# CorePCI Target+DMA Master 33/66MHz

## Product Summary

### Intended Use

- High-Performance 33MHz or 66MHz PCI Target+DMA Master Applications
  - 32-Bit, 33MHz, 3.3V or 5V fully compliant solution available
  - Advanced 32-bit, 66MHz, 3.3V solution available (A54SX72 required)
- Back-End Support for Synchronous DRAM, I/O Subsystems, and SRAM

### Major Options

- Support for I/O Space
- User-Selectable Memory and I/O Address Space Size
- Interrupt Capability

### Data Transfer Rates

- Fully Compliant Zero Wait State Burst (32-Bit Transfer Each Cycle)
- Paced Burst (Wait states injected between transfers)
- Data Prefetch with Automatic Recovery for FIFOs

### Targeted Devices

- 54SX Family: A54SX16P, A54SX32AP
- 42MX Family A42MX24, A42MX36

### Design Source Provided

- VHDL or Verilog-HDL Design Source
- Actel-Developed VHDL Test Bench

### Synthesis and Simulation Support

- Synthesis: Synplicity and Synopsys FPGA Compiler
- Simulation: Vital-Compliant VHDL Simulators and OVI Compliant Verilog Simulators

### Macro Verification

- Actel-Developed Test Bench

### Compliance

- I/O Drive Compliant in Targeted Devices
- Compliant with the PCI 2.1 Specification. Compliance with 2.2 planned.

### Version

This data sheet defines the functionality of Version 5.0 pre-production macro.



## General Description

The CorePCI Target+DMA Master connects I/O, memory, and processor subsystem resources to the main system via the PCI bus. The specific implementation of the CorePCI Target+DMA macro is intended for use with a wide variety of peripherals where high-performance data transactions are required. Figure 1 shows typical system applications using the baseline macro. The CorePCI Target+DMA provides a generic set of back-end signals. This generic interface forms a bridge to specific back-end controllers like SDRAM, SRAM, and FIFO.

The CorePCI Target+DMA Master has been developed for specific devices in the 54SX and 42MX families from Actel. The targeted devices are the A54SX16P, which is a 3.3V PCI compliant device, and the A42MX24, which is a 3.3V or 5V PCI compliant device. The CorePCI Target+DMA Master can handle any transfer rate. However, most applications demanding high data through put will operate at zero wait states. When required, wait states can be automatically inserted by a slower peripheral.

The core consists of three basic pieces: the Target+DMA Master, the back-end, and the wrapper. The Target+DMA Master controller remains constant for a variety of back-ends. A back-end controller provides the necessary control for the I/O, or memory subsystem and interfaces to the Target+DMA Master controller through a generic interface. The wrapper combines the Target+DMA Master block and the back-end for implementation in a single Actel device.

