for Channel x. NOTE: When LCKREFN[A:D] = “0”, the corresponding LKI_x bit is also zero. LCKREFN[A:D]= “0” on device reset.
30801	[0]A [1]B [2]C [3]D	LOOPENB[A:D]		Enable Loopback Mode for Channel x. When LOOPEN[A:D]= “1”, the transmitter high-speed output is looped back to the receiver high-speed input. This mode is similar to high-speed loopback mode enabled by TESTMODE[A:D] except that LOOPEN[A:D] disables the high-speed serial output. LOOPEN[A:D]= “0” on device reset.
	[4]A [5]B [6]C [7]D	NOWDALIGN[A:D]		Word Align Disable Bit. When NOWDALIGN[A:D]= “1”, receiver word alignment is disabled for Channel x. NOWDALIGN[A:D]= “0” on device reset.
30810	[0]A [1]B [2]C [3]D	DOWDALIGN[A:D]	00	Word Realign Bit. When DOWDALIGN[A:D] transitions from “0” to “1”, the receiver realigns on the next comma character for Channel x. NOWDALIGN[A:D]= “0” on device reset.
	[4]A [5]B [6]C [7]D	FMPU_STR_EN[A:D]		Enable multi-channel alignment for Channel x. When FMPU_STR_EN[A:D]= “1”, the corresponding channel participates in multi-channel alignment. FMPU_STR_EN[A:D]= “0” on device reset.
30811	[0:1]A [2:3]B [4:5]C [6:7]D	FMPU_SYNMODE[A:D][0:1]	00	Sync mode for x 00 = No channel alignment 10 = Twin channel alignment 01 = Quad channel alignment
30820	[0]A [1]B [2]C [3]D	FMPU_RESYNC1[A:D]	00	Resync a Single Channel. When FMPU_RESYNC1[A:D] transitions from “0” to “1”, the corresponding channel is resynchronized (the write and read pointers are reset). FMPU_STR_EN=0[A:D] on device reset.
	[4]A & B [5]C &D	FMPU_RESYNC2[1:2]	00	Resync a Pair of Channels. When FMPU_RESYNC2[1:2] transitions from a “0” to a “1”, the corresponding channel pair is resynchronized. FFMPU_RESYNC2[1:2]= “0” on device reset.
	[6]	FMPU_RESYNC4[1:2]	00	Resync a Four-Channel Group. When FMPU_RESYNC4 transitions from a “0” to a “1”, the corresponding four-channel group is resynchronized. FMPU_RESYNC4= “0” on device reset.
	[7]	XAUI_MODE	00	Controls use of XAUI link state machine in place of Fibre-Channel state machine. When XAUI_MODE= “1”, all four channels in the SERDES quad enable their XAUI link state machines. (LINKSM_x bits are ignored). XAUI_MODE= “0” on device reset.

Table 46. Memory Map (Continued)

(0x) Absolute Address	Bit	Name	Reset Value (0x)	Description
30821	[0]	NOCHALGN		Bypass channel alignment. NOCHALGN =1 causes bypassing of multi-channel alignment FIFOs for the corresponding SERDES quad. NOCHALGN = “0” on device reset.
	[1:3]	Reserved		
	[4:5]	RCKSEL[0:1]	00	Source for RCK78 (00=A, 10=B, 01=C, 11=D)
	[6:7]	TCKSEL[0:1]	00	Source for TCK78 (00=A, 10=B, 01=C, 11=D)
30830	[0:3]	Reserved		
	[4]	SCHAR_ENA	00	Set this to 1 to enable characterization mode for the SERDES. Characterization mode also requires inputs TESTMD[1:0]N=00, and FPGA interface pin, ENABLE_SPI4_B = “0”. The outputs for the SERDES can be observed at the following ports: TBC311 → PMID20 CV → PMID21 BYTSYNC→ PMID22 WDSYNC → PMID23 RBCO → PMID24 RBC1 → PMID25 LDOUT9 → PMID26 LDOUT8 → PMID 27 LDOUT7 → PMID28 LDOUT6 → PMID29 LDOUT5 → PMID30 LDOUT4 → PMID31 LDOUT3 → PMID32 LDOUT2 → PMID33 LDOUT1→ PMID34 LDOUT0 → PMID35
	[5]	SCHAR_TXSEL		Set this to 1 to enable driving of low speed TX ports of the SERDES during characterization mode. The inputs used During this mode are: TBC ← PMIA15 LDIN9 ← PMIA14 LDIN8 ← PMIA13 LDIN7 ← PMIA11 LDIN6 ← PMIA9 LDIN5 ← PMAI8 LDIN4 ← PMIA7 LDIN3 ← PMIA6 LDIN2 ← PMIA5 LDIN1 ← PMIA4 LDIN0 ← PMIA2
	[6:7]	SCHAR_CHAN		Selects the channel to observe and control during SERDES characterization mode: 00 = A 01 = B 10 = C 11 = D
Status Registers (Read Only), x = [A, ...,D]				
30804	[0:1]A [2:3]B [4:5]C [6:7]D	XAUISTAT[A:D][0:1]	00	XAUI Status Register. Status of XAUI link state machine for Channel x 00—No synchronization. 01—No comma (see XAUI state machine) and at least 1 cell value detected 10—Synchronization done. 11—Not used. XAUISTAT_x[0:1] = 00 on device reset.

Table 46. Memory Map (Continued)

(0x) Absolute Address	Bit	Name	Reset Value (0x)	Description
30805	[0]A [1]B [2]C [3]D	DEMUXWAS[A:D]	00	Status of Word Alignment. When DEMUX_WAS[A:D] = “1”, word alignment is achieved for Channel x. DEMUX_WAS[A:D]= “0” on device reset.
	[4]A [5]B [6]C [7]D	CH248_SYNC[A:D]		Status of Channel Alignment. When CH248_SYNC[A:D] = ”1”, multi-channel alignment is achieved for Channel x. CH248_SYNC[A:D]= “0” on device reset.
30814	[0:2]	Reserved		
	[3]A&B [4]C&D	SYNC2[1:2]OOSL		Multi-Channel Out-Of-Sync Status. When SYNC2[1:2] OOS= “1”, dual-channel synchronization has failed. SYNC2[1:2] OOS on device reset.
	[5]	SYNC4OOS		Multi-Channel Out-Of-Sync Status. When SYNC4_OOS= “1”, quad-channel synchronization has failed. SYNC4_OOS= “0” on device reset.
	[6:7]	Reserved		
SPI4 Core RX Control Registers (Read and Write)				
30900 A 30A00 B	[0:7]	RX_CAL_M_MAIN	00	Number of times calendar sequence is repeated between insertions of framing pattern (RX SPI4, main)
30901 A 30A01 B	[0:5]	Reserved		
	[6:7]	RX_CAL_LEN_MAIN	00	Length of calendar sequence on the RX status frame (SPI4, main) This is the most significant 2 bits of 10 total.
30902 A 30A02 B	[0:7]	RX_CAL_LEN_MAIN	00	Length of calendar sequence on the RX status frame (SPI4, main). This is the least significant 8 bits of 10 total
30903 A 30A03 B	[0:7]	RX_CAL_M_SHDW	00	Number of times shadow calendar sequence is repeated between insertions of framing pattern for the RX side of SPI4.
30904 A 30A04 B	[0:5]	Reserved		
	[6:7]	RX_CAL_LEN_SHDW	00	Shadow calendar length for the RX status frame of SPI4. Upper 2 bits.
30905 A 30A05	[0:7]	RX_CAL_LEN_SHDW	00	Shadow calendar length for the RX status frame of SPI4. Lower 8 bits.
30906 A 30A06 B	[0:1]	RX_DPRAM_0_NUMFIFO	00	Number of virtual FIFOs in RX DPRAM bank 0 00 = 1 FIFO 01 = 2 FIFOs 10 = 4 FIFOs 11 = 8 FIFOs
	[2:3]	RX_DPRAM_1_NUMFIFO		Number of virtual FIFOs in RX DPRAM bank 1
	[4:5]	RX_DPRAM_2_NUMFIFO		Number of virtual FIFOs in RX DPRAM bank 2
	[6:7]	RX_DPRAM_3_NUMFIFO		Number of virtual FIFOs in RX DPRAM bank 3

Table 46. Memory Map (Continued)

(0x) Absolute Address	Bit	Name	Reset Value (0x)	Description
30907 A 30A07 B	[0:1]	RX_DPRAM_0_AGGR_MODE	00	RX bank 0 aggregation mode
	[2:3]	RX_DPRAM_1_AGGR_MODE		RX bank 1 aggregation mode
	[4:5]	RX_DPRAM_2_AGGR_MODE		RX bank 2 aggregation mode
	[6:7]	RX_DPRAM_3_AGGR_MODE		RX bank 3 aggregation mode
30910 A 30A10 B	[0]	Reserved		
	[1:3]	RX_DIP4_ERR_TH	00	Number of consecutive bad/good DIP-4 code words for SPI4 RX to lose/gain alignment. Loss of alignment is indicated by ALGN_OFF_STS
	[4]	RX_DISABLE_STATUS		Causes 11 word to be sent on RX status frame. In turn causes far-end TX to cancel all credits and send a training pattern.
	[5]	RX_MASK_DIP4		Allows SPI4 link to be used in presence of errors on data input.
	[6]	RX_EN_TRAINING		Enable detection of training patterns on the RX side.
	[7]	RX_EN_BASELINE		Enable RX baseline process
30911 A 30A11 B	[0:7]	RX_FIFO_THRESHOLD_H	00	High watermark threshold for async FIFO within RDP.
30912 A 30A12 B	[0:3]	RX_DPRAM_FIFO_OVERRUN_INT_EN	00	These bits act as an enable for the status flags and the interrupts regarding overrun of the 4 RX DPRAM FIFO banks.
	[4:7]	RX_DPRAM_FIFO_URRUN_INT_EN		These bits act as an enable for the status flags and the interrupts regarding underrun of the 4 RX DPRAM FIFO banks.
30913 A 30A13 B	[0]	RX_PLL_LOL_INT_EN	00	This bit enables the status flag and the interrupt regarding loss of lock in the SPI4 core (high-speed macrocell) PLL. See RX_PLL_LOL_STS in register 3091C[A], 3091C[B].
	[1]	RX_BAS_DONE_INT_EN		This bit enables the status flag and the interrupt for the SPI4 core when RDI completes the baselining process. Baselining is the self-alignment power-up process that the macro does after it detects lock of the PLL. See RX_BAS_DONE_STS in register 3091C[A], 3091C[B].
	[2]	RX_BAS_ERR_INT_EN		This bit enables the status flag and the interrupt for the SPI4 core when RDI fails its internal self-alignment process. See RX_BAS_ERR_STS in register 3091C[A], 3091C[B].
	[3]	RX_DSKW_DONE_INT_EN		This bit enables the status flag and the interrupt for the SPI4 core when RDI completes its dynamic alignment process. See RX_DSKW_DONE_STS in register 3091C[A], 3091C[B].
	[4]	RX_DSKW_ERR_INT_EN		This bit enables the status flag and the interrupt for the SPI4 core when RDI fails to dynamically align. See RX_DSKW_ERR_STS in register 3091C[A], 3091C[B].
	[5]	RX_TRN_DET_INT_EN		This bit enables the status flag and the interrupt for the SPI4 core when RDI detects training patterns. See RX_TRN_DET_STS in register 3091C[A], 3091C[B].
	[6]	RX_ALGN_OFF_INT_EN		This bit enables the status flag and the interrupt for the SPI4 core loss of RX alignment due to excess consecutive DIP4 errors. See RX_ALGN_OFF_STS in register 3091C[A], 3091C[B].
	[7]	TX_STATUS_LOF_INT_EN		This bit enables the status flag and the interrupt for the SPI4 core TX status from having too many consecutive DIP2 errors. See TX_STATUS_LOF_STS

Table 46. Memory Map (Continued)

(0x) Absolute Address	Bit	Name	Reset Value (0x)	Description
SPI4 Core TX/RX Control Registers (Read and Write)				
30914 A 30A14 B	[0]	Reserved	00	
	[1]	Reserved		
	[2]	RX_ILLEGAL_CTL_INT_EN		This bit enables the status flag and the interrupt when the SPI4 receives unsupported extended control words.
	[3:4]	Reserved		
	[5]	RX_DIP4_INT_EN		This bit enables the status flag and the interrupt for the SPI4 core RX detection of a DIP4 error.
	[6]	TX_DIP2_INT_EN		This bit enables the status flag and the interrupt for the SPI4 core TX detection of a status framing error (either DIP2 or unexpected 11 pattern).
	[7]	RX_ASYNC_FIFO_OVERRUN_INT_EN		This bit enables the status flag and the interrupt for the SPI4 core asynchronous FIFO in RDP block being overrun.
30915 A 30A15 B	[0]	SPI4_QUARTER_RATE	00	When set to '1', enables data rates of 100 - 200 Mbps. The PLLs in the transmit and receive SPI4 high-speed blocks are bypassed in this mode.
	[1]	SPI4_LOOPBK_FE		This control enables far-end loopback when it is set to 1. Far-end loopback sends RDAT inputs back to TDAT outputs, sends RDCLK back to TDCLK, sends TSTAT back to RSTAT outputs, and ATSCLK back to RSCLK.
	[2]	SPI4_LOW_SPEED_DATA_SEL		Forces low speed data rates of (400-622 Mbits/s). The transmit PLL is still used to synthesize the SPI4 transmit clock TDCLK. However, the dynamic alignment block in the receive side is bypassed. No training sequences are used in low speed mode.
	[3]	SPI4_STATUS_IO_SEL		'0' - Selects LVTTL I/O for SPI4 status '1' - Selects LVDS I/Os for SPI4 status (Full-rate LVDS status I/Os specified by OIF SPI4.0 is not supported).
	[4]	SPI4_LOOPBK_HS		Enables loops from high-speed SPI4 TDAT outputs to RDAT inputs and RSTAT to TSTAT status inputs just before I/O.
	[5]	SPI4_LOOPBK_LS		Enables near-end parallel loop from TDP to RDP blocks and RSP to TSP blocks bypassing the high-speed SPI4 interface logic blocks. Must enable SPI4_LOOPBK_HS for this to work.
	[6]	TX_FORCE_DIP_ERR		Causes TDP to send bad parity code words.
	[7]	RX_FORCE_DIP2_ERR		Causes RSP to send bad parity code words.
30916 A 30A16 B	[0:3]	Reserved		
	[4]	RX_CAL_SW_EN	00	Inserts the calendar select word after the frame sync pattern on the receive SPI4 status channel
	[5]	RX_CAL_SEL		0 = Selects main calendar for outgoing RX status frame (calendar select word will be "01") 1 = Selects shadow calendar (calendar select word will be "10")
	[6]	TX_CAL_SW_EN		Detect the calendar select word after the frame sync pattern on the transmit status
	[7]	TX_BURST_TERMINATION		Mode control for managing the transmission of bursts when the DPRAM partition becomes empty in the middle of a burst. 0 =continue to try to access DPRAM for the remainder of burst. 1 = abort the burst.

Table 46. Memory Map (Continued)

(0x) Absolute Address	Bit	Name	Reset Value (0x)	Description
30917 A 30A17 B	[0:1]	Reserved		
	[2]	RX_CAL_MEM_SEL	00	Enables calendar memory for SPI4 RX calendar to be read or written via MPI. The RX_CAL memory encompasses address ranges 0 to 2047. Addresses 0 to 1023 are for the main calendar; addresses 1024 to 2047 are for the shadow calendar.
	[3]	RX_PDM_MEM_SEL		Enables memory for SPI4 RX port descriptors to be read or written via MPI. Addresses 0 to 255 valid for this memory. Each location contains 3 unused bits, 2 BANK_ID bits and 3 PARTITION_ID bits.
	[4]	TX_CAL_MEM_SEL		Enables calendar memory for SPI4 TX calendar to be read or written via MPI. The TX_CAL memory encompasses address ranges 0 to 2047. Addresses 0 to 1023 are for the main calendar; addresses 1024 to 2047 are for the shadow calendar.
	[5]	TX_PDM_MEM_SEL		Enables memory for SPI4 TX port descriptors to be read or written via MPI. The valid address range is 0 to 1023. The least significant 2 address bits are used as byte enables. 00 accesses the PORT_ID (8 bits) 01 accesses PARTITION_ID (3 bits), an unused bit and BURST_VAL (4 bits) 10 accesses 4 unused bits, MB_EN (1 bit), M (1 bit), and BANK_ID (2 bits). The upper 8 bits of address represent the port number being configured.
	[6]	TX_CRED_MEM_SEL		Enables read and write access to TX credits via MPI. The valid memory address range is 0 to 511. The credit for each port is a 10-bit entity. The upper 8 of 9 bits of address select which port to read or write. The lowest bit of address acts as a byte select as follows: 0 accesses the least significant 8 bits of the credit field 1 accesses the most significant 2 bits (right-justified) of the credit field
	[7]	TX_STAT_MEM_SEL		Enables memory for SPI4 TX status to be read or written via MPI. This memory used address ranges 0 to 255. Each location has six left-most bits unused and two status bits.
30920 A 30A20 B	[0:3}	TX_DPRAM_FULL_TYPE_SEL	00	Full level select for all partitions within bank [0:3]. Valid for 32-bit, 64-bit and 128-bit mode. 0 = truly full 1 = 3/4 FIFO lines full +1. E.g.: For a FIFO size of 32, this value will be (3/4 * 32) + 1 = 25 lines. Each line in the FIFO is always 128 bits of data irrespective of the aggregation mode
	[4:7]	RX_DPRAM_EMPTY_TYPE_SEL		Empty level select for all partitions within bank [0:3]. Valid for 32-bit, 64-bit and 128-bit mode. 0 = truly empty 1 = 1/4 full - 1. E.g.: For a FIFO size of 32, this value will be (1/4 * 32) - 1 = 7 lines. Each line in the FIFO is always 128 bits of data irrespective of the aggregation mode
SPI4 Core TX Control Registers (Read and Write)				
30921 A 30A21 B	[0:7]	TX_CAL_M_MAIN	00	Number of times the transmit main calendar sequence is repeated between insertions of framing pattern.
30922 A 30A22 B	[0:5]	Reserved		
	[6:7]	TX_CAL_LEN_MAIN	00	Length of main calendar sequence on the TX status frame. These are the most significant 2 bits of 10 total.
30923 A 30A23	[0:7]	TX_CAL_LEN_MAIN		Length of main calendar sequence on the TX status frame. These are the least significant 8 bits of 10 total.
30924 A 30A24 B	[0:7]	TX_CAL_M_SHDW	00	Number of times the transmit shadow calendar sequence is repeated between insertions of framing pattern.

Table 46. Memory Map (Continued)

(0x) Absolute Address	Bit	Name	Reset Value (0x)	Description
30925 A 30A25 B	[0:5]	Reserved		
	[6:7]	TX_CAL_LEN_SHDW	00	Shadow calendar length for the transmit status frame of SPI4. Upper 2 bits
30926 30A26	[0:7]	TX_CAL_LEN_SHDW		Shadow calendar length for the transmit status frame of SPI4. Lower 8 bits
30927 A 30A27 B	[0:1]	TX_DPRAM_0_NUMFIFO	00	Number of virtual FIFOs in TX DPRAM bank 0 00 = >1 FIFO 01 = >2 FIFOs 10 = >4 FIFOs 11 = >8 FIFOs
	[2:3]	TX_DPRAM_1_NUMFIFO		Number of virtual FIFOs in TX DPRAM bank 1
	[4:5]	TX_DPRAM_2_NUMFIFO		Number of virtual FIFOs in TX DPRAM bank 2
	[6:7]	TX_DPRAM_3_NUMFIFO		Number of virtual FIFOs in TX DPRAM bank 3
30930 A 30A30 B	[0:1]	TX_DPRAM_0_AGGR_MODE	00	TX bank 0 aggregation mode 00 = 32 bit data 01 = 64-bit data 10 = 128-bit data 11 = not valid
	[2:3]	TX_DPRAM_1_AGGR_MODE		TX bank 1 aggregation mode
	[4:5]	TX_DPRAM_2_AGGR_MODE		TX bank 2 aggregation mode
	[6:7]	TX_DPRAM_3_AGGR_MODE		TX bank 3 aggregation mode
30931 A 30A31 B	[0:5]	Reserved		
	[6:7]	TX_MAX_BURST1	00	Maximum number of 16-byte bursts expected for any port when its status is HUNGRY. These are the 2 most significant bits of a 10-bit number
30932 A 30A32	[0:7]	TX_MAX_BURST1	00	Maximum number of 16-byte bursts expected for any port when its status is HUNGRY. These are the 8 least significant bits of a 10-bit number
30933 A 30A33 B	[0:5]	Reserved		
	[6:7]	TX_MAX_BURST2		Maximum number of 16-byte bursts expected for any port when its status is STARVING. These are the 2 most significant bits of a 10-bit number
30934 A 30A34 B	[0:7]	TX_MAX_BURST2	00	Maximum number of 16-byte bursts expected for any port when its status is STARVING. These are the 8 least significant bits of a 10-bit number
30935 A 30A35 B	[0:7]	TX_DATA_MAX_T (byte a)	00	Maximum interval between training sequences on the transmit data interface. A 32-bit number is formed from 4 bytes organized as {a, b, c, d} and read left to right. When this number is “0”, training sequences are disabled.
30936 A 30A36 B	[0:7]	TX_DATA_MAX_T (byte b)	00	Maximum interval between training sequences on the transmit data interface. A 32-bit number is formed from 4 bytes organized as {a, b, c, d} and read left to right. When this number is “0”, training sequences are disabled.
30937 A 30A37 B	[0:7]	TX_DATA_MAX_T (byte c)	00	Maximum interval between training sequences on the transmit data interface. A 32-bit number is formed from 4 bytes organized as {a, b, c, d} and read left to right. When this number is “0”, training sequences are disabled.
30940 A 30A40 B	[0:7]	TX_DATA_MAX_T (byte d)	00	Maximum interval between training sequences on the transmit data interface. A 32-bit number is formed from 4 bytes organized as {a, b, c, d} and read left to right. When this number is "0", training sequences are disabled.

Table 46. Memory Map (Continued)

(0x) Absolute Address	Bit	Name	Reset Value (0x)	Description
30941A 30941 B	[0:7]	TX_ALPHA (byte a)	00	Number of times training sequence needs to be repeated every TX_DATA_MAX_T cycles. The number is formed from 2 ALPHA bytes, in the order {a,b}.
30942 A 30A42 B	[0:7]	TX_ALPHA (byte a)	00	Number of times training sequence needs to be repeated every TX_DATA_MAX_T cycles. The number is formed from 2 ALPHA bytes, in the order {a,b}.
30943 A 30A43 B	[0:3]	TX_DPRAM_FIFO_OVERRUN_INT_EN	00	These bits act as enables for the status flags and the interrupts regarding an overrun of the 4 TX DPRAM FIFO banks.
	[4:7]	TX_DPRAM_FIFO_URRUN_INT_EN	00	These bits act as enables for the status flags and the interrupts regarding an underrun of the 4 TX DPRAM FIFO banks.
30944 A 30A44 B	[0:4]	Reserved		
	[5:7]	TX_DIP2_ERR_TH	00	Number of consecutive bad/good DIP-2 code words for SPI4 TX to lose/gain alignment. Loss of alignment is indicated by TX_STATUS_LOF_STS
30945 A 30A45 B	[0:2]	Reserved		
	[3:7]	TX_FIFO_THRESHOLD_H	00	High watermark threshold for async FIFO within TDP to declare a full condition. This causes writes to the TDP to be held off; meaning that an overrun condition can not occur here.
SPI4 Core RX/TX Spare Controls (Read and Write)				
30496 A 30A46 B	[0:7]	Reserved	00	
30908 A 30A08 B	[0:7]	Reserved	00	
SPI4 Core Status Registers (Read Only)				
30909 A 30A09 B	[0:7]	Reserved		
3090A A 30A0A B	[0:7]	RX_DIP4_ERR_CNT	00	Count of DIP4 errors. Count will reset to “0” on a read of this register. If count reaches maximum (255) it will hold.
3090B A 30A0B B	[0:7]	TX_DIP2_ERR_CNT	00	Count of status framing errors. See SIP2_ERR_STS. Count will reset to “0” after a read of this register. If count reaches maximum (255) it will hold.
3090C A 30A0C B	[0:7]	RX_DPRAM_0_FIFO_OVERRUN_STS	00	Overrun status of bank 0 DPRAM FIFOs 0 to 7. These are enabled by RX_DPRAM_FIFO_OVERRUN_INT_EN[0] (register 30912[A], 30A12[B]). These bits clear when read, but will set immediately if the error condition persists.
3090D A 30A0D B	[0:7]	RX_DPRAM_1_FIFO_OVERRUN_STS	00	Overrun status of bank 1 DPRAM FIFOs 0 to 7. These are enabled by RX_DPRAM_FIFO_OVERRUN_INT_EN[1] (register 30912[A], 30A12[B]). These bits clear when read, but will set immediately if the error condition persists.
3090E A 30A0E B	[0:7]	RX_DPRAM_2_FIFO_OVERRUN_STS	00	Overrun status of bank 2 DPRAM FIFOs 0 to 7. These are enabled by RX_DPRAM_FIFO_OVERRUN_INT_EN[2] (register 30912[A], 30A12[B]). These bits clear when read, but will set immediately if the error condition persists.
3090F A 30A0F B	[0:7]	RX_DPRAM_3_FIFO_OVERRUN_STS	00	Overrun status of bank 3 DPRAM FIFOs 0 to 7. These are enabled by RX_DPRAM_FIFO_OVERRUN_INT_EN[3] (register 30912[A], 30A12[B]). These bits clear when read, but will set immediately if the error condition persists.
30918 A 30A18 B	[0:7]	RX_DPRAM_0_FIFO_URRUN_STS	00	Underrun status of bank 0 DPRAM FIFOs 0 to 7. These are enabled by RX_DPRAM_FIFO_URRUN_INT_EN[0] (register 30912[A], 30A12[B]). These bits clear when read, but will set immediately if the error condition persists.

Table 46. Memory Map (Continued)

(0x) Absolute Address	Bit	Name	Reset Value (0x)	Description
30919A 30A19B	[0:7]	RX_DPRAM_1_FIFO_URRUN_STS	00	Underrun status of bank 1 DPRAM FIFOs 0 to 7. These are enabled by RX_DPRAM_FIFO_URRUN_INT_EN[1] (register 30912[A], 30A12[B]). These bits clear when read, but will set immediately if the error condition persists.
3091A A 30A1A B	[0:7]	RX_DPRAM_2_FIFO_URRUN_STS	00	Underrun status of bank 2 DPRAM FIFOs 0 to 7. These are enabled by RX_DPRAM_FIFO_URRUN_INT_EN[2] (register 30912[A], 30A12[B]). These bits clear when read, but will set immediately if the error condition persists.
3091B A 30A1B B	[0:7]	RX_DPRAM_3_FIFO_URRUN_STS	00	Underrun status of bank 3 DPRAM FIFOs 0 to 7. These are enabled by RX_DPRAM_FIFO_URRUN_INT_EN[3] (register 30912[A], 30A12[B]). These bits clear when read, but will set immediately if the error condition persists.
3091C A 30A1C B	[0]	RX_PLL_LOL_STS	00	Status flag for loss of lock in PLL within the SPI4 receive core. This flag is enabled by RX_PLL_LOL_INT_EN (register 30913[A], 30A13[B]).
	[1]	RX_BAS_DONE_STS		Status flag for SPI4 receive core completing the baselining process. Baselining is the self-alignment power-up process that the macro does after it detects lock of the PLL. This flag is enabled by RX_BAS_DONE_INT_EN. (register 30913[A], 30A13[B])
	[2]	RX_BAS_ERR_STS		Status flag for SPI4 receive core failing its internal self-alignment process. This flag is enables by RX_BAS_INT_EN (register 30913[A], 30A13[B]).
	[3]	RX_DSKW_DONE_STS		Status flag for SPI4 receive dynamic alignment complete. This flag is enabled by RX_DSKW_DONE_INT_EN.
	[4]	RX_DSKW_ERR_STS		Status flag for SPI4 receive core failing to dynamically align. This flag is enabled by RX_DSKW_ERR_INT_EN (register 30913[A], 30A13[B]).
	[5]	RX_TRN_DET_STS		Status flag for SPI4 receive core detecting training patterns. This flag is enabled by RX_TRN_DET_INT_EN (register 30913[A], 30A13[B]).
	[6]	RX_ALGN_OFF_STS		Status flag for loss of data alignment due to excess consecutive DIP4 errors. This flag is enabled by RX_ALIGN_OFF_INT_EN (register 30913[A], 30A13[B]).
	[7]	TX_STATUS_LOF_STS		Status flag for TX status frame having too many consecutive DIP2 errors. This flag is enabled by TX_STATUS_LOF_INT_EN (register 30913[A], 30A13[B]).
3091D A 30A1D B	[0:4]	Reserved		
	[5]	RX_DIP4_ERR_STS	00	Status flag for DIP4 error
	[6]	TX_DIP2_ERR_STS		Flag for error in framing TSTAT[0:1] inputs. This can be an unexpected 11 pattern or a DIP2 error.
	[7]	RX_ASYNC_FIFO_OVERRUN_STS		Status flag for RX FIFO overrun inside S4RDP block.
SPI4 Core TX Status Registers (Read and Write)				
30928 A 30A28 B	[0:7]	TX_DPRAM_0_FIFO_OVERRUN_STS	00	Overrun status of bank 0 TX DPRAM FIFOs 0 to 7. These are enabled by TX_DPRAM_FIFO_OVERRUN_INT_EN[0]. These bits clear when read, but will set immediately if the error condition persists.
30929 A 30A29 B	[0:7]	TX_DPRAM_1_FIFO_OVERRUN_STS	00	Overrun status of bank 1 TX DPRAM FIFOs 0 to 7. These are enabled by TX_DPRAM_FIFO_OVERRUN_INT_EN[1]. These bits clear when read, but will set immediately if the error condition persists.
3092A A 30A2A B	[0:7]	TX_DPRAM_2_FIFO_OVERRUN_STS	00	Overrun status of bank 2 TX DPRAM FIFOs 0 to 7. These are enabled by TX_DPRAM_FIFO_OVERRUN_INT_EN[2]. These bits clear when read, but will set immediately if the error condition persists.
3092B A 30A2B B	[0:7]	TX_DPRAM_3_FIFO_OVERRUN_STS	00	Overrun status of bank 3 TX DPRAM FIFOs 0 to 7. These are enabled by TX_DPRAM_FIFO_OVERRUN_INT_EN[3]. These bits clear when read, but will set immediately if the error condition persists.

Table 46. Memory Map (Continued)

(0x) Absolute Address	Bit	Name	Reset Value (0x)	Description
3092C A 30A2C B	[0:7]	TX_DPRAM_0_FIFO_URRUN_STS	00	Underrun status of bank 0 DPRAM FIFOs 0 to 7. These are enabled by TX_DPRAM_FIFO_URRUN_INT_EN[0]. These bits clear when read, but will set immediately if the error condition persists.
3092D A 30A2D B	[0:7]	TX_DPRAM_1_FIFO_URRUN_STS	00	Underrun status of bank 1 DPRAM FIFOs 0 to 7. These are enabled by TX_DPRAM_FIFO_URRUN_INT_EN[1]. These bits clear when read, but will set immediately if the error condition persists.
3092E A 30A2E B	[0:7]	TX_DPRAM_2_FIFO_URRUN_STS	00	Underrun status of bank 2 DPRAM FIFOs 0 to 7. These are enabled by TX_DPRAM_FIFO_URRUN_INT_EN[A:B][2]. These bits clear when read, but will set immediately if the error condition persists.
3092F A 30A2F B	[0:7]	TX_DPRAM_3_FIFO_URRUN_STS	00	Underrun status of bank 3 DPRAM FIFOs 0 to 7. These are enabled by TX_DPRAM_FIFO_URRUN_INT_EN[3]. These bits clear when read, but will set immediately if the error condition persists.
Memory Controller Control Registers (Read and Write)				
30B00	[0:3]	MC_FULL_THRESHOLD	00	This 4-bit number sets the threshold for the MC_WFIFO_FULL flag for the write data FIFO. Example: if the field is 0100, then the MC_WFIFO_FULL flag will be raised when there are 4 (or fewer) filled slots remaining in the FIFO.
	[4:7]	MC_EMPTY_THRESHOLD		This 4-bit number sets the threshold for the MC_RFIFO_EMPTY flag for the read data FIFO. Example: if the field is 1010, then the MC_RFIFO_EMPTY flag will be raised when there are 10 (or fewer) filled slots remaining in the FIFO.
30B01	[0]	MC_RD_DFIFO_URRUN_INT_EN	00	Enable for Memory Controller read data FIFO underrun status flag MC_RD_DFIFO_URRUN_STS, and associated interrupt MEM_CTRL1_INT.
	[1]	MC_RD_IFIFO_URRUN_INT_EN		Enable for Memory Controller read instruction FIFO underrun status flag MC_RD_IFIFO_URRUN_STS, and associated interrupt MEM_CTRL1_INT.
	[2]	MC_WR_DFIFO_URRUN_INT_EN		Enable for Memory Controller write data FIFO underrun status flag MC_WR_DFIFO_URRUN_STS, and associated interrupt MEM_CTRL1_INT.
	[3]	MC_WR_IFIFO_URRUN_INT_EN		Enable for Memory Controller write instruction FIFO underrun status flag MC_WR_IFIFO_URRUN_STS, and associated interrupt MEM_CTRL1_INT.
	[4]	MC_RD_DFIFO_OVERRUN_INT_EN		Enable for Memory Controller read data FIFO overrun status flag MC_RD_DFIFO_OVERRUN_STS, and associated interrupt MEM_CTRL1_INT. HW_Issue 23 indicates this should never happen.
	[5]	MC_RD_IFIFO_OVERRUN_INT_EN		Enable for Memory Controller read instruction FIFO overrun status flag MC_RD_IFIFO_OVERRUN_STS, and associated interrupt MEM_CTRL1_INT.
	[6]	MC_WR_DFIFO_OVERRUN_INT_EN		Enable for Memory Controller write data FIFO overrun status flag MC_WR_DFIFO_OVERRUN_STS, and associated interrupt MEM_CTRL1_INT.
	[7]	MC_WR_IFIFO_OVERRUN_INT_EN		Enable for Memory Controller write instruction FIFO overrun status flag MC_WR_IFIFO_OVERRUN_STS, and associated interrupt MEM_CTRL1_INT.
30B02	[0:6]	Reserved		
	[7]	MC_ID_ERR_INT_EN		Enable for incoherent data and instruction words status flag MC_ID_ERR_STS, and associated interrupt MEM_CTRL2_INT.

