



Lattice**CORE**

## Video Frame Buffer IP Core User's Guide

---

|                                                                    |           |
|--------------------------------------------------------------------|-----------|
| <b>Chapter 1. Introduction .....</b>                               | <b>4</b>  |
| Quick Facts .....                                                  | 4         |
| Features .....                                                     | 4         |
| Release Information .....                                          | 5         |
| Device Support.....                                                | 5         |
| <b>Chapter 2. Functional Description .....</b>                     | <b>6</b>  |
| Key Concepts.....                                                  | 6         |
| Block Diagram.....                                                 | 6         |
| Frame Rate Conversion .....                                        | 6         |
| Dynamic Parameter Updating .....                                   | 7         |
| Memory Bandwidth and Size .....                                    | 7         |
| Primary I/O .....                                                  | 8         |
| Interface Descriptions .....                                       | 9         |
| Video Input/Output .....                                           | 9         |
| Memory interface .....                                             | 9         |
| Parameter Register Read/Write Interface .....                      | 10        |
| Timing Description .....                                           | 10        |
| Video Input/Output Timing .....                                    | 10        |
| Video Frame Timing.....                                            | 12        |
| Memory Interface Timing .....                                      | 13        |
| Dynamic Parameter Updating .....                                   | 13        |
| <b>Chapter 3. Parameter Settings .....</b>                         | <b>14</b> |
| Architecture .....                                                 | 15        |
| Frame Dimensions .....                                             | 15        |
| I/O Specification .....                                            | 16        |
| Implementation.....                                                | 17        |
| <b>Chapter 4. IP Core Generation.....</b>                          | <b>18</b> |
| Licensing the IP Core.....                                         | 18        |
| Getting Started .....                                              | 18        |
| Configuring Video Frame Buffer Core in IPexpress .....             | 18        |
| IPexpress-Created Files and Top Level Directory Structure.....     | 21        |
| Instantiating the Core .....                                       | 22        |
| Running Functional Simulation .....                                | 22        |
| Synthesizing and Implementing the Core in a Top-Level Design ..... | 22        |
| Hardware Evaluation.....                                           | 23        |
| Updating/Regenerating the IP Core .....                            | 23        |
| Regenerating an IP Core .....                                      | 23        |
| <b>Chapter 5. Support Resources .....</b>                          | <b>25</b> |
| Lattice Technical Support.....                                     | 25        |
| Online Forums.....                                                 | 25        |
| Telephone Support Hotline .....                                    | 25        |
| E-mail Support .....                                               | 25        |
| Local Support.....                                                 | 25        |
| Internet.....                                                      | 25        |
| References.....                                                    | 25        |
| Revision History .....                                             | 25        |
| <b>Appendix A. Resource Utilization .....</b>                      | <b>26</b> |
| LatticeECP3 Devices .....                                          | 26        |
| Ordering Part Number.....                                          | 26        |

---

|                                              |    |
|----------------------------------------------|----|
| LatticeECP2M and LatticeECP2MS Devices ..... | 26 |
| Ordering Part Number.....                    | 26 |
| LatticeECP2 and LatticeECP2S Devices .....   | 26 |
| Ordering Part Number.....                    | 27 |
| LatticeXP2 Devices .....                     | 27 |
| Ordering Part Number.....                    | 27 |

# Introduction

The Video Frame Buffer IP core buffers video data in external memory to be displayed on output devices such as computer monitors, projectors, etc. The Video Frame Buffer IP core supports image sizes up to 4Kx4K with YCbCr 4:2:2, 4:4:4 and RGB video formats. It supports dynamic parameter updating via a parameter bus which can be configured to operate on a different clock from the core. Simple frame rate conversion is employed to support different input and output frame rates.

## Quick Facts

**Table 1-1. Video Frame Buffer IP Core Quick Facts**

| Video Frame Buffer IP Core Configuration |                         |                                                                                  |                   |                  |
|------------------------------------------|-------------------------|----------------------------------------------------------------------------------|-------------------|------------------|
| Core Requirements                        | FPGA Families Supported | LatticeECP2™                                                                     | LatticeECP3™      | LatticeXP2™      |
|                                          | Minimum Device Required | LFE2-6E                                                                          | LFE3-17EA         | LFXP2-5E         |
|                                          | Targeted Device         | LFE2-35E-7F672C                                                                  | LFE3-35EA-8FN672C | LFXP2-30E-7F672C |
| Resource Utilization                     | Configuration           | NTSC720x480 8-bit Serial YCbCr4:2:2                                              |                   |                  |
|                                          | Registers               | 774                                                                              | 771               | 774              |
|                                          | LUTs                    | 982                                                                              | 941               | 982              |
|                                          | EBRs                    | 2                                                                                | 2                 | 2                |
| Resource Utilization                     | Configuration           | XGA1024x768 8-bit Parallel RGB                                                   |                   |                  |
|                                          | Registers               | 834                                                                              | 838               | 834              |
|                                          | LUTs                    | 1097                                                                             | 1093              | 1097             |
|                                          | EBRs                    | 2                                                                                | 2                 | 2                |
| Resource Utilization                     | Configuration           | HD1920x1080 10-bit Parallel YCbCr4:2:2                                           |                   |                  |
|                                          | Registers               | 1093                                                                             | 1095              | 1093             |
|                                          | LUTs                    | 1365                                                                             | 1349              | 1365             |
|                                          | EBRs                    | 2                                                                                | 2                 | 2                |
| Design Tool Support                      | Lattice Implementation  | Lattice Diamond® 1.4                                                             |                   |                  |
|                                          | Synthesis               | Synopsys® Synplify™ Pro for Lattice F-2011.09L                                   |                   |                  |
|                                          | Simulation              | Aldec® Active-HDL™ 8.3 SP1 Lattice Edition<br>Mentor Graphics® ModelSim™ SE 6.3F |                   |                  |

## Features

- Supports single color, YCbCr 4:2:2, YcbCr 4:4:4 and RGB video formats
- Supports input and output resolutions of 64x64 to 4Kx4K pixels
- Supports serial and parallel pixel processing
- Supports frame rate conversion
- Supports dynamic parameter update of frame size and Keep mode
- Supports configurable parameter bus width
- Supports configurable parameter bus clock
- Supports configurable memory bus width and base address
- Supports configurable memory burst length and burst count
- Configurable internal FIFO type and depth
- Supports 8, 10 or 12-bit color depth per plane

## Release Information

- Video Frame Buffer IP core version 1.0
- Last updated February 2012

## Device Support

- LatticeECP3, LatticeECP2, LatticeECP2M™, LatticeECP2MS, LatticeECP2S, LatticeXP2

# Functional Description

## Key Concepts

The Video Frame Buffer IP core receives input video data, stores it in the external memory and outputs it based on the timing controlled by the dout\_enable signal. The core stores data into memory in a different width format as required by the application. It also provides synchronization of data across different clock domains as well as different format domains.

The core provides a simple parameter bus for dynamic frame size updating. It also implements a flexible memory interface that can be connected to Lattice memory controller IP cores.

The core also supports interlaced video stream by combining two fields into one frame outside.

## Block Diagram

**Figure 2-1. Video Frame Buffer IP Core Block Diagram**

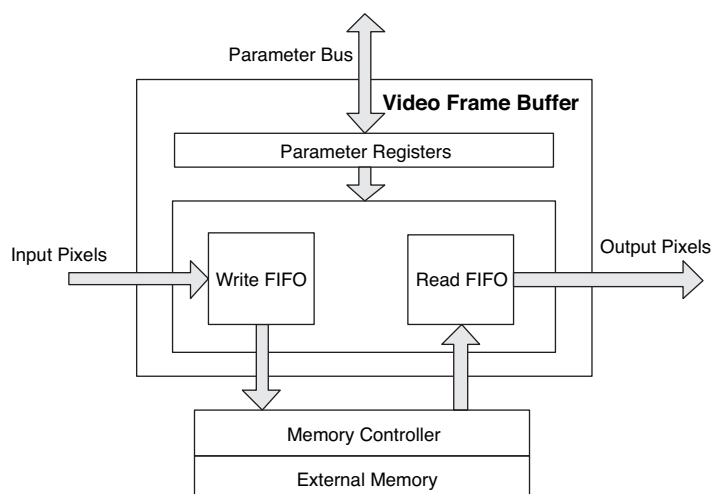


Figure 2-1 shows the block diagram of the Video Frame Buffer IP core. The core supports continuous data streams from/to external interfaces in different clock domains using asynchronous Write and Read FIFOs. Input pixels are packed and stored into the asynchronous double clock Write FIFO first. Then pixels are sent to an external memory controller to be written to the memory. After an entire video frame has been stored in the external memory, frame reading starts. The pixels read from external memory are stored in the asynchronous Read FIFO and transferred to output interface clock domain. After unpacking, pixels are output from the Video Frame Buffer IP core.

In the video frame buffer, several clock sources are involved. The memory interface operates on a separate memory clock. When frame rate conversion is enabled, there are two clocks in the video data path: input pixel sample clock and output pixel sample clock. When frame rate conversion is disabled, the video data path operates at input pixel sample clock rate. When dynamic parameter updating is enabled, the parameter bus can be configured to run on a separate clock. By default, the parameter bus runs on the input pixel sample clock.