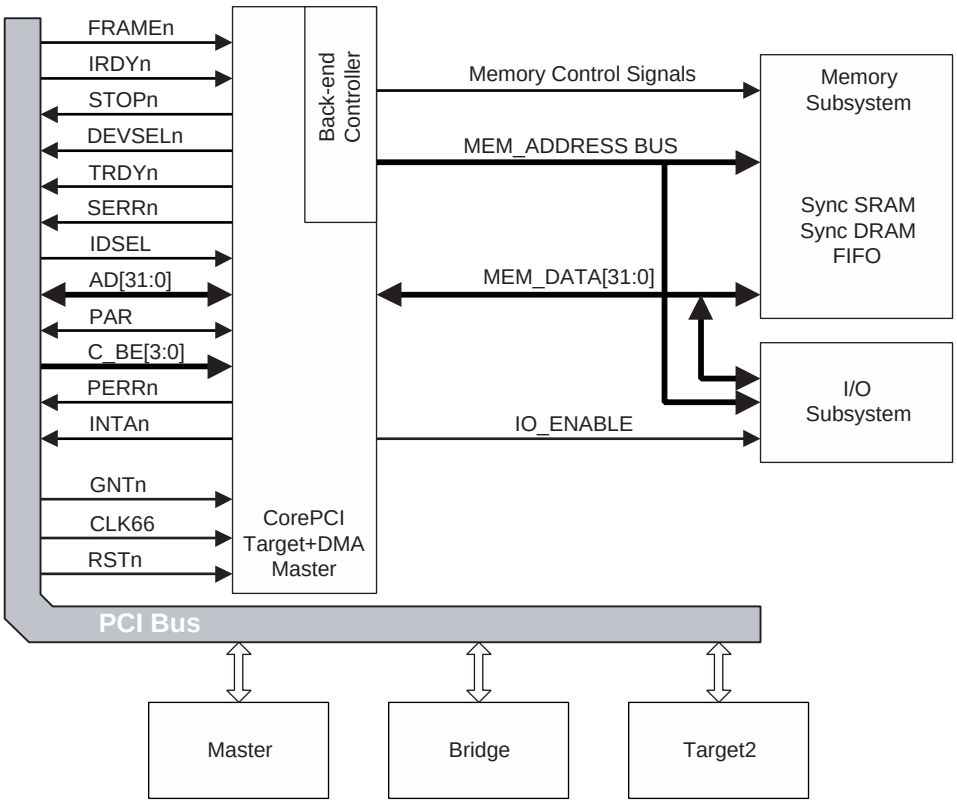



***Figure 1 •*** *System Block Diagram Depicting Target+DMA Macro Usage*

The CorePCI Target+DMA can be customized in two different ways. In the first method, a variety of variables are provided to easily change parameters such as memory and I/O sizes.The second method is to develop user specific back-end controllers for non-standard peripherals.

## Functional Block Diagram of Target+DMA Macro

The CorePCI Target+DMA macro consists of six major functional blocks, shown in Figure 2. These blocks are the DMA state machine, the address phase state machine, the dataphase state machine (Burst), datapath, parity, and the configuration block.

### DMA State Machine

The DMA state machine is responsible for gaining master ownership of the PCI bus and launching a data transfer transaction by asserting frame. Once a burst transaction has begun, the DMA state machine tracks the transfer count and terminates the burst by de-asserting the frame signals and releasing master ownership of the PCI bus.

### Address Phase State Machine

The address phase state machine is responsible for determining if the PCI bus is addressing the Target controller. When a hit is detected, the DP_START signal is activated, setting off the dataphase machine and back-end logic. The address phase state machine also determines the cycle type and expresses this information on the RD_CYC, WR_CYC, MEM_CYC, IO_CYC, and CONFIG_CYC outputs.

### Dataphase State Machine

The dataphase state machine is responsible for controlling the PCI output signals and coordinating the data transfers with the back-end logic. The PCI outputs are TRDYn, DEVSELn, and STOPn during target transfers. During master transfers, the dataphase state machine drives the IRDYn signal. Data transfers to the back-end are coordinated using the signals BE_RD_RDY, RD_BE_NOW, BE_WR_RDY, and WR_BE_NOW[3:0]. The “RDY” inputs indicate that the back-end is ready to transmit or receive data. The “NOW” signals indicate that a data transfer will occur on the next rising edge of the clock. The dataphase state machine also drives the DP_DONE output active at the end of the PCI transfer.

