3.10
Design Considerations

3.10.1 Design Philosophy

The CY7C960 and the CY7C961 each share the basic VMEbus slave circuitry, and each has a common design philosophy. The most basic foundation for the design was that of achieving high performance in whatever system the devices were placed. For the CY7C960, that philosophy determined the hierarchical partitioning of the design into several independant blocks, each with its own state machine. The coexistence of these state machines demanded a fully-synchronous internal design in order to remove the neccesity for block-to-block synchronization circuitry, or pipelining. Thus the internal operations are all designed to take place within one clock period, so that results from one state transition are available to other blocks in the cycle that they occur. No “analog” delays are employed in the design, and the standard VMEbus timings are determined by counting clock periods internally. The clock input frequency that the two parts expect to see is 80 MHz: at this frequency all the VMEbus timings are assured. The design is fully static, and the clock frequency must therefore be considered when specifying the configuration of the application. All parameters that are a function of clock period are specified as such (e.g., 3T + 4 nsec) in the AC timing tables (Chapter 3.12).

There is a comprehensive set of timing diagrams in Chapter 3.12 that should be consulted to determine the relationship of the various signals. Where possible, there are AC timing parameters specified in *Table 3–16*. In general, however, the diagrams will give a better understanding of the operation of the parts than a table of numbers.

3.10.2 CY7C964 Interface

The operation of the interface is described in Chapter 3.6, and in Section 4, The CY7C964 Bus Interface Logic Circuit. The background to this section may be of interest.

The first controller produced by Cypress for the VMEbus community was the ubiquitous VIC068A—the current industry-standard controller. This device was intended to be used in concert with the state of the art in low-cost drivers—then "ACT" logic devices such as '543's and '245's. The control outputs from the VIC068A were crafted to interface cleanly with these devices. Thus LEDI, LEDO, DENIN*, DENO*, etc from the VIC068A were signals whose timing was designed to turn the drivers on or off, and latch the data and address buses, at appropriate times for the VMEbus transfers. When the VIC64 was introduced it inherited the identical timings, because the VIC64 was socket-compatible with the VIC068A. The CY7C964 was designed as a glueless companion to VIC64, replacing the ACT logic with a higher level of integration to allow the increased functionality of the VME64 specification to be implemented in similar board space. So it also inherited the control signal timing of the VIC068A.

Several of the control signals perform double duty, such as LDS and MWB*, which are used during the comparator load sequence in the CY7C964. But in general, the signals controlling the '964 are the original buffer control signals from the VIC068A.

Therefore, when the CY7C960 and '961 were designed, they had to conform to the original control timing imposed by the VIC068A in order to use the popular CY7C964. Therefore the control signals have the same names, functions, and timings as the original VIC068A and the VIC64. One advantage of this is that applications that do not require the sophistication of the VME64 specification are not forced into employing the sophisticated CY7C964—the control signals work with the FCT family of devices that are also offered by Cypress. These devices allow simpler applications to achieve cost goals without sacrificing performance. The limitation, as should be expected, is that the more complex VMEbus transactions are not implemented unless the CY7C964 is employed.

3.10.3 Local Bus Philosophy

The CY7C960 and '961 are very flexible in the way they deal with local circuitry. However, they are not intended to compete with their more sophisticated cousins, VIC64 and VIC068A. Whereas VIC068A and VIC64 have a local bus arbitration circuit, on-chip DMA engine, and local interrupt circuitry, the '960 and '961 are conceptually much simpler. On the surface, the pinout reveals a simple local acknowledge handshake, and a local interrupt input. The '961 also provides a local bus error function. Beneath the surface, there is anoth-

er level of complexity that can be invoked by applications that need somewhat more local bus control.

The CY7C960 was intended to be the highest priority on the local bus. In other words, the ‘960 assumes that when a VMEbus slave transaction occurs, nothing will prevent it from reading or writing local transactions other than the local acknowledge handshake. The local cycle starts with no regard for other masters which may be accessing local resources. This optimizes VMEbus performance, preventing VMEbus cycles being extended by local bus contention. However, this philosophy is not beneficial in all cases. Some rudimentary control over the local bus may be needed from time to time by other devices.

To cater to this situation the CY7C960 can be prevented from starting a local cycle, or a refresh of local DRAM. To explain this operation, first consider the VMEbus activity. Whenever AS* is asserted by a VMEbus master, the CY7C960 will drive RAS* Low, and the VMEbus address is driven onto the local address bus by the CY7C964s. This happens whether the CY7C960 is addressed or not, in order to optimize performance. The problem for local bus ownership is that there is no way to predict when AS* will go active. Hence the only way to guarantee that AS* is not about to go Low, is to watch it go High. Then the AS* minimum High time provides a window of opportunity for the local master to capture the local bus from the CY7C960. The local signal which can be used to monitor AS* going High is LADI. This signal also indicates DRAM refresh activity, thus preventing a conflict between the local bus master and a local DRAM refresh cycle. The local acknowledge signal is used to capture the local bus: if enabled to do so by a bit in the configuration sequence, the LACK pin prevents the start of a ‘960 local cycle. If LACK is driven High (or is High already) when the window described earlier occurs, then no local signals are driven by the ‘960 whatever happens on the VMEbus. Control signals to the ‘964 are driven to cause the least significant ‘964 to be three-stated and the local bus master can now perform local cycles without fear of intervention or DRAM refresh. LAEN of the remaining ’964s must be controlled by external logic. The local bus is given back to the CY7C960 upon the assertion of LACK (Low).

