Features of the VIC068A VMEbus Interface Controller

This application note describes some features of the Cypress VIC068A and provides information on how to use the device.

The VIC068A was designed by a consortium of VMEbus manufacturers in partnership with Cypress. The major goals of this consortium were to achieve a standardized, reasonably priced VMEbus interface that was not dominated by any board manufacturer. Manufacturing this specialized chip requires a high-speed process (125 MHz) and high-power I/O pins (64 and 48 mA).

The VIC068A adheres to the ANSI/IEEE Standard 1014, which minimizes the problems of interfacing among the VMEbus boards of various manufacturers. A block diagram detailing the device's functional blocks appears in *Figure 1*.

VIC068A Highlights

With very precise timing, based on a 64-MHz clock that is used internally to make decisions on 8-ns intervals, you can reach the theoretical limits of the VMEbus transfer rates—a block transfer rate of 40 Mbytes/s.

Because all logic resides in a single chip, the VIC068A greatly reduces the board space necessary to interface to the VMEbus. Even a highly sophisticated interface with an A32/D32 system controller and block transfer support requires no more than 60 square cm or 20 percent of a double eurocard (6U card).

Special care has been taken to speed up the VIC068A's VMEbus access. Although many of today's CPU boards use megabytes of high-speed local RAM to limit the number of VMEbus accesses, the accesses that do occur for I/O or data reads and writes must be done efficiently to avoid slowing the rest of the system.

For both types of data transfers, the VIC068A offers special support. For single-write cycles, you can program the VIC068A to operate in the so-called master or slave write-posting mode (*Figure 2*), the local VMEbus write cycle is terminated locally as soon as data is latched in the VMEbus latches. This allows the local CPU to continue with instruction fetches or other operations while the VIC068A transfers data over the VMEbus.

In slave write-posting mode (*Figure 3*), the same function happens with write cycles form the VMEbus to the local bus. As soon as the data is latched, the VMEbus cycle is terminated and the local cycle can finish independently of further VMEbus traffic. Both modes reduce CPU overhead and VMEbus utilization, providing higher bandwidth in single-cycle writes.

The VMEbus prohibits a similar function in single-cycle reads because every read cycle on the VMEbus could turn out to be a read-modify-write (RMW) cycle. This cannot be foreseen because the only difference is that address strobe is held low between the two cycles. Therefore, if the VMEbus address strobe were released during the two portions of the same RMW cycle, another VMEbus master could break into that cycle and modify the same data.

To move blocks of data over the VMEbus, the VIC068A uses the block transfer mode. In its standard form, this mode allows a processor to transfer up to 256 bytes with just one starting address supplied to the VMEbus. Additionally, the VIC068A uses a type of pipelining to accelerate VMEbus throughput. On a block transfer read cycle, the slave VIC068A automatically prefetches the n + 1 data byte during the same read. The nth data byte is transferred across the VMEBus, and the n – 1 byte is latched in local RAM. As shown in *Figure 4*, this operation uses all three buses in overlapped and parallel operation to speed up the transfer. Write transfers use the same mechanism.

The limiting factor on the VMEbus transfer rate is either the VMEbus's many timing restrictions or the source or destination memories. If the memory consists of dynamic RAM, the restriction is probably the cycle time of the chips used, often as slow as 200 ns. To overcome this limitation, the VIC068A offers a programmable access mode so that attached DRAM can be used in page mode.

After a starting row address cycle (RAS), all subsequent cycles need only a column address (CAS) to reduce the access time, often by as much as half. For a slave interface, the VIC068A contains all the necessary counters and timing elements for local AS, DS, and address generation.

A master block transfer needs two or three additional latches for the higher address lines during the local DMA part of the block transfer. Thus, even with low-cost DRAMs, the VIC068's block transfer rate can reach 40 Mbytes/s, limited only by the VMEbus specification and the physical characteristics of the VMEbus.

This transfer rate decreases the time needed to load programs or move data to graphics boards, as well as increasing the VMEbus's bandwidth, thereby allowing more CPUs. to work together in a multiprocessor system.

Mailbox Signaling

To add greater capability to multiprocessor systems, the VIC068A has four interprocessor communication global switches (ICGS) and four interprocessor communication module switches (ICMS). These are all byte-wide mailbox registers that generate a local interrupt when accessed from the VMEbus. The ICGSs of one group reside at the same address and are accessed with a write cycle, which behaves as a broadcast to all members of the group. Because the ICMSs are at different addresses, one dedicated processor can be activated with a local interrupt request (LIRQ).

A processor can inform a logical group of processors about a new task via a broadcast using the ICGSs and can then communicate with single processors about the task using the ICMSs.