## Frame Rate Conversion

The Video Frame Buffer IP core provides a simple frame rate conversion by repeating or dropping frames.

When frame rate conversion is enabled, the core's output data path runs at the output pixel clock rate. The output frame rate is controlled by the output pixel sample clock and dout\_enable signal. When dout\_enable signal is high, the core outputs pixels continuously. When there is no new video frame, the core will output the last frame repeatedly.

When frame rate conversion is disabled, the core's output data path will run on the input pixel sample clock. The output frame is generated directly from the input video stream. If there is no new input video frame, the core stops generating output data.

## Dynamic Parameter Updating

The Video Frame Buffer IP core provides a parameter bus port for internal parameter update at run time. The parameters are double-buffered to avoid adverse effect to the core when they are changed. The new values are buffered and transferred to the working registers when the core is ready to accept a new configuration. The UPDATE register is used to indicate when the new values are consumed and when the buffers can accept new data.

**Table 2-1. Parameter Register Maps**

| Address | Register  | Size | W/R | Default | Description                                                                                                                                                                                                                        |
|---------|-----------|------|-----|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0x00    | FRMWIDTH  | 32   | W/R | 719     | <b>Frame width register.</b> The FRMWIDTH value must be (frame width – 1). The minimum value is 63, and the maximum value is the max frame width specified on the IP GUI minus 1. The default value is the maximum value.          |
| 0x04    | FRMHEIGHT | 32   | W/R | 479     | <b>Frame height register.</b> The FRMHEIGHT must be (frame height – 1). The minimum value is 63, and the maximum value is the max frame height specified on the IP GUI minus 1. The default value is the maximum value.            |
| 0x08    | KEEP      | 1    | W/R | 0       | <b>Keep mode register.</b> The value can be 0 or 1. When this bit is 1, the core is in an output locked mode. The same output frame is sent out repeatedly if frame rate conversion is active; otherwise there is no output frame. |
| 0x0C    | UPDATE    | 1    | W/R | 0       | <b>Update parameter enable register.</b> The value can be 0 or 1. Setting this bit to 1 triggers the update of the above parameters. The core resets it to 0 after the parameters have been updated.                               |

All the parameter registers can be written to only when the UPDATE register bit is 0. The parameter KEEP will take effect at the next output frame. When the UPDATE bit is set to 1, the parameters FRMWIDTH and FRMHEIGHT will take effect when the frmsync\_in signal is active, indicating a new input frame is arriving. After updating its internal parameters, the core resets the UPDATE bit to indicate that the parameter registers are now empty and can take on new values.

## Memory Bandwidth and Size

The Video Frame Buffer IP core stores and retrieves pixels to/from external memory using memory burst write and read commands. When DDR2 memory is used for external memory, one burst operation, (burst\_length\*burst\_count/2)\*memory\_data\_width bit, cannot exceed the size of a single video line. A single video line will be transferred through multiple burst write/read transactions internally.

When frame rate conversion is inactive the video frame buffer core needs a 2-frame memory storage space; when frame rate conversion is active, a 3-frame storage space is required. The total external memory size the core requires can be viewed on the video frame buffer IP GUI.

The required memory bandwidth is input pixel data rate plus output pixel data rate.

For example, for parallel 8-bit YCbCr 4:2:2 pixels, if the input pixel sample clock is 74.25 MHz and output pixel sample clock is 148.5 MHz, the required bandwidth is  $2 \times 8 \times (74.25 + 148.5) = 3564 \text{ bit} \times \text{MHz}$ . If the memory data width is 32, the required memory clock will be  $(3564/32) = 111.375 \text{ MHz}$ .

## Primary I/O

Figure 2-2. Video Frame Buffer IP Core I/O

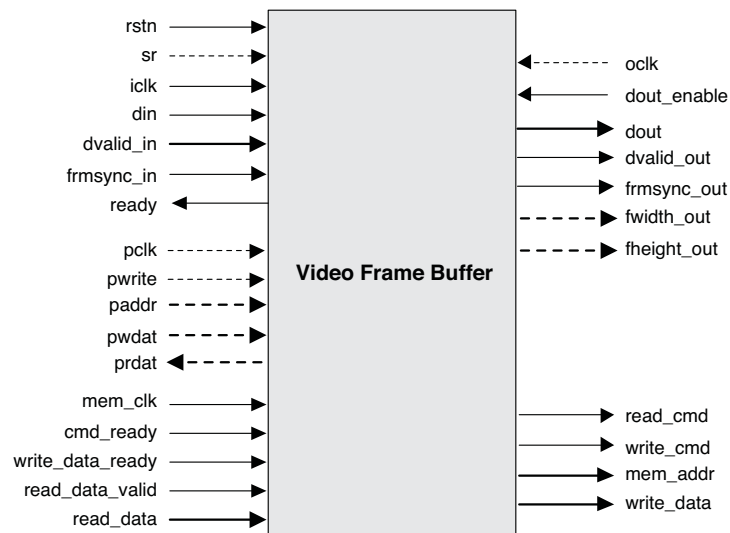


Table 2-2. Primary I/O

| Port                    | Size           | I/O | Description                                                                                    |
|-------------------------|----------------|-----|------------------------------------------------------------------------------------------------|
| <b>General I/Os</b>     |                |     |                                                                                                |
| rstn                    | 1              | I   | Asynchronous active-low reset signal.                                                          |
| sr                      | 1              | I   | Synchronous reset signal (optional).                                                           |
| iclk                    | 1              | I   | Input pixel sample clock.                                                                      |
| frmsync_in              | 1              | I   | New input video frame indicator, active-high.                                                  |
| dvalid_in               | 1              | I   | Input video data valid signal, active-high.                                                    |
| din                     | 8/10/12        | I   | Input video data in frame format.                                                              |
| ready                   | 1              | O   | When high, indicates the Video Frame Buffer IP core can accept more input data.                |
| oclk                    | 1              | I   | Output sample clock, present when frame rate conversion is active.                             |
| dout_enable             | 1              | I   | Input from down-stream module to enable output data, active high.                              |
| dout                    | 8/10/12        | O   | Output video pixel in frame format.                                                            |
| dvalid_out              | 1              | O   | Output video pixel valid signal, active high                                                   |
| frmsync_out             | 1              | O   | New output frame indicator, active high.                                                       |
| <b>Memory Interface</b> |                |     |                                                                                                |
| mem_clk                 | 1              | I   | Memory write/read clock.                                                                       |
| cmd_ready               | 1              | I   | Input from memory controller indicating it's ready to accept a new command, active high.       |
| mem_addr                | 32             | O   | Memory read/write address.                                                                     |
| read_cmd                | 1              | O   | Memory read command, active-high.                                                              |
| read_data               | 8/16/32/64/128 | I   | Read data output from memory.                                                                  |
| read_data_valid         | 1              | I   | Read data valid indicator from memory controller, active high.                                 |
| write_cmd               | 1              | O   | Memory write command, active-high.                                                             |
| write_data              | 8/16/32/64/128 | O   | Write data to memory.                                                                          |
| write_data_ready        | 1              | I   | Input from memory controller indicating that it's ready to accept new write data, active high. |



**Table 2-2. Primary I/O (Continued)**

| Port                 | Size    | I/O | Description                                         |
|----------------------|---------|-----|-----------------------------------------------------|
| <b>Optional I/Os</b> |         |     |                                                     |
| fwidth_out           | 16      | O   | Current output frame width, only for dynamic mode.  |
| fheight_out          | 16      | O   | Current output frame height, only for dynamic mode. |
| pclk                 | 1       | I   | Parameter bus clock, configurable.                  |
| pwrite               | 1       | I   | Parameter bus write enable, active-high.            |
| paddr                | 5       | I   | Parameter bus address.                              |
| pwdat                | 8/16/32 | I   | Parameter bus write data.                           |
| prdat                | 8/16/32 | O   | Parameter bus read data.                            |

## Interface Descriptions

### Video Input/Output

The Video Frame Buffer IP core uses a simple handshaking method to pass pixel data into and out of the core. The core asserts its ready output when it is ready to receive data. When the driving module has data to pass to the video frame buffer, it asserts the core's dvalid\_in port and at the same time placing the input video data on the din port. The frmsync\_in input should be driven to a 1 during the clock cycle when the very first active pixel is placed on the din bus.

Similarly, dvalid\_out is active when valid output pixels are available on dout, and frmsync\_out marks the first pixel in an output frame.

The output ports fwidth\_out and fheight\_out indicate the current output frame's width and height respectively, which are valid only when frmsync\_out is asserted. The input signal dout\_enable enables the core to generate output pixels. When dout\_enable is low (inactive), the core stops generating output pixels.

### Memory interface

The Video Frame Buffer IP core implements a flexible memory interface which operates on a separate clock from the main core.

When connecting to DDR2 memory controller, the burst length and burst count of video frame buffer should have the same values as those of DDR2 memory controller; when connecting to DDR3 memory controller, the burst length of video frame buffer should have the same value with DDR3 memory controller, while the burst count of video frame buffer should be half as that of DDR3 memory controller.