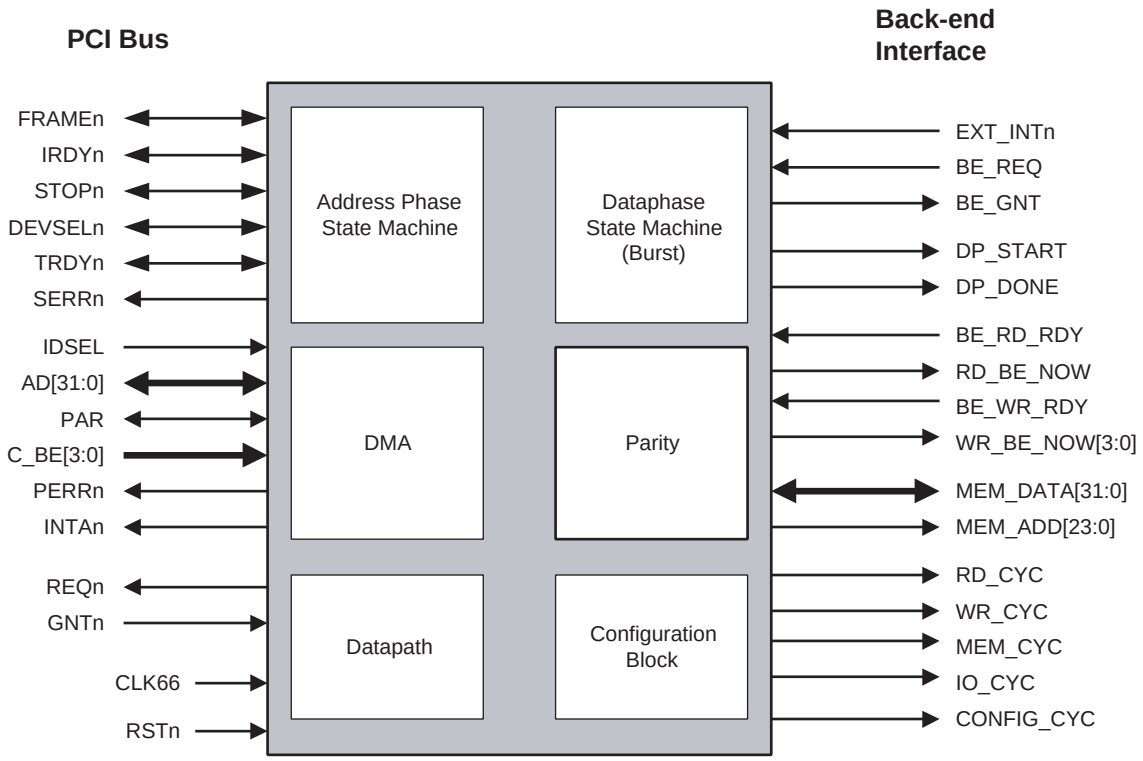



***Figure 2 •*** *Block Diagram of the CorePCI Target+DMA Macro*

### Datapath

Datapath provides the steering and registers for the data between the PCI bus and the back-end. In addition, datapath contains the address counters and increments the value after each data transaction. Datapath is also responsible for preserving FIFO read data when pre-fetched data from the FIFO is not read immediately by the PCI bus.

### Parity

Parity generates read parity and checks write parity on the PCI bus.

### Configuration Block

The configuration block contains the configuration space for the Target+DMA Master controller. These registers include the ID registers, PCI status and control registers, the base address registers, and DMA configuration and status registers.

## I/O Signal Descriptions

For the purposes of this data sheet, the following signal type definitions are used.

- Input: Standard input-only signal.
- Output: Standard active driver that drives continuously.
- T/S Output: Standard active driver that can be tri-stated.
- Bi-Directional (referred to as t/s in the PCI specification): A combination input and t/s output pin.
- STS: Sustained Tri-State (s/t/s in the PCI specification) is a term used to describe either bi-directional or t/s output pins. The STS term indicates that the signal should always be driven to a logic ‘1’ before the pin is tri-stated.
- Open Drain: Drive to ‘0’ only output. A pull-up is required to sustain the high-impedance state to a logic ‘1’ and must be provided by the central resources.