In some applications, VMEbus traffic may not provide enough opportunities—AS* may not cycle frequently. In such a case, the local master can still acquire the local bus by driving LACK High, waiting a period of time (2T) then examining LADI . If LADI is still inactive then the local bus is available. If LADI is active, then the VMEbus transaction has begun and the ‘960 is about to start a local bus access.

Note that, while the local bus is not available to the CY7C960, any VMEbus accesses to the CY7C960 will hang until the local bus is again available. Also, the refresh engine will be unable to refresh the DRAM; when LACK is driven Low, the refresh engine has priority and will burst all the missed refresh cycles (256 bursts) before the CY7C960 will respond to a VMEbus cycle.

Note that this function is enabled by a bit in the configuration bit stream. If the bit is not enabled, then the CY7C960 cannot be prevented from DRAM refresh, or from starting a local cycle. The function of LACK is then simply to control the completion of local cycles allowing for slow or asynchronous local peripherals.

3.10.4 Read-Ahead Cycles

The CY7C960 does not read ahead single-cycle transfers, but it does perform read-ahead cycles when acting as a slave to a VMEbus BLT read operation. There are two ways to prevent CY7C960 performing the read-ahead, both involving the VMEbus Master: terminate the block transfer at a 256-byte address boundary; or remove AS* with the final DS* deassertion.

Read-ahead is implemented to allow a higher performance to be achieved, as otherwise the slave board will be unable to provide data within the specified timing for minimum transaction period.

There are considerations when designing FIFO-based slave circuitry in order to guarantee that the master will be given an error-free block transfer. The FIFO going empty during the progress of a block transfer Read will not be a problem, assuming that the circuitry external to CY7C960 intercepts the read strobe until the FIFO has data, and does not acknowledge the local cycle until the data has in fact been transferred to the CY7C960. The problem occurs when a data block is desired that is not aligned to 256-byte address boundaries. The final transfer of desired data will be followed by a read-ahead cycle which will, unless intercepted in some way, clock exactly one more data location from the FIFO.

In most applications this will not be a problem, as the system design is usually organized so that the FIFO goes empty after the final desired data byte has been read, and the following read-ahead will not be relevant. Once AS* has gone inactive on the VMEbus, the CY7C960

understands that the data transfer is complete. It drives all local signals inactive, whether or not the cycle has been locally acknowledged. This effectively discards the read-ahead.

In some cases the FIFO may contain relevant data at all times, and a read-ahead cycle would result in loss of data. These applications must either ensure that the data is aligned to 256 byte address boundaries, or ensure that AS* and DS* go inactive together, thereby ensuring that the final cycle is not followed by a read-ahead.

3.10.5 Write Posting

A similar consideration is inherent in the BLT Write operation. A common system configuration on VMEbus is that the slave "posts" the write data and acknowledges the VMEbus cycle prior to the local acknowledge. This optimizes performance: without BLT write posting, performance is substantially degraded, approximating that of a well-designed single cycle transfer.

During a BLT Write transfer, if the local circuitry detects an error or is unable to complete the transaction, then the local circuitry will not acknowledge the posted data. This causes the VMEbus to extend the current cycle (the one immediately AFTER the posted transaction) and the system timer will timeout, causing a BERR* and allowing the Master to perform error recovery.

In the event that the final local cycle of a block does not terminate correctly, there is no mechanism for communicating this error to the Master. The cycle has already been DTACKed, and the bus ownership has been relinquished. This is a problem common to all VMEbus Slave Write Posting applications.

The CY7C960 posts all BLT write transactions. If the system designer has a problem with write posting due to the last-cycle issue, then single cycle transfers must be used. CY7C960 does not post single cycle write data.

3.10.6 VMEbus Error Considerations

Because of pinout limitations the CY7C960 does not monitor or drive BERR*, the VMEbus error handshake. In the case where the VMEbus cycle terminates with a BERR*, indi-

cating that the local data was in fact needed but was not provided before the VMEbus timer expired, CY7C960 cannot see BERR*, but still sees AS* go inactive. The CY7C960 always terminates the local cycle in response to AS* going inactive. In this case it is the responsibility of the Master to take action following the BERR*. The CY7C960 continues to operate normally