The video frame buffer assumes a memory byte addressing scheme. When connecting the memory controller, the address connection should be adjusted according to the DDR memory data width.

Normally the address connection can be (Assuming the DDR2/3 memory has 27 bits address width):

```

ddr_addr      = {1'b0,mem_addr[25:0]}; // for 8-bit DDR memory only
ddr_addr      = {2'b00,mem_addr[25:1]}; // for 16-bit DDR memory only
ddr_addr      = {3'b000,mem_addr[25:2]}; // for 32-bit DDR memory only

```

And the corresponding command should be:

```

ddr_cmd       = (write_cmd) ? 4'b0010 : 4'b0001; // (write_cmd) ? WRITE : READ
ddr_cmd_valid = (write_cmd || read_cmd);

```

In order to get the best throughput (about 5% increase compared to normal connection), the users should fine tune the combination of row/bank/column/ address to get the best throughput.

To get maximum throughput on the memory bus (about 5% increase compared to normal connection), the user needs to steer clear of write-to-precharge/read-to-precharge/precharge-to-active during row switching. That is, to access each bank in turn by using write/read with auto-precharge command to close current bank immediately after current burst write/read. In order to adopt this policy, the users should fine tune the combination of Row/Bank/Column address to get the best throughput.

For example, when DDR2 memory data width is 16, row size is 14, column size is 10, bank size is 8, burst length is 8 and burst count is 1, which means memory controller user interface data width is 32, row address is 14 bits, column address is 10 bits, bank address is  $\log_2(\text{bank size})=3$  bits and  $\log_2(\text{burst length} * \text{burst count})= 3$  bits. The memory controller address mapping inside the memory controller IP core is

```
ddr_addr = {row_addr[13:0], bank_addr[2:0], col_addr[9:0]}
```

In order to get the best throughput, we increase the col\_addr[2:0] first to utilize the burst operation, then we increase the bank\_addr[2:0], followed by increasing the rest of col\_addr[9:3], and finally with the row\_addr. So the interface to the DDR2/3 memory will look like:

```
ddr_addr = {2'b00, mem_addr[25:14], mem_addr[6:4], mem_addr[13:7], mem_addr[3:1]};
```

The corresponding command should be:

```
ddr_cmd = (write_cmd) ? 4'b0100 : 4'b0011; // (write_cmd) ? WRITEA : READA
```

As the memory controller user interface data width is 32, mem\_addr[1:0] will always be zero.

The core arbitrates between memory write and read operations, ensuring only a *write\_cmd* or a *read\_cmd* is asserted at any given time.

Two clock cycles after the *write\_data\_ready* is asserted, data will be available on the *write\_data* port. That means the parameter "Data\_rdy to Write Data Delay" of memory controller must be set to 2

The *cmd\_ready* can be asserted once every two clock cycles, and it should have at least a one-cycle interval, which is consistent with Lattice DDR Memory Controller IP cores.

## Parameter Register Read/Write Interface

The Video Frame Buffer IP core implements a simple register read/write interface for run-time parameter updates. The parameter bus interface can be configured to run on a separate clock. It operates at the input pixel clock rate by default.

When pwrite is high, pwrdat and paddr must contain valid data. The contents of all parameter registers will be transferred to the core's internal storage when UPDATE is asserted. If a parameter has not been written to before the assertion of UPDATE, its old value will be transferred into the internal storage.

prdat contains register read data corresponding to the address value placed on the paddr in the previous clock cycle.

When the parameter bus data width is equal to 32, paddr[1:0] should be fixed to 0. When parameter bus data width equals to 16, paddr[0] should be fixed to 0.

The parameter bus data width should be configured based on the system CPU's data width.

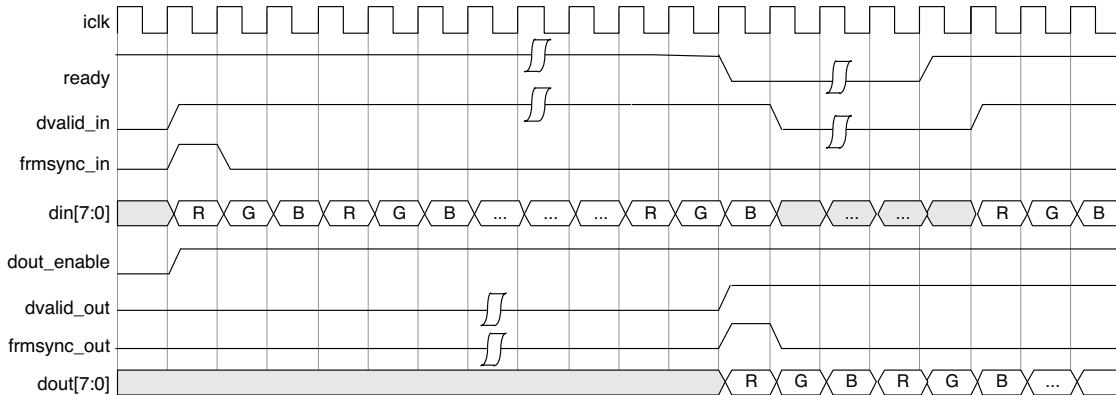
## Timing Description

### Video Input/Output Timing

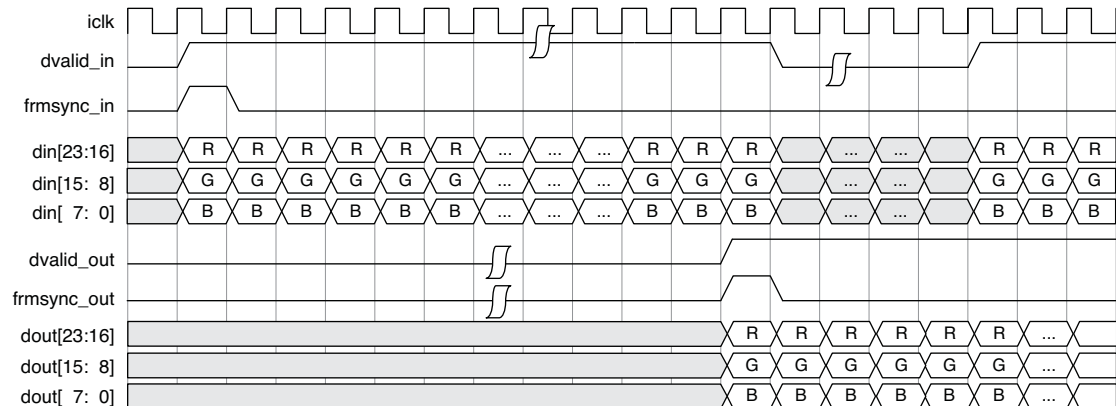
The Video Frame Buffer IP core supports single color, YCbCr 4:2:2, YCbCr 4:4:4 or RGB video format.

For YCbCr 4:4:4 or RGB video format, the three planes are interleaved for serial processing and combined on the din and dout ports for parallel processing. Figures 2-3 and 2-4 show the timing of RGB serial processing and parallel processing.

**Figure 2-3. RGB Serial Processing (8-bit pixel)**

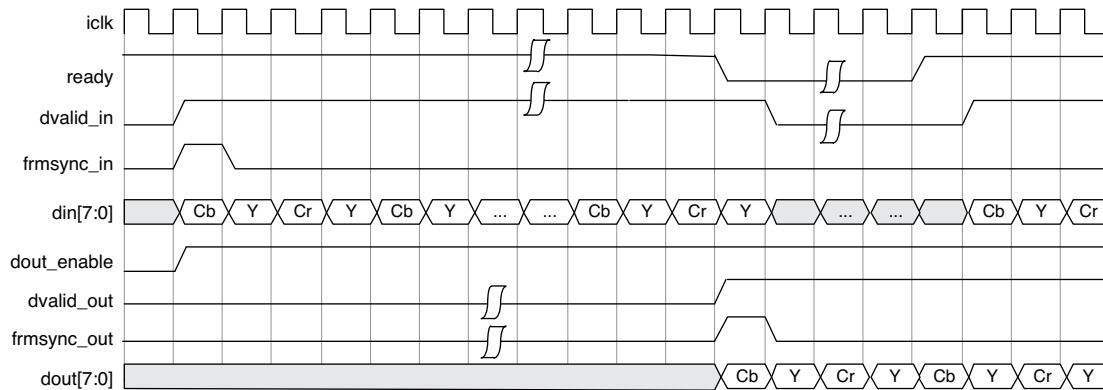


**Figure 2-4. RGB Parallel Processing (8-bit pixel)**



For YCbCr 4:2:2 video serial processing, the input and output sequence should be Cb, Y, Cr, Y, .... For parallel processing, the Y plane occupies the upper bits of the din and dout ports, and the Cb and Cr planes the lower bits. Cb and Cr planes are interleaved in the lower half, and Cb comes before Cr. Figures 2-5 and 2-6 show the timing of YCbCr 4:2:2 serial processing and parallel processing.