Table 46. Memory Map (Continued)

(0x) Absolute Address	Bit	Name	Reset Value (0x)	Description
30B03	[0]	MC_BURST_MODE	00	0 = 2-word burst 1 = 4-word burst
	[1]	Reserved		
	[2:3]	MC_ICK_SEL		Input clock selector: 00 = MCREFCLK (HSTL) 01 = BTREFCLK (SPIA reference, LVTTL) 10 = ATREFCLK (SPIB reference, LVTTL) 11 = F_MC_REFCLK (FPGA)
	[4:6]	Reserved		
	[7]	MC_OCK_SEL	00	Selects between input clock divided by 2, or PLL output for source of PMIC/PMICN: 0 = input clock 1 = PLL
30B04	[0]	Reserved		
	[1:3]	PLL_N	00	Numerator of multiplier factor for frequency of PLL output clock: CK2X = MC_ICK * (PLL_N / PLL_M)
	[4]	Reserved		
	[5:7]	PLL_M	00	Denominator of multiplier factor for frequency of PLL Output clock: CK2X = MC_ICK * (PLL_N / PLL_M)
30B05	[0:7}	Reserved		
30B06	[0:7]	Reserved		
Memory Controller Status Registers (Read Only)				
30B08	[0]	MC_RD_DFIFO_URRUN_STS	00	Status flag for read data FIFO underrun. Enabled by MC_RD_DFIFO_URRUN_INT_EN
	[1]	MC_RD_IFIFO_URRUN_STS		Status flag for read instruction FIFO underrun. Enabled by MC_RD_IFIFO_URRUN_INT_EN
	[2]	MC_WR_DFIFO_URRUN_STS		Status flag for write data FIFO underrun. Enabled by MC_WR_DFIFO_URRUN_INT_EN
	[3]	MC_WR_IFIFO_URRUN_STS		Status flag for write instruction FIFO underrun. Enabled by MC_WR_IFIFO_URRUN_INT_EN
	[4]	MC_RD_DFIFO_OVERRUN_STS		Status flag for read data FIFO overrun. Enabled by MC_RD_DFIFO_OVERRUN_INT_EN
	[5]	MC_RD_IFIFO_OVERRUN_STS		Status flag for read instruction FIFO overrun. Enabled by MC_RD_IFIFO_OVERRUN_INT_EN
	[6]	MC_WR_DFIFO_OVERRUN_STS		Status flag for write data FIFO overrun. Enabled by MC_WR_DFIFO_OVERRUN_INT_EN
	[7]	MC_WR_IFIFO_OVERRUN_STS		Status flag for write instruction FIFO overrun. Enabled by MC_WR_IFIFO_OVERRUN_INT_EN
30B09	[0:6]	Reserved		
	[7]	MC_ID_ERR_STS	00	Status flag for incoherent data and instruction words. Enabled by MC_ID_ERR_INT_EN.
30B0A	[0:7]	Reserved		

Table 46. Memory Map (Continued)

(0x) Absolute Address	Bit	Name	Reset Value (0x)	Description
Common Control Registers (Read and Write)				
30B20	[0:3]	Reserved		
	[4]	HARD_RESET_FC	00	Hard reset for SERDES block. When set to 1, disabled SERDES logic by resetting it. Same functionality as interface signal FPGA_RESET_FC
	[5]	SOFT_RESET_MC		Software reset for Memory Controller block
	[6]	SOFT_RESET_S4A		Software reset for SPIA block
	[7]	SOFT_RESET_S4B		Software reset for SPIB block
30B21	[0:6]	COM_SPARE_C	00	Spare common control bits
	[7]	FORCE_INT		Force all the enabled status and interrupt bits
Common Status Registers (Read and Write)				
30B28	[0:7]	Reserved		Reserved
30B29	[0]	SERDES_INT	00	Signals that an enabled interrupt has come from the SERDES. This is cleared when read and will only reassert when the underlying event goes away and comes back.
	[1]	MEM_CTRL2_INT		Signals that an enabled interrupt has come from the Memory Controller other than the FIFOs. This is cleared when read and will only reassert when the underlying event goes away and comes back.
	[2]	MEM_CTRL1_INT		Signals that an enabled interrupt has come from the Memory Controller FIFOs. This is cleared when read and will only reassert when the underlying event goes away and comes back.
	[3]	DPRAM_INT		Signals that an enabled interrupt has come from one of the DPRAM FIFO banks. This is cleared when read and will only reassert when the underlying event goes away and comes back.
	[4]	SPI_B_COR2_INT		Signals that an enabled interrupt has come from the register 30A1D. This is cleared when read and will only reassert when the underlying event goes away and comes back.
	[5]	SPI_B_COR1_INT		Signals that an enabled interrupt has come from register 30A1C. This is cleared when read and will only reassert when the underlying event goes away and comes back.
	[6]	SPI_A_COR2_INT		Signals that an enabled interrupt has come from register 3091D. This is cleared when read and will only reassert when the underlying event goes away and comes back.
	[7]	SPI_A_COR1_INT		Signals that an enabled interrupt has come from register 3091C. This is cleared when read and will only reassert when the underlying event goes away and comes back.

Table 46. Memory Map (Continued)

(0x) Absolute Address	Bit	Name	Reset Value (0x)	Description
30B2A	[0]	TX_INT_B	00	Signals that an enabled interrupt has come from TX DPRAM FIFOs for SPI4 core B. This is cleared when read and will only reassert when the underlying event goes away and comes back.
	[1]	RX_INT_B		Signals that an enabled interrupt has come from RX DPRAM FIFOs for SPI4 core B. This is cleared when read and will only reassert when the underlying event goes away and comes back.
	[2]	TX_INT_A		Signals that an enabled interrupt has come from TX DPRAM FIFOs for SPI4 core A. This is cleared when read and will only reassert when the underlying event goes away and comes back.
	[3]	RX_INT_A		Signals that an enabled interrupt has come from RX DPRAM FIFOs for SPI4 core B. This is cleared when read and will only reassert when the underlying event goes away and comes back.
	[4]	BANK_3_INT		Signals that an enabled interrupt has come from DPRAM bank 3 FIFOs (or both SPI4 blocks, both TX and RX flags). This is cleared when read and will only reassert when the underlying event goes away and comes back.
	[5]	BANK_2_INT		Signals that an enabled interrupt has come from DPRAM bank 2 FIFOs (or both SPI4 blocks, both TX and RX flags). This is cleared when read and will only reassert when the underlying event goes away and comes back.
	[6]	BANK_1_INT		Signals that an enabled interrupt has come from DPRAM bank 1 FIFOs (or both SPI4 blocks, both TX and RX flags). This is cleared when read and will only reassert when the underlying event goes away and comes back.
	[7]	BANK_0_INT		Signals that an enabled interrupt has come from DPRAM bank 0 FIFOs (or both SPI4 blocks, both TX and RX flags). This is cleared when read and will only reassert when the underlying event goes away and comes back.
30B2B	[0:7]	COM_SPARE_S	00	Spare common status bits
Embedded Control Memory Access (31000-31FFF)				
31000	[0:7]	data[7:0]	00	Embedded control memories are accessed when register addresses 3100-31FFF are written or read. The address passed to the embedded memories is the lower 12 bits of the register address. Writing to a register address in this range causes data to be transferred to the selected embedded memories. More than one memory can be written with a single operation by having multiple select bits active. The select bits are 30917 and 30A17 registers. Reading from a register address in this range causes the data in the memory selected to appear. The following priority rules apply to the selection of which memory is read from. (1) 30917 bits take precedence over bits in the 30A17 register. (2) lower bit numbers take precedence over higher bit numbers within a register.

Absolute Maximum Ratings

Stresses in excess of the absolute maximum ratings can cause permanent damage to the device. These are absolute stress ratings only. Functional operation of the device is not implied at these or any other conditions in excess of those given in the operations sections of this data sheet. Exposure to absolute maximum ratings for extended periods can adversely affect device reliability.

The ORCA Series 4 FPSCs include circuitry designed to protect the chips from damaging substrate injection currents and to prevent accumulations of static charge. Nevertheless, conventional precautions should be observed during storage, handling, and use to avoid exposure to excessive electrical stress

Table 47. Absolute Maximum Ratings.

Parameter	Symbol	Min.	Max.	Unit
Storage Temperature	T_{STG}	-65	150	°C
Power Supply Voltage with Respect to Ground	V_{DD33}	- 0.3	4.2	V
	$V_{DD33\_FPGAPLL}$	-0.3	4.2	V
	V_{DD15}	-0.3	2.0	V
	V_{DDIO}	- 0.3	4.2	V
SPI4 Voltages with respect to Ground	V_{DDA_SPIA}	-0.3	2.0	V
	V_{DDA_SPIB}	-0.3	2.0	V
	V_{DDA_SPIC}	-0.3	2.0	V
	V_{DDA_SPID}	-0.3	2.0	V
Memory Controller Voltages with respect to Ground	V_{DDH}	-0.3	2.0	V
	V_{DD_PLL}	-0.3	4.2	V
SERDES Supply Voltages	V_{DD_ANA}	-0.3	2.0	V
	V_{DDGB}	-0.3	2.0	V
FPGA Input Signal with Respect to Ground	V_{IN}	V_{SS} – 0.3	V_{DD}IO + 0.3	V
FPGA Signal Applied to High-impedance Output	—	V_{SS} – 0.3	V_{DD}IO + 0.3	V
Maximum Package Body (Soldering) Temperature	—	—	220	°C

Recommended Operating Conditions

Table 48. Recommended Operating Conditions

Parameter	Symbol	Min.	Max.	Unit
Power Supply Voltage with respect to Ground[1]	V_{DD33}	3.0	3.6	V
	$V_{DD33\_FPGAPLL}$	3.0	3.6	V
	V_{DD15}	1.425	1.575	V
SPI4 Voltages with respect to Ground	V_{DDA_SPIA}	1.425	1.575	V
	V_{DDA_SPIB}	1.425	1.575	V
	V_{DDA_SPIC}	1.425	1.575	V
	V_{DDA_SPID}	1.425	1.575	V
Memory Controller Voltages with respect to Ground	V_{DDH}[2]	1.425	1.575	V
	REF_1	0.68	0.9	V
	REF_2	0.68	0.9	V
	REF_3	0.68	0.9	V
	REF_4	0.68	0.9	V
	V_{DDA_PLL}	3.0	3.6	V
SERDES Supply Voltage	V_{DD_ANA}	1.425	1.575	V
	V_{DDGB}	1.425	1.575	V
SERDES CML I/O Supply Voltages	V_{DDIB}	1.425	1.89	V
	V_{DDOB}	1.425	1.89	V
FPGA Input Voltage	$V_{\_IN}$	V_{ss}-0.3	V_{DDIO} to +0.3	V
Memory Controller Input Voltage[2]	$V_{\_IWMCTRL}$	V_{ss}-0.3	1.89	V
Junction Temperature	T_J	– 40	125	°C

1. For FPGA Recommended Operating Conditions and Electrical Characteristics, see the Recommended Operating Conditions and Electrical Characteristics tables in the ORCA Series 4 FPGA data sheet (OR4E06) and the ORCA Series 4 I/O Buffer Technical Note. FPSC Standby Currents (IDDSB15 and IDDSB33) are tested with the Embedded Core in the powered down state.

2 Memory Controller 1.8V tolerant with VDDH = 1.5V ± 5%

ORSPI4 Power Tables

Table 49. ORSPI4 SPI4 Worst Case (per interface) Power Table

Parameter	Reference Clock Frequency	VDD15 VDDA_SPI[A,D]	VDD33	Total Power	Unit
500 MHz (1 Gbps) with Dynamic Alignment					
Frequency	125 MHz				
Total		1.123	1.030	2.15	W
425 MHz (850 Mbps) with Dynamic Alignment					
Frequency	106.25 MHz				
Total		0.996	0.952	1.95	W
350 MHz (700 Mbps) with Dynamic Alignment					
Frequency	87.5 MHz				
Total		0.869	0.828	1.70	W
350 MHz (700 Mbps) with Static Alignment					
Frequency	87.5 MHz				
Total		0.761	0.704	1.47	W
311 MHz (622 Mbps) with Static Alignment					
Frequency	156 MHz				
Total		0.543	0.468	1.01	W
100 MHz (200 Mbps) Quarter Rate					
Frequency	200 MHz				
Total		0.173	0.331	0.50	W
78 MHz (156 Mbps) Quarter Rate					
Frequency	156 MHz				
Total		0.102	0.181	0.28	W

Temperature: -40°C to 125°C, Power Supplies: VDD15, VDDA_SPI[A,D] = 1.575V, VDD33 = 3.6V, Data Pattern: PRBS 2^31-1

Table 50. ORSPI4 QDR Memory Controller Worst Case Power Table

Parameter	Clock Frequency	VDD15 VDDH	VDD33	Total Power	Units
Frequency	200 MHz				
Total		291.38	64.80	356.18	mW
Frequency	160 MHz				
Total		226.80	46.80	273.60	mW

Note: In many applications the PLL can be disabled to reduce power.
Temperature: -40°C to 125°C, Power Supplies: VDD15, VDDH = 1.575V, VDD33 = 3.6V, Data Pattern: PRBS 2^31-1

Table 51. ORSPI4 SERDES Worst Case Power Table

Parameter	Operating Frequency	VDD15[1]	Units
SERDES, MUX/DEMUX, Align FIFO and I/O (per channel)	1.25 GHz	195	mW
SERDES, MUX/DEMUX, Align FIFO and I/O (per channel)	2.5 GHz	210	mW

Parameter	Operating Frequency	VDD15[1]	Units
SERDES, MUX/DEMUX, Align FIFO and I/O (per channel)	3.125 GHz	225	mW
8b/10b Encoder/Decoder (per channel)	3.125 GHz	50	mW

1. With all channels operating, Temperature: -40°C to 125°C, Power Supplies: VDDA, VDDB, VDDIB = 1.575V, Data Pattern: PRBS 2^31-1

SPI4 Electrical and Timing Characteristics

SPI4 LVDS I/O

Table 52. Driver DC Data*

Parameter	Symbol	Test Conditions	Min.	Typ.	Max.	Units
Output Voltage High, VOA or VOB	VOH	RLOAD = 100 Ω ± 1%	—	—	1.475 †	V
Output Voltage Low, VOA or VOB	VOL	RLOAD = 100 Ω ± 1%	0.925†	—	—	V
Output Differential Voltage	|VOD|	RLOAD = 100 Ω ± 1%	0.25	—	0.45 †	V
Output Offset Voltage	VOS	RLOAD = 100 Ω ± 1%	1.125*	—	1.275 †	V
Output Impedance, Differential	RO	VCM = 1.0 V and 1.4 V	80	100	120	W
RO Mismatch Between A and B	Δ RO	VCM = 1.0 V and 1.4 V	—	—	10	%
Change in Differential Voltage Between Complementary States	|Δ VOD|	RLOAD = 100 Ω ± 1%	—	—	25	mV
Change in Output Offset Voltage Between Complementary States	Δ VOS	RLOAD = 100 Ω ± 1%	—	—	25	mV
Output Current	ISA, ISB	Driver shorted to GND	—	—	24	mA
Output Current	ISAB	Drivers shorted together	—	—	12	mA
Power-off Output Leakage	|Ixa|, |Ixb|	VDD = 0 V VPAD, VPADN = 0 V—2.5 V	—	—	10	mA

1. VDD33 = 3.1 V—3.5 V, VDD15 = 1.4 V—1.6 V, –40 ˚C.
2. External reference, REF10 = 1.0 V ± 3%, REF14 = 1.4 V ± 3%.

Table 53. Receiver DC Data[1]

Parameter	Symbol	Test Conditions	Min.	Typ.	Max.	Units
Input Voltage Range, VIA or VIB	VI	|VGPD| < 925 mV DC – 1 MHz	0.0	1.2	2.4	V
Input Differential Threshold	VIDTH	|VGPD| < 925 mV 450 MHz	–100	—	100	mV
Input Differential Hysteresis	VHYST	(+VIDTHH) – (–VIDTHL)	25	—	—	mV
Receiver Differential Input Impedance	RIN	With build-in termination, center-tapped	80	100	120	Ω

1. VDD = 3.1V - 3.5V, 0 °C -125 °C.

Table 54. SPI4 LVDS Operating Parameters

Parameter	Test Conditions	Min.	Normal	Max.	Units
Transmit Termination Resistor	—	80	100	120	Ω.
Receiver Termination Resistor	—	80	100	120	Ω.

Note: Under worst-case operating conditions, the LVDS driver will withstand a disabled or unpowered receiver for an unlimited period of time without being damaged. Similarly, when outputs are short-circuited to each other or to ground, the LVDS will not suffer permanent damage. The LVDS driver supports hot insertion. Under a well-controlled environment, the LVDS I/O can drive backplane as well as cable.

Figure 78. Output Buffer Delays

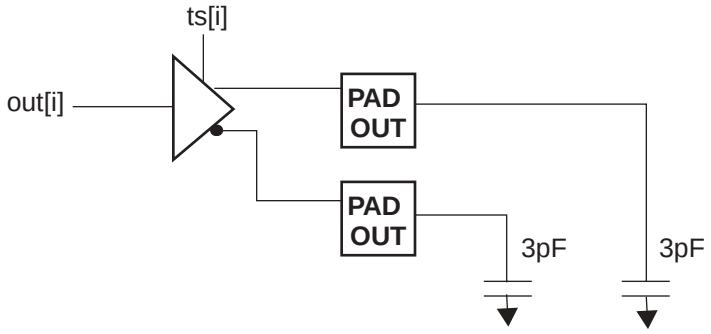


Table 55. LVDS Driver AC Data[1]

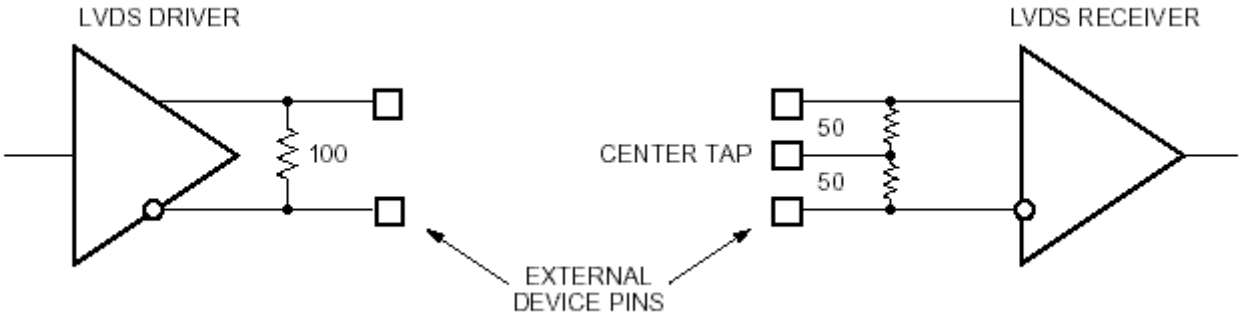
Parameter	Symbol	Test Conditions	Min.	Typ.	Max.	Units
V_{OD} Fall Time, 80% to 20%	t_F	Z_L = 100 Ω ± 1% C_{PAD} = 3.0 pF, C_{PAD} = 3.0 pF	100	—	210	ps
V_{OD} Rise Time, 20% to 80%	t_R	Z_L = 100 Ω ± 1% C_{PAD} = 3.0 pF, C_{PAD} = 3.0 pF	100	—	210	ps
Differential Skew |tPHLA – tPLHB| or |tPHLB – tPLHA|	t_{SKEW1}	Any differential pair on package at 50% point of the transition	—	—	50	ps

1. $V_{DD}33$ = 3.1V - 3.5 V, $V_{DD}15$ = 1.4V - 1.6 V, -40˚C.

Termination Resistor

The LVDS drivers and receivers operate at 100 Ω differential impedance, as shown below. External resistors are not required. The differential driver and receiver buffers include termination resistors inside the device package, as shown in Figure 79 below. The center tap inputs should be connected to ground via a .01 pF capacitor.

Figure 79. SPI4 LVDS Driver and Receiver and Associated Internal Components



SPI4 AC Timing

Supported Data Rates

The SPI4 interfaces (SPIA and SPIB) on the ORSPI4 device support the following data rates

Table 56. Supported Data Rates

Data Rate	Clock Frequency (DDR)	Transmit REFCLK Frequency	Recommended Operating Mode
500 -900 Mbps	250 - 450 MHz	62.5 - 112.5 MHz	Dynamic
500 -700 Mbps	250 - 350 MHz	62.5 - 87.5 MHz	Static
100 - 200 Mbps	50 - 100 MHz	100 - 200 MHz[1]	Quarter-rate static mode

1 SPI[A,B]_TREFCLK_x8 internal signal is used in quarter-rate mode

System Timing Reference Points

Figure 80 shows the system timing reference points for timing parameters discussed in Table 57. Figure 81 shows the reference points at the transmitter and receiver with respect to the clock edges. Figure 82 shows the system timing reference points for status channel timing parameters discussed in Table 59. Figure 83 shows the reference points with respect to the clock edges

Figure 80. System Timing Reference Points.

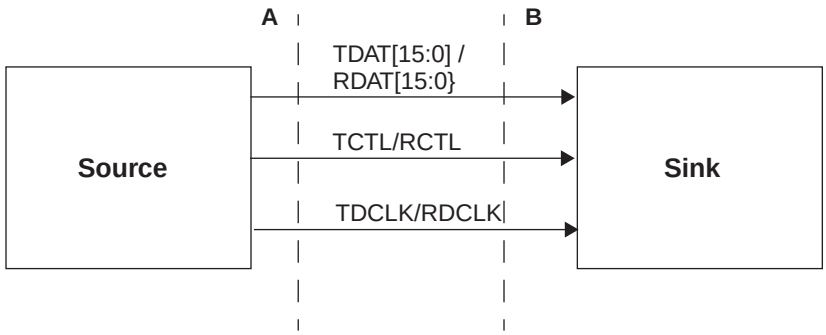
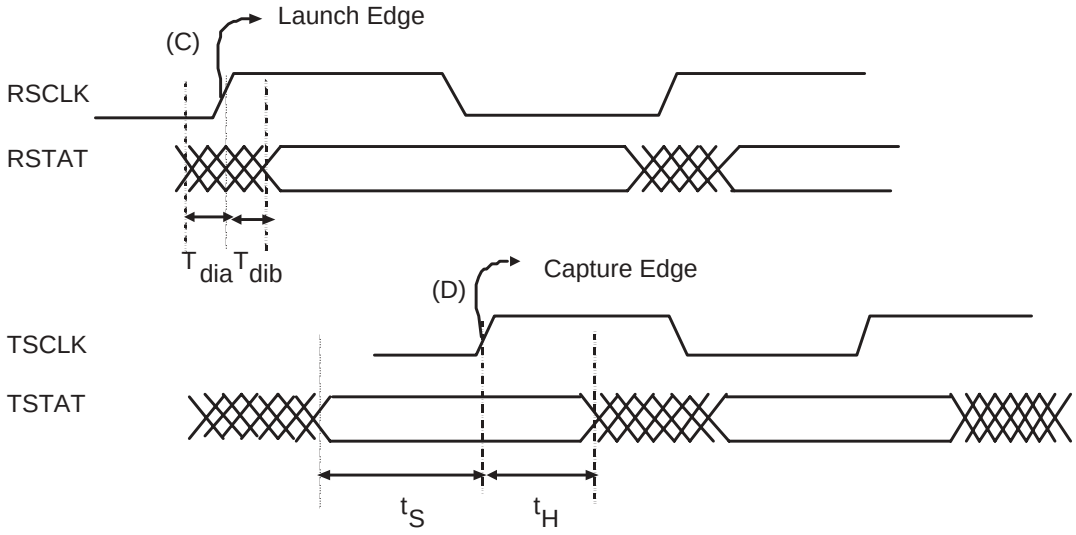


Figure 81. Timing Reference Points with Respect to Clock Edge



The static timing budget for a data rate of 700 Mbps is shown inTable 57.

Table 57. Data Path Interface Timing for Static Alignment

Description		Value	Unit
	TDCLK/RDCLK frequency	350	MHz
Before reference point A[1]	Clock to data skew, clock duty cycle distortion, data duty cycle distortion, data jitter $T_{dia} + T_{dib}$	560	ps
Between reference points A and B	Clock to data skew, relative jitter	230	ps
	Subtotal G_{max}	790	ps
After reference point B	Sampling error (device setup and hold)[2]	200	ps
	Relative jitter	80	ps
	Subtotal	1090	ps
	Total 700 Mbps bit period	1420	ps
	Available margin	330	ps

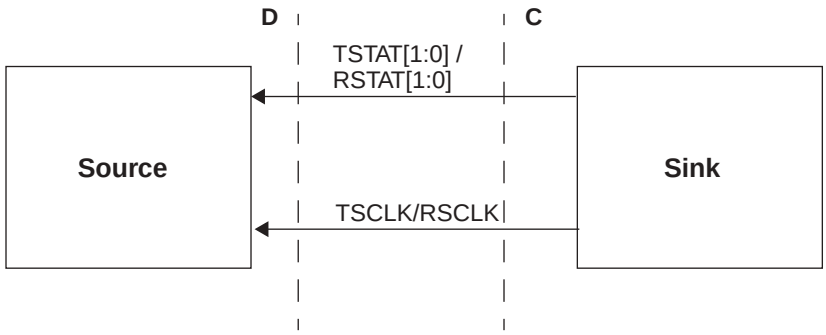
1 The maximum value for ORSPI4 is TBD over process, voltage and temperature
2 The maximum value for ORSPI4 is TBD over process, voltage and temperature

The dynamic alignment timing budget is shown in Table 58.

Table 58. Data Path Interface Timing for Dynamic Alignment.

Description		Value (UI peak-to-peak)
At reference point A	Clock to data skew, clock duty cycle distortion	0.1
	Data jitter	0.24
Between reference points A and B	Data jitter	0.20
After reference point B	On-chip jitter due to routing	0.01
	PLL output jitter	TBD
	Sampling granularity (16 clock phases)	0.0625

Figure 82. Status Channel Reference Points



Note: Only data signals are shown

Figure 83. Status Channel Reference Points with Respect to Clock Edge (LVDS and LVTTL I/Os)

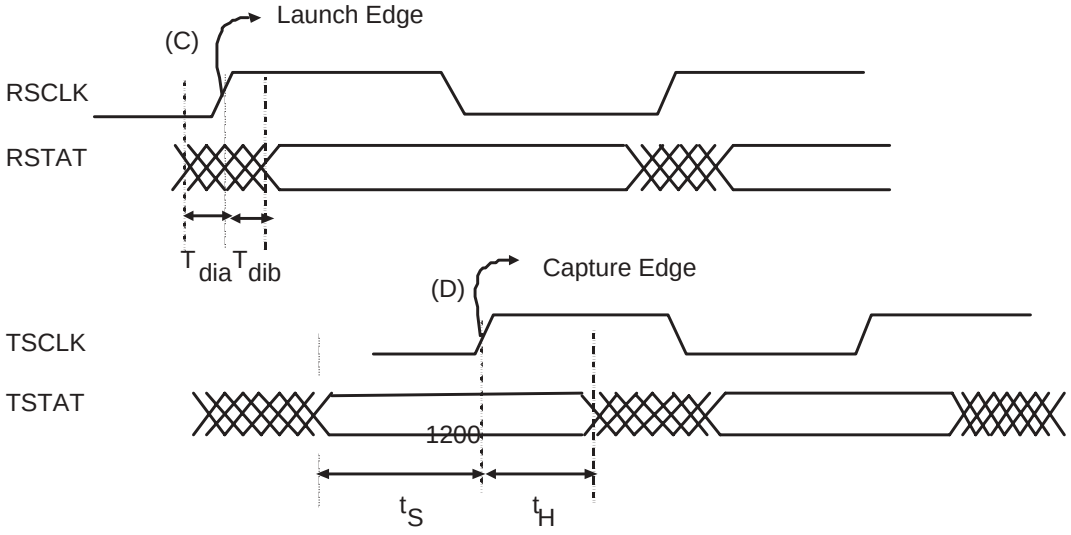


Table 59. Status Path Interface OIF-SPI4-02.0 Specification Timing (Reference)

Symbol	Description	Min	Max	Units
f_D	TDCLK/RDCLK frequency	—	f_D	MHz
f_S	TSCLK/RSCLK frequency	—	$f_D/4$	MHz
—	TSCLK/RSCLK duty cycle	40	60	%
T_{-dia}	Data invalid window with respect to clock edge (Reference point C)	2.5	—	ns
T_{-dib}	Data invalid window with respect to clock edge (Reference point C)	1	—	ns
t_S	Setup time for TSTAT with respect to TSCLK (Reference point D)	2	—	ns
t_H	Hold time for TSTAT with respect to TSCLK (Reference point D)	0.5	—	ns

Table 60. Status Path Interface ORSPI4 Timing in Centered (OIF) Mode

Symbol	Description	Min	Max	Units
f_D	TDCLK/RDCLK frequency	—	f_D	MHz
f_S	TSCLK/RSCLK frequency	—	$f_D/4$	MHz
—	TSCLK/RSCLK duty cycle	40	60	%
T_{-dia}	Data invalid window with respect to clock edge (Reference point C)	—	TBD	ns
T_{-dib}	Data invalid window with respect to clock edge (Reference point C)	—	TBD	ns
t_S	Setup time for TSTAT with respect to TSCLK (Reference point D)	LVDS 0.4 LVTTL 1.5	—	ns
t_H	Hold time for TSTAT with respect to TSCLK (Reference point D)	LVDS 0.4 LVTTL 0.4	—	ns

Figure 84. ORSPI4 Static Mode Status Signals Data Capture (Reference Point D)

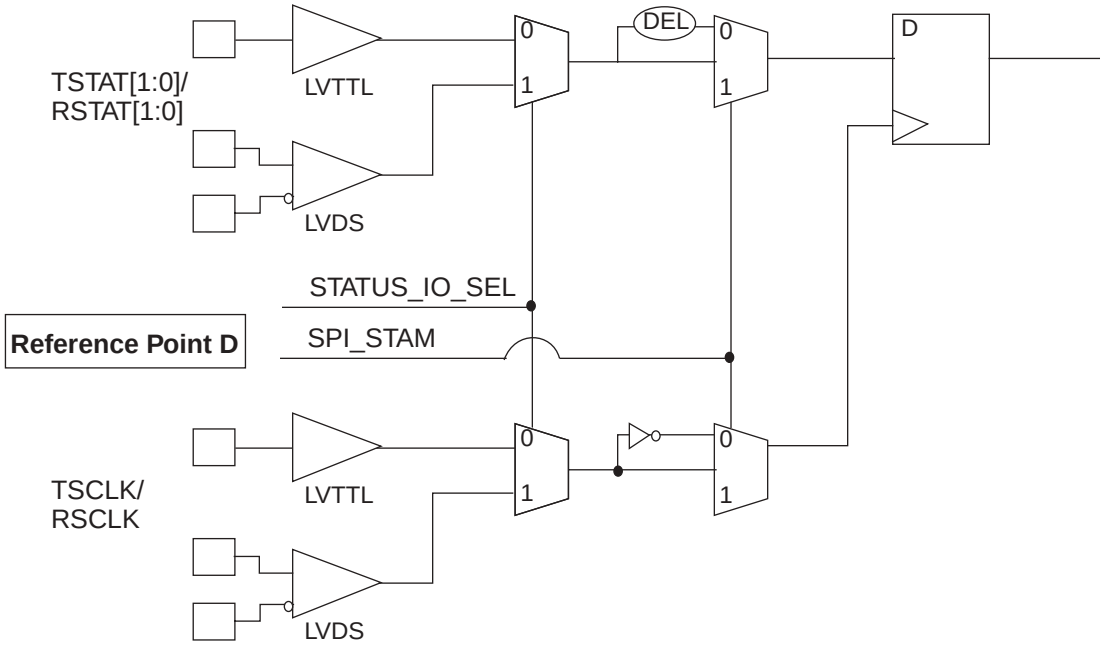


Figure 85. Status Channel Reference Points with Respect to Clock Edge (Edge Aligned Legacy Mode)

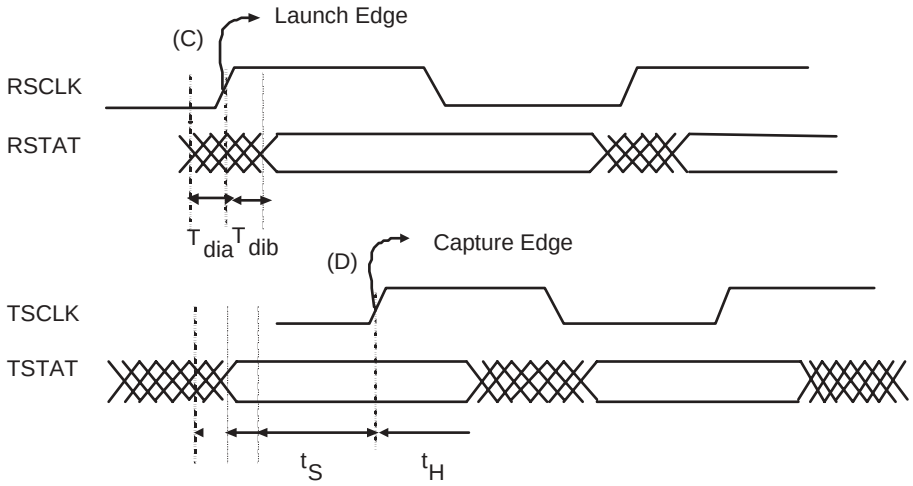


Table 61. Status Path Interface ORSPI4 Timing in Legacy Mode

Symbol	Description	Min	Max	Units
f_D	TDCLK/RDCLK frequency		f_D	MHz
f_S	TSCLK/RSCLK frequency		$f_D/4$	MHz
—	TSCLK/RSCLK duty cycle	40	60	%
T_{-}^{dia}	Data invalid window with respect to clock edge (Reference point C)	TBD	—	ns
T_{-dib}	Data invalid window with respect to clock edge (Reference point C)	TBD	—	ns
t_S*	Setup time for TSCLK with respect to RSCLK (Reference point D)	0.9	—	ns
t_H*	Hold time for TSCLK with respect to RSCLK (Reference point D)	0.9	—	ns

* Referencing back to the rising edge of the clock.
$t_S = 0.4T = 0.9$ ns
$t_H = 0.6T = 0.9$ ns
Where T = period of the clock in hs.
These formulas account for 40/60 duty cycle clock requirement.