***Table 1 •*** *CorePCI Target+DMA Signals*

| Name[1] | Type | Description |
|---|---|---|
| CLK | Input | Clock input for the PCI macro. |
| RSTn | Input | Active LOW asynchronous reset. |
| REQn | Output | Active LOW signal requesting mastership of the PCI Bus. |
| GNTn | Input | Active LOW signal granting mastership of the PCI Bus. |
| AD[31:0] | Bi-Directional | Multiplexed address and data bus. Valid address is indicated by FRAMEn assertion. |
| C_BE[3:0] | Bi-Directional | Bus command and byte enable information. During the address phase, these signals define the bus command. During the data phase, they define the byte enables. |
| PAR | Bi-Directional | Parity signal. Parity is even across AD[31:0] and C_BE[3:0] |
| FRAMEn | Bi-Directional(STS) | Active LOW signal indicating the beginning and duration of an access. While FRAMEn is asserted, data transfers continue. |
| IRDYn | Bi-Directional(STS) | Active LOW signal indicating that the bus master is ready to complete the current data phase transaction. |
| TRDYn | Bi-Directional(STS) | Active LOW signal indicating that the target is ready to complete the current data phase transaction. |
| STOPn | Bi-Directional(STS) | Active LOW signal from the target requesting termination of the current transaction. |
| IDSEL | Input | Active HIGH target select used during configuration read and write transactions. |
| DEVSELn | Bi-Directional(STS) | Active LOW output from the target indicating that it is the target of the current access. |
| PERRn | T/S Output (STS) | Active LOW parity error signal. |
| SERRn | Open Drain | Active LOW system error signal. This signal reports PCI address parity errors and back-end system errors reported on the FATAL_ERROR input. |
| INTAn | Open Drain | Active LOW interrupt request. |

***Notes:***

*1. Active LOW signals are designated with a trailing lower-case n instead of #.*

***Table 2 •*** *Back-End Interface Signals*

| Name[1] | Type | Description |
|---|---|---|
| IO_CYC | Output | Active high signal indicating a transaction to I/O space. |
| MEM_CYC | Output | Active high signal indicating a transaction to memory space. |
| CONFIG_CYC | Output | Active high signal indicating a transaction to configuration space. |
| READ_CYC | Output | Active high signal indicating a read transaction. |
| WRITE_CYC | Output | Active high signal indicating a write transaction. |
| MEM_DATA | Bi-Directional | 32-bit bi-directional data bus. |
| MEM_ADD[N:0] | Output | Memory address bus. N is defined by the variable MADDR_WIDTH-1. |
| DP_START | Output | Active high pulse indicating that a PCI transaction to this target is beginning. |

***Notes:***

*1. Active LOW signals are designated with a trailing lower-case n instead of #.*

***Table 2 •*** *Back-End Interface Signals (Continued)*

| Name[1] | Type | Description |
|---|---|---|
| DP_DONE | Output | Active high pulse indicating that a PCI transaction to this target has finished. |
| RD_BE_NOW | Output | This signal is only activated when RD_BE_RDY is active. At the beginning of a PCI transfer, this signal indicates that the PCI bus is ready to read data from the back-end. For fast back-ends like SRAM and FIFO, valid data will be expected immediately on the following rising clock edge. For slow response devices, like SDRAM, a latency timer will hold off the controller until the back-end is initialized. Once the initialization is complete, data will be expected on all subsequent rising clock edges. This signal will go inactive whenever IRDYn or RD_BE_RDY becomes inactive. |
| RD_BE_RDY | Input | Active high signal indicating that the back-end is ready to send data to the Target+DMA interface. If the ready signal does not become active within the limits defined by the PCI bus, then a disconnect without data will be initiated. |
| WR_BE_NOW[3:0] | Output | This signal is only activated when WR_BE_RDY is active. At the beginning of a PCI transfer, this signal indicates that the PCI bus is ready to write data to the back-end. For fast back-ends like SRAM and FIFO, valid data will be provided to the back-end on the next clock edge. For slow response devices, like SDRAM, a latency timer will hold off the controller until the back-end is initialized. Once the initialization is complete, and when this data signal is valid, data will be provided on all subsequent clock edges. This signal will go inactive whenever IRDYn or WR_BE_RDY becomes inactive. Each bit in the RD_BE_NOW bus represents a byte enable. |
| WR_BE_RDY | Input | Active high signal indicating that the back-end is ready to receive data from the Target+DMA interface. If the ready signal does not become active within the limits defined by the PCI bus, then a disconnect without data will be initiated. |
| BE_REQ | Input | A request from the back-end to the PCI Target+DMA Controller to take control of the back-end. This signal is active high. |
| BE_GNT | Output | A grant from the PCI Target+DMA Controller to the back-end granting control. When the BE_GNT signal is active and a transaction to the PCI Target+DMA controller occurs, the PCI controller will respond with Retry cycle. If a cycle is in progress when the BE_REQ is asserted, the BE_GNT will not assert until completion of the current PCI cycle. If the back-end must take control during a cycle, then the ready signals can be de-asserted, causing a PCI time-out and resultant disconnect. |
| EXT_INTn | Input | Active low interrupt from the back-end. When PCI interrupts are enabled, this should cause an INTAn signal to be asserted. |