**Figure 2-5. YCbCr 4:2:2 Serial Processing (8-bit pixel)**



**Figure 2-6. YCbCr 4:2:2 Parallel Processing (8-bit pixel)**

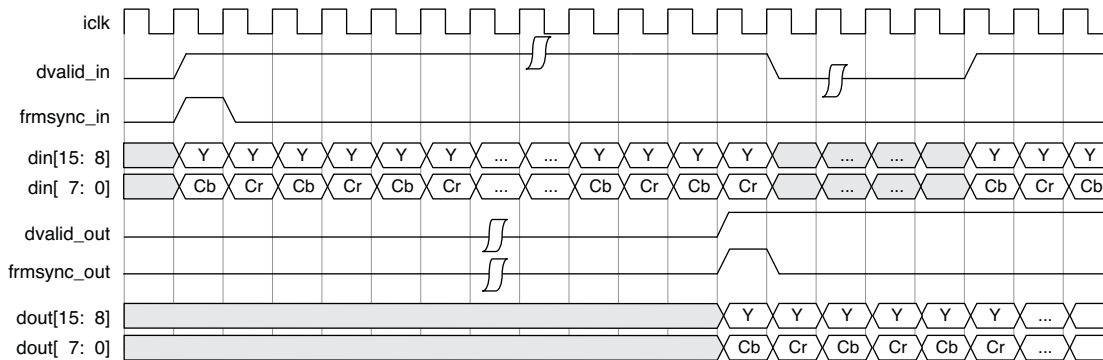
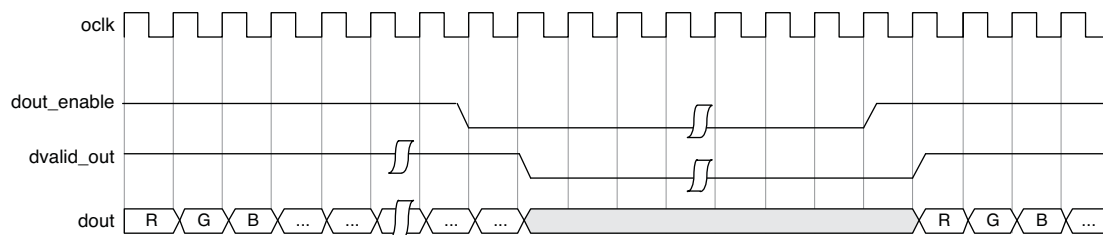


Figure 2-7 shows the **dout\_enable** control timing. When **dout\_enable** is de-asserted, the core will stop outputting data. Similarly, when **dout\_enable** is asserted, the core will begin outputting data. The assertion and de-assertion of **dout\_enable** can be used to generate a horizontal blank and a vertical blank depending on the output video format.

**Figure 2-7. **dout\_enable** Control Timing**



## Video Frame Timing

**Figure 2-8. Output Frame Rate is the Same as the Input Frame Rate**

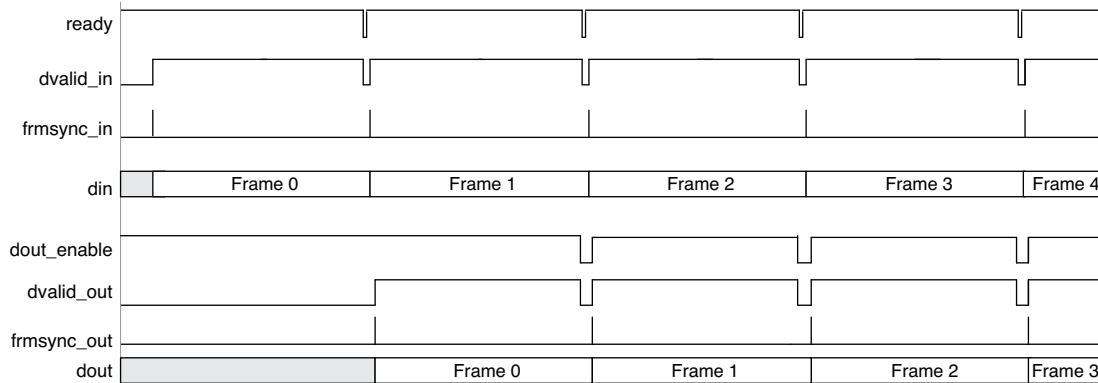


Figure 2-8 shows the frame timing when frame rate conversion is disabled. After accepting one input frame, the core starts generating frames. The output frame is driven by the input frame. When a new input frame arrives before the current output frame is finished, the current output frame is dropped and a new output frame is started. After the last input frame, the video frame buffer stops generating output frames.

**Figure 2-9. Output Frame Rate is Twice the Input Frame Rate**

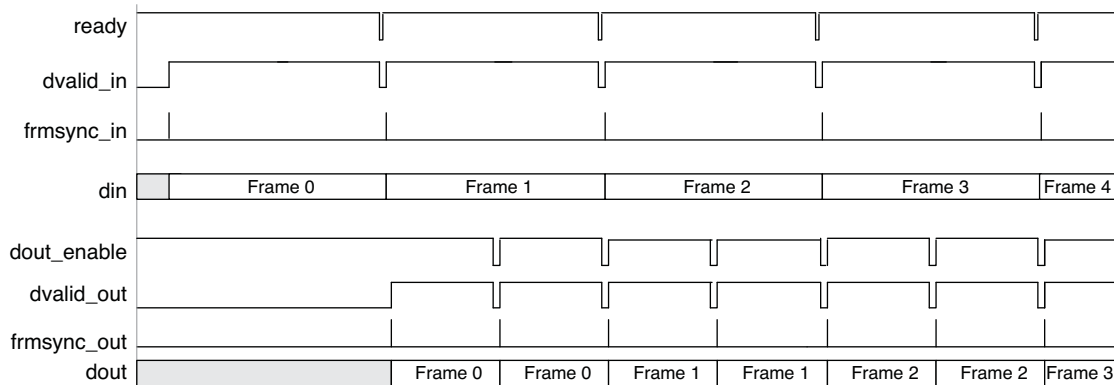


Figure 2-9 shows the frame timing when the output frame rate is twice the input frame rate. Output frames run on a separate output pixel clock. The output frame rate is determined by the output pixel clock and dout\_enable signal. The core generates output frames based on the oldest pending input frame(s). When a new input frame is received, the core pushes out the oldest frame and exports the second oldest frame. When there is no new frame input, the core exports the stored frames and then keeps exporting the last frame repeatedly. When the external memory is fully occupied by the unconsumed frames (no more space for new frames), the core will overwrite the latest input frame.

## Memory Interface Timing

Figure 2-10 shows the timing of the memory write operation. When the internal write FIFO is half full, it triggers memory write operation cycles. The memory write operation will continue until the write FIFO is empty.

**Figure 2-10. Timing Diagram for Memory Write**

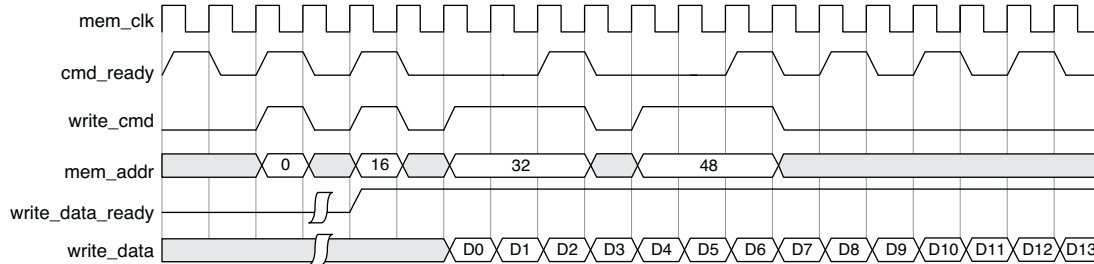
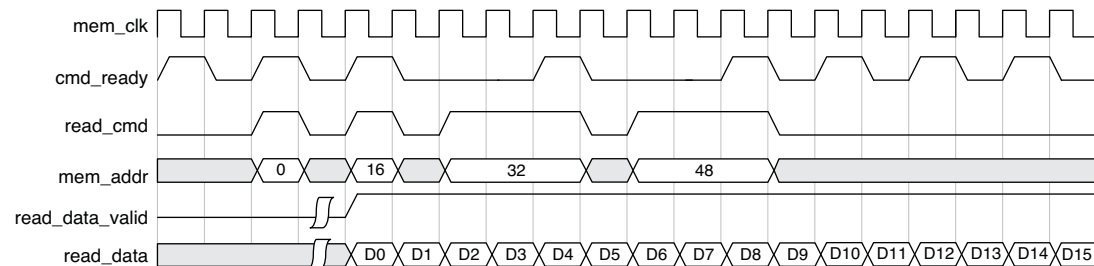


Figure 2-11 shows the timing of memory read operation. When the internal read FIFO is less than half full, and there is no ongoing write operation, the memory read burst operation is started. The length of the read burst operation and once read burst makes the read FIFO half full.

The memory write and read operations are in bursts. The memory write and read operations switch at the end of the burst cycle.