SERDES Electrical and Timing Characteristics

SERDES High Speed Data Transmitter

Table 62 specifies serial data output buffer parameters measured on devices with typical and worst case process parameters and over the full range of operation conditions.

Table 62. Serial Output Timing and Levels (CML I/O)

Parameter	Min.	Typ.	Max.	Units
Rise Time (20%—80%)	50	80	110	ps
Fall Time (80%—20%)	50	80	110	ps
Common Mode	V_{DDOB} – 0.30	V_{DDOB} – 0.25	V_{DDOB} – 0.15	V
Differential Swing (Full Amplitude)[1]	600	700	1000	mV_{p-p}
Differential Swing (Half Amplitude)[1]	300	350	500	mV_{p-p}
Output Load (external)	—	86	—	Ω.

1. Differential swings are measured at the end of 3 inches of FR-4 and 12 inches of coax cable.

Transmitter output jitter is a critical parameter to systems with high-speed data links. Table 63 and Table 64 specify the transmitter output jitter for typical and worst case devices over the full range of operating conditions.

Table 63. Channel Output Jitter (3.125 Gbits/s)

Parameter	Min.	Typ.[1]	Max.[1]	Units
Deterministic	—	0.12	0.16	UI_{p-p}
Random	—	0.05	0.08	UI_{p-p}
Total[2, 3]	—	0.17	0.24	UI_{p-p}

1. With PRBS 2^7-1 data pattern, all channels operating, FPGA logic active, REFCLK jitter of 30 ps., 0ºC to 85ºC, 1.425 V to 1.575 V supply.
2. Wavecrest SIA-3000 instrument used to measure one-sigma (rms) random jitter component value. This value is multiplied by 14 to provide the peak-to-peak value that corresponds to a BER of 10^{-12}.
3. Total jitter measurement performed with Wavecrest SIA-3000 at a BER of 10^{-12}. See instrument documentation and other Wavecrest publications for a detailed discussion of jitter types included in this measurement.

Table 64. Channel Output Jitter (2.5 Gbits/s)

Parameter	Min.	Typ.[1]	Max.[1]	Units
Deterministic	—	0.11	0.13	UI_{p-p}
Random	—	0.05	0.07	UI_{p-p}
Total[2, 3]	—	0.16	0.20	UI_{p-p}

1. With PRBS 2^7-1 data pattern, all channels operating, FPGA logic active, REFCLK jitter of 30 ps., 0ºC to 85ºC, 1.425 V to 1.575 V supply.
2. Wavecrest SIA-3000 instrument used to measure one-sigma (rms) random jitter component value. This value is multiplied by 14 to provide the peak-to-peak value that corresponds to a BER of 10^{-12}.
3. Total jitter measurement performed with Wavecrest SIA-3000 at a BER of 10^{-12}. See instrument documentation and other Wavecrest publications for a detailed discussion of jitter types included in this measurement.

SERDES High Speed Data Receiver

Table 65 specifies receiver parameters measured on devices with worst case process parameters and over the full range of operation conditions.

Table 65. External Data Input Specifications

Parameter	Conditions	Min.	Typ.	Max.	Units
Input Data					
Stream of Nontransitions	8b/10b encode/decode off	—	—	72	Bits
Sensitivity (differential), worst-case[1]	3.125 Gbps	80	—	—	mVp-p
Input Levels[2]	—	V_{SS} - 0.3	—	V_{DD_ANA} + 0.3	V
Internal Buffer Resistance (Each input to VDDIB)	—	40	50	60	Ω.
PLL Lock Time[3]	—	—	—	Note 3	—

1. With PRBS 2^7-1 data pattern, all channels operating, FPGA logic active, REFCLK jitter of 30 ps., T_A = 0°C to 85°C, 1.425 V to 1.575 V supply.
2. Input level min + (input peak to peak swing)/2 ≤ common mode input voltage ≤ input level max - (input peak to peak swing)/2
3. The ORSPI4 SERDES receiver performs four levels of synchronization on the incoming serial data stream, providing first bit, then byte (character), then channel (32-bit word), and finally optional multi-channel alignment as described in TN1025. The PLL Lock Time is the time required for the CDR PLL to lock to the transitions in the incoming high-speed serial data stream. If the PLL is unable to lock to the serial data stream, it instead locks to REFCLK to stabilize the voltage-controlled oscillator (VCO), and periodically switches back to the serial data stream to again attempt synchronization.

Input Data Jitter Tolerance

A receiver's ability to tolerate incoming signal jitter is very dependent on jitter type. High speed serial interface standards have recognized the dependency on jitter type and have recently modified specifications to indicate tolerance levels for different jitter types as they relate to specific protocols (e.g XAUI, FC, Infiniband etc.). Sinusoidal jitter is considered to be a worst case jitter type. Table 66 shows receiver specifications with 10 MHz sinusoidal jitter injection. XAUI specific jitter tolerance measurements were measured in a separate experiment detailed in technical note TN1032, *SERDES Test Chip Jitter,* and are not reflected in these results.

Table 66. Receiver Sinusoidal Jitter Tolerance Specifications

Parameter	Conditions	Max.	Unit
Input Data			
Jitter Tolerance @ 3.125 Gbps, Typical	600 mV diff eye[1]	0.75	UIP-P
Jitter Tolerance @ 3.125 Gbps, Worst case	600 mV diff eye[1]	0.65	UIP-P
Jitter Tolerance @ 2.5 Gbps, Typical	600 mV diff eye[1]	0.79	UIP-P
Jitter Tolerance @ 2.5 Gbps, Worst case	600 mV diff eye[1]	0.67	UIP-P

1. With PRBS 2^7-1 data pattern, all channels operating, FPGA logic active, REFCLK jitter of 30 ps., T_A = 0°C to 85°C, 1.425 V to 1.575 V supply. Jitter measured with a Wavecrest SIA-3000.

Input Eye-Mask Characterization

Figure 86 provides an eye-mask characterization of the SERDES receiver input. The eye-mask is specified below for two different eye-mask heights. It provides guidance on a number of input parameters, including signal amplitude and rise time limits, noise and jitter limits, and P and N input skew tolerance. Almost all detrimental characteristics of transmit signal and the interconnection link design result in eye-closure. This, combined with the eye-opening limitations of the line receiver, can provide a good indication of a link's ability to transfer data error-free.

The Clock and Data Recovery (CDR) portion of the ORSPI4 SERDES receiver has the ability to filter incoming signal jitter that is below the clock recovery PLL bandwidth (about 3 MHz). The eye-mask specifications of Table 67 are for jitter frequencies above the PLL bandwidth of the CDR, which is a worst case condition. When jitter occurs at frequencies below the PLL bandwidth, the receiver jitter tolerance is significantly better. For this case error-free data detection can occur even with a completely closed eye-mask.

Figure 86. Receive Data Eye-Diagram Template (Differential)

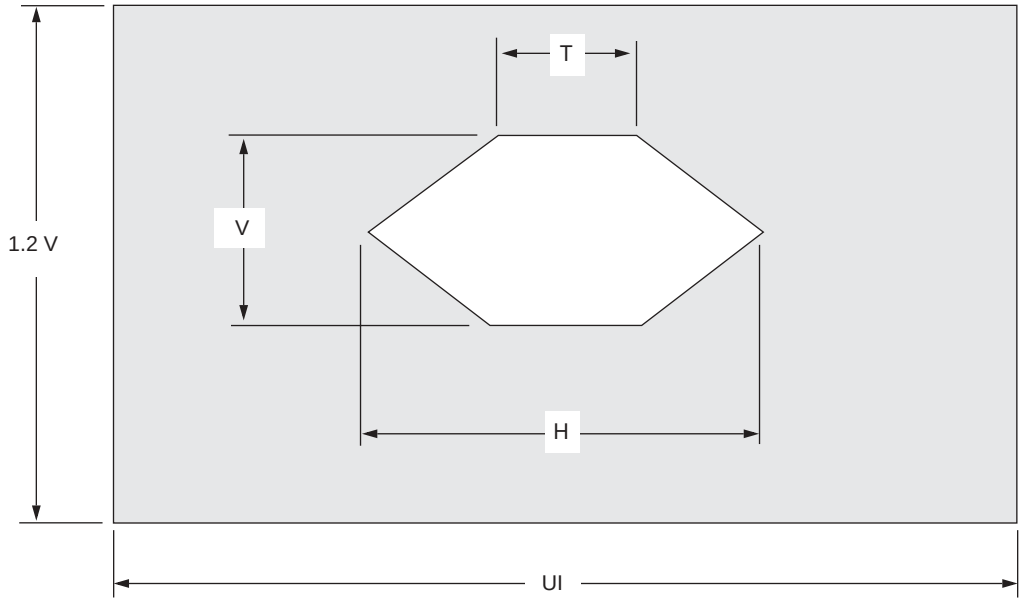


Table 67. Receiver Eye-Mask Specifications[1]

Parameter	Conditions	Value	Unit
Input Data			
Eye Opening Width (H) @ 3.125 Gbps	V=175 mV diff[1]	0.55	UIP-P
Eye Opening Width (T) @ 3.125 Gbps	V=175 mV diff[1]	0.15	UIP-P
Eye Opening Width (H) @ 3.125 Gbps	V=600 mV diff[1]	0.35	UIP-P
Eye Opening Width (T) @ 3.125 Gbps	V=600 mV diff[1]	0.10	UIP-P
Eye Opening Width (H) @ 2.5 Gbps	V=175 mV diff[1]	0.42	UIP-P
Eye Opening Width (T) @ 2.5 Gbps	V=175 mV diff[1]	0.15	UIP-P
Eye Opening Width (H) @ 2.5 Gbps	V=600 mV diff[1]	0.33	UIP-P
Eye Opening Width (T) @ 2.5 Gbps	V=600 mV diff[1]	0.10	UIP-P

1. With PRBS 2^7-1 data pattern, 10 MHz sinusoidal jitter, all channels operating, FPGA logic active, REFCLK jitter of 30 ps., T_A = 0°C to 85°C, 1.425 V to 1.575 V supply.

SERDES External Reference Clock

The external reference clock selection and its interface are a critical part of system applications for this product. Table 68 specifies reference clock requirements, over the full range of operating conditions. The designer is encourage to read TN1040, *SERDES Reference Clock,* which discusses various aspects of this system element and its interconnection.

Table 68. Reference Clock Specifications (REFCLKP and REFCLKN)

Parameter	Min.	Typ.	Max.	Units
Frequency Range	60	—	185	MHz
Frequency Tolerance[1]	-350	—	350	ppm
Duty Cycle (Measured at 50% Amplitude Point)	40	50	60	%
Rise Time	—	500	1000	ps
Fall Time	—	500	1000	ps
P–N Input Skew	—	—	75	ps
Differential Amplitude	500	800	2 x V_{DDIB}	mV_{p-p}
Common Mode Level	$V_{single-ended/2}$	0.75	$V_{DD15} – (V_{single-ended/2})$	V
Single-Ended Amplitude	250	400	V_{DDIB}	mV_{p-p}
Input Capacitance (at REFCLKP)	—	—	5	pF
Input Capacitance (at RECLKN)	—	—	5	pF

1. This specification indicates the capability of the high-speed receiver CDR PLL to acquire lock when the reference clock frequency and incoming data rate are not synchronized.

SERDES Core Timing Characteristics

Table 69 summarizes the end-to-end latencies through the embedded core for the various modes. All latencies are given in clock cycles for system clocks at half the REFCLK_[A:B] frequency. For a REFCLK_[A:B] of 156.25 MHz, a system clock cycle is 6.4 ns.

Table 69. Signal Latencies, Embedded Core

Operating Mode	Signal Latency (max.)
Transmit Path	5 clock cycles
Receive Path	
Multi-Channel Alignment Bypassed[1]	4.5 clock cycles
With Multi-Channel Alignment[1]	13.5-22.5 clock cycles

1. With multi-channel alignment, the latency is largest when the skew between channels is at the maximum that can be correctly compensated for (18 clock cycles). The latency specified in the table is for data from the channel received first.

Memory Controller Electrical and Timing Characteristics

HSTL Class II

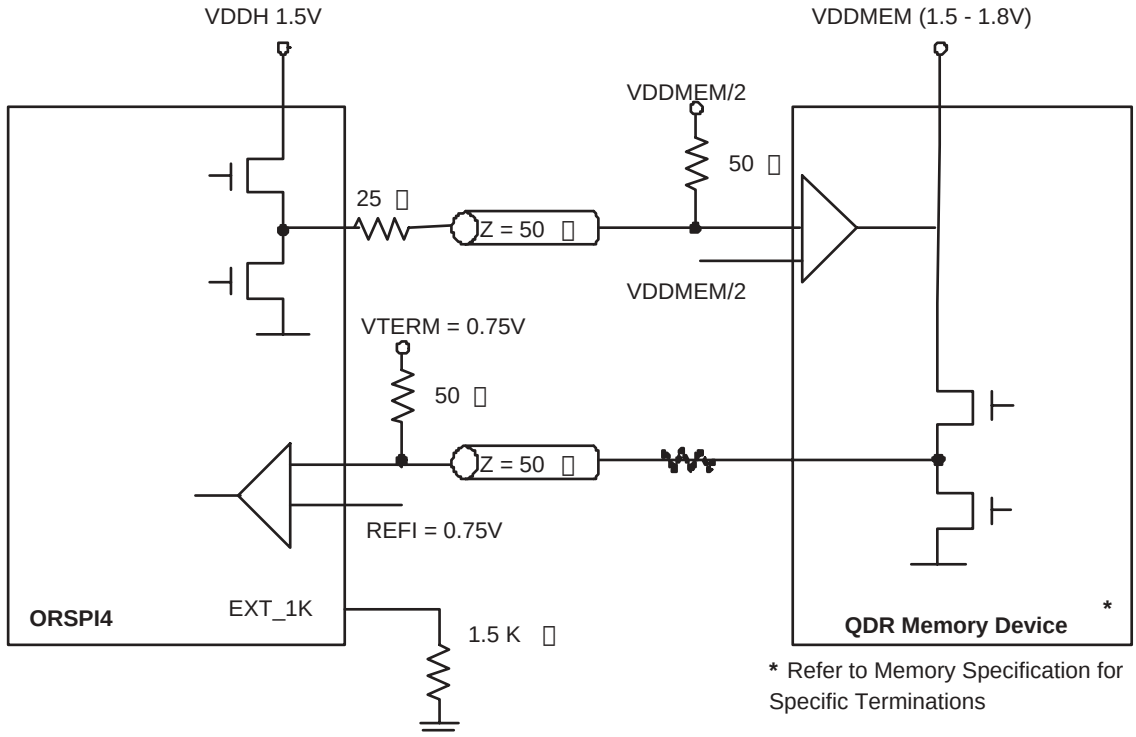
HSTL (High-Speed Transceiver Logic) - JEDEC standard JES 8-6 (August 1995), is a technology independent interface standard for digital integrated circuits. It is a voltage scalable and technology independent I/O structure. The I/O structures required by this standard contain a reference receiver (typically 50% of the VDDH), described as a REFI for single-ended inputs and outputs using power supply inputs (VDDH) that may differ from those operating the device itself. This buffer is designed to receive HSTL signal levels from either 1.5V or 1.8V devices. HSTL also allows chips with different power supplies to easily communicate with each other.

Table 70. HSTL Class II DC Operating Specifications

Parameter	Condition	Min	Typical	Max	Unit
VDDH	--	1.425	1.5	1.575	V
REFI[1]	--	0.68	0.75	0.9	V
VTERM[1]	--	--	0.75	-	V
VIH	--	REFI + 100mV	0.85	VDDH + 0.3	V
ViL	--	-0.3	0.65	REFI -100 mV	V
VOH[2]	IOH > 16 mA	VDDH - 0.4	1.1	--	V
VOL	IOL > -16 mA	--	--	0.4	V

1. 50% VDDH
2. VDDH - 400 mV.

Figure 87. HSTL Class II Termination Scheme



Supported Data Rates

The Memory Controller on the ORSPI4 device can support the following data rates.

Table 71. Supported Data Rates (36-Bit QDR-II, 32 bit Considered Data)

Data Rate			
-1		**-2**	
Clock Frequency (DDR)	Data Rate	Clock Frequency (DDR)	Data Rate
160 MHz	10.24 Gbps	TBD	TBD

Power Supplies for ORSPI4

Power Supply Descriptions

Table 72 shows the ORSPI4 FPGA and embedded core power supply groupings. VDD33 Is a 3.3V positive power supply used for 3.3 V configuration RAMs. VDD33_FPGAPLL is a 3.3V positive power supply for internal PLLs. When using PLLs, this power supply should be well isolated from all other power supplies on the board for proper operation. The five VDDIO supplies are positive power supply used by the FPGA I/O banks. The 1.5 volt digital power supplies are used for the FPGA and the embedded core transmit and receive digital logic including the microprocessor logic. The 1.5 volt analog power supply is used for SERDES high-speed analog circuitry in the embedded core between the I/O buffers and the digital logic. The SERDES VDDIB and VDDOB power supplies can be independently set to 1.5 V or 1.8 V, depending on the end application. The SERDES guard band supplies are independent connection brought out to pins.

Table 72. Power Supplies

FPGA Supplies
VDD15
VDD33
VDD33_FPGAPLL
VDDIO0
VDDIO1
VDDIO5
VDDIO6
VDDIO7

The ORSPI4 SPI4 embedded core requires an isolated 1.5V supply connected to four dedicated VDDA_SPI[A:D] pins. This supply is used to power the analog circuitry of the core between the LVDS I/O and the digital logic. The LVDS bus also requires connections to pins used for AC center-tap termination. The dedicated LVCTAP pins should be connected to GND through a 0.01 uFd capacitor.

The QDR memory controller portion of the embedded core has dedicated HSTL I/O buffers which require an additional 1.5V supply known as VDDH. The HSTL input buffers uses an external reference voltage. The reference voltage connects to the REFI_[1:4] pins. The REFI pins should ideally be connected to a noise immune source that is exactly 1/2 the value of the VDDH supply.

Table 73. Embedded Core Power Supplies

Supply Description	Name			
SERDES Input Buffers (1.5/1.8V)	VDDIB_A	VDDIB_B	VDDIB_C	VDDIB_D
SERDES Guardband (1.5V)	VDDGB			
SERDES Output Buffers (1.5/1.8V)	VDDOB_A	VDDOB_B	VDDOB_C	VDDOB_D
SERDES Analog (1.5V)	VDD_ANA			
SPI (1.5V)	VDDA_SPIA	VDDA_SPIB	VDDA_SPIC	VDDA_SPID
Memory Controller PLL (3.3V)	VDDA_PLL			
HSTL Input Buffer Reference Voltage (VDDH/2)	REFI_1			
	REFI_2			
	REFI_3			
	REFI_4			
HSTL Output Buffer Supply (1.5V)	VDDH			

Recommended Power Supply Connections

Ideally, a board should have the power supplies described below:

- VDD33, VDD33_FPGAPLL and VDDIO supplies for the FPGA Logic
- A single 1.5 V source to supply power to FPGA and core digital logic. (VDD15)
- A dedicated 1.5 V power supply for the SERDES analog power pins. This will allow the end user to minimize noise. The guard band pins can also be sourced from the analog power supplies. (VDD_ANA, VDDGB)
- SERDES TX output buffer power. The power supplies to the SERDES TX output buffers should be isolated from the rest of the board power supplies. Special care must be taken to minimize noise when providing board level power to these output buffers. The power supply can be 1.5 V or 1.8 V depending on the end application. (VDDOB)
- SERDES RX input buffer power. The power supplies to the SERDES RX input buffers should be isolated from the rest of the board power supplies. Special care must be taken to minimize noise when providing board level power to these input buffers. The power supply can be 1.5 V or 1.8 V depending on the end application. (VDDIB)
- An isolated 1.5V supply for the VDDA_SPI to minimize noise from the common 1.5V board supply. (VDD_SPI[A:D])
- The memory controller bus requires a dedicated 1.5V supply connected to the VDDH pins for HSTL output buffer supply voltage. (VDDH)
- The HSTL input buffers of the Memory Controller require a voltage reference connected to the REFI pins which is exactly half the VDDH supply. This supply should be filtered, and should not exceed a peak-to-peak AC noise of 2% of the VREF (DC). The HSTL buffer scheme also requires a termination resistor per signal. It is recommended that the clock pin termination be filtered separately from the data/control pin termination to minimize noise.
- The VDDA_PLL supply pin requires a noise minimized 3.3V supply.

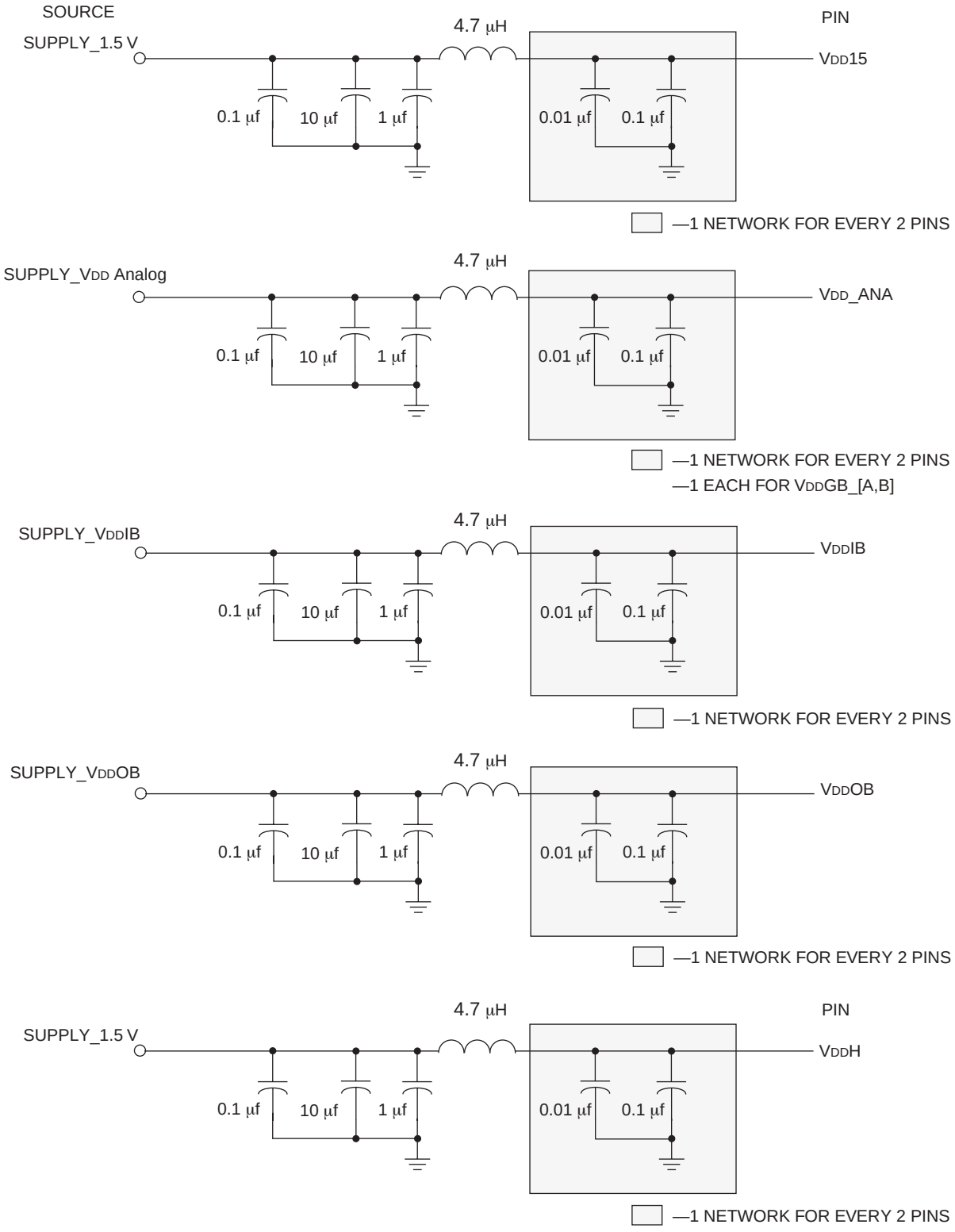
Recommended Power Supply Filtering Scheme

The board connections of the various SERDES V_{DD} and V_{SS} pins are critical to system performance. An example demonstration board schematic is available at www.latticesemi.com.

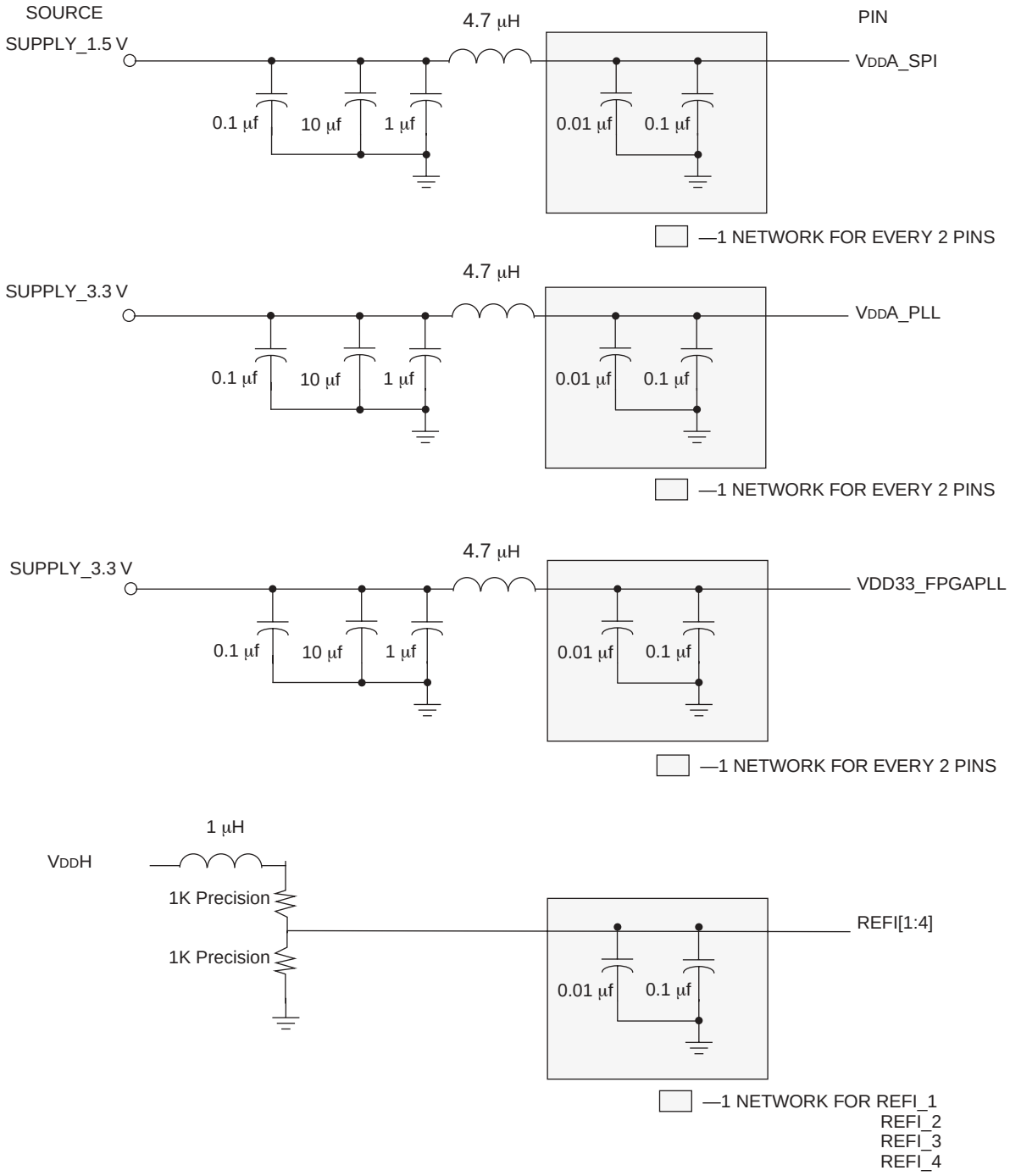
Power supply filtering is in the form of:

- A parallel bypass capacitor network consisting of 10 µf, 0.1 µf, and 1.0 µf caps close to the power source.
- A parallel bypass capacitor network consisting of 0.01 µf and 0.1 µf close to the pin.
- Example connections are shown in Figure 88. The naming convention for the power supply sources shown in the figure are as follows:
  - Supply_1.5 V – All digital, auxiliary power pins.
  - Supply_V_{DD}IB – Input RX buffer power pins for SERDES.
  - Supply_V_{DD}OB – Output TX buffer power pins for SERDES.
  - Supply_V_{DD}ANA – TX analog power pins, RX analog power pins, guard band power pins for SERDES.
  - Supply_V_{DD}33, V_{DD}A_PLL - FPGA and Embedded PLL power pins.
  - Supply V_{DD}A_SPIA, V_{DD}A_SPIB, V_{DD}A_SPIC, V_{DD}A_SPID - Analog core 1.5V SPI supplies.
  - Supply V_{DD}H - HSTL output buffer power supply of the QDR Memory Controller.
  - Supply REFI_1, REFI_2, REFI_3, REFI_4 - Voltage reference for HSTL input buffer of the QDR Memory Controller should be exactly one-half V_{DD}H

Figure 88. Power Supply Filtering



Power Supply Filtering (Continued)



Pin Descriptions

This section describes the pins found on the Series 4 FPGAs. Any pin not described in this table is a user-programmable I/O. During configuration, the user-programmable I/Os are 3-stated with an internal pull-up resistor. If any pin is not used (or not bonded to a package pin), it is also 3-stated with an internal pull-up resistor after configuration. The pin descriptions in Table and throughout this data sheet show active-low signals with an overscore. The package pinout tables that follow, show this as a signal ending with _N. For example $\overline{LDC}$ and LDC_N are equivalent.