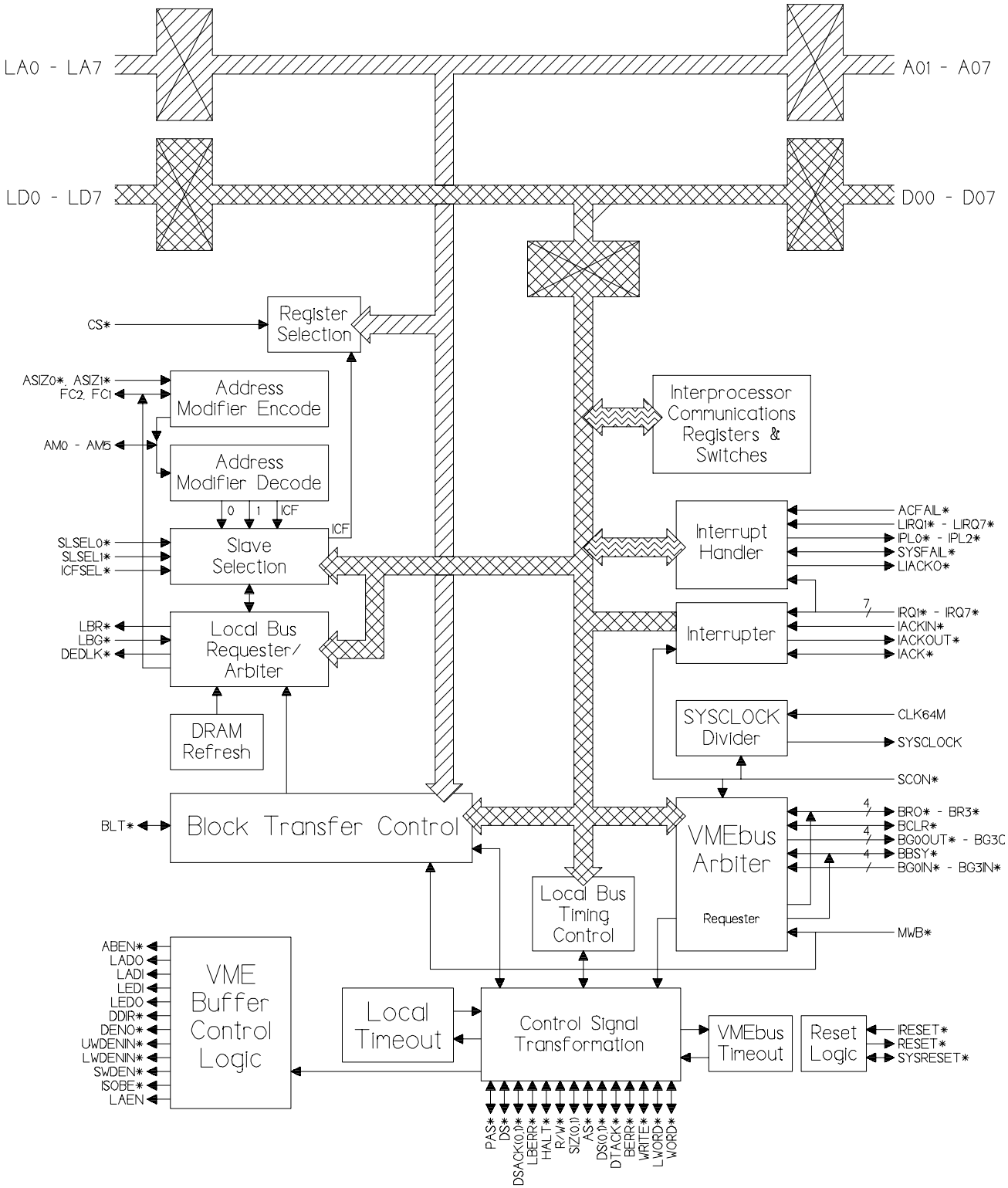

Figure 1. VIC068A Functional Block Diagram

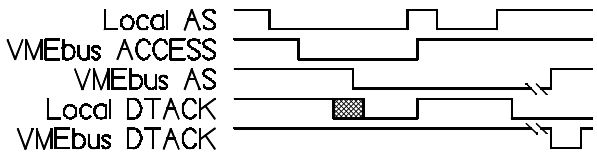

Figure 2. Master Write Posting

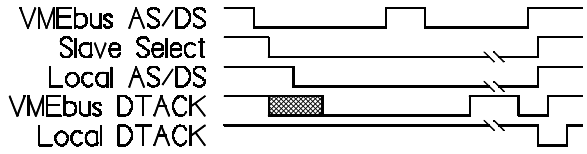

Figure 3. Slave Write Posting

Eight-byte-wide interprocessor communication registers (ICR) are also available. Five of these registers serve as general-purpose read/write registers, and three are dedicated to control local activities (Halt, Reset, Mask ID, etc.). The ICRs can be read and written from the local side or the VMEbus without interfering with each other.

Interrupt Generator

The VIC068A handles up to seven simultaneous pending IRQs with separate vectors. The VIC068A also provides independent local IRQ vectors, if external IRQs are served.