**Figure 2-11. Timing Diagram for Memory Read**

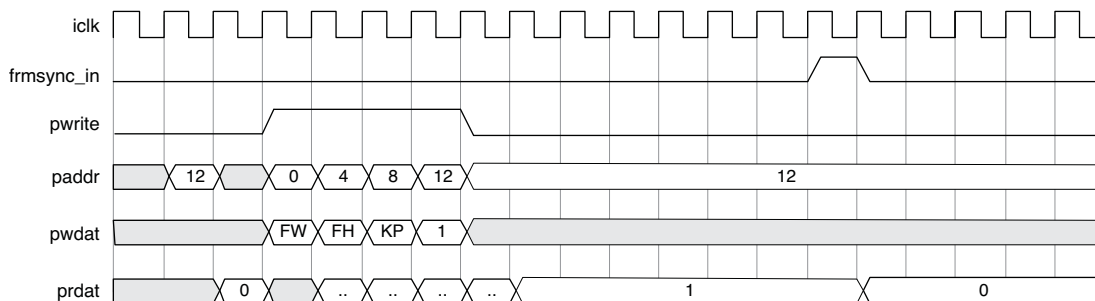


## Dynamic Parameter Updating

Figure 2-12 shows the timing of dynamic parameter updating. FW and FH are the video frame width and height. KP is the KEEP register value.

The parameter bus can be configured to operate on a separate clock. By default, it operates at the input pixel clock rate. The parameter registers are writable only when the UPDATE register is 0. When the UPDATE register bit is set to 1, the core will update its internal parameter registers with the new values at the next active frmsync\_in and reset the UPDATE register bit.

**Figure 2-12. Timing Diagram for Dynamic Parameter Updating (parameter bus width = 32)**



# Parameter Settings

This section describes how to generate the Lattice Video Frame Buffer IP core using the IPexpress™ tool. Refer to [“IP Core Generation” on page 18](#) for a description of how to generate the IP.

The Video Frame Buffer configuration GUI is accessed via the IPexpress tool and provides an interface for setting the desired parameters and invoking the IP core generator. Since the values of some parameters affect the size of the resultant core, the maximum value for these parameters may be limited by the size of the target device. [Table 3-1](#) provides a list of user-configurable parameters for the Video Frame Buffer IP core.

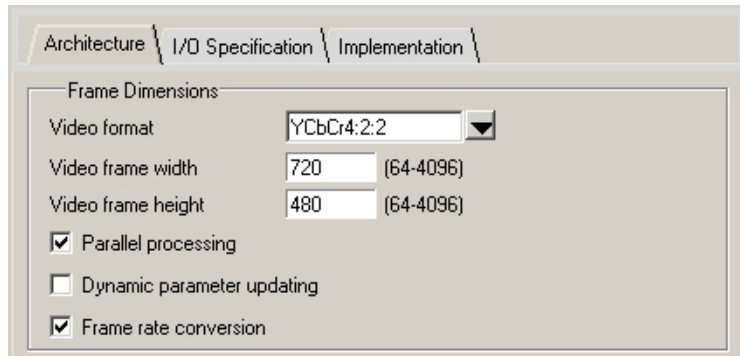
**Table 3-1. Video Frame Buffer IP Core Parameters**

| Parameter                     | Range/Options                               | Default       |
|-------------------------------|---------------------------------------------|---------------|
| <b>Video Frame In and Out</b> |                                             |               |
| Video format                  | Single color, YCbCr4:2:2, YcbCr4:4:4 or RGB | YCbCr4:2:2    |
| (Max) Video frame width       | 64-4096                                     | 720           |
| (Max) Video frame height      | 64-4096                                     | 480           |
| Parallel processing           | Yes, No                                     | Yes           |
| Dynamic parameter updating    | Yes, No                                     | No            |
| Frame rate conversion         | Yes, No                                     | Yes           |
| <b>I/O Specifications</b>     |                                             |               |
| Input pixel width             | 8, 10, 12                                   | 8             |
| Memory bus width              | 8, 16, 32, 64, 128                          | 32            |
| Memory base address           | 0x00000000-0xFFFFFFFF                       | 0x00000000    |
| Memory address width          | Read only                                   | 21            |
| Memory size needed            | Read only                                   | 2073600 bytes |
| Parameter bus width           | 8, 16, 32                                   | 32            |
| Separate parameter bus clock  | Yes, No                                     | No            |
| Synchronous reset             | Yes, No                                     | No            |
| Output frame size ports       | Yes, No                                     | No            |
| <b>Memory Type</b>            |                                             |               |
| Write FIFO type               | EBR, Distributed                            | EBR           |
| Read FIFO type                | EBR, Distributed                            | EBR           |
| Write FIFO depth              | 32, 64, 128, 256, 512                       | 64            |
| Read FIFO depth               | 32, 64, 128, 256, 512                       | 64            |
| DDR memory burst length       | 2, 4, 8                                     | 8             |
| Command burst count           | 1, 2, 4, 8                                  | 1             |
| <b>Synthesis Options</b>      |                                             |               |
| Frequency constraint (MHz)    | 1-400                                       | 250           |
| Fanout limit                  | 1-200                                       | 100           |
| Resource sharing              | Yes, No                                     | Yes           |
| Pipelining and retiming       | Yes, No                                     | No            |

## Architecture

The Architecture tab provides settings for video frames and algorithm options.

**Figure 3-1. Architecture Tab**



### Frame Dimensions

This section provides settings that define the video format, input/output frame dimensions and dynamic parameter updating.

**Video format** defines the format of video stream. It can be single color, YCbCr4:2:2, YCbCr4:4:4 or RGB.

**Video frame width** and **Video frame height** define the video frame size for buffering.

The **Parallel processing** checkbox determines whether the core processes video color planes in parallel.

The **Dynamic parameter updating** checkbox determines whether the core supports parameters updating at run-time. Refer to [“Dynamic Parameter Updating” on page 14](#) for more information.

The **Frame rate conversion** checkbox enables frame rate conversion. When it is enabled, the output video stream runs on a separate clock.

When dynamic parameter updating is enabled, the **Max video frame width** and **Max video frame height** specify the largest frame size the core needs to support.

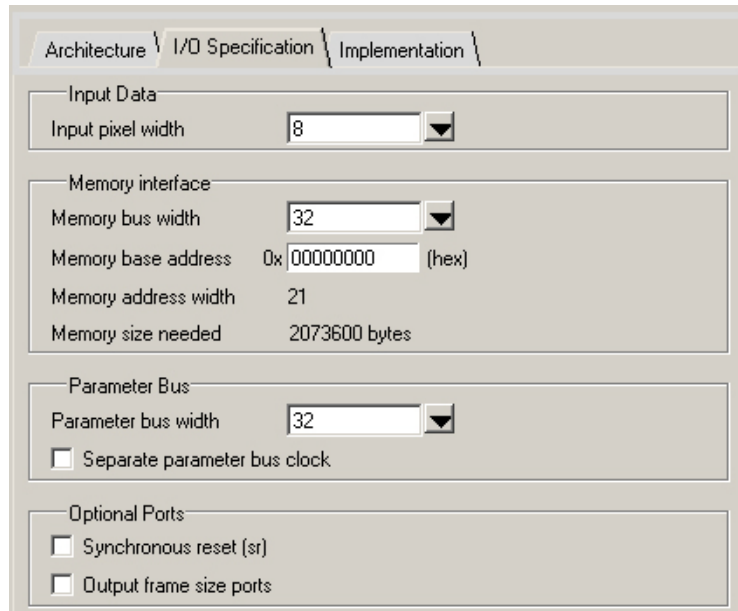
The range for the frame dimensions is 64 to 4096.



## I/O Specification

The I/O Specification tab provides settings for pixel data width, memory bus width, memory base address, parameter bus width and optional ports.

**Figure 3-2. I/O Specification Tab**



**Input pixel width** sets the bit width of the incoming pixel. Values can be 8, 10 and 12. Default value is 8.

**Memory bus width** sets the data width of memory interface. Values can be 8, 16, 32, 64 and 128. Default value is 32.

**Memory base address** sets the base address of the external memory space used. Value must be in hex format and its bit width cannot exceed 32 bits.

**Memory address width** specifies the address width of the memory interface.

**Memory space needed** specifies the size of the external memory space needed.

**Memory address width** and **memory size needed** are generated automatically by the GUI.

**Parameter bus width** sets the bus width of the parameter bus interface. This parameter is only available when dynamic parameter updating is selected. Values can be 8, 16 and 32. Default value is 32.

The **Separate parameter bus clock** checkbox determines whether the core uses a separate clock to run the parameter update interface. This parameter is only available when dynamic parameter updating is selected.