Table 74. Pin Descriptions

Symbol	I/O	Description
Dedicated Pins		
VDD33	—	3.3V positive power supply. This power supply is used for 3.3 V configuration RAMs.
VDD33_FPGAPLL	—	3.3V positive power supply. This power supply is used for 3.3 V internal PLLs. This power supply should be well isolated from all other power supplies on the board for proper operation.
VDD15	—	1.5 V positive power supply for internal logic.
VDDIO	—	Positive power supply used by I/O banks.
VSS	—	Ground.
PTEMP	I	Temperature sensing diode pin. Dedicated input.
$\overline{RESET}$	I	During configuration, $\overline{RESET}$ forces the restart of configuration and a pull-up is enabled. After configuration, $\overline{RESET}$ can be used as a general FPGA input or as a direct input, which causes all PLC latches/FFs to be asynchronously set/reset.
CCLK	O	In the master and asynchronous peripheral modes, CCLK is an output which strobes configuration data in.
	I	In the slave or readback after configuration, CCLK is input synchronous with the data on DIN or D[7:0]. CCLK is an output for daisy-chain operation when the lead device is in master, peripheral, or system bus modes.
DONE	I	As an input, a low level on DONE delays FPGA start-up after configuration.[1]
	O	As an active-high, open-drain output, a high level on this signal indicates that configuration is complete. DONE has an optional pull-up resistor.
$\overline{PRGRM}$	I	$\overline{PRGRM}$ is an active-low input that forces the restart of configuration and resets the boundary-scan circuitry. This pin always has an active pull-up.
$\overline{RD\_CFG}$	I	This pin must be held high during device initialization until the $\overline{INIT}$ pin goes high. This pin always has an active pull-up. During configuration, $\overline{RD\_CFG}$ is an active-low input that activates the TS_ALL function and 3-states all of the I/O. After configuration, $\overline{RD\_CFG}$ can be selected (via a bit stream option) to activate the TS_ALL function as described above, or, if readback is enabled via a bit stream option, a high-to-low transition on $\overline{RD\_CFG}$ will initiate readback of the configuration data, including PFU output states, starting with frame address 0.
RD_DATA/TDO	O	RD_DATA/TDO is a dual-function pin. If used for readback, RD_DATA provides configuration data out. If used in boundary-scan, TDO is test data out.
$\overline{CFG\_IRQ}$/MPI_IRQ	O	During JTAG, slave, master, and asynchronous peripheral configuration assertion on this $\overline{CFG\_IRQ}$ (active-low) indicates an error or errors for block RAM or FPSC initialization. MPI active-low interrupt request output, when the MPI is used.
LVDS_R	-	Reference resistor connection for controlled impedance termination of Series 4 FPGA LVDS inputs.
Special-Purpose Pins		
M[3:0]	I	During power up and initialization, M0—M3 are used to select the configuration mode with their values latched on the rising edge of $\overline{INIT}$. During configuration, a pull-up is enabled.
	I/O	After configuration, these pins are user-programmable I/O.[1]
PLL_CK[0:7][TC]	I	Semi-dedicated PLL clock pins. During configuration they are 3-stated with a pull up.
	I/O	These pins are user-programmable I/O pins if not used by PLLs after configuration.
P[TBLR]CLK[1:0][TC]	I	Pins dedicated for the primary clock. Input pins on the middle of each side with differential pairing.
	I/O	After configuration these pins are user programmable I/O, if not used for clock inputs.

Table 74. Pin Descriptions (Continued)

Symbol	I/O	Description
TDI, TCK, TMS	I	If boundary-scan is used, these pins are test data in, test clock, and test mode select inputs. If boundary-scan is not selected, all boundary-scan functions are inhibited once configuration is complete. Even if boundary-scan is not used, either TCK or TMS must be held at logic 1 during configuration. Each pin has a pull-up enabled during configuration.
	I/O	After configuration, these pins are user-programmable I/O if boundary scan is not used.[1]
RDY/$\overline{\text{BUSY}}$/RCLK	O	During configuration in asynchronous peripheral mode, RDY/RCLK indicates another byte can be written to the FPGA. If a read operation is done when the device is selected, the same status is also available on D7 in asynchronous peripheral mode. During the master parallel configuration mode, RCLK is a read output signal to an external memory. This output is not normally used.
	I/O	After configuration this pin is a user-programmable I/O pin.[1]
HDC	O	High During Configuration is output high until configuration is complete. It is used as a control output, indicating that configuration is not complete.
	I/O	After configuration, this pin is a user-programmable I/O pin.[1]
$\overline{\text{LDC}}$	O	$\overline{\text{Low During Configuration}}$ is output low until configuration is complete. It is used as a control output, indicating that configuration is not complete.
	I/O	After configuration, this pin is a user-programmable I/O pin.[1]
$\overline{\text{INIT}}$	I/O	$\overline{\text{INIT}}$ is a bidirectional signal before and during configuration. During configuration, a pull-up is enabled, but an external pull-up resistor is recommended. As an active-low open-drain output, $\overline{\text{INIT}}$ is held low during power stabilization and internal clearing of memory. As an active-low input, $\overline{\text{INIT}}$ holds the FPGA in the wait-state before the start of configuration. After configuration, this pin is a user-programmable I/O pin.[1]
$\overline{\text{CS0}}$, CS1	I	$\overline{\text{CS0}}$ and CS1 are used in the asynchronous peripheral, slave parallel, and microprocessor configuration modes. The FPGA is selected when $\overline{\text{CS0}}$ is low and CS1 is high. During configuration, a pull-up is enabled.
	I/O	After configuration, if MPI is not used, these pins are user-programmable I/O pins.[1]
$\overline{\text{RD}}$/$\overline{\text{MPI\_STRB}}$	I	$\overline{\text{RD}}$ is used in the asynchronous peripheral configuration mode. A low on $\overline{\text{RD}}$ changes D[7:3] into a status output. $\overline{\text{WR}}$ and $\overline{\text{RD}}$ should not be used simultaneously. If they are, the write strobe overrides. This pin is also used as the MPI data transfer strobe. As a status indication, a high indicates ready, and a low indicates busy.
$\overline{\text{WR}}$/MPI_RW	I	$\overline{\text{WR}}$ is used in asynchronous peripheral mode. A low on $\overline{\text{WR}}$ transfers data on D[7:0] to the FPGA. In MPI mode, a high on MPI_RW allows a read from the data bus, while a low causes a write transfer to the FPGA.
	I/O	After configuration, if the MPI is not used, $\overline{\text{WR}}$/MPI_RW is a user-programmable I/O pin.[1]
PPC_A[14:31]	I	During MPI mode the PPC_A[14:31] are used as the address bus driven by the PowerPC bus master utilizing the least-significant bits of the PowerPC 32-bit address.
$\overline{\text{MPI\_BURST}}$	I	$\overline{\text{MPI\_BURST}}$ is driven low to indicate a burst transfer is in progress in MPI mode. Driven high indicates that the current transfer is not a burst.
$\overline{\text{MPI\_BDIP}}$	I	$\overline{\text{MPI\_BDIP}}$ is driven by the PowerPC processor in MPI mode. Assertion of this pin indicates that the second beat in front of the current one is requested by the master. Negated before the burst transfer ends to abort the burst data phase.
MPI_TSZ[0:1]	I	MPI_TSZ[0:1] signals are driven by the bus master in MPI mode to indicate the data transfer size for the transaction. Set 01 for byte, 10 for half-word, and 00 for word.
A[21:0]	O	During master parallel mode A[21:0] address the configuration EPROMs up to 4M bytes.
	I/O	If not used for MPI these pins are user-programmable I/O pins after configuration.[1]
MPI_ACK	O	In MPI mode this is driven low indicating the MPI received the data on the write cycle or returned data on a read cycle.
	I/O	If not used for MPI these pins are user-programmable I/O pins after configuration.[1]

Table 74. Pin Descriptions (Continued)

Symbol	I/O	Description
MPI_CLK	I	This is the PowerPC synchronous, positive-edge bus clock used for the MPI interface. It can be a source of the clock for the Embedded System Bus. If MPI is used this will be the AMBA bus clock.
	I/O	If not used for MPI these pins are user-programmable I/O pins after configuration.[1]
MPI_TEA	O	A low on the MPI transfer error acknowledge indicates that the MPI detects a bus error on the internal system bus for the current transaction.
	I/O	If not used for MPI these pins are user-programmable I/O pins after configuration.[1]
MPI_RTRY	O	This pin requests the MPC860 to relinquish the bus and retry the cycle.
	I/O	If not used for MPI these pins are user-programmable I/O pins after configuration.[1]
D[0:31]	I/O	Selectable data bus width from 8, 16, 32-bit in MPI mode. Driven by the bus master in a write transaction and driven by MPI in a read transaction.
	I	D[7:0] receive configuration data during master parallel, peripheral, and slave parallel configuration modes when $\overline{WR}$ is low and each pin has a pull-up enabled. During serial configuration modes, D0 is the DIN input.
	O	D[7:3] output internal status for asynchronous peripheral mode when $\overline{RD}$ is low.
	I/O	After configuration, if MPI is not used, the pins are user-programmable I/O pins.[1]
DP[0:3]	I/O	Selectable parity bus width in MPI mode from 1, 2, 4-bit, DP[0] for D[0:7], DP[1] for D[8:15], DP[2] for D[16:23], and DP[3] for D[24:31]. After configuration, if MPI is not used, the pins are user-programmable I/O pin.[1]
DIN	I	During slave serial or master serial configuration modes, DIN accepts serial configuration data synchronous with CCLK. During parallel configuration modes, DIN is the D0 input. During configuration, a pull-up is enabled.
	I/O	After configuration, this pin is a user-programmable I/O pin.[1]
DOUT	O	During configuration, DOUT is the serial data output that can drive the DIN of daisy-chained slave devices. Data out on DOUT changes on the rising edge of CCLK.
	I/O	After configuration, DOUT is a user-programmable I/O pin.[1]
TESTCFG	I	During configuration this pin should be held high, to allow configuration to occur. A pull up is enabled during configuration.
	I/O	After configuration, TESTCFG is a user programmable I/O pin.[1]

1. The FPGA States of Operation section in the ORCA Series 4 FPGAs data sheet contains more information on how to control these signals during start-up. The timing of DONE release is controlled by one set of bit stream options, and the timing of the simultaneous release of all other configuration pins (and the activation of all user I/Os) is controlled by a second set of options.

ORSPI4 SPI4 External I/O Description

This section describes device I/O signals to/from the SPI4 interface. Table 75 and Table 76 lists the external signals that interface to the SPI4 block.

Table 75. SPI4 External Transmit Path Interface Signals

Pin Name (j = A or B)	Direction O = FPSC Output I = FPSC Input	Description
TSCLKj	I	LVTTL transmit status clock input.
TSTATj[1:0]	I	LVTTL transmit status input
jTSCLKN	I	LVDS (negative) transmit status clock input
jTSCLKP	I	LVDS (positive) transmit status clock input
jTSTATN[1:0]	I	LVDS (negative) transmit status data input
jTSTATP[1:0]	I	LVDS (positive) transmit status data input

Pin Name (j = A or B)	Direction O = FPSC Output I = FPSC Input	Description
jTDATN[15:0]	O	LVDS (negative) transmit SPI4 data output
jTDATP[15:0]	O	LVDS (positive) transmit SPI4 data output
jTCTLN	O	LVDS (negative) transmit SPI4 control signal output
jTCTLP	O	LVDS (positive) transmit SPI4 control signal output
jTDCLKN	O	LVDS (negative) transmit SPI4 clock reference. (100 - > 450 MHz)
jTDCLKP	O	LVDS (positive) transmit SPI4 clock reference. (100 - >450 MHz)
jTREFCLK	I	SPI4 clock reference input.

Table 76. SPI4 External Receive Path Interface Signals

Pin Name (j = A or B)	Direction O = FPSC Output I = FPSC Input	Description
RSCLKj	O	LVTTL receive status clock output.
RSTATj[1:0]	O	LVTTL receive status output
jRSCLKN	O	LVDS (negative) receive status clock output
jRSCLKP]	O	LVDS (positive) receive status clock output
jRSTATN[1:0]	O	LVDS (negative) receive status data output
jRSTATP[1:0	O	LVDS (positive) receive status data output
jRDATN[15:0]	I	LVDS (negative) receive SPI4 data input
jRDATP[15:0]	I	LVDS (positive) receive SPI4 data input
jRCTLN	I	LVDS (negative) receive SPI4 control signal input
jRCTLP	I	LVDS (positive) receive SPI4 control signal input
jRDCLKN	I	LVDS (negative) receive SPI4 clock reference. (100 - >450 MHz)
jRDCLKP	I	LVDS (positive) receive SPI4 clock reference. (100 - >450 MHz)

Table 77. SPI4 Miscellaneous System Signals

Pin Name (j = A or B)	Direction O = FPSC Output I = FPSC Input	Description
REF10	I	LVDS Reference voltage: 1.0 V +/- 3%
REF14	I	LVDS Reference voltage: 1.4 V +/- 3%
RESHI	I	LVDS resistor high pin (100 Ω. in series with RESLO)
RESLO	I	LVDS resistor low pin (100 Ω. in series with RESHI)
ALVCTAP1	I/O	LVDS input center tap (use 0.01 µF to GND)
ALVCTAP2	I/O	LVDS input center tap (use 0.01 µF to GND)
ALVCTAP3	I/O	LVDS input center tap (use 0.01 µF to GND)
ALVCTAP4	I/O	LVDS input center tap (use 0.01 µF to GND)
ALVCTAP5	I/O	LVDS input center tap (use 0.01 µF to GND)
BLVCTAP1	I/O	LVDS input center tap (use 0.01 µF to GND)
BLVCTAP2	I/O	LVDS input center tap (use 0.01 µF to GND)

Pin Name (j = A or B)	Direction O = FPSC Output I = FPSC Input	Description
BLVCTAP3	I/O	LVDS input center tap (use 0.01 μF to GND)
BLVCTAP4	I/O	LVDS input center tap (use 0.01 μF to GND)
BLVCTAP5	I/O	LVDS input center tap (use 0.01 μF to GND)
SPARE_1	I/O	Reserved
SPARE_2	I/O	Reserved

ORSPI4 SERDES External I/O Description

This section describes device I/O signals to/from the SERDES. Table 78 lists the external signals that interface to the SERDES block.

Table 78. SERDES External Interface Signals

Pin Name	Direction O = FPSC Output I = FPSC Input	Description
SERDES Interface Pins		
HDINP_A	I	Serial Input for channel A
HDINN_A		
HDINP_B	I	Serial Input for channel B
HDINN_B		
HDINP_C	I	Serial Input for channel C
HDINN_C		
HDINP_D	I	Serial Input for channel D
HDINN_D		
HDOUTP_A	O	Serial Output for channel A
HDOUTN_A		
HDOUTP_B	O	Serial Output for channel B
HDOUTN_B		
HDOUTP_C	O	Serial Output for channel C
HDOUTN_C		
HDOUTP_D	O	Serial Output for channel D
HDOUTN_D		
REFCLKP	I	Reference clock to SERDES quad
REFCLKN		
REXT	--	Reference resistor
REXTN	--	Reference resistor. A 3.32 KW ± 1% resistor must be connected across REXT and REXTN. This current should handle a total of 300μA.
Misc System Signals		
RESETN	I	Global Reset Integrated 50K pull-down allows chip to stay in reset state when external driver loses power
TESTMD[1:0]N	I	LVTTL test mode pins with integrated 50K pull-ups that default chip into operational mode when un-driven.

PDN	I	Power-down active low. Puts Core into low-power (non-functional) state.
TRISTN	I	Active low control input causes all output pins to be disabled.
TESTCLK	I	Chip test pin. Integrated 50K pull-up

ORSPI4 Memory Controller External I/O Description

This section describes device I/O signals to/from the Memory Controller. Table 79 lists the external signals that interface to the Memory Controller block.

Table 79. Memory Controller External Interface Signals

Pin Name	QDRII SDRAM Pin Name	Direction O = FPSC Output I = FPSC Input	Description
Interface to QDRII SDRAM			
PMIK, PMIKN	K,K#	O	Clock for write data D, address SA, and enables W# and R#
PMID(35:0)	D(35:0)	O	Write data bus
PMIA(17:0)	SA(17:0)	O	Address bus
PMIWN	W#	O	Write enable (active-LO)
PMIRN	R#	O	Read enable (active-LO)
PMIC, PMICN	CQ,CQ#	I	Clock for read data Q
PMIQ(35:0)	Q(35:0)	I	Read data bus
REFCLK Source Inputs			
MCREFCLK	-	I	Dedicated Memory Controller reference clock (HSTL)
ATREFCLK	-	I	SPIA reference clock (LVTTL). Note that this signal also is fed to the SPIA block.
BTREFCLK	-	I	SPIB reference clock, (LVTTL). Note that this signal also is fed to the SPIB block.
Other Signals			
EXT_1K	-	-	Reference resistor. Connect to a 1.5 K Ω ± 1% precision resistor to ground. This current should handle a total of 700 μA.

Package I/O

Table 80. I/O Summary

	Package	
I/O Type	**FE1036**	**F1156**
User programmable I/O	498	356
Available programmable differential pair pins	498	356
FPGA configuration pins	7	7
FPGA dedicated function pins	2	2
Core function pins	322	301
VDD15	42	80
VDD33	14	30
VDD33_FPGAPLL	8	8
VDDIO	30	50
VSS	79	187
VDDGB	1	0
VDDIB	4	0
VDDOB	8	0
VDD_ANA	4	0
VDDA_SPI[A:D]	4	4
VDDA_PLL	1	1
VDDH	8	30
HSTL VREF	4	4
No connect	0	96
Total package pins	1036	1156

Pin Tables

The ORSPI4 FPSC is available in two package types; a 1156-pin fpBGA package, and a 1036-pin fpSBGA package. Both packages are 1.0 mm pitch packages.

The 1036-pin package offers two SPI4 interfaces, or one SPI4 interface and a quad 0.6-3.7 Gbps SERDES, a high-speed QDR-II SRAM Memory Controller, and 498 user I/Os on the FPGA array.

The 1156-pin package offers two SPI4 interfaces (no SERDES available on this package offering), a high-speed QDR-II SRAM Memory Controller, and 356 user I/Os on the FPGA array.

Table 81. 1156 fpBGA Pin Table

F1156 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	F1156 Pair
F4	-	-	O	PRD_DATA	RD_DATA/TDO	-
G4	-	-	I	PRESET_N	RESET_N	-
H4	-	-	I	PRD_CFG_N	RD_CFG_N	-
J4	-	-	I	PPRGRM_N	PRGRM_N	-
H5	0 (TL)	7	IO	PL2D	PLL_CK0C/HPPLL	L1C
H6	0 (TL)	7	IO	PL2C	PLL_CK0T/HPPLL	L1T
J6	0 (TL)	7	IO	PL3D	-	L2C
J5	0 (TL)	7	IO	PL3C	VREF_0_07	L2T
H3	0 (TL)	7	IO	PL4D	D5	L3C
J3	0 (TL)	7	IO	PL4C	D6	L3T
K4	0 (TL)	8	IO	PL5D	-	L4C
K5	0 (TL)	8	IO	PL5C	VREF_0_08	L4T
G2	0 (TL)	8	IO	PL6D	HDC	L5C
G1	0 (TL)	8	IO	PL6C	LDC_N	L5T
L5	0 (TL)	8	IO	PL7D	-	L6C
L4	0 (TL)	8	IO	PL7C	-	L6T
H2	0 (TL)	9	IO	PL8D	TESTCFG	L7C
H1	0 (TL)	9	IO	PL8C	D7	L7T
J2	0 (TL)	9	IO	PL9D	VREF_0_09	L8C
J1	0 (TL)	9	IO	PL9C	A17/PPC_A31	L8T
K2	0 (TL)	9	IO	PL10D	CS0_N	L9C
K1	0 (TL)	9	IO	PL10C	CS1	L9T
K3	0 (TL)	10	IO	PL11D	-	L10C
L3	0 (TL)	10	IO	PL11C	-	L10T
L2	0 (TL)	10	IO	PL12D	INIT_N	L11C
L1	0 (TL)	10	IO	PL12C	DOUT	L11T
M4	0 (TL)	10	IO	PL13D	VREF_0_10	L12C
M5	0 (TL)	10	IO	PL13C	A16/PPC_A30	L12T
N5	7 (CL)	1	IO	PL14D	A15/PPC_A29	L13C
N4	7 (CL)	1	IO	PL14C	A14/PPC_A28	L13T
M3	7 (CL)	1	IO	PL15D	-	L14C
N3	7 (CL)	1	IO	PL15C	-	L14T
P4	7 (CL)	1	IO	PL16D	VREF_7_01	L15C

Table 81. 1156 fpBGA Pin Table (Continued)

F1156 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	F1156 Pair
P3	7 (CL)	1	IO	PL16C	D4	L15T
M2	7 (CL)	2	IO	PL17D	-	L16C
M1	7 (CL)	2	IO	PL17C	-	L16T
N2	7 (CL)	2	IO	PL18D	RDY/BUSY_N/RCLK	L17C
N1	7 (CL)	2	IO	PL18C	VREF_7_02	L17T
P5	7 (CL)	2	IO	PL19D	A13/PPC_A27	L18C
R5	7 (CL)	2	IO	PL19C	A12/PPC_A26	L18T
R4	7 (CL)	3	IO	PL20D	-	L19C
R3	7 (CL)	3	IO	PL20C	-	L19T
P2	7 (CL)	3	IO	PL21D	A11/PPC_A25	L20C
P1	7 (CL)	3	IO	PL21C	VREF_7_03	L20T
T3	7 (CL)	3	IO	PL22D	-	L21C
T4	7 (CL)	3	IO	PL22C	-	L21T
R1	7 (CL)	4	IO	PL23D	RD_N/MPI_STRB_N	L22C
R2	7 (CL)	4	IO	PL23C	VREF_7_04	L22T
R6	7 (CL)	4	IO	PL24D	PLCK0C	L23C
T6	7 (CL)	4	IO	PL24B	-	L178C
T5	7 (CL)	4	IO	PL24C	PLCK0T	L23T
U5	7 (CL)	4	IO	PL24A	-	L178T
U3	7 (CL)	5	IO	PL25D	A10/PPC_A24	L24C
U4	7 (CL)	5	IO	PL25C	A9/PPC_A23	L24T
T1	7 (CL)	5	IO	PL26D	A8/PPC_A22	L25C
T2	7 (CL)	5	IO	PL26C	VREF_7_05	L25T
V3	7 (CL)	5	IO	PL27D	-	L26C
V4	7 (CL)	5	IO	PL27C	-	L26T
U1	7 (CL)	6	IO	PL28D	PLCK1C	L27C
U2	7 (CL)	6	IO	PL28C	PLCK1T	L27T
V5	7 (CL)	6	IO	PL29D	VREF_7_06	L28C
W5	7 (CL)	6	IO	PL29C	A7/PPC_A21	L28T
W3	7 (CL)	6	IO	PL30D	A6/PPC_A20	L29C
W4	7 (CL)	6	IO	PL30C	A5/PPC_A19	L29T
V1	7 (CL)	7	IO	PL31D	-	L30C
V2	7 (CL)	7	IO	PL31C	-	L30T
W1	7 (CL)	7	IO	PL32D	WR_N/MPI_RW	L31C
W2	7 (CL)	7	IO	PL32C	VREF_7_07	L31T
Y1	7 (CL)	7	IO	PL33D	-	L32C
Y2	7 (CL)	7	IO	PL33C	-	L32T
AA1	7 (CL)	8	IO	PL34D	A4/PPC_A18	L33C
AA2	7 (CL)	8	IO	PL34C	VREF_7_08	L33T
Y3	7 (CL)	8	IO	PL35D	A3/PPC_A17	L34C
W6	7 (CL)	8	IO	PL35B	-	L35C
AA3	7 (CL)	8	IO	PL35C	A2/PPC_A16	L34T

Table 81. 1156 fpBGA Pin Table (Continued)

F1156 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	F1156 Pair
Y6	7 (CL)	8	IO	PL35A	-	L35T
AB1	7 (CL)	8	IO	PL36D	A1/PPC_A15	L36C
Y5	7 (CL)	8	IO	PL36B	-	L37C
AB2	7 (CL)	8	IO	PL36C	A0/PPC_A14	L36T
Y4	7 (CL)	8	IO	PL36A	-	L37T
AC1	7 (CL)	8	IO	PL37D	DP0	L38C
AA5	7 (CL)	8	IO	PL37B	-	L39C
AC2	7 (CL)	8	IO	PL37C	DP1	L38T
AA4	7 (CL)	8	IO	PL37A	-	L39T
AD1	6 (BL)	1	IO	PL38D	D8	L40C
AA6	6 (BL)	1	IO	PL38B	-	L41C
AD2	6 (BL)	1	IO	PL38C	VREF_6_01	L40T
AB6	6 (BL)	1	IO	PL38A	-	L41T
AB3	6 (BL)	1	IO	PL39D	D9	L42C
AB4	6 (BL)	1	IO	PL39B	-	L43C
AC3	6 (BL)	1	IO	PL39C	D10	L42T
AB5	6 (BL)	1	IO	PL39A	-	L43T
AE1	6 (BL)	2	IO	PL40D	-	L44C
AC4	6 (BL)	2	IO	PL40B	-	L45C
AE2	6 (BL)	2	IO	PL40C	VREF_6_02	L44T
AC5	6 (BL)	2	IO	PL40A	-	L45T
AF1	6 (BL)	2	IO	PL41D	-	L46C
AD5	6 (BL)	2	IO	PL41B	-	L47C
AF2	6 (BL)	2	IO	PL41C	-	L46T
AD4	6 (BL)	2	IO	PL41A	-	L47T
AD3	6 (BL)	3	IO	PL42D	D11	L48C
AE4	6 (BL)	3	IO	PL42B	-	L49C
AE3	6 (BL)	3	IO	PL42C	D12	L48T
AE5	6 (BL)	3	IO	PL42A	-	L49T
AG1	6 (BL)	3	IO	PL43D	-	L50C
AF4	6 (BL)	3	IO	PL43B	-	L51C
AG2	6 (BL)	3	IO	PL43C	-	L50T
AF5	6 (BL)	3	IO	PL43A	-	L51T
AH1	6 (BL)	3	IO	PL44D	VREF_6_03	L52C
AC6	6 (BL)	4	IO	PL44B	-	L53C
AH2	6 (BL)	3	IO	PL44C	D13	L52T
AD6	6 (BL)	4	IO	PL44A	-	L53T
AF3	6 (BL)	4	IO	PL45D	-	L54C
AE6	6 (BL)	4	IO	PL45B	-	L55C
AG3	6 (BL)	4	IO	PL45C	VREF_6_04	L54T
AF6	6 (BL)	4	IO	PL45A	-	L55T
AJ1	6 (BL)	4	IO	PL46D	-	L56C
AJ2	6 (BL)	4	IO	PL46C	-	L56T

Table 81. 1156 fpBGA Pin Table (Continued)

F1156 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	F1156 Pair
AH3	6 (BL)	4	IO	PL47D	PLL_CK7C/HPPLL	L57C
AJ3	6 (BL)	4	IO	PL47C	PLL_CK7T/HPPLL	L57T
AG4	-	-	I	PTEMP	PTEMP	-
AH4	-	-	IO	LVDS_R	LVDS_R	-
AH6	6 (BL)	5	IO	PB2A	DP2	L58T
AN5	6 (BL)	5	IO	PB2C	PLL_CK6T/PPLL	L59T
AH7	6 (BL)	5	IO	PB2B	-	L58C
AP5	6 (BL)	5	IO	PB2D	PLL_CK6C/PPLL	L59C
AN6	6 (BL)	5	IO	PB3C	-	L60T
AP6	6 (BL)	5	IO	PB3D	-	L60C
AK6	6 (BL)	5	IO	PB4C	VREF_6_05	L61T
AK7	6 (BL)	5	IO	PB4D	DP3	L61C
AJ6	6 (BL)	6	IO	PB5A	-	L62T
AL6	6 (BL)	6	IO	PB5C	-	L63T
AJ7	6 (BL)	6	IO	PB5B	-	L62C
AL7	6 (BL)	6	IO	PB5D	-	L63C
AG7	6 (BL)	6	IO	PB6A	-	L64T
AM6	6 (BL)	6	IO	PB6C	VREF_6_06	L65T
AG8	6 (BL)	6	IO	PB6B	-	L64C
AM7	6 (BL)	6	IO	PB6D	D14	L65C
AG9	6 (BL)	6	IO	PB7A	-	L66T
AN7	6 (BL)	6	IO	PB7C	-	L67T
AG10	6 (BL)	6	IO	PB7B	-	L66C
AP7	6 (BL)	6	IO	PB7D	-	L67C
AH8	6 (BL)	7	IO	PB8A	-	L68T
AN8	6 (BL)	7	IO	PB8C	D15	L69T
AJ8	6 (BL)	7	IO	PB8B	-	L68C
AP8	6 (BL)	7	IO	PB8D	D16	L69C
AH9	6 (BL)	7	IO	PB9A	-	L70T
AK8	6 (BL)	7	IO	PB9C	D17	L71T
AJ9	6 (BL)	7	IO	PB9B	-	L70C
AK9	6 (BL)	7	IO	PB9D	D18	L71C
AJ10	6 (BL)	7	IO	PB10A	-	L72T
AL8	6 (BL)	7	IO	PB10C	VREF_6_07	L73T
AH10	6 (BL)	7	IO	PB10B	-	L72C
AL9	6 (BL)	7	IO	PB10D	D19	L73C
AM8	6 (BL)	8	IO	PB11C	D20	L74T
AM9	6 (BL)	8	IO	PB11D	D21	L74C
AJ11	6 (BL)	8	IO	PB12A	-	L75T
AN9	6 (BL)	8	IO	PB12C	VREF_6_08	L76T
AH11	6 (BL)	8	IO	PB12B	-	L75C
AP9	6 (BL)	8	IO	PB12D	D22	L76C
AG11	6 (BL)	9	IO	PB13A	-	L77T

Table 81. 1156 fpBGA Pin Table (Continued)

F1156 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	F1156 Pair
AK10	6 (BL)	9	IO	PB13C	D23	L78T
AG12	6 (BL)	9	IO	PB13B	-	L77C
AK11	6 (BL)	9	IO	PB13D	D24	L78C
AJ12	6 (BL)	9	IO	PB14A	-	L79T
AN10	6 (BL)	9	IO	PB14C	VREF_6_09	L80T
AH12	6 (BL)	9	IO	PB14B	-	L79C
AP10	6 (BL)	9	IO	PB14D	D25	L80C
AJ13	6 (BL)	9	IO	PB15A	-	L81T
AL10	6 (BL)	9	IO	PB15C	-	L82T
AH13	6 (BL)	9	IO	PB15B	-	L81C
AL11	6 (BL)	9	IO	PB15D	-	L82C
AJ14	6 (BL)	10	IO	PB16A	-	L83T
AM10	6 (BL)	10	IO	PB16C	D26	L84T
AH14	6 (BL)	10	IO	PB16B	-	L83C
AM11	6 (BL)	10	IO	PB16D	D27	L84C
AJ16	6 (BL)	10	IO	PB17A	-	L85T
AN11	6 (BL)	10	IO	PB17C	-	L86T
AH16	6 (BL)	10	IO	PB17B	-	L85C
AP11	6 (BL)	10	IO	PB17D	-	L86C
AJ17	6 (BL)	10	IO	PB18A	-	L87T
AK12	6 (BL)	10	IO	PB18C	VREF_6_10	L88T
AH17	6 (BL)	10	IO	PB18B	-	L87C
AK13	6 (BL)	10	IO	PB18D	D28	L88C
AG13	6 (BL)	11	IO	PB19A	-	L89T
AN12	6 (BL)	11	IO	PB19C	D29	L90T
AG14	6 (BL)	11	IO	PB19B	-	L89C
AP12	6 (BL)	11	IO	PB19D	D30	L90C
AH15	6 (BL)	11	IO	PB20A	-	L91T
AL12	6 (BL)	11	IO	PB20C	VREF_6_11	L92T
AJ15	6 (BL)	11	IO	PB20B	-	L91C
AL13	6 (BL)	11	IO	PB20D	D31	L92C
AJ18	5 (BC)	1	IO	PB21A	-	L93T
AM12	5 (BC)	1	IO	PB21C	-	L94T
AH18	5 (BC)	1	IO	PB21B	-	L93C
AM13	5 (BC)	1	IO	PB21D	-	L94C
AJ19	5 (BC)	1	IO	PB22A	-	L95T
AN13	5 (BC)	1	IO	PB22C	VREF_5_01	L96T
AH19	5 (BC)	1	IO	PB22B	-	L95C
AP13	5 (BC)	1	IO	PB22D	-	L96C
AP14	5 (BC)	2	IO	PB23C	PBCK0T	L97T
AN14	5 (BC)	2	IO	PB23D	PBCK0C	L97C
AG15	5 (BC)	2	IO	PB24A	-	L99T
AK14	5 (BC)	2	IO	PB24C	VREF_5_02	L98T