***Notes:***

*1. Active LOW signals are designated with a trailing lower-case n instead of #.*

## Supported Commands

Table 3 lists the PCI commands supported in the current CorePCI Target+DMA implementation. If required, I/O support, and thus I/O commands, can be eliminated from the design by setting the appropriate customization options.

### I/O Read (0010) and Write (0011)

The I/O read command is used to read data mapped into I/O address space. The target will not check to verify the consistency of the address and byte enables. This, and any additional error checking, is left for implementation by the user. The default I/O space size is 256 bytes.

### Memory Read (0110) and Write (0111)

The memory read command is used to read data in memory-mapped address space. The baseline memory macro supports 16 megabytes which can be located anywhere in 32-bit address space.

### Configuration Read (1010) and Write (1011)

The configuration read command is used to read the configuration space of each device. The device is selected if its IDSEL signal is asserted and AD[1:0] are '00'. Additional address bits are defined as follows:

- AD[7:2] contains one of 64 DWORD addresses for the configuration registers.
- AD[10:8] indicates which device of a multi-function agent is addressed. The macro does not currently support multi-function devices and these signals should be '00'.
- AD[31:11] are "don't cares."

***Table 3 •*** *Supported PCI Commands*

| C_BE[3:0] | Command Type |
|---|---|
| 0010 | I/O Read |
| 0011 | I/O Write |
| 0110 | Memory Read |
| 0111 | Memory Write |
| 1010 | Configuration Read |
| 1011 | Configuration Write |

## Data Transaction

The CorePCI Target+DMA is designed to be fully compliant for all transfer types, including single DWORD and burst transactions. DMA Master transactions use the Memory Read and Memory write commands to transfer DWORDS. Burst transactions can operate with zero, one, or more wait states. Either the PCI bus client or the back end can pace the rate of transactions. If both can operate at zero wait state, then no wait states will be injected. However, if either the PCI bus client or the back end requires more than one cycle to complete the transaction, then wait states will be added automatically.

The PCI bus client can insert wait states by de-asserting the IRDYn (when master), or the TRDYn (when the client is a target) signal, at any point during a burst transaction.

## Device Utilization

Utilization statistics for targeted devices are listed in Table 4. The Target controller requires a certain amount of logic regardless of the back-end type. Each back-end requires different amounts of logic, depending on the complexity of the controller.

***Table 4 •*** *Utilization for the CorePCI Target+DMA Macro*

| Function | 42MX[1] | 54SX[2] |
|---|---|---|
| Target Controller Only | 440/1050 | 407/841 |
| SDRAM Controller | N/A | 63/135 |
| SRAM Controller | 3/6 | 3/7 |

***Notes:***

1. *The first number represents the number of S-modules and the second number is the total modules.*
2. *The first number represents the R-modules required and the second number is the total number of C-modules required.*

## Power Consumption

Actel devices are standard CMOS and consume power only when logic levels are being switched. There are three distinct power consumption situations: static PCI bus, active PCI bus not addressing the Actel device, and transactions to/from the Actel device. When the PCI bus is static, the Actel device consumes no power except for the clock network. When the PCI bus is active to another device, only a small percentage of the control function is active. When the Actel device is being actively addressed, then a major portion of the design is switching, causing the highest power consumption.

## Configuration Space

The PCI specification defines a 64-byte space (configuration header) to define various attributes of the PCI target, as shown in Table 5. All registers in bold are implemented in this target, including the two base address registers for memory and I/O spaces. None of the remaining registers are included in the baseline implementation.