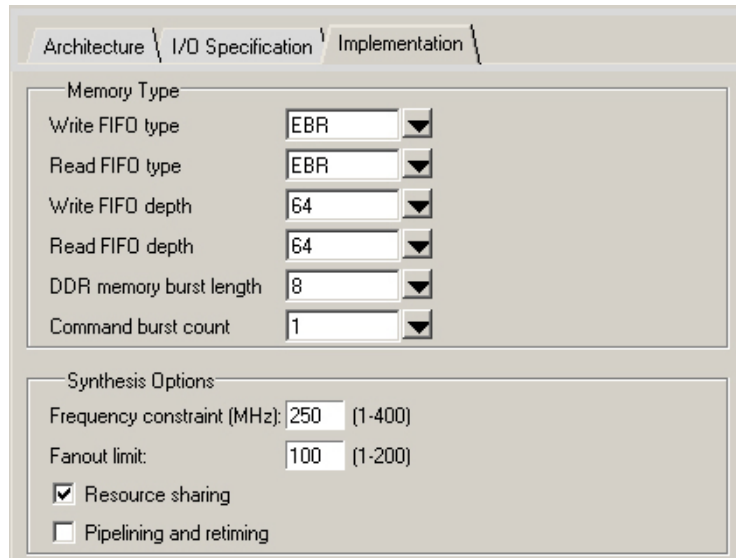
The **Synchronous reset checkbox** determines whether the core has a synchronous reset port.

The **Output frame size ports** checkbox determines whether the core provides output frame size ports. This parameter is only available when dynamic parameter updating is selected.

## Implementation

The Implementation tab provides settings for the memory type and synthesis constraints.

**Figure 3-3. Implementation Tab**



| Category          | Parameter                  | Value                               | Range   |
|-------------------|----------------------------|-------------------------------------|---------|
| Memory Type       | Write FIFO type            | EBR                                 |         |
|                   | Read FIFO type             | EBR                                 |         |
|                   | Write FIFO depth           | 64                                  |         |
|                   | Read FIFO depth            | 64                                  |         |
|                   | DDR memory burst length    | 8                                   |         |
|                   | Command burst count        | 1                                   |         |
| Synthesis Options | Frequency constraint (MHz) | 250                                 | (1-400) |
|                   | Fanout limit               | 100                                 | (1-200) |
|                   | Resource sharing           | <input checked="" type="checkbox"/> |         |
|                   | Pipelining and retiming    | <input type="checkbox"/>            |         |

**Write FIFO type** selects EBR or Distributed RAM for the internal write FIFO type of the frame buffer module. **Read FIFO type** selects EBR or Distributed RAM for the internal read FIFO type of the frame buffer module. **Write FIFO depth** selects the depth of the internal write FIFO and **Read FIFO depth** selects the depth of internal read FIFO. The FIFO depth can be 32, 64, 128, 256 and 512. The default value is 64.

**DDR memory burst length** selects the burst length value. **Command burst count** selects the burst count value for working with the memory controller. The burst length can be 2, 4 and 8. Its default value is 8. The burst count can be 1, 2, 4 and 8. Its default value is 1.

**Frequency constraint** sets the required clock frequency in MHz. This option applies to all the clocks in the core. **Fanout limit** sets the fanout limit value for the synthesis tool. **Resource sharing** and **Pipelining and retiming**, if enabled, are synthesis directives that are used in the core generation. Users can adjust these options to get a better timing result.



# IP Core Generation

This section provides information on how to generate the Lattice Video Frame Buffer IP core using the Diamond IPexpress tool, and how to include the core in a top-level design.

## Licensing the IP Core

An IP core- and device-specific license is required to enable full, unrestricted use of the Lattice Video Frame Buffer IP core in a complete, top-level design. Instructions on how to obtain licenses for Lattice IP cores are given at [www.latticesemi.com/products/intellectualproperty/aboutip/index.cfm](http://www.latticesemi.com/products/intellectualproperty/aboutip/index.cfm).

Users may download and generate the Lattice Video Frame Buffer IP core and fully evaluate the core through functional simulation and implementation (synthesis, map, place and route) without an IP license. The Video Frame Buffer IP core supports Lattice's IP hardware evaluation capability, which makes it possible to create versions of the IP core which operate in hardware for a limited time (approximately four hours) without requiring an IP license. See "Hardware Evaluation" on page 23 for further details. However, a license is required to enable timing simulation, to open the design in Diamond and to generate bitstreams that do not include the hardware evaluation timeout limitation.

## Getting Started

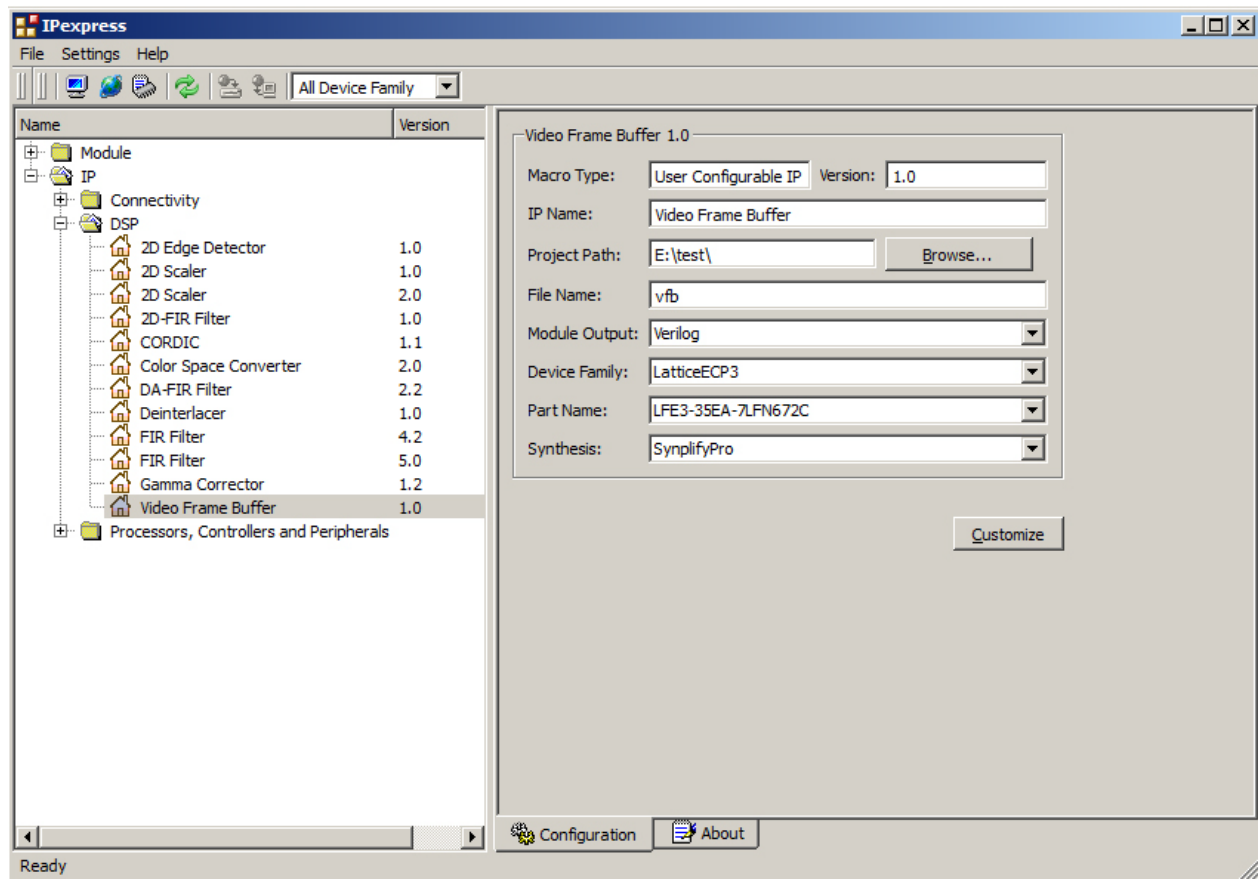
The Video Frame Buffer IP core is available for download from the Lattice IP Server using the IPexpress tool. The IP files are automatically installed using ispUPDATE technology in any user-specified directory. After the IP core has been installed, the IP core will be available in the IPexpress GUI dialog box shown in Figure 4-1. To generate a specific IP core configuration, the user specifies:

- **Project Path** – Path to the directory where the generated IP files will be located.
- **File Name** – "username" designation given to the generated IP core and corresponding folders and files.
- **Module Output** – Verilog or VHDL.
- **Device Family** – Device family to which IP is to be targeted. Only families that support the particular IP core are listed.
- **Part Name** – Specific targeted part within the selected device family.

## Configuring Video Frame Buffer Core in IPexpress

The Video Frame Buffer configuration GUI is accessed via the Diamond IPexpress tool, and provides an interface for setting the desired parameters and invoking the IP core generator. The start-up IPexpress page allows the user to select the IP to be generated, project directory, user-designated module name, design entry type and target device. The File Name is used as the username in the core generation. The Video Frame Buffer IP core is found under **IP > DSP** as shown below.