Table 81. 1156 fpBGA Pin Table (Continued)

F1156 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	F1156 Pair
AG16	5 (BC)	2	IO	PB24B	-	L99C
AK15	5 (BC)	2	IO	PB24D	-	L98C
AL14	5 (BC)	2	IO	PB25C	-	L100T
AL15	5 (BC)	2	IO	PB25D	-	L100C
AM14	5 (BC)	3	IO	PB26C	-	L101T
AM15	5 (BC)	3	IO	PB26D	VREF_5_03	L101C
AG18	5 (BC)	3	IO	PB27A	-	L102T
AN15	5 (BC)	3	IO	PB27C	-	L103T
AG17	5 (BC)	3	IO	PB27B	-	L102C
AP15	5 (BC)	3	IO	PB27D	-	L103C
AL16	5 (BC)	3	IO	PB28C	PBCK1T	L104T
AM16	5 (BC)	3	IO	PB28D	PBCK1C	L104C
AN16	5 (BC)	4	IO	PB29C	-	L105T
AP16	5 (BC)	4	IO	PB29D	-	L105C
AK16	5 (BC)	4	IO	PB30C	-	L106T
AK17	5 (BC)	4	IO	PB30D	VREF_5_04	L106C
AN17	5 (BC)	4	IO	PB31C	-	L107T
AP17	5 (BC)	4	IO	PB31D	-	L107C
AL17	5 (BC)	5	IO	PB32C	-	L108T
AM17	5 (BC)	5	IO	PB32D	VREF_5_05	L108C
AK18	5 (BC)	5	IO	PB33C	-	L109T
AL18	5 (BC)	5	IO	PB33D	-	L109C
AN18	5 (BC)	5	IO	PB34C	-	L110T
AP18	5 (BC)	5	IO	PB34D	-	L110C
AN19	5 (BC)	6	IO	PB35C	-	L111T
AP19	5 (BC)	6	IO	PB35D	VREF_5_06	L111C
AM18	5 (BC)	6	IO	PB36C	-	L112T
AM19	5 (BC)	6	IO	PB36D	-	L112C
AN20	5 (BC)	7	IO	PB37C	-	L113T
AP20	5 (BC)	7	IO	PB37D	-	L113C
AL19	5 (BC)	7	IO	PB38C	VREF_5_07	L114T
AL20	5 (BC)	7	IO	PB38D	-	L114C
AK19	5 (BC)	7	IO	PB39C	-	L115T
AK20	5 (BC)	7	IO	PB39D	-	L115C
AM20	5 (BC)	8	IO	PB40C	-	L116T
AM21	5 (BC)	8	IO	PB40D	VREF_5_08	L116C
AN21	5 (BC)	8	IO	PB41C	-	L117T
AP21	5 (BC)	8	IO	PB41D	-	L117C
AN22	5 (BC)	8	IO	PB42C	-	L118T
AP22	5 (BC)	8	IO	PB42D	-	L118C
AL21	5 (BC)	9	IO	PB43C	-	L119T
AL22	5 (BC)	9	IO	PB43D	-	L119C
AK21	5 (BC)	9	IO	PB44C	-	L120T

Table 81. 1156 fpBGA Pin Table (Continued)

F1156 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	F1156 Pair
AK22	5 (BC)	9	IO	PB44D	VREF_5_09	L120C
AM22	5 (BC)	9	IO	PB45C	-	L121T
AM23	5 (BC)	9	IO	PB45D	-	L121C
AN23	5 (BC)	10	IO	PB46C	-	L122T
AP23	5 (BC)	10	IO	PB46D	VREF_5_10	L122C
AL23	-	-	I	RESETN	-	-
AK23	-	-	I	TRISTN	-	-
AP24	-	-	I	TESTMD1N	-	-
AN24	-	-	I	TESTMD0N	-	-
AM24	-	-	I	PDN	-	-
AL24	-	-	I	ATREFCLK	-	-
AK24	-	-	I	TESTCLK	-	-
AP25	-	-	I	BTREFCLK	-	-
AN25	-	-	I	TSTAT1B	-	-
AG22	-	-	I	TSTAT0B	-	-
AM25	-	-	I	TSCLKB	-	-
AH23	-	-	O	RSTAT1B	-	-
AL25	-	-	O	RSTAT0B	-	-
AH24	-	-	O	RSCLKB	-	-
AK30	-	-	I	ATSTAT1N	-	R1C
AL31	-	-	I	ATSTAT0N	-	R2C
AJ30	-	-	I	ATSTAT1P	-	R1T
AK31	-	-	I	ATSTAT0P	-	R2T
AM31	-	-	I	ATSCLKN	-	R3C
AE27	-	-	I/O	ALVCTAP5	-	-
AL32	-	-	I	ATSCLKP	-	R3T
AF29	-	-	I	BTSCLKN	-	R4C
AH30	-	-	I/O	BLVCTAP5	-	-
AG28	-	-	I	BTSCLKP	-	R4T
AM32	-	-	I	BTSTAT1N	-	R5C
AF28	-	-	I	BTSTAT0N	-	R6C
AM33	-	-	I	BTSTAT1P	-	R5T
AE28	-	-	I	BTSTAT0P	-	R6T
AK32	-	-	O	BRSTAT1N	-	R7C
AL33	-	-	O	BRSTAT0N	-	R8C
AJ32	-	-	O	BRSTAT1P	-	R7T
AL34	-	-	O	BRSTAT0P	-	R8T
AJ31	-	-	O	BRSCLKN	-	R9C
AE26	-	-	O	ARSCLKN	-	R10C
AH31	-	-	O	BRSCLKP	-	R9T
AD26	-	-	O	ARSCLKP	-	R10T
AK33	-	-	O	ARSTAT1N	-	R11C
AJ33	-	-	O	ARSTAT0N	-	R12C

Table 81. 1156 fpBGA Pin Table (Continued)

F1156 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	F1156 Pair
AK34	-	-	O	ARSTAT1P	-	R11T
AJ34	-	-	O	ARSTAT0P	-	R12T
AF31	-	-	I	RESLO	-	-
AD28	-	-	I	RESHI	-	-
AG31	-	-	I	REF14	-	-
AE29	-	-	I	REF10	-	-
AH32	-	-	O	BTDAT15N	-	R13C
AG29	-	-	O	BTDAT14N	-	R14C
AG32	-	-	O	BTDAT15P	-	R13T
AG30	-	-	O	BTDAT14P	-	R14T
AH33	-	-	O	BTDAT13N	-	R15C
AD27	-	-	O	BTDAT12N	-	R16C
AH34	-	-	O	BTDAT13P	-	R15T
AC27	-	-	O	BTDAT12P	-	R16T
AG33	-	-	O	BTDAT11N	-	R17C
AE30	-	-	O	BTDAT10N	-	R18C
AG34	-	-	O	BTDAT11P	-	R17T
AF30	-	-	O	BTDAT10P	-	R18T
AF32	-	-	O	BTDAT9N	-	R19C
AD30	-	-	O	BTDAT8N	-	R20C
AE32	-	-	O	BTDAT9P	-	R19T
AD29	-	-	O	BTDAT8P	-	R20T
AF33	-	-	O	BTCTLN	-	R21C
AB26	-	-	O	BTDCLKN	-	R22C
AF34	-	-	O	BTCTLP	-	R21T
AC26	-	-	O	BTDCLKP	-	R22T
AE31	-	-	VDDA_SPIA	VDDA_SPIA	-	-
AD31	-	-	VSS	VSS	-	-
AC28	-	-	O	BTDAT7N	-	R23C
AE33	-	-	O	BTDAT6N	-	R24C
AC29	-	-	O	BTDAT7P	-	R23T
AE34	-	-	O	BTDAT6P	-	R24T
AD32	-	-	O	BTDAT5N	-	R25C
AB28	-	-	O	BTDAT4N	-	R26C
AC32	-	-	O	BTDAT5P	-	R25T
AB29	-	-	O	BTDAT4P	-	R26T
AD33	-	-	O	BTDAT3N	-	R27C
AC30	-	-	O	BTDAT2N	-	R28C
AD34	-	-	O	BTDAT3P	-	R27T
AB30	-	-	O	BTDAT2P	-	R28T
AC31	-	-	O	BTDAT1N	-	R29C
AB27	-	-	O	BTDAT0N	-	R30C
AB31	-	-	O	BTDAT1P	-	R29T

Table 81. 1156 fpBGA Pin Table (Continued)

F1156 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	F1156 Pair
AA27	-	-	O	BTDAT0P	-	R31T
AC33	-	-	I	BRDCLKN	-	R32C
AA29	-	-	I	BRDAT15N	-	R33C
AC34	-	-	I	BRDCLKP	-	R32T
AA30	-	-	I	BRDAT15P	-	R33T
AA31	-	-	I/O	BLVCTAP1	-	-
AA26	-	-	I	BRDAT14N	-	R34C
AB33	-	-	I	BRDAT13N	-	R35C
Y26	-	-	I	BRDAT14P	-	R34T
AB34	-	-	I	BRDAT13P	-	R35T
AA28	-	-	I	BRDAT12N	-	R36C
AB32	-	-	I	BRDAT11N	-	R37C
Y28	-	-	I	BRDAT12P	-	R36T
AA32	-	-	I	BRDAT11P	-	R37T
Y30	-	-	I	BRDAT10N	-	R38C
AA33	-	-	I/O	BLVCTAP2	-	-
Y29	-	-	I	BRDAT10P	-	R38T
Y31	-	-	I	BRDAT9N	-	R39C
Y27	-	-	I	BRDAT8N	-	R40C
Y32	-	-	I	BRDAT9P	-	R39T
W27	-	-	I	BRDAT8P	-	R40T
AA34	-	-	I	BRCTLN	-	R41C
W28	-	-	I/O	BLVCTAP3	-	-
Y34	-	-	I	BRCTLP	-	R41T
W30	-	-	I	BRDAT7N	-	R42C
Y33	-	-	I	BRDAT6N	-	R43C
W29	-	-	I	BRDAT7P	-	R42T
W33	-	-	I	BRDAT6P	-	R43T
W26	-	-	I	BRDAT5N	-	R44C
W31	-	-	I	BRDAT4N	-	R45C
V26	-	-	I	BRDAT5P	-	R44T
W32	-	-	I	BRDAT4P	-	R45T
V28	-	-	I/O	BLVCTAP4	-	-
W34	-	-	I	BRDAT3N	-	R46C
V30	-	-	I	BRDAT2N	-	R47C
V34	-	-	I	BRDAT3P	-	R46T
V29	-	-	I	BRDAT2P	-	R47T
V33	-	-	I	BRDAT1N	-	R48C
V27	-	-	I	BRDAT0N	-	R49C
V32	-	-	I	BRDAT1P	-	R48T
U27	-	-	I	BRDAT0P	-	R49T
V31	-	-	VSS	VSS	-	-
U31	-	-	VDDA_SPIB	VDDA_SPIB	-	-

Table 81. 1156 fpBGA Pin Table (Continued)

F1156 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	F1156 Pair
U28	-	-	O	ATDAT15N	-	R50C
U32	-	-	VSS	VSS	-	-
U29	-	-	O	ATDAT15P	-	R50T
U33	-	-	O	ATDAT14N	-	R51C
U34	-	-	O	ATDAT14P	-	R51T
U30	-	-	O	ATDAT13N	-	R52C
T34	-	-	O	ATDAT12N	-	R53C
T29	-	-	O	ATDAT13P	-	R52T
T33	-	-	O	ATDAT12P	-	R53T
T32	-	-	O	ATDAT11N	-	R54C
U26	-	-	O	ATDAT10N	-	R55C
T31	-	-	O	ATDAT11P	-	R54T
T26	-	-	O	ATDAT10P	-	R55T
R34	-	-	O	ATDAT9N	-	R56C
T28	-	-	O	ATDAT8N	-	R57C
R33	-	-	O	ATDAT9P	-	R56T
T30	-	-	O	ATDAT8P	-	R57T
R32	-	-	O	ATCTLN	-	R58C
T27	-	-	O	ATDCLKN	-	R59C
R31	-	-	O	ATCTLP	-	R58T
R27	-	-	O	ATDCLKP	-	R59T
P34	-	-	O	ATDAT7N	-	R60C
R28	-	-	O	ATDAT6N	-	R61C
P33	-	-	O	ATDAT7P	-	R60T
R29	-	-	O	ATDAT6P	-	R61T
N34	-	-	O	ATDAT5N	-	R62C
R30	-	-	O	ATDAT4N	-	R63C
N33	-	-	O	ATDAT5P	-	R62T
P30	-	-	O	ATDAT4P	-	R63T
P32	-	-	O	ATDAT3N	-	R64C
R26	-	-	O	ATDAT2N	-	R65C
P31	-	-	O	ATDAT3P	-	R64T
P26	-	-	O	ATDAT2P	-	R65T
M34	-	-	O	ATDAT1N	-	R66C
P28	-	-	O	ATDAT0N	-	R67C
M33	-	-	O	ATDAT1P	-	R66T
P29	-	-	O	ATDAT0P	-	R67T
N31	-	-	VSS	VSS	-	-
M32	-	-	VDDA_SPIC	VDDA_SPIC	-	-
N32	-	-	VSS	VSS	-	-
L34	-	-	I	ARDCLKN	-	R68C
P27	-	-	I	ARDAT15N	-	R69C
L33	-	-	I	ARDCLKP	-	R68T

Table 81. 1156 fpBGA Pin Table (Continued)

F1156 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	F1156 Pair
N27	-	-	I	ARDAT15P	-	R69T
L32	-	-	I/O	ALVCTAP1	-	-
N30	-	-	I	ARDAT14N	-	R70C
K34	-	-	I	ARDAT13N	-	R71C
M30	-	-	I	ARDAT14P	-	R70T
K33	-	-	I	ARDAT13P	-	R71T
N28	-	-	I	ARDAT12N	-	R72C
J32	-	-	I	ARDAT11N	-	R73C
N29	-	-	I	ARDAT12P	-	R72T
K32	-	-	I	ARDAT11P	-	R73T
L30	-	-	I/O	ALVCTAP2	-	-
M31	-	-	I	ARDAT10N	-	R74C
N26	-	-	I	ARDAT9N	-	R75C
L31	-	-	I	ARDAT10P	-	R74T
M26	-	-	I	ARDAT9P	-	R75T
J34	-	-	I	ARDAT8N	-	R76C
M28	-	-	I	ARCTLN	-	R77C
J33	-	-	I	ARDAT8P	-	R76T
M29	-	-	I	ARCTLP	-	R77T
K31	-	-	I/O	ALVCTAP3	-	-
L27	-	-	I	ARDAT7N	-	R78C
H34	-	-	I	ARDAT6N	-	R79C
M27	-	-	I	ARDAT7P	-	R78T
H33	-	-	I	ARDAT6P	-	R79T
L29	-	-	I	ARDAT5N	-	R80C
J31	-	-	I	ARDAT4N	-	R81C
K30	-	-	I	ARDAT5P	-	R80T
H31	-	-	I	ARDAT4P	-	R81T
L28	-	-	I/O	ALVCTAP4	-	-
G34	-	-	I	ARDAT3N	-	R82C
J30	-	-	I	ARDAT2N	-	R83C
G33	-	-	I	ARDAT3P	-	R82T
H30	-	-	I	ARDAT2P	-	R83T
F34	-	-	I	ARDAT1N	-	R84C
L26	-	-	I	ARDAT0N	-	R85C
F33	-	-	I	ARDAT1P	-	R84T
K26	-	-	I	ARDAT0P	-	R85T
H32	-	-	VDDA_SPID	VDDA_SPID	-	-
G32	-	-	VSS	VSS	-	-
G30	-	-	I	TSTAT1A	-	-
G31	-	-	I	TSTAT0A	-	-
H29	-	-	I	TSCLKA	-	-
F32	-	-	O	RSTAT1A	-	-

Table 81. 1156 fpBGA Pin Table (Continued)

F1156 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	F1156 Pair
K29	-	-	O	RSTAT0A	-	-
E34	-	-	O	RSCLKA	-	-
E33	-	-	I/O	PMIA17	-	-
D34	-	-	I/O	PMIA16	-	-
K28	-	-	I/O	PMIA15	-	-
D33	-	-	I/O	PMIA14	-	-
G29	-	-	I/O	PMIA13	-	-
C33	-	-	I/O	PMIA12	-	-
K27	-	-	I/O	PMIA11	-	-
E32	-	-	I/O	PMIA10	-	-
D32	-	-	I/O	PMIA9	-	-
C32	-	-	I/O	PMIA8	-	-
G28	-	-	I/O	PMIWN	-	-
B32	-	-	I/O	PMIRN	-	-
J29	-	-	I/O	PMIA7	-	-
C31	-	-	VREF	REFI_1	-	-
J28	-	-	I/O	PMIA6	-	-
D31	-	-	I/O	PMIA5	-	-
J27	-	-	I/O	PMIA4	-	-
F30	-	-	I/O	PMIA3	-	-
H27	-	-	I/O	PMIA2	-	-
E31	-	-	I/O	PMIA1	-	-
H28	-	-	I/O	PMIA0	-	-
F31	-	-	I	EXT_1K	-	-
F29	-	-	I/O	PMID35	-	-
E30	-	-	I/O	PMID34	-	-
F28	-	-	I/O	PMID33	-	-
D30	-	-	I/O	PMID32	-	-
E27	-	-	I/O	PMID31	-	-
C30	-	-	I/O	PMID30	-	-
F27	-	-	I/O	PMID29	-	-
B31	-	-	I/O	PMID28	-	-
G27	-	-	I/O	PMID27	-	-
A31	-	-	I/O	PMID26	-	-
E26	-	-	I/O	PMID25	-	-
B30	-	-	I/O	PMID24	-	-
F26	-	-	I/O	PMID23	-	-
A30	-	-	I/O	PMID22	-	-
G26	-	-	I/O	PMID21	-	-
E29	-	-	I/O	PMID20	-	-
H26	-	-	I/O	PMID19	-	-
E28	-	-	I/O	PMID18	-	-
E25	-	-	I/O	PMID17	-	-

Table 81. 1156 fpBGA Pin Table (Continued)

F1156 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	F1156 Pair
D29	-	-	I/O	PMID16	-	-
C29	-	-	VREF	REFI_2	-	-
F25	-	-	I/O	PMIK	-	-
B29	-	-	I/O	PMIKN	-	-
G25	-	-	I/O	PMID15	-	-
A29	-	-	I/O	PMID14	-	-
H25	-	-	I/O	PMID13	-	-
D28	-	-	I/O	PMID12	-	-
E24	-	-	I/O	PMID11	-	-
C28	-	-	I/O	PMID10	-	-
F24	-	-	I/O	PMID9	-	-
B28	-	-	I/O	PMID8	-	-
G24	-	-	I/O	PMID7	-	-
A28	-	-	I/O	PMID6	-	-
H24	-	-	I/O	PMID5	-	-
D27	-	-	I/O	PMID4	-	-
E23	-	-	I/O	PMID3	-	-
C27	-	-	I/O	PMID2	-	-
F23	-	-	I/O	PMID1	-	-
B27	-	-	I/O	PMID0	-	-
A27	-	-	VREF	REFI_3	-	-
G23	-	-	I/O	PMIC	-	-
D26	-	-	I/O	PMICN	-	-
H23	-	-	I/O	PMIQ35	-	-
C26	-	-	I/O	PMIQ34	-	-
E22	-	-	I/O	PMIQ33	-	-
B26	-	-	I/O	PMIQ32	-	-
F22	-	-	I/O	PMIQ31	-	-
A26	-	-	I/O	PMIQ30	-	-
E21	-	-	I/O	PMIQ29	-	-
D25	-	-	I/O	PMIQ28	-	-
G22	-	-	I/O	PMIQ27	-	-
C25	-	-	I/O	PMIQ26	-	-
F21	-	-	I/O	PMIQ25	-	-
B25	-	-	I/O	PMIQ24	-	-
G20	-	-	I/O	PMIQ23	-	-
A25	-	-	I/O	PMIQ22	-	-
F20	-	-	I/O	PMIQ21	-	-
D24	-	-	I/O	PMIQ20	-	-
E20	-	-	I/O	PMIQ19	-	-
C24	-	-	I/O	PMIQ18	-	-
G21	-	-	I/O	PMIQ17	-	-
B24	-	-	I/O	PMIQ16	-	-

Table 81. 1156 fpBGA Pin Table (Continued)

F1156 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	F1156 Pair
H22	-	-	I/O	MCREFCLK	-	-
A24	-	-	I/O	SPARE_2	-	-
E19	-	-	I/O	PMIQ15	-	-
D23	-	-	VREF	REFI_4	-	-
F19	-	-	I/O	PMIQ14	-	-
C23	-	-	I/O	PMIQ13	-	-
G19	-	-	I/O	PMIQ12	-	-
B23	-	-	VDDA_PLL	VDDA_PLL	-	-
H21	-	-	I/O	PMIQ11	-	-
A23	-	-	VSS	VSS	-	-
H20	-	-	I/O	PMIQ10	-	-
D22	-	-	I/O	PMIQ9	-	-
H19	-	-	I/O	PMIQ8	-	-
C22	-	-	I/O	PMIQ7	-	-
H18	-	-	I/O	PMIQ6	-	-
B22	-	-	I/O	PMIQ5	-	-
E18	-	-	I/O	PMIQ4	-	-
A22	-	-	I/O	PMIQ3	-	-
F18	-	-	I/O	PMIQ2	-	-
C21	-	-	I/O	PMIQ1	-	-
G18	-	-	I/O	PMIQ0	-	-
B21	1 (TC)	7	IO	PT46D	-	L123C
A21	1 (TC)	7	IO	PT46C	-	L123T
A20	1 (TC)	7	IO	PT45D	VREF_1_07	L124C
A19	1 (TC)	7	IO	PT45C	-	L124T
C20	1 (TC)	8	IO	PT44D	-	L125C
B20	1 (TC)	8	IO	PT44C	-	L125T
B19	1 (TC)	8	IO	PT43D	-	L126C
A18	1 (TC)	8	IO	PT43C	-	L126T
C19	1 (TC)	8	IO	PT42D	VREF_1_08	L127C
D19	1 (TC)	8	IO	PT42C	-	L127T
B18	1 (TC)	9	IO	PT41D	-	L128C
A17	1 (TC)	9	IO	PT41C	-	L128T
C18	1 (TC)	9	IO	PT40D	-	L129C
D18	1 (TC)	9	IO	PT40C	VREF_1_09	L129T
B17	1 (TC)	9	IO	PT39D	-	L130C
C17	1 (TC)	9	IO	PT39C	-	L130T
A16	1 (TC)	10	IO	PT38D	-	L131C
B16	1 (TC)	10	IO	PT38C	VREF_1_10	L131T
A15	1 (TC)	10	IO	PT37D	-	L132C
B15	1 (TC)	10	IO	PT37C	-	L132T
A14	1 (TC)	10	IO	PT36D	-	L133C
B14	1 (TC)	10	IO	PT36C	-	L133T

Table 81. 1156 fpBGA Pin Table (Continued)

F1156 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	F1156 Pair
E17	1 (TC)	1	IO	PT35D	-	L134C
H17	1 (TC)	1	IO	PT35B	-	L135C
E16	1 (TC)	1	IO	PT35C	-	L134T
G17	1 (TC)	1	IO	PT35A	-	L135T
C16	1 (TC)	1	IO	PT34D	VREF_1_01	L136C
G15	1 (TC)	1	IO	PT34B	-	L137C
C15	1 (TC)	1	IO	PT34C	-	L136T
F15	1 (TC)	1	IO	PT34A	-	L137T
D17	1 (TC)	1	IO	PT33D	-	L138C
F14	1 (TC)	1	IO	PT33B	-	L139C
D16	1 (TC)	1	IO	PT33C	-	L138T
G14	1 (TC)	1	IO	PT33A	-	L139T
F17	1 (TC)	2	IO	PT32D	-	L140C
H16	1 (TC)	2	IO	PT32B	-	L141C
F16	1 (TC)	2	IO	PT32C	VREF_1_02	L140T
G16	1 (TC)	2	IO	PT32A	-	L141T
A13	1 (TC)	2	IO	PT31D	-	L142C
F13	1 (TC)	2	IO	PT31B	-	L143C
B13	1 (TC)	2	IO	PT31C	-	L142T
F12	1 (TC)	2	IO	PT31A	-	L143T
C14	1 (TC)	2	IO	PT30D	-	L144C
G13	1 (TC)	2	IO	PT30B	-	L145C
C13	1 (TC)	2	IO	PT30C	-	L144T
H13	1 (TC)	2	IO	PT30A	-	L145T
D15	1 (TC)	3	IO	PT29D	-	L146C
G12	1 (TC)	3	IO	PT29B	-	L147C
D14	1 (TC)	3	IO	PT29C	VREF_1_03	L146T
H12	1 (TC)	3	IO	PT29A	-	L147T
A12	1 (TC)	3	IO	PT28D	-	L148C
H15	1 (TC)	3	IO	PT28B	-	L149C
B12	1 (TC)	3	IO	PT28C	-	L148T
H14	1 (TC)	3	IO	PT28A	-	L149T
D13	1 (TC)	3	IO	PT27D	-	L150C
D12	1 (TC)	3	IO	PT27C	-	L150T
E15	1 (TC)	4	IO	PT26D	-	L151C
E14	1 (TC)	4	IO	PT26C	-	L151T
A11	1 (TC)	4	IO	PT25D	-	L152C
J14	1 (TC)	4	IO	PT25B	-	L153C
A10	1 (TC)	4	IO	PT25C	-	L152T
J13	1 (TC)	4	IO	PT25A	-	L153T
C12	1 (TC)	4	IO	PT24D	-	L154C
C11	1 (TC)	4	IO	PT24C	VREF_1_04	L154T
B11	1 (TC)	5	IO	PT23D	PTCK1C	L155C

Table 81. 1156 fpBGA Pin Table (Continued)

F1156 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	F1156 Pair
B10	1 (TC)	5	IO	PT23C	PTCK1T	L155T
E13	1 (TC)	5	IO	PT22D	PTCK0C	L156C
E12	1 (TC)	5	IO	PT22C	PTCK0T	L156T
A9	1 (TC)	5	IO	PT21D	VREF_1_05	L157C
B9	1 (TC)	5	IO	PT21C	-	L157T
C10	1 (TC)	6	IO	PT20D	-	L158C
C9	1 (TC)	6	IO	PT20C	-	L158T
D11	1 (TC)	6	IO	PT19D	-	L159C
D10	1 (TC)	6	IO	PT19C	VREF_1_06	L159T
A8	0 (TL)	1	IO	PT18D	MPI_RTRY_N	L160C
B8	0 (TL)	1	IO	PT18C	MPI_ACK_N	L160T
H11	0 (TL)	1	IO	PT17D	-	L161C
G11	0 (TL)	1	IO	PT17C	VREF_0_01	L161T
H10	0 (TL)	1	IO	PT16D	M0	L162C
G10	0 (TL)	1	IO	PT16C	M1	L162T
A7	0 (TL)	2	IO	PT15D	MPI_CLK	L163C
B7	0 (TL)	2	IO	PT15C	A21/MPI_BURST_N	L163T
C8	0 (TL)	2	IO	PT14D	M2	L164C
C7	0 (TL)	2	IO	PT14C	M3	L164T
E11	0 (TL)	2	IO	PT13D	VREF_0_02	L165C
E10	0 (TL)	2	IO	PT13C	MPI_TEA_N	L165T
F11	0 (TL)	3	IO	PT12D	-	L166C
F10	0 (TL)	3	IO	PT12C	-	L166T
D9	0 (TL)	3	IO	PT11D	VREF_0_03	L167C
D8	0 (TL)	3	IO	PT11C	-	L167T
H9	0 (TL)	3	IO	PT10D	D0	L168C
G9	0 (TL)	3	IO	PT10C	TMS	L168T
H7	0 (TL)	4	IO	PT9D	A20/MPI_BDIP_N	L169C
G7	0 (TL)	4	IO	PT9C	A19/MPI_TSZ1	L169T
A6	0 (TL)	4	IO	PT8D	A18/MPI_TSZ0	L170C
B6	0 (TL)	4	IO	PT8C	D3	L170T
E9	0 (TL)	4	IO	PT7D	VREF_0_04	L171C
E8	0 (TL)	4	IO	PT7C	-	L171T
F9	0 (TL)	5	IO	PT6D	D1	L172C
F8	0 (TL)	5	IO	PT6C	D2	L172T
C6	0 (TL)	5	IO	PT5D	-	L173C
C5	0 (TL)	5	IO	PT5C	VREF_0_05	L173T
D7	0 (TL)	5	IO	PT4D	TDI	L174C
D6	0 (TL)	5	IO	PT4C	TCK	L174T
E7	0 (TL)	6	IO	PT3D	-	L175C
G8	0 (TL)	6	IO	PT3B	-	L176C
E6	0 (TL)	6	IO	PT3C	VREF_0_06	L175T
H8	0 (TL)	6	IO	PT3A	-	L176T

Table 81. 1156 fpBGA Pin Table (Continued)

F1156 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	F1156 Pair
F7	0 (TL)	6	IO	PT2D	PLL_CK1C/PPLL	L177C
F6	0 (TL)	6	IO	PT2C	PLL_CK1T/PPLL	L177T
G5	-	-	O	PCFG_MPI_IRQ	CFG_IRQ_N/MPI_IRQ_N	-
G6	-	-	IO	PCCLK	CCLK	-
F5	-	-	IO	PDONE	DONE	-
AA24	-	-	VDD15	VDD15	-	-
AA8	-	-	VDD15	VDD15	-	-
AA9	-	-	VDD15	VDD15	-	-
AB8	-	-	VDD15	VDD15	-	-
AB9	-	-	VDD15	VDD15	-	-
AC13	-	-	VDD15	VDD15	-	-
AC14	-	-	VDD15	VDD15	-	-
AC15	-	-	VDD15	VDD15	-	-
AC16	-	-	VDD15	VDD15	-	-
AC17	-	-	VDD15	VDD15	-	-
AC18	-	-	VDD15	VDD15	-	-
AC19	-	-	VDD15	VDD15	-	-
AC23	-	-	VDD15	VDD15	-	-
AC8	-	-	VDD15	VDD15	-	-
AC9	-	-	VDD15	VDD15	-	-
AD10	-	-	VDD15	VDD15	-	-
AD11	-	-	VDD15	VDD15	-	-
AD12	-	-	VDD15	VDD15	-	-
AD13	-	-	VDD15	VDD15	-	-
AD14	-	-	VDD15	VDD15	-	-
AD15	-	-	VDD15	VDD15	-	-
AD16	-	-	VDD15	VDD15	-	-
AD17	-	-	VDD15	VDD15	-	-
AD18	-	-	VDD15	VDD15	-	-
AD19	-	-	VDD15	VDD15	-	-
AD20	-	-	VDD15	VDD15	-	-
AD21	-	-	VDD15	VDD15	-	-
AD22	-	-	VDD15	VDD15	-	-
AD23	-	-	VDD15	VDD15	-	-
AD8	-	-	VDD15	VDD15	-	-
AE10	-	-	VDD15	VDD15	-	-
AE11	-	-	VDD15	VDD15	-	-
AE12	-	-	VDD15	VDD15	-	-
AE13	-	-	VDD15	VDD15	-	-
AE14	-	-	VDD15	VDD15	-	-
AE15	-	-	VDD15	VDD15	-	-
AE16	-	-	VDD15	VDD15	-	-