Miscellaneous Features

The VIC068A furnishes several features for VMEbus support:

- SYSFAIL generation
- Software reset
- ACFAIL
- BERR register for detailed information

For local support, the VIC068A provides these features:

- Seven local IRQ sources, all level, polarity, edge and vector programmable
- Local bus timeout (2–512 ms)
- With /without VMEbus request time included
- 21 different local IRQ vectors
- VIC ID register

In addition to the VIC068, the following parts or equivalents are required for a minimum hardware interface:

- Three address latches and drivers (74xx543)
- Three data latches and drivers (74xx543)
- Four isolation buffers (74xx245)

You might also need the following:

- One to two PLDs for slave address decoding
- Two to three latches for master block transfer
- 1⁄2 PLD for block transfer glue logic

Interfacing

To connect a processor other than the 680x0 to the VIC068A, it is often easiest to map the processor control signals into the control signals available on a 680x0-type of processor. This type of transition interface offers the advantage of compatibility with a large family of 680x0-compatible peripheral parts, which you can then use elsewhere in the design.

Figure 5 shows a sample interface, whose four address latches store the multiplexed Mbus of the MC88000 processor. Four data latches store the data bytes after the acknowledge of the 680x0 bus and then start calculating parity the processor's MBus. The reason for this approach lies in some older peripheral I/O chips, which change their data lines when they should remain stable (i.e., transmit data buffer empty, etc.).

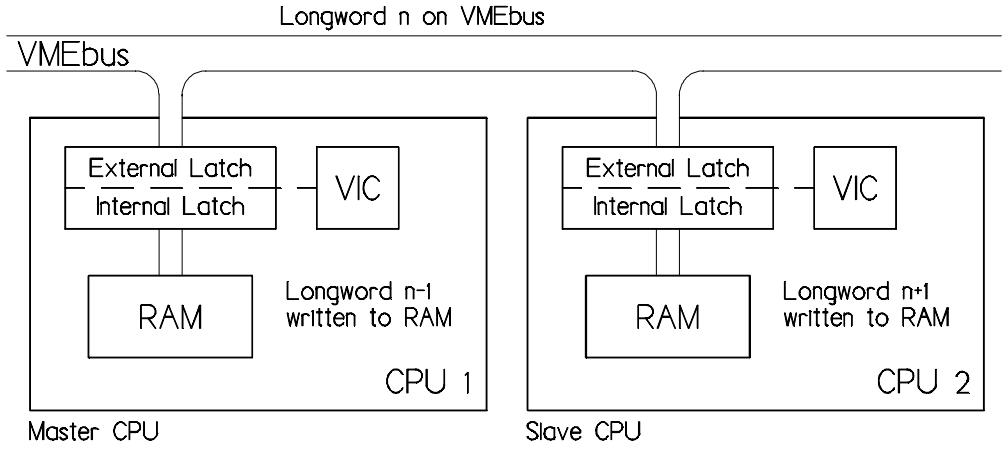

Figure 4. Block Transfer Read Cycle

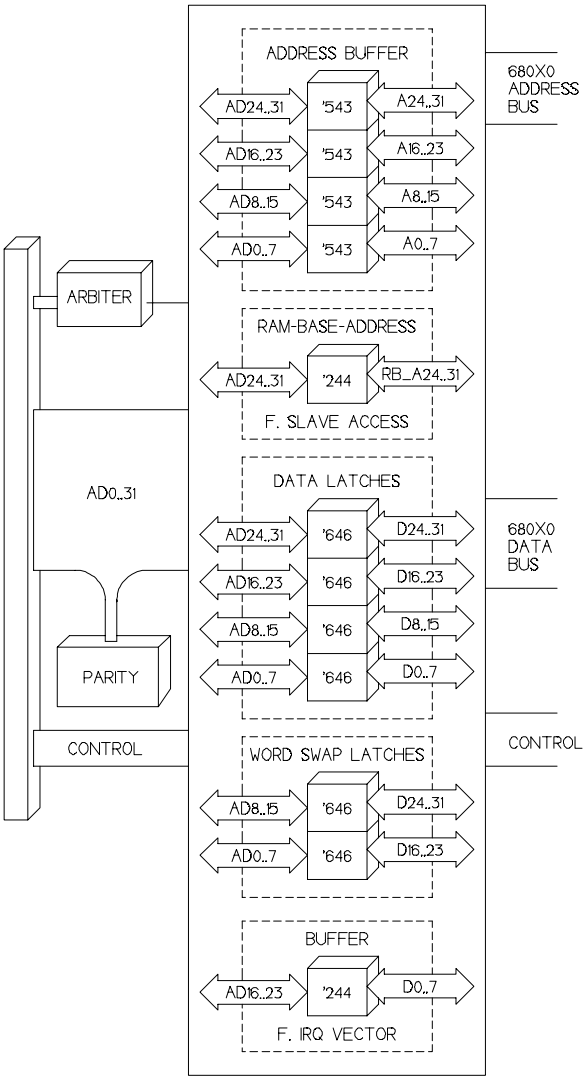

Figure 5. Sample Interface

Two other data latches emulate the MC68020's dynamic bus sizing. The last buffer, between D0–D7 of the 680x0 bus and AD16–AD23 of the MBus, emulates the 680x0 bus's IRQ cycles with normal read cycles of the MC88000.

Acknowledgment

Cypress Semiconductor wishes to thank Jürgen Bullacher of Eltec GMbH and Eltec International S.A.R.L. for submitting this article.