Figure 4-1. IPexpress Dialog Box

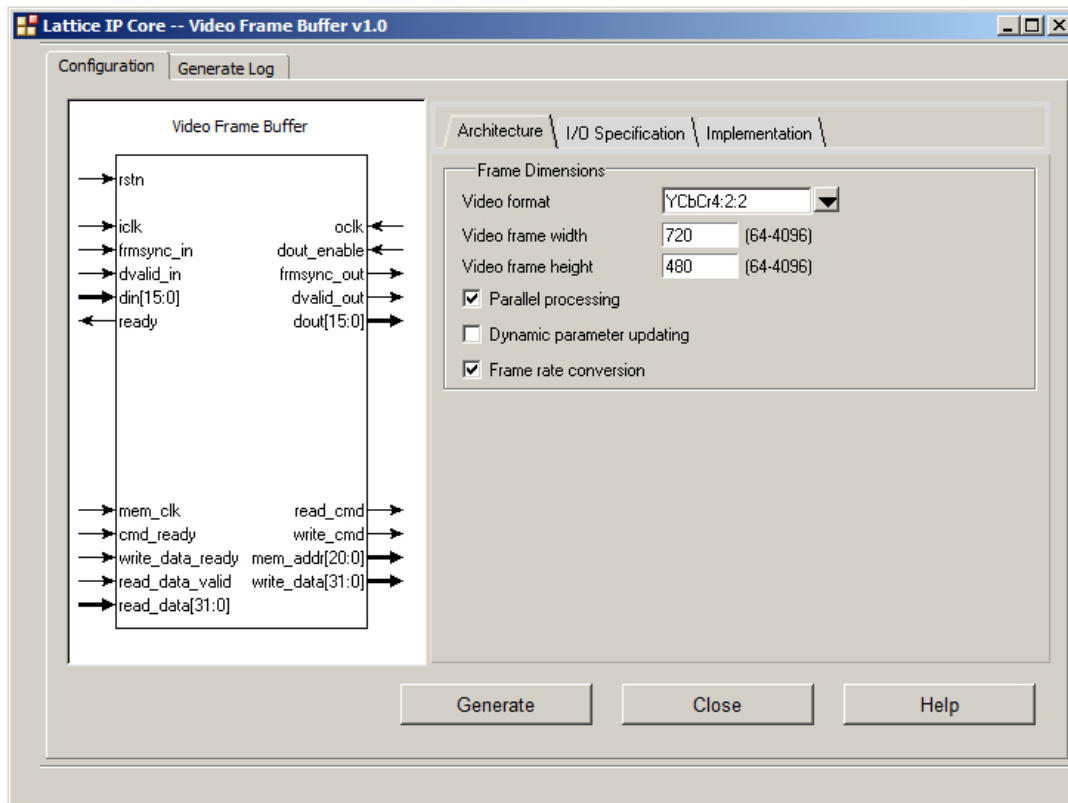


**Important Note:** File Name cannot be “frame\_buffer\_core,” as this name has been used in the internal design of the core.

Note that if the IPexpress tool is called from within an existing project, Project Path, Module Output, Device Family and Part Name default to the specified project parameters. Refer to the IPexpress tool online help for further information.

To create a custom configuration, the user clicks the **Customize** button in the IPexpress tool dialog box to display the Video Frame Buffer IP core Configuration GUI, as shown in Figure 4-2. From this dialog box, the user can select the IP parameter options specific to their application. Refer to “[Parameter Settings](#)” on page 14 for more information on Video Frame Buffer IP core parameter settings.

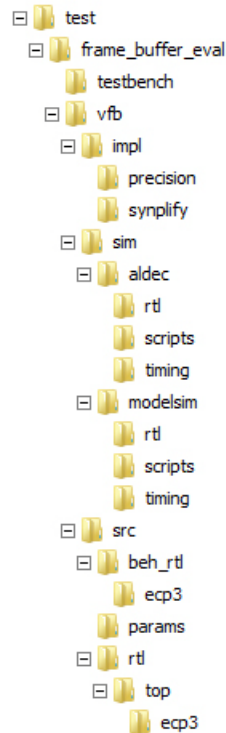
Figure 4-2. Configuration GUI



## IPexpress-Created Files and Top Level Directory Structure

When the user clicks the **Generate** button in the IP Configuration dialog box, the IP core and supporting files are generated in the specified Project Path directory. The directory structure of the generated files is shown in [Figure 4-3](#). This example shows the directory structure generated with the Video Frame Buffer for a LatticeECP3 device.

**Figure 4-3. Video Frame Buffer IP Core Directory Structure**



[Table 4-1](#) provides a list of key files and directories created by the IPexpress tool and how they are used. The IPexpress tool creates several files that are used throughout the design cycle. The names of most of the created files are customized to the user's module name specified in the IPexpress tool.

**Table 4-1. File List**

| File              | Description                                                                                                                                                                                                                                                                                                                                       |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <username>.lpc    | This file contains the IPexpress tool options used to recreate or modify the core in the IPexpress tool.                                                                                                                                                                                                                                          |
| <username>.ipx    | The IPX file holds references to all of the elements of an IP or module after it is generated from the IPexpress tool. The file is used to bring in the appropriate files during the design implementation and analysis. It is also used to re-load parameter settings into the IP/module generation GUI when an IP/module is being re-generated. |
| <username>.ngo    | This file provides the synthesized IP core.                                                                                                                                                                                                                                                                                                       |
| <username>_bb.v   | This file provides the synthesis black box for the user's synthesis.                                                                                                                                                                                                                                                                              |
| <username>_inst.v | This file provides an instance template for Video Frame Buffer IP core.                                                                                                                                                                                                                                                                           |
| <username>_beh.v  | This file provides the front-end simulation library for Video Frame Buffer IP core.                                                                                                                                                                                                                                                               |

[Table 4-2](#) provides a list of key additional files providing IP core generation status information and command line generation capability are generated in the user's project directory.

**Table 4-2. Additional Files**

| File                    | Description                                                                                                 |
|-------------------------|-------------------------------------------------------------------------------------------------------------|
| <username>_generate.tcl | This file is created when the GUI <b>Generate</b> button is pushed. This file may be run from command line. |
| <username>_generate.log | This is the synthesis and map log file.                                                                     |
| <username>_gen.log      | This is the IPexpress IP generation log file.                                                               |

## Instantiating the Core

The generated Video Frame Buffer IP core package includes black-box (<username>\_bb.v) and instance (<username>\_inst.v) templates that can be used to instantiate the core in a top-level design. An example RTL top-level reference source file that can be used as an instantiation template for the IP core is provided in \<project\_dir>\frame\_buffer\_eval\<username>\src\rtl\top. Users may also use this top-level reference as the starting template for the top-level for their complete design.

## Running Functional Simulation

Simulation support for the Video Frame Buffer IP core is provided for the Aldec Active-HDL (Verilog and VHDL) simulator and the Mentor Graphics ModelSim simulator. The functional simulation includes a configuration-specific behavioral model of the Video Frame Buffer IP core. The test bench sources stimulus to the core, and monitors output from the core. The generated IP core package includes the configuration-specific behavior model (<username>\_beh.v) for functional simulation in the Project Path root directory. The simulation scripts supporting ModelSim evaluation simulation is provided in \<project\_dir>\frame\_buffer\_eval\<username>\sim\modelsim\scripts. The simulation script supporting Active-HDL evaluation simulation is provided in \<project\_dir>\frame\_buffer\_eval\<username>\sim\aldec\scripts. Both Modelsim and Active-HDL simulation is supported via test bench files provided in \<project\_dir>\frame\_buffer\_eval\testbench. Models required for simulation are provided in the corresponding \models folder. Users may run the Active-HDL evaluation simulation by doing the following:

1. Open Active-HDL.
2. Under the **Tools** tab, select **Execute Macro**.
3. Browse to folder \<project\_dir>\frame\_buffer\_eval\<username>\sim\aldec\scripts and execute one of the “do” scripts shown.

Users may run the ModelSim evaluation simulation by doing the following:

1. Open ModelSim.
2. Under the File tab, select **Change Directory** and choose the folder <project\_dir>\frame\_buffer\_eval\<username>\sim\modelsim\scripts.
3. Under the **Tools** tab, select **Execute Macro** and execute ..\scripts\<username>\_rtl\_se.do.

*Note: When the simulation is complete, a pop-up window will appear asking “Are you sure you want to finish?” Choose **No** to analyze the results. Choosing **Yes** closes ModelSim.*

## Synthesizing and Implementing the Core in a Top-Level Design

Synthesis support for the Video Frame Buffer IP core is provided for Mentor Graphics Precision RTL or Synopsys Synplify. The Video Frame Buffer IP core itself is synthesized and is provided in NGO format when the core is generated in IPexpress. Users may synthesize the core in their own top-level design by instantiating the core in their top level as described previously and then synthesizing the entire design with either Synplify or Precision RTL synthesis.

The top-level file `<username>_eval_top.v` provided in `\<project_dir>\frame_buffer_eval\<username>\src\top` supports the ability to implement Video Frame Buffer core in isolation. Push-button implementation of this top-level design with either Synplify or Precision RTL Synthesis is supported via the project files `<username>_eval.idf` located in the `\<project_dir>\frame_buffer_eval\<username>\impl\synplify` and the `\<project_dir>\frame_buffer_eval\<username>\impl\precision` directories, respectively.