Table 81. 1156 fpBGA Pin Table (Continued)

F1156 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	F1156 Pair
AE17	-	-	VDD15	VDD15	-	-
AE18	-	-	VDD15	VDD15	-	-
AE19	-	-	VDD15	VDD15	-	-
AE20	-	-	VDD15	VDD15	-	-
AE21	-	-	VDD15	VDD15	-	-
AE22	-	-	VDD15	VDD15	-	-
AE23	-	-	VDD15	VDD15	-	-
AE9	-	-	VDD15	VDD15	-	-
K10	-	-	VDD15	VDD15	-	-
K11	-	-	VDD15	VDD15	-	-
K12	-	-	VDD15	VDD15	-	-
K9	-	-	VDD15	VDD15	-	-
L10	-	-	VDD15	VDD15	-	-
L11	-	-	VDD15	VDD15	-	-
L12	-	-	VDD15	VDD15	-	-
L13	-	-	VDD15	VDD15	-	-
L14	-	-	VDD15	VDD15	-	-
L15	-	-	VDD15	VDD15	-	-
L16	-	-	VDD15	VDD15	-	-
L8	-	-	VDD15	VDD15	-	-
M13	-	-	VDD15	VDD15	-	-
M14	-	-	VDD15	VDD15	-	-
M8	-	-	VDD15	VDD15	-	-
M9	-	-	VDD15	VDD15	-	-
N8	-	-	VDD15	VDD15	-	-
N9	-	-	VDD15	VDD15	-	-
P8	-	-	VDD15	VDD15	-	-
P9	-	-	VDD15	VDD15	-	-
R8	-	-	VDD15	VDD15	-	-
R9	-	-	VDD15	VDD15	-	-
T8	-	-	VDD15	VDD15	-	-
T9	-	-	VDD15	VDD15	-	-
U8	-	-	VDD15	VDD15	-	-
U9	-	-	VDD15	VDD15	-	-
V24	-	-	VDD15	VDD15	-	-
V8	-	-	VDD15	VDD15	-	-
V9	-	-	VDD15	VDD15	-	-
W24	-	-	VDD15	VDD15	-	-
W8	-	-	VDD15	VDD15	-	-
W9	-	-	VDD15	VDD15	-	-
Y24	-	-	VDD15	VDD15	-	-
Y8	-	-	VDD15	VDD15	-	-
Y9	-	-	VDD15	VDD15	-	-

Table 81. 1156 fpBGA Pin Table (Continued)

F1156 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	F1156 Pair
AA25	-	-	VDD33	VDD33	-	-
AB24	-	-	VDD33	VDD33	-	-
AB25	-	-	VDD33	VDD33	-	-
AC24	-	-	VDD33	VDD33	-	-
AC25	-	-	VDD33	VDD33	-	-
AD24	-	-	VDD33	VDD33	-	-
AD25	-	-	VDD33	VDD33	-	-
AE24	-	-	VDD33	VDD33	-	-
AE25	-	-	VDD33	VDD33	-	-
AF22	-	-	VDD33	VDD33	-	-
AF23	-	-	VDD33	VDD33	-	-
AF24	-	-	VDD33	VDD33	-	-
AF25	-	-	VDD33	VDD33	-	-
AG23	-	-	VDD33	VDD33	-	-
AG24	-	-	VDD33	VDD33	-	-
AG25	-	-	VDD33	VDD33	-	-
AH22	-	-	VDD33	VDD33	-	-
N24	-	-	VDD33	VDD33	-	-
N25	-	-	VDD33	VDD33	-	-
P24	-	-	VDD33	VDD33	-	-
P25	-	-	VDD33	VDD33	-	-
R24	-	-	VDD33	VDD33	-	-
R25	-	-	VDD33	VDD33	-	-
T24	-	-	VDD33	VDD33	-	-
T25	-	-	VDD33	VDD33	-	-
U24	-	-	VDD33	VDD33	-	-
U25	-	-	VDD33	VDD33	-	-
V25	-	-	VDD33	VDD33	-	-
W25	-	-	VDD33	VDD33	-	-
Y25	-	-	VDD33	VDD33	-	-
J7	-	-	VDD33	VDD33_FPGAPLL	-	-
K8	-	-	VDD33	VDD33_FPGAPLL	-	-
AE8	-	-	VDD33	VDD33_FPGAPLL	-	-
AF7	-	-	VDD33	VDD33_FPGAPLL	-	-
AH20	-	-	VDD33	VDD33_FPGAPLL	-	-
AH21	-	-	VDD33	VDD33_FPGAPLL	-	-
D20	-	-	VDD33	VDD33_FPGAPLL	-	-
D21	-	-	VDD33	VDD33_FPGAPLL	-	-
J19	-	-	VDDH	VDDH	-	-
J20	-	-	VDDH	VDDH	-	-
J21	-	-	VDDH	VDDH	-	-
J23	-	-	VDDH	VDDH	-	-
J25	-	-	VDDH	VDDH	-	-

Table 81. 1156 fpBGA Pin Table (Continued)

F1156 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	F1156 Pair
K19	-	-	VDDH	VDDH	-	-
K20	-	-	VDDH	VDDH	-	-
K21	-	-	VDDH	VDDH	-	-
K23	-	-	VDDH	VDDH	-	-
K25	-	-	VDDH	VDDH	-	-
L18	-	-	VDDH	VDDH	-	-
L19	-	-	VDDH	VDDH	-	-
L20	-	-	VDDH	VDDH	-	-
L21	-	-	VDDH	VDDH	-	-
L23	-	-	VDDH	VDDH	-	-
M18	-	-	VDDH	VDDH	-	-
M19	-	-	VDDH	VDDH	-	-
M20	-	-	VDDH	VDDH	-	-
M21	-	-	VDDH	VDDH	-	-
M23	-	-	VDDH	VDDH	-	-
M25	-	-	VDDH	VDDH	-	-
N18	-	-	VDDH	VDDH	-	-
N19	-	-	VDDH	VDDH	-	-
N20	-	-	VDDH	VDDH	-	-
N21	-	-	VDDH	VDDH	-	-
N23	-	-	VDDH	VDDH	-	-
P18	-	-	VDDH	VDDH	-	-
P19	-	-	VDDH	VDDH	-	-
P20	-	-	VDDH	VDDH	-	-
P21	-	-	VDDH	VDDH	-	-
J10	0 (TL)	-	VDDIO0	VDDIO0	-	-
J11	0 (TL)	-	VDDIO0	VDDIO0	-	-
J12	0 (TL)	-	VDDIO0	VDDIO0	-	-
J8	0 (TL)	-	VDDIO0	VDDIO0	-	-
J9	0 (TL)	-	VDDIO0	VDDIO0	-	-
K7	0 (TL)	-	VDDIO0	VDDIO0	-	-
L7	0 (TL)	-	VDDIO0	VDDIO0	-	-
M7	0 (TL)	-	VDDIO0	VDDIO0	-	-
N7	0 (TL)	-	VDDIO0	VDDIO0	-	-
P7	0 (TL)	-	VDDIO0	VDDIO0	-	-
J15	0 (TL)	-	VDDIO1	VDDIO1	-	-
J16	0 (TL)	-	VDDIO1	VDDIO1	-	-
J17	0 (TL)	-	VDDIO1	VDDIO1	-	-
J18	0 (TL)	-	VDDIO1	VDDIO1	-	-
K13	0 (TL)	-	VDDIO1	VDDIO1	-	-
K14	0 (TL)	-	VDDIO1	VDDIO1	-	-
K15	0 (TL)	-	VDDIO1	VDDIO1	-	-
K16	0 (TL)	-	VDDIO1	VDDIO1	-	-

Table 81. 1156 fpBGA Pin Table (Continued)

F1156 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	F1156 Pair
K17	0 (TL)	-	VDDIO1	VDDIO1	-	-
K18	0 (TL)	-	VDDIO1	VDDIO1	-	-
AF15	0 (TL)	-	VDDIO5	VDDIO5	-	-
AF16	0 (TL)	-	VDDIO5	VDDIO5	-	-
AF17	0 (TL)	-	VDDIO5	VDDIO5	-	-
AF18	0 (TL)	-	VDDIO5	VDDIO5	-	-
AF19	0 (TL)	-	VDDIO5	VDDIO5	-	-
AF20	0 (TL)	-	VDDIO5	VDDIO5	-	-
AF21	0 (TL)	-	VDDIO5	VDDIO5	-	-
AG19	0 (TL)	-	VDDIO5	VDDIO5	-	-
AG20	0 (TL)	-	VDDIO5	VDDIO5	-	-
AG21	0 (TL)	-	VDDIO5	VDDIO5	-	-
AC7	0 (TL)	-	VDDIO6	VDDIO6	-	-
AD7	0 (TL)	-	VDDIO6	VDDIO6	-	-
AE7	0 (TL)	-	VDDIO6	VDDIO6	-	-
AF10	0 (TL)	-	VDDIO6	VDDIO6	-	-
AF11	0 (TL)	-	VDDIO6	VDDIO6	-	-
AF12	0 (TL)	-	VDDIO6	VDDIO6	-	-
AF13	0 (TL)	-	VDDIO6	VDDIO6	-	-
AF14	0 (TL)	-	VDDIO6	VDDIO6	-	-
AF8	0 (TL)	-	VDDIO6	VDDIO6	-	-
AF9	0 (TL)	-	VDDIO6	VDDIO6	-	-
AA7	0 (TL)	-	VDDIO7	VDDIO7	-	-
AB7	0 (TL)	-	VDDIO7	VDDIO7	-	-
R7	0 (TL)	-	VDDIO7	VDDIO7	-	-
T7	0 (TL)	-	VDDIO7	VDDIO7	-	-
U6	0 (TL)	-	VDDIO7	VDDIO7	-	-
U7	0 (TL)	-	VDDIO7	VDDIO7	-	-
V6	0 (TL)	-	VDDIO7	VDDIO7	-	-
V7	0 (TL)	-	VDDIO7	VDDIO7	-	-
W7	0 (TL)	-	VDDIO7	VDDIO7	-	-
Y7	0 (TL)	-	VDDIO7	VDDIO7	-	-
B5	-	-	NC		-	-
D5	-	-	NC		-	-
C34	-	-	NC		-	-
D4	-	-	NC		-	-
D3	-	-	NC		-	-
D2	-	-	NC		-	-
E5	-	-	NC		-	-
E4	-	-	NC		-	-
E3	-	-	NC		-	-
E2	-	-	NC		-	-
E1	-	-	NC		-	-

Table 81. 1156 fpBGA Pin Table (Continued)

F1156 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	F1156 Pair
F3	-	-	NC		-	-
B4	-	-	NC		-	-
F2	-	-	NC		-	-
F1	-	-	NC		-	-
G3	-	-	NC		-	-
P6	-	-	NC		-	-
N6	-	-	NC		-	-
M6	-	-	NC		-	-
A5	-	-	NC		-	-
L6	-	-	NC		-	-
K6	-	-	NC		-	-
AL5	-	-	NC		-	-
AL4	-	-	NC		-	-
AL3	-	-	NC		-	-
AL2	-	-	NC		-	-
AL1	-	-	NC		-	-
AK5	-	-	NC		-	-
AK4	-	-	NC		-	-
AK3	-	-	NC		-	-
A4	-	-	NC		-	-
AK2	-	-	NC		-	-
AK1	-	-	NC		-	-
AM5	-	-	NC		-	-
AM4	-	-	NC		-	-
AM3	-	-	NC		-	-
AM2	-	-	NC		-	-
AN4	-	-	NC		-	-
AN3	-	-	NC		-	-
AP4	-	-	NC		-	-
AJ5	-	-	NC		-	-
B3	-	-	NC		-	-
AJ4	-	-	NC		-	-
AH5	-	-	NC		-	-
AG5	-	-	NC		-	-
AG6	-	-	NC		-	-
AJ20	-	-	NC		-	-
AJ25	-	-	NC		-	-
AJ24	-	-	NC		-	-
AJ23	-	-	NC		-	-
AJ22	-	-	NC		-	-
AJ21	-	-	NC		-	-
C4	-	-	NC		-	-
AJ29	-	-	NC		-	-

Table 81. 1156 fpBGA Pin Table (Continued)

F1156 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	F1156 Pair
AJ28	-	-	NC		-	-
AJ27	-	-	NC		-	-
AJ26	-	-	NC		-	-
AH27	-	-	NC		-	-
AH26	-	-	NC		-	-
AG27	-	-	NC		-	-
AG26	-	-	NC		-	-
AF27	-	-	NC		-	-
AF26	-	-	NC		-	-
C3	-	-	NC		-	-
AH25	-	-	NC		-	-
AH29	-	-	NC		-	-
AH28	-	-	NC		-	-
AK29	-	-	NC		-	-
AK28	-	-	NC		-	-
AK27	-	-	NC		-	-
AK26	-	-	NC		-	-
AK25	-	-	NC		-	-
AP29	-	-	NC		-	-
AP28	-	-	NC		-	-
C2	-	-	NC		-	-
AP27	-	-	NC		-	-
AP26	-	-	NC		-	-
AN29	-	-	NC		-	-
AN28	-	-	NC		-	-
AN27	-	-	NC		-	-
AN26	-	-	NC		-	-
AM29	-	-	NC		-	-
AM28	-	-	NC		-	-
AM27	-	-	NC		-	-
AM26	-	-	NC		-	-
D1	-	-	NC		-	-
AL29	-	-	NC		-	-
AL28	-	-	NC		-	-
AL27	-	-	NC		-	-
AL26	-	-	NC		-	-
A32	-	-	NC		-	-
A33	-	-	NC		-	-
A34	-	-	NC		-	-
AL30	-	-	NC		-	-
B34	-	-	NC		-	-
B33	-	-	NC		-	-
A1	-	-	VSS	VSS	-	-

Table 81. 1156 fpBGA Pin Table (Continued)

F1156 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	F1156 Pair
A2	-	-	VSS	VSS	-	-
A3	-	-	VSS	VSS	-	-
AA10	-	-	VSS	VSS	-	-
AA11	-	-	VSS	VSS	-	-
AA12	-	-	VSS	VSS	-	-
AA13	-	-	VSS	VSS	-	-
AA14	-	-	VSS	VSS	-	-
AA15	-	-	VSS	VSS	-	-
AA16	-	-	VSS	VSS	-	-
AA17	-	-	VSS	VSS	-	-
AA18	-	-	VSS	VSS	-	-
AA19	-	-	VSS	VSS	-	-
AA20	-	-	VSS	VSS	-	-
AA21	-	-	VSS	VSS	-	-
AA22	-	-	VSS	VSS	-	-
AA23	-	-	VSS	VSS	-	-
AB10	-	-	VSS	VSS	-	-
AB11	-	-	VSS	VSS	-	-
AB12	-	-	VSS	VSS	-	-
AB13	-	-	VSS	VSS	-	-
AB14	-	-	VSS	VSS	-	-
AB15	-	-	VSS	VSS	-	-
AB16	-	-	VSS	VSS	-	-
AB17	-	-	VSS	VSS	-	-
AB18	-	-	VSS	VSS	-	-
AB19	-	-	VSS	VSS	-	-
AB20	-	-	VSS	VSS	-	-
AB21	-	-	VSS	VSS	-	-
AB22	-	-	VSS	VSS	-	-
AB23	-	-	VSS	VSS	-	-
AC10	-	-	VSS	VSS	-	-
AC11	-	-	VSS	VSS	-	-
AC12	-	-	VSS	VSS	-	-
AC20	-	-	VSS	VSS	-	-
AC21	-	-	VSS	VSS	-	-
AC22	-	-	VSS	VSS	-	-
AD9	-	-	VSS	VSS	-	-
AM1	-	-	VSS	VSS	-	-
AM30	-	-	VSS	VSS	-	-
AM34	-	-	VSS	VSS	-	-
AN1	-	-	VSS	VSS	-	-
AN2	-	-	VSS	VSS	-	-
AN30	-	-	VSS	VSS	-	-

Table 81. 1156 fpBGA Pin Table (Continued)

F1156 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	F1156 Pair
AN31	-	-	VSS	VSS	-	-
AN32	-	-	VSS	VSS	-	-
AN33	-	-	VSS	VSS	-	-
AN34	-	-	VSS	VSS	-	-
AP1	-	-	VSS	VSS	-	-
AP2	-	-	VSS	VSS	-	-
AP3	-	-	VSS	VSS	-	-
AP30	-	-	VSS	VSS	-	-
AP31	-	-	VSS	VSS	-	-
AP32	-	-	VSS	VSS	-	-
AP33	-	-	VSS	VSS	-	-
AP34	-	-	VSS	VSS	-	-
B1	-	-	VSS	VSS	-	-
B2	-	-	VSS	VSS	-	-
C1	-	-	VSS	VSS	-	-
L17	-	-	VSS	VSS	-	-
L9	-	-	VSS	VSS	-	-
M10	-	-	VSS	VSS	-	-
M11	-	-	VSS	VSS	-	-
M12	-	-	VSS	VSS	-	-
M15	-	-	VSS	VSS	-	-
M16	-	-	VSS	VSS	-	-
M17	-	-	VSS	VSS	-	-
N10	-	-	VSS	VSS	-	-
N11	-	-	VSS	VSS	-	-
N12	-	-	VSS	VSS	-	-
N13	-	-	VSS	VSS	-	-
N14	-	-	VSS	VSS	-	-
N15	-	-	VSS	VSS	-	-
N16	-	-	VSS	VSS	-	-
N17	-	-	VSS	VSS	-	-
P10	-	-	VSS	VSS	-	-
P11	-	-	VSS	VSS	-	-
P12	-	-	VSS	VSS	-	-
P13	-	-	VSS	VSS	-	-
P14	-	-	VSS	VSS	-	-
P15	-	-	VSS	VSS	-	-
P16	-	-	VSS	VSS	-	-
P17	-	-	VSS	VSS	-	-
R10	-	-	VSS	VSS	-	-
R11	-	-	VSS	VSS	-	-
R12	-	-	VSS	VSS	-	-
R13	-	-	VSS	VSS	-	-

Table 81. 1156 fpBGA Pin Table (Continued)

F1156 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	F1156 Pair
R14	-	-	VSS	VSS	-	-
R15	-	-	VSS	VSS	-	-
R16	-	-	VSS	VSS	-	-
R17	-	-	VSS	VSS	-	-
T10	-	-	VSS	VSS	-	-
T11	-	-	VSS	VSS	-	-
T12	-	-	VSS	VSS	-	-
T13	-	-	VSS	VSS	-	-
T14	-	-	VSS	VSS	-	-
T15	-	-	VSS	VSS	-	-
T16	-	-	VSS	VSS	-	-
T17	-	-	VSS	VSS	-	-
U10	-	-	VSS	VSS	-	-
U11	-	-	VSS	VSS	-	-
U12	-	-	VSS	VSS	-	-
U13	-	-	VSS	VSS	-	-
U14	-	-	VSS	VSS	-	-
U15	-	-	VSS	VSS	-	-
U16	-	-	VSS	VSS	-	-
U17	-	-	VSS	VSS	-	-
U18	-	-	VSS	VSS	-	-
V10	-	-	VSS	VSS	-	-
V11	-	-	VSS	VSS	-	-
V12	-	-	VSS	VSS	-	-
V13	-	-	VSS	VSS	-	-
V14	-	-	VSS	VSS	-	-
V15	-	-	VSS	VSS	-	-
V16	-	-	VSS	VSS	-	-
V17	-	-	VSS	VSS	-	-
V18	-	-	VSS	VSS	-	-
V19	-	-	VSS	VSS	-	-
V20	-	-	VSS	VSS	-	-
V21	-	-	VSS	VSS	-	-
V22	-	-	VSS	VSS	-	-
V23	-	-	VSS	VSS	-	-
W10	-	-	VSS	VSS	-	-
W11	-	-	VSS	VSS	-	-
W12	-	-	VSS	VSS	-	-
W13	-	-	VSS	VSS	-	-
W14	-	-	VSS	VSS	-	-
W15	-	-	VSS	VSS	-	-
W16	-	-	VSS	VSS	-	-
W17	-	-	VSS	VSS	-	-

Table 81. 1156 fpBGA Pin Table (Continued)

F1156 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	F1156 Pair
W18	-	-	VSS	VSS	-	-
W19	-	-	VSS	VSS	-	-
W20	-	-	VSS	VSS	-	-
W21	-	-	VSS	VSS	-	-
W22	-	-	VSS	VSS	-	-
W23	-	-	VSS	VSS	-	-
Y10	-	-	VSS	VSS	-	-
Y11	-	-	VSS	VSS	-	-
Y12	-	-	VSS	VSS	-	-
Y13	-	-	VSS	VSS	-	-
Y14	-	-	VSS	VSS	-	-
Y15	-	-	VSS	VSS	-	-
Y16	-	-	VSS	VSS	-	-
Y17	-	-	VSS	VSS	-	-
Y18	-	-	VSS	VSS	-	-
Y19	-	-	VSS	VSS	-	-
Y20	-	-	VSS	VSS	-	-
Y21	-	-	VSS	VSS	-	-
Y22	-	-	VSS	VSS	-	-
Y23	-	-	VSS	VSS	-	-
J22	-	-	VSS	VSS	-	-
J24	-	-	VSS	VSS	-	-
J26	-	-	VSS	VSS	-	-
K22	-	-	VSS	VSS	-	-
K24	-	-	VSS	VSS	-	-
L22	-	-	VSS	VSS	-	-
L24	-	-	VSS	VSS	-	-
L25	-	-	VSS	VSS	-	-
M22	-	-	VSS	VSS	-	-
M24	-	-	VSS	VSS	-	-
N22	-	-	VSS	VSS	-	-
P22	-	-	VSS	VSS	-	-
P23	-	-	VSS	VSS	-	-
R18	-	-	VSS	VSS	-	-
R19	-	-	VSS	VSS	-	-
R20	-	-	VSS	VSS	-	-
R21	-	-	VSS	VSS	-	-
R22	-	-	VSS	VSS	-	-
R23	-	-	VSS	VSS	-	-
T18	-	-	VSS	VSS	-	-
T19	-	-	VSS	VSS	-	-
T20	-	-	VSS	VSS	-	-
T21	-	-	VSS	VSS	-	-

Table 81. 1156 fpBGA Pin Table (Continued)

F1156 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	F1156 Pair
T22	-	-	VSS	VSS	-	-
T23	-	-	VSS	VSS	-	-
U19	-	-	VSS	VSS	-	-
U20	-	-	VSS	VSS	-	-
U21	-	-	VSS	VSS	-	-
U22	-	-	VSS	VSS	-	-
U23	-	-	VSS	VSS	-	-
Note: All differential pairs use adjacent balls.						

Table 82. 1036 fpSBGA Pin Table

FE1036 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	FE1036 Pair
J44	-	-	VDD33	VDD33_FPGAPLL	-	-
K42	-	-	O	PRD_DATA	RD_DATA/TDO	-
J43	-	-	I	PRESET_N	RESET_N	-
K43	-	-	I	PRD_CFG_N	RD_CFG_N	-
K44	-	-	I	PPRGRM_N	PRGRM_N	-
M42	0 (TL)	7	IO	PL2D	PLL_CK0C/HPPLL	L1C
M41	0 (TL)	7	IO	PL2C	PLL_CK0T/HPPLL	L1T
M40	0 (TL)	7	IO	PL3D	-	L3C
L38	0 (TL)	7	IO	PL3B	-	L4C
A1	-	-	VSS	VSS	-	-
M39	0 (TL)	7	IO	PL3C	VREF_0_07	L3T
L39	0 (TL)	7	IO	PL3A	-	L4T
N42	0 (TL)	7	IO	PL4D	D5	L5C
G30	0 (TL)	-	VDDIO0	VDDIO0	-	-
N41	0 (TL)	7	IO	PL4C	D6	L5T
N40	0 (TL)	8	IO	PL5D	-	L7C
L40	0 (TL)	8	IO	PL5B	-	L8C
A2	-	-	VSS	VSS	-	-
N39	0 (TL)	8	IO	PL5C	VREF_0_08	L7T
L41	0 (TL)	8	IO	PL5A	-	L8T
P44	0 (TL)	8	IO	PL6D	HDC	L9C
C42	-	-	VDD15	VDD15	-	-
P43	0 (TL)	8	IO	PL6C	LDC_N	L9T
P42	0 (TL)	8	IO	PL7D	-	L11C
L42	0 (TL)	8	IO	PL7B	-	L12C
G31	0 (TL)	-	VDDIO0	VDDIO0	-	-
P41	0 (TL)	8	IO	PL7C	-	L11T
L43	0 (TL)	8	IO	PL7A	-	L12T
R44	0 (TL)	9	IO	PL8D	TESTCFG	L13C
A11	-	-	VSS	VSS	-	-
R43	0 (TL)	9	IO	PL8C	D7	L13T
R42	0 (TL)	9	IO	PL9D	VREF_0_09	L15C
M43	0 (TL)	9	IO	PL9B	-	L16C
D42	-	-	VDD15	VDD15	-	-
R41	0 (TL)	9	IO	PL9C	A17/PPC_A31	L15T
M44	0 (TL)	9	IO	PL9A	-	L16T
T44	0 (TL)	9	IO	PL10D	CS0_N	L17C
G36	0 (TL)	-	VDDIO0	VDDIO0	-	-
T43	0 (TL)	9	IO	PL10C	CS1	L17T
T42	0 (TL)	10	IO	PL11D	-	L19C
N44	0 (TL)	10	IO	PL11B	-	L20C

Table 82. 1036 fpSBGA Pin Table (Continued)

FE1036 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	FE1036 Pair
A22	-	-	VSS	VSS	-	-
T41	0 (TL)	10	IO	PL11C	-	L19T
N43	0 (TL)	10	IO	PL11A	-	L20T
U44	0 (TL)	10	IO	PL12D	INIT_N	L21C
E40	-	-	VDD15	VDD15	-	-
U43	0 (TL)	10	IO	PL12C	DOUT	L21T
U42	0 (TL)	10	IO	PL13D	VREF_0_10	L23C
P40	0 (TL)	10	IO	PL13B	-	L24C
U41	0 (TL)	10	IO	PL13C	A16/PPC_A30	L23T
P39	0 (TL)	10	IO	PL13A	-	L24T
V44	7 (CL)	1	IO	PL14D	A15/PPC_A29	L25C
B2	-	-	VSS	VSS	-	-
V43	7 (CL)	1	IO	PL14C	A14/PPC_A28	L25T
G34	-	-	VSS	VSS	-	-
V42	7 (CL)	1	IO	PL15D	-	L27C
R40	7 (CL)	1	IO	PL15B	-	L28C
G35	-	-	VDD15	VDD15	-	-
V41	7 (CL)	1	IO	PL15C	-	L27T
R39	7 (CL)	1	IO	PL15A	-	L28T
W44	7 (CL)	1	IO	PL16D	VREF_7_01	L29C
T40	7 (CL)	1	IO	PL16B	-	L30C
W38	7 (CL)	-	VDDIO7	VDDIO7	-	-
W43	7 (CL)	1	IO	PL16C	D4	L29T
T39	7 (CL)	1	IO	PL16A	-	L30T
W42	7 (CL)	2	IO	PL17D	-	L31C
U40	7 (CL)	2	IO	PL17B	-	L32C
B43	-	-	VSS	VSS	-	-
W41	7 (CL)	2	IO	PL17C	-	L31T
U39	7 (CL)	2	IO	PL17A	-	L32T
Y44	7 (CL)	2	IO	PL18D	RDY/BUSY_N/RCLK	L33C
V40	7 (CL)	2	IO	PL18B	-	L34C
G37	-	-	VDD15	VDD15	-	-
Y43	7 (CL)	2	IO	PL18C	VREF_7_02	L33T
V39	7 (CL)	2	IO	PL18A	-	L34T
Y42	7 (CL)	2	IO	PL19D	A13/PPC_A27	L35C
W40	7 (CL)	2	IO	PL19B	-	L36C
Y38	7 (CL)	-	VDDIO7	VDDIO7	-	-
Y41	7 (CL)	2	IO	PL19C	A12/PPC_A26	L35T
W39	7 (CL)	2	IO	PL19A	-	L36T
AA44	7 (CL)	3	IO	PL20D	-	L37C
Y40	7 (CL)	3	IO	PL20B	-	L38C
B44	-	-	VSS	VSS	-	-

Table 82. 1036 fpSBGA Pin Table (Continued)

FE1036 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	FE1036 Pair
AA43	7 (CL)	3	IO	PL20C	-	L37T
Y39	7 (CL)	3	IO	PL20A	-	L38T
AA42	7 (CL)	3	IO	PL21D	A11/PPC_A25	L39C
AA40	7 (CL)	3	IO	PL21B	-	L40C
H38	-	-	VDD15	VDD15	-	-
AA41	7 (CL)	3	IO	PL21C	VREF_7_03	L39T
AA39	7 (CL)	3	IO	PL21A	-	L40T
AB44	7 (CL)	3	IO	PL22D	-	L41C
AB40	7 (CL)	3	IO	PL22B	-	L42C
AB38	7 (CL)	-	VDDIO7	VDDIO7	-	-
AB43	7 (CL)	3	IO	PL22C	-	L41T
AB39	7 (CL)	3	IO	PL22A	-	L42T
AB42	7 (CL)	4	IO	PL23D	RD_N/MPI_STRB_N	L43C
AC40	7 (CL)	4	IO	PL23B	-	L44C
G20	-	-	VSS	VSS	-	-
AB41	7 (CL)	4	IO	PL23C	VREF_7_04	L43T
AC39	7 (CL)	4	IO	PL23A	-	L44T
AC44	7 (CL)	4	IO	PL24D	PLCK0C	L45C
AD40	7 (CL)	4	IO	PL24B	-	L46C
AC43	7 (CL)	4	IO	PL24C	PLCK0T	L45T
AD39	7 (CL)	4	IO	PL24A	-	L46T
AC42	7 (CL)	5	IO	PL25D	A10/PPC_A24	L47C
AE40	7 (CL)	5	IO	PL25B	-	L48C
AC38	7 (CL)	-	VDDIO7	VDDIO7	-	-
AC41	7 (CL)	5	IO	PL25C	A9/PPC_A23	L47T
AE39	7 (CL)	5	IO	PL25A	-	L48T
AD44	7 (CL)	5	IO	PL26D	A8/PPC_A22	L49C
AF40	7 (CL)	5	IO	PL26B	-	L50C
G23	-	-	VSS	VSS	-	-
AD43	7 (CL)	5	IO	PL26C	VREF_7_05	L49T
AF39	7 (CL)	5	IO	PL26A	-	L50T
AD42	7 (CL)	5	IO	PL27D	-	L51C
AG44	7 (CL)	5	IO	PL27B	-	L52C
J38	-	-	VDD15	VDD15	-	-
AD41	7 (CL)	5	IO	PL27C	-	L51T
AG43	7 (CL)	5	IO	PL27A	-	L52T
AE44	7 (CL)	6	IO	PL28D	PLCK1C	L53C
AG42	7 (CL)	6	IO	PL28B	-	L54C
AE38	7 (CL)	-	VDDIO7	VDDIO7	-	-
AE43	7 (CL)	6	IO	PL28C	PLCK1T	L53T
AG41	7 (CL)	6	IO	PL28A	-	L54T
AE42	7 (CL)	6	IO	PL29D	VREF_7_06	L55C

Table 82. 1036 fpSBGA Pin Table (Continued)