In addition to the configuration header, three configuration register, 40h, 44h, and 48h, are used to define DMA and interrupt specific status and controls.

### Read-Only Configuration Registers

The read-only registers listed below have default values, but should be modified by the designer. See the PCI specification for setting these values.

- Vendor ID
- Device ID
- Revision ID
- Class Code
- Subsystem ID
- Subsystem Vendor ID

The latter two registers are included in anticipation of adoption of Revision 2.2 of the PCI specification. The header type register is also read-only, but should not be modified (pre-set to a constant value of '00h').

### Read/Write Configuration Registers

The following register locations have at least one bit that is both read and write capable. For a complete description, refer to the appropriate table.

- Command Register (Table 6 on page 8)
- Status Registers (Table 7 on page 8)
- Base Address Register for Memory (Table 8 on page 9)
- Base Address Register for I/O (Table 9 on page 9)
- Interrupt Register (Table 10 on page 9)
- User Configuration Registers (Tables 11, 12, 13, and 14)

***Table 5 •*** *PCI Configuration Header*

| 31-24 | 23-16 | 15-8 | 7-0 | Address |
|---|---|---|---|---|
| **Device ID** | | **Vendor ID** | | 00h |
| **Status** | | **Command** | | 04h |
| **Class Code** | | | **Revision ID** | 08h |
| BIST | **Header Type** | Latency Timer | Cache Line Size | 0Ch |
| **Base Address #1 (Memory Location for Baseline Target)** | | | | 10h |
| **Base Address #2 (I/O Location for Baseline Target)** | | | | 14h |
| Base Address #3 | | | | 18h |
| Base Address #4 | | | | 1Ch |
| Base Address #5 | | | | 20h |
| Base Address #6 | | | | 24h |
| CardBus CIS Pointer | | | | 28h |
| **Subsystem ID** | | **Subsystem Vendor ID** | | 2Ch |
| Expansion ROM Base Address | | | | 30h |
| Reserved | | | | 34h |
| Reserved | | | | 38h |
| Max_Lat | Min_Gnt | **Interrupt Pin** | **Interrupt Line** | 3Ch |
| **PCI Start Address** | | | | 40h |
| **Ram Start Address** | | | | 44h |
| **DMA Control Register** | | | | 48h |

***Table 6 •*** *Command Register (04h)*

| Bit | Type | Description |
|---|---|---|
| 0 | R/W | I/O Space<br>A value of '0' disables the device's response to I/O space addresses. Should be '0' after reset. |
| 1 | R/W | Memory Space<br>A value of '0' disables the device's response to memory space addresses. Should be '0' after reset. |
| 2 | R/O | Bus Master<br>Target-only implementation. It is set to '1' indicating bus master capability. |
| 3 | R/O | Special Cycles<br>No response to special cycles. It is set to '0'. |
| 4 | R/O | Memory Write and Invalidate Enable<br>Memory write and invalidate not supported. It is set to '0'. |
| 5 | R/O | VGA Palette Snoop<br>Assumes non-VGA peripheral. It is set to '0'. |
| 6 | R/W | Parity Error Response<br>When '0', the device ignores parity errors. When '1', normal parity checking is performed. '0' after reset. |
| 7 | R/O | Wait Cycle Control<br>No data-stepping supported. It is set to '0'. |
| 8 | R/W | SERRn Enable<br>When '0', the SERRn driver is disabled. '0' after reset. |
| 9 | R/O | Fast Back-to-Back Enable<br>Set to '0'. Only fast back-to-back transactions to same agent are allowed. |
| 15-10 | R/O | Reserved and set to all '0's. |