To use this project file in Diamond:

1. Choose **File > Open > Project**.
2. Browse to `\<project_dir>\frame_buffer_eval\<username>\impl\<synplify or precision>` in the Open Project dialog box.
3. Select and open `<username>.idf`. At this point, all of the files needed to support top-level synthesis and implementation will be imported to the project.
4. Select the Process tab in the left-hand GUI window.
5. Implement the complete design via the standard Diamond GUI flow.

## Hardware Evaluation

The Video Frame Buffer IP core supports the Lattice IP hardware evaluation capability, which makes it possible to create versions of the IP core that operate in hardware for a limited period of time (approximately four hours) without requiring the purchase of an IP license. It may also be used to evaluate the core in hardware in user-defined designs.

Choose **Project > Active Strategy > Translate Design Settings**. The hardware evaluation capability may be enabled/disabled in the Strategy dialog box. It is enabled by default.

## Updating/Regenerating the IP Core

By regenerating an IP core with the IPexpress tool, you can modify any of its settings including device type, design entry method, and any of the options specific to the IP core. Regenerating can be done to modify an existing IP core or to create a new but similar one.

### Regenerating an IP Core

To regenerate an IP core:

1. In IPexpress, click the **Regenerate** button.
2. In the Regenerate view of IPexpress, choose the IPX source file of the module or IP you wish to regenerate.
3. IPexpress shows the current settings for the module or IP core in the Source box. Make your new settings in the Target box.
4. If you want to generate a new set of files in a new location, set the new location in the IPX Target File box. The base of the file name will be the base of all the new file names. The IPX Target File must end with an `.ipx` extension.
5. Click **Regenerate**. The module's dialog box opens showing the current option settings.
6. In the dialog box, choose the desired options. To get information about the options, click **Help**. Also, check the **About** tab in IPexpress for links to technical notes and user's guides. The IP core may come with additional information. As the options change, the schematic diagram of the module changes to show the I/O and the device resources the module will need.



7. To import the module into your project, if it's not already there, select Import **IPX to Diamond Project** (not available in stand-alone mode).
8. Click **Generate**.
9. Check the **Generate Log** tab to check for warnings and error messages.
10. Click **Close**.

The IPexpress package file (.ipx) supported by Diamond holds references to all of the elements of the generated IP core required to support simulation, synthesis and implementation. The IP core may be included in a user's design by importing the .ipx file to the associated Diamond project. To change the option settings of a module or IP that is already in a design project, double-click the module's .ipx file in the File List view. This opens IPexpress and the module's dialog box showing the current option settings. Then go to step 6 above.



# Support Resources

This chapter contains information about Lattice Technical Support, additional references, and document revision history.

## Lattice Technical Support

There are a number of ways to receive technical support.

### E-mail Support

[techsupport@latticesemi.com](mailto:techsupport@latticesemi.com)

### Local Support

Contact your nearest Lattice sales office.

### Internet

[www.latticesemi.com](http://www.latticesemi.com)

## References

- HB1009, [LatticeECP3 Family Handbook](#)
- HB1003, [LatticeECP2/M Family Handbook](#)
- HB1004, [LatticeXP2 Family Handbook](#)

## Revision History

| Date           | Document Version | IP Core Version | Change Summary                                                                                           |
|----------------|------------------|-----------------|----------------------------------------------------------------------------------------------------------|
| March 2012     | 01.0             | 1.0             | Initial release.                                                                                         |
| June 2012      | 01.1             | 1.0             | Appendix A, Resource Utilization – Corrected ordering part numbers.                                      |
| September 2013 | 01.2             | 1.2             | Added information in the Memory Interface section. Provided the scheme on improving data bus throughput. |
|                |                  |                 | Updated Lattice Technical Support information.                                                           |

# Resource Utilization

This appendix gives resource utilization information for Lattice FPGAs using the Video Frame Buffer IP core.

IPexpress is the Lattice IP configuration utility, and is included as a standard feature of the Diamond design tools. Details regarding the usage of IPexpress can be found in the IPexpress and Diamond help systems. For more information on the Diamond design tools, visit the Lattice web site at [www.latticesemi.com/software](http://www.latticesemi.com/software).

## LatticeECP3 Devices

**Table A-1. Performance and Resource Utilization<sup>1</sup>**

| Video Format | Frame Width | Frame Height | Parallel Processing | Pixel Width | Memory Bus Width | Registers | LUT4s | EBR | Fmax (iclk) | Fmax (mem_clk) | Fmax (oclk) |
|--------------|-------------|--------------|---------------------|-------------|------------------|-----------|-------|-----|-------------|----------------|-------------|
| YCbCr422     | 720         | 480          | No                  | 8           | 32               | 771       | 941   | 2   | 333         | 292            | 291         |
| RGB          | 1024        | 768          | Yes                 | 8           | 32               | 838       | 1093  | 2   | 309         | 272            | 277         |
| YCbCr422     | 1920        | 1080         | Yes                 | 8           | 32               | 1095      | 1349  | 2   | 293         | 283            | 289         |

1. Performance and utilization data are generated targeting a LFE3-35EA-8FN672C device using Lattice Diamond 1.4 and Synplify Pro F-2011.09L software. Performance may vary when using a different software version or targeting a different device density or speed grade within the LatticeECP3 family.

## Ordering Part Number

The Ordering Part Number (OPN) for Video Frame Buffer IP core targeting LatticeECP3 devices is VFB-E3-U.

## LatticeECP2M and LatticeECP2MS Devices

**Table A-2. Performance and Resource Utilization<sup>1</sup>**

| Video Format | Frame Width | Frame Height | Parallel Processing | Pixel Width | Memory Bus Width | Registers | LUT4s | EBR | Fmax (iclk) | Fmax (mem_clk) | Fmax (oclk) |
|--------------|-------------|--------------|---------------------|-------------|------------------|-----------|-------|-----|-------------|----------------|-------------|
| YCbCr422     | 720         | 480          | No                  | 8           | 32               | 774       | 982   | 2   | 330         | 268            | 280         |
| RGB          | 1024        | 768          | Yes                 | 8           | 32               | 834       | 1097  | 2   | 281         | 269            | 303         |
| YCbCr422     | 1920        | 1080         | Yes                 | 8           | 32               | 1093      | 1365  | 2   | 284         | 259            | 280         |

1. Performance and utilization data are generated targeting a LFE2M35E-7F672C device using Lattice Diamond 1.4 and Synplify Pro F-2011.09L software. Performance may vary when using a different software version or targeting a different device density or speed grade within the LatticeECP2M family.

## Ordering Part Number

The Ordering Part Number (OPN) for Video Frame Buffer IP core targeting LatticeECP2M/S devices is VFB-PM-U.

## LatticeECP2 and LatticeECP2S Devices

**Table A-3. Performance and Resource Utilization<sup>1</sup>**

| Video Format | Frame Width | Frame Height | Parallel Processing | Pixel Width | Memory Bus Width | Registers | LUT4s | EBR | Fmax (iclk) | Fmax (mem_clk) | Fmax (oclk) |
|--------------|-------------|--------------|---------------------|-------------|------------------|-----------|-------|-----|-------------|----------------|-------------|
| YCbCr422     | 720         | 480          | No                  | 8           | 32               | 774       | 982   | 2   | 322         | 286            | 264         |
| RGB          | 1024        | 768          | Yes                 | 8           | 32               | 834       | 1097  | 2   | 290         | 289            | 283         |
| YCbCr422     | 1920        | 1080         | Yes                 | 8           | 32               | 1093      | 1365  | 2   | 273         | 277            | 294         |

1. Performance and utilization data are generated targeting a LFE2-35E-7F672C device using Lattice Diamond 1.4 and Synplify Pro F-2011.09L software. Performance may vary when using a different software version or targeting a different device density or speed grade within the LatticeECP2 family.

## Ordering Part Number

The Ordering Part Number (OPN) for Video Frame Buffer IP core targeting LatticeECP2/S devices is VFB-P2-U.

## LatticeXP2 Devices

**Table A-4. Performance and Resource Utilization<sup>1</sup>**

| Video Format | Frame Width | Frame Height | Parallel Processing | Pixel Width | Memory Bus Width | Registers | LUT4s | EBR | Fmax (iclk) | Fmax (mem_clk) | Fmax (oclk) |
|--------------|-------------|--------------|---------------------|-------------|------------------|-----------|-------|-----|-------------|----------------|-------------|
| YCbCr422     | 720         | 480          | No                  | 8           | 32               | 774       | 982   | 2   | 289         | 240            | 250         |
| RGB          | 1024        | 768          | Yes                 | 8           | 32               | 834       | 1097  | 2   | 260         | 254            | 253         |
| YCbCr422     | 1920        | 1080         | Yes                 | 8           | 32               | 1093      | 1365  | 2   | 244         | 240            | 254         |

1. Performance and utilization data are generated targeting a LFXP2-30E-7F672C device using Lattice Diamond 1.4 and Synplify Pro F-2011.09L software. Performance may vary when using a different software version or targeting a different device density or speed grade within the LatticeXP2 family.

## Ordering Part Number

The Ordering Part Number (OPN) for Video Frame Buffer IP core targeting LatticeXP2 devices is VFB-X2-U.