FE1036 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	FE1036 Pair
AG40	7 (CL)	6	IO	PL29B	-	L56C
G26	-	-	VSS	VSS	-	-
AE41	7 (CL)	6	IO	PL29C	A7/PPC_A21	L55T
AG39	7 (CL)	6	IO	PL29A	-	L56T
AF44	7 (CL)	6	IO	PL30D	A6/PPC_A20	L57C
AH40	7 (CL)	6	IO	PL30B	-	L58C
N38	-	-	VDD15	VDD15	-	-
AF43	7 (CL)	6	IO	PL30C	A5/PPC_A19	L57T
AH39	7 (CL)	6	IO	PL30A	-	L58T
AF42	7 (CL)	7	IO	PL31D	-	L59C
AJ40	7 (CL)	7	IO	PL31B	-	L60C
AF38	7 (CL)	-	VDDIO7	VDDIO7	-	-
AF41	7 (CL)	7	IO	PL31C	-	L59T
AJ39	7 (CL)	7	IO	PL31A	-	L60T
AH44	7 (CL)	7	IO	PL32D	WR_N/MPI_RW	L61C
AK42	7 (CL)	7	IO	PL32B	-	L62C
G29	-	-	VSS	VSS	-	-
AH43	7 (CL)	7	IO	PL32C	VREF_7_07	L61T
AK41	7 (CL)	7	IO	PL32A	-	L62T
AH42	7 (CL)	7	IO	PL33D	-	L63C
AK40	7 (CL)	7	IO	PL33B	-	L64C
R7	-	-	VDD15	VDD15	-	-
AH41	7 (CL)	7	IO	PL33C	-	L63T
AK39	7 (CL)	7	IO	PL33A	-	L64T
AJ44	7 (CL)	8	IO	PL34D	A4/PPC_A18	L65C
AJ43	7 (CL)	8	IO	PL34C	VREF_7_08	L65T
AJ42	7 (CL)	8	IO	PL35D	A3/PPC_A17	L67C
AM44	7 (CL)	8	IO	PL35B	-	L68C
G32	-	-	VSS	VSS	-	-
AJ41	7 (CL)	8	IO	PL35C	A2/PPC_A16	L67T
AM43	7 (CL)	8	IO	PL35A	-	L68T
AK44	7 (CL)	8	IO	PL36D	A1/PPC_A15	L69C
AL40	7 (CL)	8	IO	PL36B	-	L70C
T38	-	-	VDD15	VDD15	-	-
AK43	7 (CL)	8	IO	PL36C	A0/PPC_A14	L69T
AL39	7 (CL)	8	IO	PL36A	-	L70T
AL44	7 (CL)	8	IO	PL37D	DP0	L71C
AN43	7 (CL)	8	IO	PL37B	-	L72C
AL43	7 (CL)	8	IO	PL37C	DP1	L71T
AP43	7 (CL)	8	IO	PL37A	-	L72T
AL42	6 (BL)	1	IO	PL38D	D8	L73C
AR43	6 (BL)	1	IO	PL38B	-	L74C

Table 82. 1036 fpSBGA Pin Table (Continued)

FE1036 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	FE1036 Pair
AB1	-	-	VSS	VSS	-	-
AL41	6 (BL)	1	IO	PL38C	VREF_6_01	L73T
AR42	6 (BL)	1	IO	PL38A	-	L74T
AM42	6 (BL)	1	IO	PL39D	D9	L75C
AN40	6 (BL)	1	IO	PL39B	-	L76C
U7	-	-	VDD15	VDD15	-	-
AM41	6 (BL)	1	IO	PL39C	D10	L75T
AN39	6 (BL)	1	IO	PL39A	-	L76T
AM40	6 (BL)	2	IO	PL40D	-	L77C
AP40	6 (BL)	2	IO	PL40B	-	L78C
AJ38	6 (BL)	-	VDDIO6	VDDIO6	-	-
AM39	6 (BL)	2	IO	PL40C	VREF_6_02	L77T
AP39	6 (BL)	2	IO	PL40A	-	L78T
AP44	6 (BL)	2	IO	PL41D	-	L79C
AN42	6 (BL)	2	IO	PL41B	-	L80C
AD7	-	-	VSS	VSS	-	-
AR44	6 (BL)	2	IO	PL41C	-	L79T
AN41	6 (BL)	2	IO	PL41A	-	L80T
AP42	6 (BL)	3	IO	PL42D	D11	L81C
AT44	6 (BL)	3	IO	PL42B	-	L82C
Y7	-	-	VDD15	VDD15	-	-
AP41	6 (BL)	3	IO	PL42C	D12	L81T
AT43	6 (BL)	3	IO	PL42A	-	L82T
AU44	6 (BL)	3	IO	PL43D	-	L83C
AV44	6 (BL)	3	IO	PL43B	-	L84C
AM38	6 (BL)	-	VDDIO6	VDDIO6	-	-
AU43	6 (BL)	3	IO	PL43C	-	L83T
AV43	6 (BL)	3	IO	PL43A	-	L84T
AU42	6 (BL)	3	IO	PL44D	VREF_6_03	L85C
AW44	6 (BL)	4	IO	PL44B	-	L86C
AD38	-	-	VSS	VSS	-	-
AT42	6 (BL)	3	IO	PL44C	D13	L85T
AY44	6 (BL)	4	IO	PL44A	-	L86T
AR41	6 (BL)	4	IO	PL45D	-	L87C
BB44	6 (BL)	4	IO	PL45B	-	L88C
AC7	-	-	VDD15	VDD15	-	-
AR40	6 (BL)	4	IO	PL45C	VREF_6_04	L87T
BA44	6 (BL)	4	IO	PL45A	-	L88T
AU41	6 (BL)	4	IO	PL46D	-	L89C
AR38	6 (BL)	-	VDDIO6	VDDIO6	-	-
AT41	6 (BL)	4	IO	PL46C	-	L89T
AT40	6 (BL)	4	IO	PL47D	PLL_CK7C/HPPLL	L91C

Table 82. 1036 fpSBGA Pin Table (Continued)

FE1036 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	FE1036 Pair
AG7	-	-	VSS	VSS	-	-
AT39	6 (BL)	4	IO	PL47C	PLL_CK7T/HPPLL	L91T
AG38	-	-	VSS	VSS	-	-
AR39	-	-	I	PTEMP	PTEMP	-
AV29	6 (BL)	-	VDDIO6	VDDIO6	-	-
AE7	-	-	VDD15	VDD15	-	-
BC33	-	-	IO	LVDS_R	LVDS_R	-
BA42	-	-	VDD33	VDD33_FPGAPLL	-	-
AK7	-	-	VSS	VSS	-	-
AV42	-	-	VDD33	VDD33_FPGAPLL	-	-
AH38	-	-	VDD15	VDD15	-	-
BB43	6 (BL)	5	IO	PB2A	DP2	L93T
BC42	6 (BL)	5	IO	PB2C	PLL_CK6T/PPLL	L94T
AV32	6 (BL)	-	VDDIO6	VDDIO6	-	-
BA43	6 (BL)	5	IO	PB2B	-	L93C
BD42	6 (BL)	5	IO	PB2D	PLL_CK6C/PPLL	L94C
BC41	6 (BL)	5	IO	PB3C	-	L96T
AK38	-	-	VSS	VSS	-	-
BD41	6 (BL)	5	IO	PB3D	-	L96C
BC40	6 (BL)	5	IO	PB4C	VREF_6_05	L98T
AJ7	-	-	VDD15	VDD15	-	-
BD40	6 (BL)	5	IO	PB4D	DP3	L98C
AW43	6 (BL)	6	IO	PB5A	-	L99T
BA37	6 (BL)	6	IO	PB5C	-	L100T
AV35	6 (BL)	-	VDDIO6	VDDIO6	-	-
AY43	6 (BL)	6	IO	PB5B	-	L99C
BB37	6 (BL)	6	IO	PB5D	-	L100C
AW42	6 (BL)	6	IO	PB6A	-	L101T
BC37	6 (BL)	6	IO	PB6C	VREF_6_06	L102T
AN1	-	-	VSS	VSS	-	-
AY42	6 (BL)	6	IO	PB6B	-	L101C
BD37	6 (BL)	6	IO	PB6D	D14	L102C
AW41	6 (BL)	6	IO	PB7A	-	L103T
BA36	6 (BL)	6	IO	PB7C	-	L104T
AL38	-	-	VDD15	VDD15	-	-
AV41	6 (BL)	6	IO	PB7B	-	L103C
BB36	6 (BL)	6	IO	PB7D	-	L104C
AY41	6 (BL)	7	IO	PB8A	-	L105T
BC36	6 (BL)	7	IO	PB8C	D15	L106T
BA41	6 (BL)	7	IO	PB8B	-	L105C
BD36	6 (BL)	7	IO	PB8D	D16	L106C
AU40	6 (BL)	7	IO	PB9A	-	L107T

Table 82. 1036 fpSBGA Pin Table (Continued)

FE1036 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	FE1036 Pair
BA35	6 (BL)	7	IO	PB9C	D17	L108T
AN7	-	-	VSS	VSS	-	-
AU39	6 (BL)	7	IO	PB9B	-	L107C
BB35	6 (BL)	7	IO	PB9D	D18	L108C
AW40	6 (BL)	7	IO	PB10A	-	L109T
BC35	6 (BL)	7	IO	PB10C	VREF_6_07	L110T
AV40	6 (BL)	7	IO	PB10B	-	L109C
BD35	6 (BL)	7	IO	PB10D	D19	L110C
BA34	6 (BL)	8	IO	PB11C	D20	L112T
BB34	6 (BL)	8	IO	PB11D	D21	L112C
BA40	6 (BL)	8	IO	PB12A	-	L113T
BC34	6 (BL)	8	IO	PB12C	VREF_6_08	L114T
BB40	6 (BL)	8	IO	PB12B	-	L113C
BD34	6 (BL)	8	IO	PB12D	D22	L114C
AV39	6 (BL)	9	IO	PB13A	-	L115T
BA33	6 (BL)	9	IO	PB13C	D23	L116T
AW39	6 (BL)	9	IO	PB13B	-	L115C
BB33	6 (BL)	9	IO	PB13D	D24	L116C
BB39	6 (BL)	9	IO	PB14A	-	L117T
BA32	6 (BL)	9	IO	PB14C	VREF_6_09	L118T
BA39	6 (BL)	9	IO	PB14B	-	L117C
BB32	6 (BL)	9	IO	PB14D	D25	L118C
BC39	6 (BL)	9	IO	PB15A	-	L119T
BC32	6 (BL)	9	IO	PB15C	-	L120T
BD39	6 (BL)	9	IO	PB15B	-	L119C
BD32	6 (BL)	9	IO	PB15D	-	L120C
BB38	6 (BL)	10	IO	PB16A	-	L121T
BA31	6 (BL)	10	IO	PB16C	D26	L122T
BA38	6 (BL)	10	IO	PB16B	-	L121C
BB31	6 (BL)	10	IO	PB16D	D27	L122C
AW38	6 (BL)	10	IO	PB17A	-	L123T
BC31	6 (BL)	10	IO	PB17C	-	L124T
AY38	6 (BL)	10	IO	PB17B	-	L123C
BD31	6 (BL)	10	IO	PB17D	-	L124C
BC38	6 (BL)	10	IO	PB18A	-	L125T
BA30	6 (BL)	10	IO	PB18C	VREF_6_10	L126T
BD38	6 (BL)	10	IO	PB18B	-	L125C
BB30	6 (BL)	10	IO	PB18D	D28	L126C
AW37	6 (BL)	11	IO	PB19A	-	L127T
BC30	6 (BL)	11	IO	PB19C	D29	L128T
AY37	6 (BL)	11	IO	PB19B	-	L127C
BD30	6 (BL)	11	IO	PB19D	D30	L128C

Table 82. 1036 fpSBGA Pin Table (Continued)

FE1036 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	FE1036 Pair
AW36	6 (BL)	11	IO	PB20A	-	L129T
BA29	6 (BL)	11	IO	PB20C	VREF_6_11	L130T
AY36	6 (BL)	11	IO	PB20B	-	L129C
BB29	6 (BL)	11	IO	PB20D	D31	L130C
AW35	5 (BC)	1	IO	PB21A	-	L131T
BC29	5 (BC)	1	IO	PB21C	-	L132T
AN38	-	-	VSS	VSS	-	-
AY35	5 (BC)	1	IO	PB21B	-	L131C
BD29	5 (BC)	1	IO	PB21D	-	L132C
AW34	5 (BC)	1	IO	PB22A	-	L133T
BA28	5 (BC)	1	IO	PB22C	VREF_5_01	L134T
AM7	-	-	VDD15	VDD15	-	-
AY34	5 (BC)	1	IO	PB22B	-	L133C
BB28	5 (BC)	1	IO	PB22D	-	L134C
AW33	5 (BC)	2	IO	PB23A	-	L135T
BC28	5 (BC)	2	IO	PB23C	PBCK0T	L136T
AV17	5 (BC)	-	VDDIO5	VDDIO5	-	-
BD28	5 (BC)	2	IO	PB23D	PBCK0C	L136C
BA27	5 (BC)	2	IO	PB24C	VREF_5_02	L138T
AY33	5 (BC)	2	IO	PB24B	-	L137C
BB27	5 (BC)	2	IO	PB24D	-	L138C
AW32	5 (BC)	2	IO	PB25A	-	L139T
BC27	5 (BC)	2	IO	PB25C	-	L140T
AP38	-	-	VDD15	VDD15	-	-
AY32	5 (BC)	2	IO	PB25B	-	L139C
BD27	5 (BC)	2	IO	PB25D	-	L140C
AW31	5 (BC)	3	IO	PB26A	-	L141T
BA26	5 (BC)	3	IO	PB26C	-	L142T
AV20	5 (BC)	-	VDDIO5	VDDIO5	-	-
AY31	5 (BC)	3	IO	PB26B	-	L141C
BB26	5 (BC)	3	IO	PB26D	VREF_5_03	L142C
AW30	5 (BC)	3	IO	PB27A	-	L143T
BC26	5 (BC)	3	IO	PB27C	-	L144T
AN44	-	-	VSS	VSS	-	-
AY30	5 (BC)	3	IO	PB27B	-	L143C
BD26	5 (BC)	3	IO	PB27D	-	L144C
AW29	5 (BC)	3	IO	PB28A	-	L145T
BA25	5 (BC)	3	IO	PB28C	PBCK1T	L146T
AR7	-	-	VDD15	VDD15	-	-
AY29	5 (BC)	3	IO	PB28B	-	L145C
BB25	5 (BC)	3	IO	PB28D	PBCK1C	L146C
AW28	5 (BC)	3	IO	PB29A	-	L147T

Table 82. 1036 fpSBGA Pin Table (Continued)

FE1036 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	FE1036 Pair
BC25	5 (BC)	4	IO	PB29C	-	L148T
AV21	5 (BC)	-	VDDIO5	VDDIO5	-	-
AY28	5 (BC)	3	IO	PB29B	-	L147C
BD25	5 (BC)	4	IO	PB29D	-	L148C
AW27	5 (BC)	4	IO	PB30A	-	L149T
BA24	5 (BC)	4	IO	PB30C	-	L150T
AV7	-	-	VSS	VSS	-	-
AY27	5 (BC)	4	IO	PB30B	-	L149C
BB24	5 (BC)	4	IO	PB30D	VREF_5_04	L150C
AW26	5 (BC)	4	IO	PB31A	-	L151T
BC24	5 (BC)	4	IO	PB31C	-	L152T
AT7	-	-	VDD15	VDD15	-	-
AY26	5 (BC)	4	IO	PB31B	-	L151C
BD24	5 (BC)	4	IO	PB31D	-	L152C
AW25	5 (BC)	5	IO	PB32A	-	L153T
BA23	5 (BC)	5	IO	PB32C	-	L154T
AV23	5 (BC)	-	VDDIO5	VDDIO5	-	-
AY25	5 (BC)	5	IO	PB32B	-	L153C
BB23	5 (BC)	5	IO	PB32D	VREF_5_05	L154C
AW23	5 (BC)	5	IO	PB33A	-	L155T
BC23	5 (BC)	5	IO	PB33C	-	L156T
AV18	-	-	VSS	VSS	-	-
AY23	5 (BC)	5	IO	PB33B	-	L155C
BD23	5 (BC)	5	IO	PB33D	-	L156C
AW24	5 (BC)	5	IO	PB34A	-	L157T
BA22	5 (BC)	5	IO	PB34C	-	L158T
AT38	-	-	VDD15	VDD15	-	-
AY24	5 (BC)	5	IO	PB34B	-	L157C
BB22	5 (BC)	5	IO	PB34D	-	L158C
AW22	5 (BC)	6	IO	PB35A	-	L159T
BD21	5 (BC)	6	IO	PB35C	-	L160T
AV25	5 (BC)	-	VDDIO5	VDDIO5	-	-
AY22	5 (BC)	6	IO	PB35B	-	L159C
BC21	5 (BC)	6	IO	PB35D	VREF_5_06	L160C
BD20	5 (BC)	6	IO	PB36C	-	L162T
AV22	-	-	VSS	VSS	-	-
BC20	5 (BC)	6	IO	PB36D	-	L162C
AY21	5 (BC)	7	IO	PB37A	-	L163T
BB20	5 (BC)	7	IO	PB37C	-	L164T
AU7	-	-	VDD15	VDD15	-	-
AW21	5 (BC)	7	IO	PB37B	-	L163C
BA20	5 (BC)	7	IO	PB37D	-	L164C

Table 82. 1036 fpSBGA Pin Table (Continued)

FE1036 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	FE1036 Pair
BA21	5 (BC)	7	IO	PB38A	-	L165T
BD19	5 (BC)	7	IO	PB38C	VREF_5_07	L166T
AV26	5 (BC)	-	VDDIO5	VDDIO5	-	-
BB21	5 (BC)	7	IO	PB38B	-	L165C
BC19	5 (BC)	7	IO	PB38D	-	L166C
AY20	5 (BC)	7	IO	PB39A	-	L167T
BB19	5 (BC)	7	IO	PB39C	-	L168T
AV24	-	-	VSS	VSS	-	-
AW20	5 (BC)	7	IO	PB39B	-	L167C
BA19	5 (BC)	7	IO	PB39D	-	L168C
AY19	5 (BC)	8	IO	PB40A	-	L169T
BD18	5 (BC)	8	IO	PB40C	-	L170T
AW19	5 (BC)	8	IO	PB40B	-	L169C
BC18	5 (BC)	8	IO	PB40D	VREF_5_08	L170C
AY18	5 (BC)	8	IO	PB41A	-	L171T
BB18	5 (BC)	8	IO	PB41C	-	L172T
AW18	5 (BC)	8	IO	PB41B	-	L171C
BA18	5 (BC)	8	IO	PB41D	-	L172C
AY17	5 (BC)	8	IO	PB42A	-	L173T
BD17	5 (BC)	8	IO	PB42C	-	L174T
AV27	-	-	VSS	VSS	-	-
AW17	5 (BC)	8	IO	PB42B	-	L173C
BC17	5 (BC)	8	IO	PB42D	-	L174C
AY16	5 (BC)	9	IO	PB43A	-	L175T
BB17	5 (BC)	9	IO	PB43C	-	L176T
AW16	5 (BC)	9	IO	PB43B	-	L175C
BA17	5 (BC)	9	IO	PB43D	-	L176C
BD13	5 (BC)	9	IO	PB44A	-	L177T
BD16	5 (BC)	9	IO	PB44C	-	L178T
BC13	5 (BC)	9	IO	PB44B	-	L177C
BC16	5 (BC)	9	IO	PB44D	VREF_5_09	L178C
BB16	5 (BC)	9	IO	PB45C	-	L180T
AV30	-	-	VSS	VSS	-	-
BA16	5 (BC)	9	IO	PB45D	-	L180C
BD12	5 (BC)	10	IO	PB46A	-	L181T
BD14	5 (BC)	10	IO	PB46C	-	L182T
BC12	5 (BC)	10	IO	PB46B	-	L181C
BD15	5 (BC)	10	IO	PB46D	VREF_5_10	L182C
BC15	5 (BC)	10	IO	PB47C	PLL_CK5T/PPLL	L184T
BC14	5 (BC)	10	IO	PB47D	PLL_CK5C/PPLL	L184C
BC11	-	-	VDD33	VDD33_FPGAPLL	-	-
AV33	-	-	VSS	VSS	-	-

Table 82. 1036 fpSBGA Pin Table (Continued)

FE1036 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	FE1036 Pair
BD7	-	-	VDD33	VDD33_FPGAPLL	-	-
BD8	-	-	I	SPARE_1	-	-
BB1	-	-	I	RESETN	-	-
BB2	-	-	I	TRISTN	-	-
AV15	-	-	I	TESTMD1N	-	-
BD6	-	-	VDDGB	VDDGB	-	-
BC8	-	-	I	TESTMD0N	-	-
AV38	-	-	VSS	VSS	-	-
BC7	-	-	I	PDN	-	-
BD5	-	-	I	ATREFCLK	-	-
BD4	-	-	I	TESTCLK	-	-
BD3	-	-	I	BTREFCLK	-	-
BC3	-	-	I	TSTAT1B	-	-
AV13	-	-	I	TSTAT0B	-	-
BB7	-	-	I	TSCLKB	-	-
BC6	-	-	O	RSTAT1B	-	-
BC4	-	-	O	RSTAT0B	-	-
BC5	-	-	O	RSCLKB	-	-
AW8	-	-	VSS	VSS	-	-
AW15	-	-	VDDOB	VDDOB_D	-	-
AY15	-	-	O	HDOUTP_D	-	HSP_1
BA15	-	-	O	HDOUTN_D	-	HSN_1
BB15	-	-	VDDOB	VDDOB_D	-	-
AY14	-	-	I	HDINP_D	-	HSP_2
AW10	-	-	VSS	VSS	-	-
BA14	-	-	I	HDINN_D	-	HSN_2
AV10	-	-	VDD_ANA	VDD_ANA	-	-
BB14	-	-	VDDIB	VDDIB_D	-	-
AW12	-	-	VSS	VSS	-	-
AW13	-	-	VDDOB	VDDOB_C	-	-
AY13	-	-	O	HDOUTP_C	-	HSP_3
BA13	-	-	O	HDOUTN_C	-	HSN_3
BB13	-	-	VDDOB	VDDOB_C	-	-
AY12	-	-	I	HDINP_C	-	HSP_4
AW14	-	-	VSS	VSS	-	-
BA12	-	-	I	HDINN_C	-	HSN_4
AV12	-	-	VDD_ANA	VDD_ANA	-	-
BB12	-	-	VDDIB	VDDIB_C	-	-
BC9	-	-	VSS	VSS	-	-
AW11	-	-	VDDOB	VDDOB_B	-	-
AY11	-	-	O	HDOUTP_B	-	HSP_5
BA11	-	-	O	HDOUTN_B	-	HSN_5

Table 82. 1036 fpSBGA Pin Table (Continued)

FE1036 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	FE1036 Pair
BB11	-	-	VDDOB	VDDOB_B	-	-
AV14	-	-	VDD_ANA	VDD_ANA	-	-
AY10	-	-	I	HDINP_B	-	HSP_6
BC10	-	-	VSS	VSS	-	-
BA10	-	-	I	HDINN_B	-	HSN_6
BB10	-	-	VDDIB	VDDIB_B	-	-
BD9	-	-	VSS	VSS	-	-
AW9	-	-	VDDOB	VDDOB_A	-	-
AY9	-	-	O	HDOUTP_A	-	HSP_7
BA9	-	-	O	HDOUTN_A	-	HSN_7
BB9	-	-	VDDOB	VDDOB_A	-	-
AV8	-	-	VDD_ANA	VDD_ANA	-	-
AY8	-	-	I	HDINP_A	-	HSP_8
BD10	-	-	VSS	VSS	-	-
BA8	-	-	I	HDINN_A	-	HSN_8
BB8	-	-	VDDIB	VDDIB_A	-	-
AW7	-	-	I	REFCLKP	-	HSP_9
AY7	-	-	I	REFCLKN	-	HSN_9
BA7	-	-	O	REXTN	-	-
AV11	-	-	O	REXT	-	-
AW4	-	-	I	ATSTAT1N	-	R1C
AU6	-	-	I	ATSTAT0N	-	R2C
AW3	-	-	I	ATSTAT1P	-	R1T
AV6	-	-	I	ATSTAT0P	-	R2T
AV3	-	-	I	ATSCLKN	-	R3C
T6	-	-	VDD33	VDD33	-	-
AV9	-	-	I/O	ALVCTAP5	-	-
T7	-	-	VDD33	VDD33	-	-
AV2	-	-	I	ATSCLKP	-	R3T
AY2	-	-	I	BTSCLKN	-	R4C
AR6	-	-	I/O	BLVCTAP5	-	-
AW2	-	-	I	BTSCLKP	-	R4T
BC1	-	-	VSS	VSS	-	-
AT4	-	-	I	BTSTAT1N	-	R5C
BC2	-	-	VSS	VSS	-	-
BB5	-	-	I	BTSTAT0N	-	R6C
AT3	-	-	I	BTSTAT1P	-	R5T
BB6	-	-	I	BTSTAT0P	-	R6T
AU38	-	-	VDD15	VDD15	-	-
AV16	-	-	VDD15	VDD15	-	-
AR5	-	-	O	BRSTAT1N	-	R7C
BC22	-	-	VSS	VSS	-	-

Table 82. 1036 fpSBGA Pin Table (Continued)

FE1036 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	FE1036 Pair
BA1	-	-	O	BRSTAT0N	-	R8C
AR4	-	-	O	BRSTAT1P	-	R7T
AY1	-	-	O	BRSTAT0P	-	R8T
AP5	-	-	O	BRSCLKN	-	R9C
W6	-	-	VDD33	VDD33	-	-
AW1	-	-	O	ARSCLKN	-	R10C
W7	-	-	VDD33	VDD33	-	-
AP4	-	-	O	BRSCLKP	-	R9T
AV1	-	-	O	ARSCLKP	-	R10T
BC43	-	-	VSS	VSS	-	-
AN6	-	-	O	ARSTAT1N	-	R11C
BA5	-	-	O	ARSTAT0N	-	R12C
BC44	-	-	VSS	VSS	-	-
AN5	-	-	O	ARSTAT1P	-	R11T
BA6	-	-	O	ARSTAT0P	-	R12T
AV19	-	-	VDD15	VDD15	-	-
AV28	-	-	VDD15	VDD15	-	-
AU1	-	-	I	RESLO	-	-
AP1	-	-	I	RESHI	-	-
BD1	-	-	VSS	VSS	-	-
AR1	-	-	I	REF14	-	-
BA2	-	-	I	REF10	-	-
AB6	-	-	VDD33	VDD33	-	-
AB7	-	-	VDD33	VDD33	-	-
AM6	-	-	O	BTDAT15N	-	R13C
BD2	-	-	VSS	VSS	-	-
BA4	-	-	O	BTDAT14N	-	R14C
AM5	-	-	O	BTDAT15P	-	R13T
BA3	-	-	O	BTDAT14P	-	R14T
AM4	-	-	O	BTDAT13N	-	R15C
AV31	-	-	VDD15	VDD15	-	-
AY4	-	-	O	BTDAT12N	-	R16C
AM3	-	-	O	BTDAT13P	-	R15T
AV34	-	-	VDD15	VDD15	-	-
AY3	-	-	O	BTDAT12P	-	R16T
AL5	-	-	O	BTDAT11N	-	R17C
BD11	-	-	VSS	VSS	-	-
AW5	-	-	O	BTDAT10N	-	R18C
AL4	-	-	O	BTDAT11P	-	R17T
AW6	-	-	O	BTDAT10P	-	R18T
AF6	-	-	VDD33	VDD33	-	-
AL3	-	-	O	BTDAT9N	-	R19C

Table 82. 1036 fpSBGA Pin Table (Continued)

FE1036 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	FE1036 Pair
AF7	-	-	VDD33	VDD33	-	-
AV5	-	-	O	BTDAT8N	-	R20C
AL2	-	-	O	BTDAT9P	-	R19T
AV4	-	-	O	BTDAT8P	-	R20T
BD22	-	-	VSS	VSS	-	-
AK6	-	-	O	BTCTLN	-	R21C
BD33	-	-	VSS	VSS	-	-
AU5	-	-	O	BTDCLKN	-	R22C
AK5	-	-	O	BTCTLP	-	R21T
AU4	-	-	O	BTDCLKP	-	R22T
AV36	-	-	VDD15	VDD15	-	-
AK4	-	-	VDDA_SPIA	VDDA_SPIA	-	-
AV37	-	-	VDD15	VDD15	-	-
AK3	-	-	VSS	VSS	-	-
AU3	-	-	O	BTDAT7N	-	R23C
AJ6	-	-	O	BTDAT6N	-	R24C
BD43	-	-	VSS	VSS	-	-
AU2	-	-	O	BTDAT7P	-	R23T
AJ5	-	-	O	BTDAT6P	-	R24T
AJ4	-	-	O	BTDAT5N	-	R25C
AH6	-	-	VDD33	VDD33	-	-
AT6	-	-	O	BTDAT4N	-	R26C
AJ3	-	-	O	BTDAT5P	-	R25T
AH7	-	-	VDD33	VDD33	-	-
AT5	-	-	O	BTDAT4P	-	R26T
AH5	-	-	O	BTDAT3N	-	R27C
AT2	-	-	O	BTDAT2N	-	R28C
BD44	-	-	VSS	VSS	-	-
AH4	-	-	O	BTDAT3P	-	R27T
AT1	-	-	O	BTDAT2P	-	R28T
AH3	-	-	O	BTDAT1N	-	R29C
AY5	-	-	VDD15	VDD15	-	-
AR3	-	-	O	BTDAT0N	-	R30C
AH2	-	-	O	BTDAT1P	-	R29T
AY6	-	-	VDD15	VDD15	-	-
AP3	-	-	O	BTDAT0P	-	R31T
AG6	-	-	I	BRDCLKN	-	R32C
AR2	-	-	I	BRDAT15N	-	R33C
AG5	-	-	I	BRDCLKP	-	R32T
AP2	-	-	I	BRDAT15P	-	R33T
AF1	-	-	I/O	BLVCTAP1	-	-
AL6	-	-	VDD33	VDD33	-	-

Table 82. 1036 fpSBGA Pin Table (Continued)

FE1036 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	FE1036 Pair
AN4	-	-	I	BRDAT14N	-	R34C
AL7	-	-	VDD33	VDD33	-	-
AG4	-	-	I	BRDAT13N	-	R35C
AN3	-	-	I	BRDAT14P	-	R34T
AG3	-	-	I	BRDAT13P	-	R35T
AM2	-	-	I	BRDAT12N	-	R36C
AF3	-	-	I	BRDAT11N	-	R37C
AM1	-	-	I	BRDAT12P	-	R36T
AF2	-	-	I	BRDAT11P	-	R37T
AL1	-	-	I	BRDAT10N	-	R38C
AF4	-	-	I/O	BLVCTAP2	-	-
AK1	-	-	I	BRDAT10P	-	R38T
AE6	-	-	I	BRDAT9N	-	R39C
AK2	-	-	I	BRDAT8N	-	R40C
AE5	-	-	I	BRDAT9P	-	R39T
AJ2	-	-	I	BRDAT8P	-	R40T
AE4	-	-	I	BRCTLN	-	R41C
AN2	-	-	I/O	BLVCTAP3	-	-
AE3	-	-	I	BRCTLP	-	R41T
AP6	-	-	VDD33	VDD33	-	-
AJ1	-	-	I	BRDAT7N	-	R42C
AP7	-	-	VDD33	VDD33	-	-
AD6	-	-	I	BRDAT6N	-	R43C
AH1	-	-	I	BRDAT7P	-	R42T
AD5	-	-	I	BRDAT6P	-	R43T
AG2	-	-	I	BRDAT5N	-	R44C
AD4	-	-	I	BRDAT4N	-	R45C
AG1	-	-	I	BRDAT5P	-	R44T
AD3	-	-	I	BRDAT4P	-	R45T
AF5	-	-	I/O	BLVCTAP4	-	-
AC6	-	-	I	BRDAT3N	-	R46C
AE2	-	-	I	BRDAT2N	-	R47C
AC5	-	-	I	BRDAT3P	-	R46T
AE1	-	-	I	BRDAT2P	-	R47T
AC4	-	-	I	BRDAT1N	-	R48C
AD2	-	-	I	BRDAT0N	-	R49C
AC3	-	-	I	BRDAT1P	-	R48T
AD1	-	-	I	BRDAT0P	-	R49T
W1	-	-	VSS	VSS	-	-
V1	-	-	VDDA_SPIB	VDDA_SPIB	-	-
AC1	-	-	O	ATDAT15N	-	R50C
U1	-	-	VSS	VSS	-	-

Table 82. 1036 fpSBGA Pin Table (Continued)