***Table 7 •*** *Status Register (04h)*

| Bit | Type | Description |
|---|---|---|
| 4-0 | R/O | Reserved—set to '00000'b. |
| 5 | R/O | 66 MHz Capable<br>Set as necessary to indicate a 33 MHz or 66MHz Target+DMA Master |
| 6 | R/O | UDF Supported<br>Set to '0'—no user definable features. |
| 7 | R/O | Fast Back-to-Back Capable<br>Set to '0'—fast back-to-back to same agent only. |
| 8 | R/W | Data Parity Error Detected<br>Set to '0' at system reset. This bit is set to a '1' by internal logic whenever a data parity error is detected. |
| 10-9 | R/O | DEVSELn Timing<br>Set to '10'—slow DEVSELn response. |
| 11 | R/W | Signaled Target Abort<br>Set to '0' at system reset. This bit is set to a '1' by internal logic whenever a target abort cycle is executed. |

***Note:*** *The R/W capability in the status register is restricted to clearing the bit by writing a '1' into the bit location.*

***Table 7 •*** *Status Register (04h) (Continued)*

| Bit | Type | Description |
|---|---|---|
| 12 | R/W | Received Target Abort<br>Set to '0' at system reset. This bit is set to a '1' when the DMA master detects a target abort. |
| 13 | R/W | Received Master Abort<br>Set to '0' at system reset. This bit is set to a '1' when the DMA master terminates the cycle with a master abort cycle. |
| 14 | R/W | Signaled System Error<br>Set to '0' at system reset. This bit is set to '1' by internal logic whenever the SERRn signal is asserted by the Target. |
| 15 | R/W | Detected Parity Error<br>Set to '0' at system reset. This bit is set to '1' by internal logic whenever the Target asserts the PERRn signal. |

***Note:*** *The R/W capability in the status register is restricted to clearing the bit by writing a '1' into the bit location.*

***Table 8 •*** *Base Address Register—Memory (10h)*

| Bit | Type | Description |
|---|---|---|
| 0 | R/O | Indicates memory location. It is set to '0'. |
| 2-1 | R/O | Indicates mapping into any 32-bit address space. It is set to '00'. |
| 3 | R/O | Set to a '1' Indicating prefetch allowed on reads. |
| 23-4 | R/O | Indicates a 16 MB address space. It is set to all '0's. |
| 31-24 | R/W | Programmable location for 16 MB address space. To determine a hit, these bits must be compared to PCI address bits 31-24. |

***Note:*** *The description for bit values 31-24 and 23-4 will vary depending on the actual memory size defined in the customization options. See "Customization Options" on page 12 for more information.*

***Table 9 •*** *Base Address Register—I/O (14h)*

| Bit | Type | Description |
|---|---|---|
| 0 | R/O | Indicates I/O space. It is set to '1'. |
| 1 | R/O | Reserved. It is set to '0'. |
| 7-2 | R/O | 256-byte I/O space for this peripheral. It is set to '0'. |
| 31-8 | R/W | Programmable address for this peripheral's I/O space. To determine a hit, these bits must be compared to PCI address bits 31-8. |

***Note:*** *The description for bit values 31-8 and 7-2 will vary depending on the actual memory size defined in the customization options. See "Customization Options" on page 12 for more information.*

***Table 10 •*** *Interrupt Register (3Ch)*

| Bit | Type | Description |
|---|---|---|
| 7-0 | R/O | Set to '00000001'b to indicate INTAn. |
| 15-8 | R/W | Required read/write register. This register has no impact on internal logic. |

***Note:*** *This register is not required if no interrupt is required. See "Customization Options" on page 12 for more information.*

***Table 11 •*** *PCI Start Address (40h)*

| Bit | Type | Description |
|---|---|---|
| 1-0 | R/O | Set to '00'b. PCI transfers must be on a DWORD boundary. |
| 31-2 | R/W | PCI start address.<br>This location will not increment during the DMA transfer. At the end of a transfer, this register value will hold the initial starting address. |

***Table 12 •*** *RAM Start Address (44h)*

| Bit | Type | Description |
|---|---|---|
| 1-0 | R/O | Set to '00'b. PCI transfers must be on a DWORD boundary. |
| 23-2 | R/W | RAM start address.<br>This location will increment during the DMA transfer. At the end of a transfer, this register value will be unchanged. |
| 31-24 | R/O | Set to all zeros. |