FE1036 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	FE1036 Pair
AC2	-	-	O	ATDAT15P	-	R50T
AA1	-	-	O	ATDAT14N	-	R51C
AA2	-	-	O	ATDAT14P	-	R51T
AB2	-	-	O	ATDAT13N	-	R52C
AA3	-	-	O	ATDAT12N	-	R53C
AB3	-	-	O	ATDAT13P	-	R52T
AA4	-	-	O	ATDAT12P	-	R53T
Y1	-	-	O	ATDAT11N	-	R54C
V2	-	-	O	ATDAT10N	-	R55C
Y2	-	-	O	ATDAT11P	-	R54T
U2	-	-	O	ATDAT10P	-	R55T
Y3	-	-	O	ATDAT9N	-	R56C
AB5	-	-	O	ATDAT8N	-	R57C
Y4	-	-	O	ATDAT9P	-	R56T
AB4	-	-	O	ATDAT8P	-	R57T
W2	-	-	O	ATCTLN	-	R58C
AA5	-	-	O	ATDCLKN	-	R59C
W3	-	-	O	ATCTLP	-	R58T
Y5	-	-	O	ATDCLKP	-	R59T
W4	-	-	O	ATDAT7N	-	R60C
AA6	-	-	O	ATDAT6N	-	R61C
W5	-	-	O	ATDAT7P	-	R60T
Y6	-	-	O	ATDAT6P	-	R61T
T1	-	-	O	ATDAT5N	-	R62C
V3	-	-	O	ATDAT4N	-	R63C
R1	-	-	O	ATDAT5P	-	R62T
U3	-	-	O	ATDAT4P	-	R63T
T2	-	-	O	ATDAT3N	-	R64C
V4	-	-	O	ATDAT2N	-	R65C
R2	-	-	O	ATDAT3P	-	R64T
U4	-	-	O	ATDAT2P	-	R65T
T3	-	-	O	ATDAT1N	-	R66C
R4	-	-	O	ATDAT0N	-	R67C
R3	-	-	O	ATDAT1P	-	R66T
R5	-	-	O	ATDAT0P	-	R67T
T4	-	-	VSS	VSS	-	-
U5	-	-	VDDA_SPIC	VDDA_SPIC	-	-
U6	-	-	VSS	VSS	-	-
P1	-	-	I	ARDCLKN	-	R68C
V5	-	-	I	ARDAT15N	-	R69C
P2	-	-	I	ARDCLKP	-	R68T
V6	-	-	I	ARDAT15P	-	R69T

Table 82. 1036 fpSBGA Pin Table (Continued)

FE1036 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	FE1036 Pair
T5	-	-	I/O	ALVCTAP1	-	-
M3	-	-	I	ARDAT14N	-	R70C
P3	-	-	I	ARDAT13N	-	R71C
M4	-	-	I	ARDAT14P	-	R70T
P4	-	-	I	ARDAT13P	-	R71T
P5	-	-	I	ARDAT12N	-	R72C
N1	-	-	I	ARDAT11N	-	R73C
P6	-	-	I	ARDAT12P	-	R72T
N2	-	-	I	ARDAT11P	-	R73T
R6	-	-	I/O	ALVCTAP2	-	-
N3	-	-	I	ARDAT10N	-	R74C
N5	-	-	I	ARDAT9N	-	R75C
N4	-	-	I	ARDAT10P	-	R74T
N6	-	-	I	ARDAT9P	-	R75T
M1	-	-	I	ARDAT8N	-	R76C
L2	-	-	I	ARCTLN	-	R77C
M2	-	-	I	ARDAT8P	-	R76T
L3	-	-	I	ARCTLP	-	R77T
G1	-	-	I/O	ALVCTAP3	-	-
M6	-	-	I	ARDAT7N	-	R78C
K1	-	-	I	ARDAT6N	-	R79C
M5	-	-	I	ARDAT7P	-	R78T
K2	-	-	I	ARDAT6P	-	R79T
K3	-	-	I	ARDAT5N	-	R80C
J1	-	-	I	ARDAT4N	-	R81C
J3	-	-	I	ARDAT5P	-	R80T
J2	-	-	I	ARDAT4P	-	R81T
H3	-	-	I/O	ALVCTAP4	-	-
H1	-	-	I	ARDAT3N	-	R82C
L5	-	-	I	ARDAT2N	-	R83C
H2	-	-	I	ARDAT3P	-	R82T
L4	-	-	I	ARDAT2P	-	R83T
G2	-	-	I	ARDAT1N	-	R84C
K4	-	-	I	ARDAT0N	-	R85C
G3	-	-	I	ARDAT1P	-	R84T
J4	-	-	I	ARDAT0P	-	R85T
L6	-	-	VDDA_SPID	VDDA_SPID	-	-
K5	-	-	VSS	VSS	-	-
F1	-	-	I	TSTAT1A	-	-
L7	-	-	I	TSTAT0A	-	-
E1	-	-	I	TSCLKA	-	-
D4	-	-	VSS	VSS	-	-

Table 82. 1036 fpSBGA Pin Table (Continued)

FE1036 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	FE1036 Pair
H4	-	-	O	RSTAT1A	-	-
D1	-	-	O	RSTAT0A	-	-
K6	-	-	O	RSCLKA	-	-
F2	-	-	I/O	PMIA17	-	-
J5	-	-	I/O	PMIA16	-	-
C4	-	-	I/O	PMIA15	-	-
G4	-	-	I/O	PMIA14	-	-
E2	-	-	I/O	PMIA13	-	-
E5	-	-	VSS	VSS	-	-
G5	-	-	I/O	PMIA12	-	-
F3	-	-	I/O	PMIA11	-	-
G9	-	-	VSS	VSS	-	-
J6	-	-	I/O	PMIA10	-	-
F4	-	-	I/O	PMIA9	-	-
H5	-	-	I/O	PMIA8	-	-
F6	-	-	VDDH	VDDH	-	-
E3	-	-	I/O	PMIWN	-	-
D2	-	-	I/O	PMIRN	-	-
E4	-	-	I/O	PMIA7	-	-
D3	-	-	VREF	REFI_1	-	-
G13	-	-	VSS	VSS	-	-
G6	-	-	I/O	PMIA6	-	-
G15	-	-	VSS	VSS	-	-
H6	-	-	I/O	PMIA5	-	-
H7	-	-	I/O	PMIA4	-	-
C1	-	-	I/O	PMIA3	-	-
G7	-	-	VDDH	VDDH	-	-
C3	-	-	I/O	PMIA2	-	-
F5	-	-	I/O	PMIA1	-	-
E6	-	-	I/O	PMIA0	-	-
K7	-	-	VSS	VSS	-	-
C2	-	-	I	EXT_1K	-	-
G10	-	-	VDDH	VDDH	-	-
M7	-	-	VSS	VSS	-	-
P7	-	-	VSS	VSS	-	-
D7	-	-	I/O	PMID35	-	-
D5	-	-	I/O	PMID34	-	-
G12	-	-	VDDH	VDDH	-	-
C5	-	-	I/O	PMID33	-	-
D6	-	-	I/O	PMID32	-	-
G8	-	-	I/O	PMID31	-	-
F7	-	-	I/O	PMID30	-	-

Table 82. 1036 fpSBGA Pin Table (Continued)

FE1036 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	FE1036 Pair
C6	-	-	I/O	PMID29	-	-
E7	-	-	I/O	PMID28	-	-
G14	-	-	VDDH	VDDH	-	-
D8	-	-	I/O	PMID27	-	-
B3	-	-	I/O	PMID26	-	-
E8	-	-	I/O	PMID25	-	-
B4	-	-	I/O	PMID24	-	-
B9	-	-	I/O	PMID23	-	-
A3	-	-	I/O	PMID22	-	-
C7	-	-	I/O	PMID21	-	-
A4	-	-	I/O	PMID20	-	-
G16	-	-	VDDH	VDDH	-	-
C8	-	-	I/O	PMID19	-	-
A9	-	-	I/O	PMID18	-	-
B5	-	-	I/O	PMID17	-	-
C10	-	-	I/O	PMID16	-	-
D10	-	-	VREF	REFI_2	-	-
F8	-	-	I/O	PMIK	-	-
B10	-	-	I/O	PMIKN	-	-
D9	-	-	I/O	PMID15	-	-
A10	-	-	I/O	PMID14	-	-
E9	-	-	I/O	PMID13	-	-
B11	-	-	I/O	PMID12	-	-
C9	-	-	I/O	PMID11	-	-
C11	-	-	I/O	PMID10	-	-
J7	-	-	VDDH	VDDH	-	-
F9	-	-	I/O	PMID9	-	-
D11	-	-	I/O	PMID8	-	-
B6	-	-	I/O	PMID7	-	-
E11	-	-	I/O	PMID6	-	-
B7	-	-	I/O	PMID5	-	-
C12	-	-	I/O	PMID4	-	-
N7	-	-	VDDH	VDDH	-	-
A5	-	-	I/O	PMID3	-	-
D12	-	-	I/O	PMID2	-	-
B8	-	-	I/O	PMID1	-	-
B12	-	-	I/O	PMID0	-	-
A12	-	-	VREF	REFI_3	-	-
C13	-	-	I/O	PMIC	-	-
D13	-	-	I/O	PMICN	-	-
A7	-	-	I/O	PMIQ35	-	-
B13	-	-	I/O	PMIQ34	-	-

Table 82. 1036 fpSBGA Pin Table (Continued)

FE1036 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	FE1036 Pair
A8	-	-	I/O	PMIQ33	-	-
A13	-	-	I/O	PMIQ32	-	-
A6	-	-	I/O	PMIQ31	-	-
C14	-	-	I/O	PMIQ30	-	-
F11	-	-	I/O	PMIQ29	-	-
D14	-	-	I/O	PMIQ28	-	-
G11	-	-	I/O	PMIQ27	-	-
B14	-	-	I/O	PMIQ26	-	-
F10	-	-	I/O	PMIQ25	-	-
A14	-	-	I/O	PMIQ24	-	-
E10	-	-	I/O	PMIQ23	-	-
C15	-	-	I/O	PMIQ22	-	-
F12	-	-	I/O	PMIQ21	-	-
D15	-	-	I/O	PMIQ20	-	-
E12	-	-	I/O	PMIQ19	-	-
B15	-	-	I/O	PMIQ18	-	-
F13	-	-	I/O	PMIQ17	-	-
A15	-	-	I/O	PMIQ16	-	-
E13	-	-	I/O	MCREFCLK	-	-
C16	-	-	I/O	SPARE_2	-	-
F14	-	-	I/O	PMIQ15	-	-
D16	-	-	VREF	REFI_4	-	-
E14	-	-	I/O	PMIQ14	-	-
B16	-	-	I/O	PMIQ13	-	-
F15	-	-	I/O	PMIQ12	-	-
A16	-	-	VDDA_PLL	VDDA_PLL	-	-
E15	-	-	I/O	PMIQ11	-	-
C17	-	-	VSS	VSS	-	-
F16	-	-	I/O	PMIQ10	-	-
D17	-	-	I/O	PMIQ9	-	-
E16	-	-	I/O	PMIQ8	-	-
B17	-	-	I/O	PMIQ7	-	-
F17	-	-	I/O	PMIQ6	-	-
A17	-	-	I/O	PMIQ5	-	-
E17	-	-	I/O	PMIQ4	-	-
A18	-	-	I/O	PMIQ3	-	-
D18	-	-	I/O	PMIQ2	-	-
B18	-	-	I/O	PMIQ1	-	-
C18	-	-	I/O	PMIQ0	-	-
A19	-	-	VDD33	VDD33_FPGAPLL	-	-
B19	-	-	VDD33	VDD33_FPGAPLL	-	-
E18	1 (TC)	7	IO	PT47D	PLL_CK2C/PPLL	L185C

Table 82. 1036 fpSBGA Pin Table (Continued)

FE1036 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	FE1036 Pair
G17	1 (TC)	-	VDDIO1	VDDIO1	-	-
F18	1 (TC)	7	IO	PT47C	PLL_CK2T/PPLL	L185T
C19	1 (TC)	7	IO	PT46D	-	L187C
AY39	-	-	VDD15	VDD15	-	-
E21	1 (TC)	7	IO	PT46B	-	L188C
D19	1 (TC)	7	IO	PT46C	-	L187T
F21	1 (TC)	7	IO	PT46A	-	L188T
E19	1 (TC)	7	IO	PT45D	VREF_1_07	L189C
L1	-	-	VSS	VSS	-	-
F19	1 (TC)	7	IO	PT45C	-	L189T
C20	1 (TC)	8	IO	PT44D	-	L191C
G18	1 (TC)	-	VDDIO1	VDDIO1	-	-
E23	1 (TC)	8	IO	PT44B	-	L192C
D20	1 (TC)	8	IO	PT44C	-	L191T
F23	1 (TC)	8	IO	PT44A	-	L192T
E20	1 (TC)	8	IO	PT43D	-	L193C
AY40	-	-	VDD15	VDD15	-	-
E24	1 (TC)	8	IO	PT43B	-	L194C
F20	1 (TC)	8	IO	PT43C	-	L193T
F24	1 (TC)	8	IO	PT43A	-	L194T
B20	1 (TC)	8	IO	PT42D	VREF_1_08	L195C
L44	-	-	VSS	VSS	-	-
F22	1 (TC)	8	IO	PT42B	-	L196C
A20	1 (TC)	8	IO	PT42C	-	L195T
G22	1 (TC)	8	IO	PT42A	-	L196T
C21	1 (TC)	9	IO	PT41D	-	L197C
G21	1 (TC)	-	VDDIO1	VDDIO1	-	-
E26	1 (TC)	9	IO	PT41B	-	L198C
D21	1 (TC)	9	IO	PT41C	-	L197T
F26	1 (TC)	9	IO	PT41A	-	L198T
B21	1 (TC)	9	IO	PT40D	-	L199C
BB3	-	-	VDD15	VDD15	-	-
E25	1 (TC)	9	IO	PT40B	-	L200C
A21	1 (TC)	9	IO	PT40C	VREF_1_09	L199T
F25	1 (TC)	9	IO	PT40A	-	L200T
D22	1 (TC)	9	IO	PT39D	-	L201C
M38	-	-	VSS	VSS	-	-
E28	1 (TC)	9	IO	PT39B	-	L202C
E22	1 (TC)	9	IO	PT39C	-	L201T
F28	1 (TC)	9	IO	PT39A	-	L202T
C22	1 (TC)	10	IO	PT38D	-	L203C
G24	1 (TC)	-	VDDIO1	VDDIO1	-	-

Table 82. 1036 fpSBGA Pin Table (Continued)

FE1036 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	FE1036 Pair
E29	1 (TC)	10	IO	PT38B	-	L204C
B22	1 (TC)	10	IO	PT38C	VREF_1_10	L203T
F29	1 (TC)	10	IO	PT38A	-	L204T
C23	1 (TC)	10	IO	PT37D	-	L205C
BB4	-	-	VDD15	VDD15	-	-
E27	1 (TC)	10	IO	PT37B	-	L206C
D23	1 (TC)	10	IO	PT37C	-	L205T
F27	1 (TC)	10	IO	PT37A	-	L206T
B23	1 (TC)	10	IO	PT36D	-	L207C
R38	-	-	VSS	VSS	-	-
A23	1 (TC)	10	IO	PT36C	-	L207T
C24	1 (TC)	1	IO	PT35D	-	L209C
G27	1 (TC)	-	VDDIO1	VDDIO1	-	-
C31	1 (TC)	1	IO	PT35B	-	L210C
D24	1 (TC)	1	IO	PT35C	-	L209T
D31	1 (TC)	1	IO	PT35A	-	L210T
B24	1 (TC)	1	IO	PT34D	VREF_1_01	L211C
BB41	-	-	VDD15	VDD15	-	-
E31	1 (TC)	1	IO	PT34B	-	L212C
A24	1 (TC)	1	IO	PT34C	-	L211T
F31	1 (TC)	1	IO	PT34A	-	L212T
C25	1 (TC)	1	IO	PT33D	-	L213C
V7	-	-	VSS	VSS	-	-
E30	1 (TC)	1	IO	PT33B	-	L214C
D25	1 (TC)	1	IO	PT33C	-	L213T
F30	1 (TC)	1	IO	PT33A	-	L214T
B25	1 (TC)	2	IO	PT32D	-	L215C
G28	1 (TC)	-	VDDIO1	VDDIO1	-	-
E32	1 (TC)	2	IO	PT32B	-	L216C
A25	1 (TC)	2	IO	PT32C	VREF_1_02	L215T
F32	1 (TC)	2	IO	PT32A	-	L216T
C26	1 (TC)	2	IO	PT31D	-	L217C
BB42	-	-	VDD15	VDD15	-	-
D33	1 (TC)	2	IO	PT31B	-	L218C
D26	1 (TC)	2	IO	PT31C	-	L217T
E33	1 (TC)	2	IO	PT31A	-	L218T
B26	1 (TC)	2	IO	PT30D	-	L219C
V38	-	-	VSS	VSS	-	-
F33	1 (TC)	2	IO	PT30B	-	L220C
A26	1 (TC)	2	IO	PT30C	-	L219T
G33	1 (TC)	2	IO	PT30A	-	L220T
C27	1 (TC)	3	IO	PT29D	-	L221C

Table 82. 1036 fpSBGA Pin Table (Continued)

FE1036 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	FE1036 Pair
E34	1 (TC)	3	IO	PT29B	-	L222C
D27	1 (TC)	3	IO	PT29C	VREF_1_03	L221T
F34	1 (TC)	3	IO	PT29A	-	L222T
B27	1 (TC)	3	IO	PT28D	-	L223C
E35	1 (TC)	3	IO	PT28B	-	L224C
A27	1 (TC)	3	IO	PT28C	-	L223T
F35	1 (TC)	3	IO	PT28A	-	L224T
C28	1 (TC)	3	IO	PT27D	-	L225C
AA7	-	-	VSS	VSS	-	-
C35	1 (TC)	3	IO	PT27B	-	L226C
D28	1 (TC)	3	IO	PT27C	-	L225T
D35	1 (TC)	3	IO	PT27A	-	L226T
B28	1 (TC)	4	IO	PT26D	-	L227C
A37	1 (TC)	4	IO	PT26B	-	L228C
A28	1 (TC)	4	IO	PT26C	-	L227T
B37	1 (TC)	4	IO	PT26A	-	L228T
C29	1 (TC)	4	IO	PT25D	-	L229C
E36	1 (TC)	4	IO	PT25B	-	L230C
D29	1 (TC)	4	IO	PT25C	-	L229T
F36	1 (TC)	4	IO	PT25A	-	L230T
B29	1 (TC)	4	IO	PT24D	-	L231C
AA38	-	-	VSS	VSS	-	-
C37	1 (TC)	4	IO	PT24B	-	L232C
A29	1 (TC)	4	IO	PT24C	VREF_1_04	L231T
D37	1 (TC)	4	IO	PT24A	-	L232T
A30	1 (TC)	5	IO	PT23D	PTCK1C	L233C
E37	1 (TC)	5	IO	PT23B	-	L234C
B30	1 (TC)	5	IO	PT23C	PTCK1T	L233T
F37	1 (TC)	5	IO	PT23A	-	L234T
D30	1 (TC)	5	IO	PT22D	PTCK0C	L235C
A38	1 (TC)	5	IO	PT22B	-	L236C
C30	1 (TC)	5	IO	PT22C	PTCK0T	L235T
B38	1 (TC)	5	IO	PT22A	-	L236T
A31	1 (TC)	5	IO	PT21D	VREF_1_05	L237C
C38	1 (TC)	5	IO	PT21B	-	L238C
B31	1 (TC)	5	IO	PT21C	-	L237T
D38	1 (TC)	5	IO	PT21A	-	L238T
A32	1 (TC)	6	IO	PT20D	-	L239C
E38	1 (TC)	6	IO	PT20B	-	L240C
B32	1 (TC)	6	IO	PT20C	-	L239T
F38	1 (TC)	6	IO	PT20A	-	L240T
C32	1 (TC)	6	IO	PT19D	-	L241C

Table 82. 1036 fpSBGA Pin Table (Continued)

FE1036 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	FE1036 Pair
E43	1 (TC)	6	IO	PT19B	-	L242C
D32	1 (TC)	6	IO	PT19C	VREF_1_06	L241T
E44	1 (TC)	6	IO	PT19A	-	L242T
A34	0 (TL)	1	IO	PT18D	MPI_RTRY_N	L243C
A33	-	-	VSS	VSS	-	-
F44	0 (TL)	1	IO	PT18B	-	L244C
B34	0 (TL)	1	IO	PT18C	MPI_ACK_N	L243T
F43	0 (TL)	1	IO	PT18A	-	L244T
B33	0 (TL)	1	IO	PT17D	-	L245C
G38	0 (TL)	-	VDDIO0	VDDIO0	-	-
F41	0 (TL)	1	IO	PT17B	-	L246C
C33	0 (TL)	1	IO	PT17C	VREF_0_01	L245T
F42	0 (TL)	1	IO	PT17A	-	L246T
C34	0 (TL)	1	IO	PT16D	M0	L247C
F40	-	-	VDD15	VDD15	-	-
G39	0 (TL)	2	IO	PT16B	-	L248C
D34	0 (TL)	1	IO	PT16C	M1	L247T
G40	0 (TL)	2	IO	PT16A	-	L248T
A35	0 (TL)	2	IO	PT15D	MPI_CLK	L249C
A43	-	-	VSS	VSS	-	-
H40	0 (TL)	2	IO	PT15B	-	L250C
B35	0 (TL)	2	IO	PT15C	A21/MPI_BURST_N	L249T
H39	0 (TL)	2	IO	PT15A	-	L250T
A36	0 (TL)	2	IO	PT14D	M2	L251C
P38	0 (TL)	-	VDDIO0	VDDIO0	-	-
B36	0 (TL)	2	IO	PT14C	M3	L251T
C36	0 (TL)	2	IO	PT13D	VREF_0_02	L253C
G19	-	-	VDD15	VDD15	-	-
G41	0 (TL)	2	IO	PT13B	-	L254C
D36	0 (TL)	2	IO	PT13C	MPI_TEA_N	L253T
G42	0 (TL)	2	IO	PT13A	-	L254T
A39	0 (TL)	3	IO	PT12D	-	L255C
A44	-	-	VSS	VSS	-	-
A40	0 (TL)	3	IO	PT12C	-	L255T
A41	0 (TL)	3	IO	PT11D	VREF_0_03	L257C
U38	0 (TL)	-	VDDIO0	VDDIO0	-	-
G43	0 (TL)	3	IO	PT11B	-	L258C
A42	0 (TL)	3	IO	PT11C	-	L257T
G44	0 (TL)	3	IO	PT11A	-	L258T
B39	0 (TL)	3	IO	PT10D	D0	L259C
G25	-	-	VDD15	VDD15	-	-
B40	0 (TL)	3	IO	PT10C	TMS	L259T

Table 82. 1036 fpSBGA Pin Table (Continued)

FE1036 Ball	VDDIO Bank	VREF Group	I/O	Pin Description	Additional Function	FE1036 Pair
B41	0 (TL)	4	IO	PT9D	A20/MPI_BDIP_N	L261C
B1	-	-	VSS	VSS	-	-
H44	0 (TL)	4	IO	PT9B	-	L262C
B42	0 (TL)	4	IO	PT9C	A19/MPI_TSZ1	L261T
H43	0 (TL)	4	IO	PT9A	-	L262T
C39	0 (TL)	4	IO	PT8D	A18/MPI_TSZ0	L263C
C40	0 (TL)	4	IO	PT8C	D3	L263T
D39	0 (TL)	4	IO	PT7D	VREF_0_04	L265C
J42	0 (TL)	4	IO	PT7B	-	L266C
D40	0 (TL)	4	IO	PT7C	-	L265T
J41	0 (TL)	4	IO	PT7A	-	L266T
E39	0 (TL)	5	IO	PT6D	D1	L267C
F39	0 (TL)	5	IO	PT6C	D2	L267T
C41	0 (TL)	5	IO	PT5D	-	L269C
J40	0 (TL)	5	IO	PT5B	-	L270C
D41	0 (TL)	5	IO	PT5C	VREF_0_05	L269T
J39	0 (TL)	5	IO	PT5A	-	L270T
E41	0 (TL)	5	IO	PT4D	TDI	L271C
E42	0 (TL)	5	IO	PT4C	TCK	L271T
C43	0 (TL)	6	IO	PT3D	-	L273C
H42	0 (TL)	6	IO	PT3B	-	L274C
D43	0 (TL)	6	IO	PT3C	VREF_0_06	L273T
H41	0 (TL)	6	IO	PT3A	-	L274T
C44	0 (TL)	6	IO	PT2D	PLL_CK1C/PPLL	L275C
D44	0 (TL)	6	IO	PT2C	PLL_CK1T/PPLL	L275T
K38	-	-	O	PCFG_MPI_IRQ	CFG_IRQ_N/MPI_IRQ_N	-
K39	-	-	IO	PCCLK	CCLK	-
K40	-	-	IO	PDONE	DONE	-
K41	-	-	VDD33	VDD33_FPGAPLL	-	-

Note: All differential pairs use adjacent balls.

Package Information

Package Thermal Characteristics Summary

There are three thermal parameters that are in common use: ΘJA, ψJC, and ΘJC. It should be noted that all the parameters are affected, to varying degrees, by package design (including paddle size) and choice of materials, the amount of copper in the test board or system board, and system airflow.

ΘJA

This is the thermal resistance from junction to ambient (theta-JA, R-theta, etc.):

$$\Theta_{JA} = \frac{T_J - T_A}{Q} \quad (1)$$

where T_J is the junction temperature, T_A, is the ambient air temperature, and Q is the chip power.

Experimentally, ΘJA is determined when a special thermal test die is assembled into the package of interest, and the part is mounted on the thermal test board. The diodes on the test chip are separately calibrated in an oven. The package/board is placed either in a JEDEC natural convection box or in the wind tunnel, the latter for forced convection measurements. A controlled amount of power (Q) is dissipated in the test chip's heater resistor, the chip's temperature (T_J) is determined by the forward drop on the diodes, and the ambient temperature (T_A) is noted. Note that ΘJA is expressed in units of °C/W.

ψJC

This JEDEC designated parameter correlates the junction temperature to the case temperature. It is generally used to infer the junction temperature while the device is operating in the system. It is not considered a true thermal resistance and it is defined by:

$$\psi_{JC} = \frac{T_J - T_C}{Q} \quad (2)$$

where T_C is the case temperature at top dead center, T_J is the junction temperature, and Q is the chip power. During the ΘJA measurements described above, besides the other parameters measured, an additional temperature reading, T_C, is made with a thermocouple attached at top-dead-center of the case. ψJC is also expressed in units of °C/W.

ΘJC

This is the thermal resistance from junction to case. It is most often used when attaching a heat sink to the top of the package. It is defined by:

$$\Theta_{JC} = \frac{T_J - T_C}{Q} \quad (3)$$

The parameters in this equation have been defined above. However, the measurements are performed with the case of the part pressed against a water-cooled heat sink to draw most of the heat generated by the chip out the top of the package. It is this difference in the measurement process that differentiates ΘJC from ψJC. ΘJC is a true thermal resistance and is expressed in units of °C/W.

ΘJB

This is the thermal resistance from junction to board. It is defined by:

$$\Theta_{JB} = \frac{T_J - T_B}{Q} \quad (4)$$

where T_B is the temperature of the board adjacent to a lead measured with a thermocouple. The other parameters on the right-hand side have been defined above. This is considered a true thermal resistance, and the measure-

ment is made with a water-cooled heat sink pressed against the board to draw most of the heat out of the leads. Note that ΘJB is expressed in units of °C/W and that this parameter and the way it is measured are still being discussed by the JEDEC committee.

FPSC Maximum Junction Temperature

Once the power dissipated by the FPSC has been determined, the maximum junction temperature of the FPSC can be found. This is needed to determine if speed derating of the device from the 85 °C junction temperature used in all of the delay tables is needed. Using the maximum ambient temperature, T_{Amax}, and the power dissipated by the device, Q (expressed in °C), the maximum junction temperature is approximated by:

$$T_{Jmax} = T_{Amax} + (Q \bullet \Theta_{JA})$$

Package Thermal Characteristics

The thermal characteristics of the 1036-ball fpSBGA and the 1156-ball fpBGA used for the ORSPI4 are available at the Thermal Management section of the Lattice Semiconductor web site at www.latticesemi.com.

Heat Sink Vendors for BGA Packages

The estimated worst-case power requirements for the ORSPI4 is in the 8 W to 10 W range. Consequently, for most applications an external heat sink will be required. Table 83 lists, in alphabetical order, heat sink vendors who advertise heat sinks aimed at the BGA market.

Table 83. Heat Sink Vendors

Vendor	Location	Phone
Aavid Thermalloy	Concord, NH	(603) 224-9988
Chip Coolers	Warwick, RI	(800) 227-0254
IERC	Burbank, CA	(818) 842-7277
R-Theta	Buffalo, NY	(800) 388-5428
Sanyo Denki	Torrance, CA	(310) 783-5400
Wakefield Thermal Solutions	Pelham, NH	(800) 325-1426

Package Parasitics

The electrical performance of an IC package, such as signal quality and noise sensitivity, is directly affected by the package parasitics. Table 84 lists eight parasitics associated with the ORCA packages. These parasitics represent the contributions of all components of a package, which include the bond wires, all internal package routing, and the external leads.

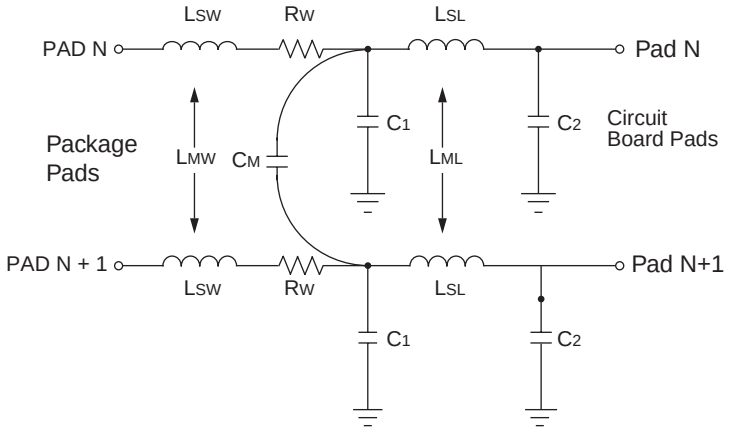
Four inductances in nH are listed: L_{SW} and L_{SL}, the self-inductance of the lead; and L_{MW} and L_{ML}, the mutual inductance to the nearest neighbor lead. These parameters are important in determining ground bounce noise and inductive crosstalk noise. Three capacitances in pF are listed: C_M, the mutual capacitance of the lead to the nearest neighbor lead; and C_1 and C_2, the total capacitance of the lead to all other leads (all other leads are assumed to be grounded). These parameters are important in determining capacitive crosstalk and the capacitive loading effect of the lead. Resistance values are in mΩ.

The parasitic values in Table 84 are for the circuit model of bond wire and package lead parasitics. If the mutual capacitance value is not used in the designer's model, then the value listed as mutual capacitance should be added to each of the C_1 and C_2 capacitors.

Table 84. ORCA Typical Package Parasitics

Package	L_{SW}	L_{MW}	R_W	C_1	C_2	C_M	L_{SL}	L_{ML}
1036	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD
1156	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD

Figure 89. Package Parasitics



Package Outline Drawings

Package Outline Drawings for the 1036-ball fpSBGA and the 1156-ball fpBGA used for the ORSPI4 are available in the Package Diagrams section of the Lattice Semiconductor web site at www.latticesemi.com.

Ordering Information

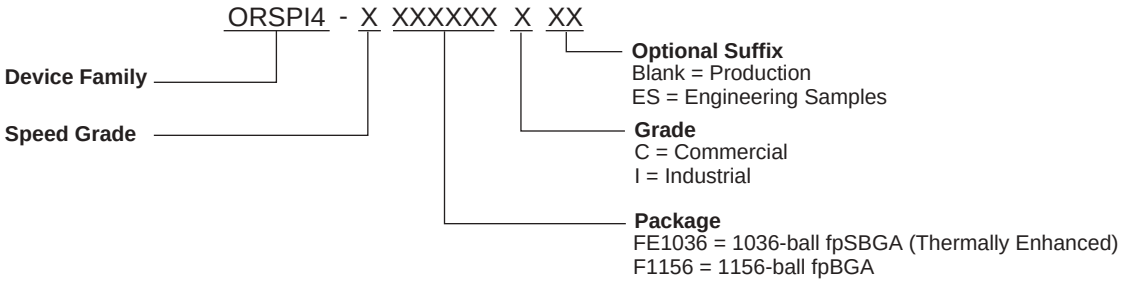


Table 85. Device Type Options

Device	Voltage
ORSPI4	1.5 V internal 3.3 V/2.5 V/1.8 V/ 1.5 V I/O

Table 86. Commercial Ordering Information[1]

Device Family	Part Number	Speed Grade	Package Type	Ball Count	Grade
ORSPI4	ORSPI4-2FE1036CES	2	fpSBGA	1036	C
	ORSPI4-1FE1036CES	1			C
	ORSPI4-2F1156CES	2	fpBGA	1156	C
	ORSPI4-1F1156CES	1			C

Table 87. Industrial Ordering Information[1]

Device Family	Part Number	Speed Grade	Package Type	Ball Count	Grade
ORSPI4	ORSPI4-1FE1036IES	1	fpSBGA	1036	I
	ORSPI4-F1156IES	1	fpBGA	1156	I

1. For all but the slowest commercial speed grade, the speed grades on these devices are dual marked. For example, the commercial speed grade -2XXXXXC is also marked with the industrial grade -1XXXXXI. The commercial grade is always one speed grade faster than the associated dual mark industrial grade. The slowest commercial speed grade is marked as commercial grade only.