***Note:*** *The description for bit values 31-24 and 23-2 will vary depending on the actual memory size defined in the customization options. See "Customization Options" on page 12 for more information. For this case, MADDR_WIDTH is set to 24.*

***Table 13 •*** *DMA Control Register (48h)*

| Bit | Type | Description |
|---|---|---|
| 0 | R/W | DMA Error<br>Bit is set when DMA is terminated because of an error condition. Reasons include a PCI master abort, PCI target abort, or the back-end interface setting the ERROR or FATAL_ERROR signals. Writing a '1' clears this bit. |
| 1 | R/W | DMA Abort<br>Writing a '1' to this bit aborts the current DMA transfer. This bit always reads as a '0'. |
| 2 | R/O | DMA Done<br>A '1' indicates that the DMA transfer is done. The bit is cleared by issuing a new DMA request. |
| 3 | R/W | DMA Direction<br>A '1' indicates a read from PCI and a write to RAM. A '0' indicates a read from RAM and a write to PCI. |
| 4 | R/W | DMA Request<br>Writing a '1' will initiate a DMA transfer and the bit will remain set until the DMA transfer completes. This bit can be cleared by issuing a DMA abort command (see bit 1). |
| 5 | R/W | DMA Enable<br>This bit must be set to '1' to enable any DMA transfers. |

***Table 13 •*** *DMA Control Register (48h) (Continued)*

| Bit | Type | Description |
|---|---|---|
| 6 | R/W | DMA Interrupt Active<br>This bit is set whenever the DMA transfer completes or a DMA error condition occurs. The bit is cleared by writing a '1' to this location. |
| 7 | R/W | DMA Interrupt Enable<br>A '1' enables the DMA interrupt to drive the PCI INTAn signal. |
| 8 | R/W | DMA Interrupt Status<br>A '1' in this bit indicates an active external interrupt condition (assertion of EXT_INTn). It is cleared by the user by writing a '1' to this bit position. Set to '0' after reset. |
| 9 | R/W | External Interrupt Enable<br>Writing a '1' to this bit enables support for the external interrupt signal. Writing a '0' to this bit disables external interrupt support. |
| 10 | R/W | Data Recovery Enable<br>Writing a '1' to this bit enables support for the macro's data recovery logic. Use of this logic with a FIFO back-end ensures that no data will be lost. |
| 15-11 | R/O | Reserved (set to '0'). |
| 27-16 | R/W | DMA Transfer Length<br>Number of bytes to be transferred. Bits 16 and 17 are set to '0' since DMA transactions must be on DWORD boundaries. During a DMA transfer, this location will decrement indicating the number of bytes remaining. To transfer 1024 DWORDs, this location should be set to all zeros. Lengths must be at least 4 DWORDs. |
| 31-28 | R/O | Reserved (set to '0'). |

***Table 14 •*** *CorePCI Target+DMA Master Customization Constants*

| Constant | Type | Description |
|---|---|---|
| USER_DEVICE_ID | Binary | Device ID constant. |
| USER_VENDOR_ID | Binary | Vendor ID constant. |
| USER_REVISION_ID | Binary | Revision ID constant. |
| USER_BASE_CLASS | Binary | Base Class constant. |
| USER_SUB_CLASS | Binary | Sub Class constant. |
| USER_PROGRAM_IF | Binary | Base Class Interface constant. |
| USER_SUBSYSTEM_ID | Binary | Subsystem ID constant. |
| USER_SUBVENDOR_ID | Binary | Subsystem Vendor ID constant. |
| MADDR_WIDTH | Integer | Defines memory space size. Allowable range is 4-31 where 4 represents 16 bytes and 24 represents 16 Mbytes of memory space. |
| IADDR_WIDTH | Integer | Defines the address space size for I/O. The valid range is 2 to 8, where 2 indicates 4 bytes and 8 indicates 256 bytes. |

## Customization Options

The PCI Target+DMA Master has a variety of options for user customization. Included with the source design files is a special package that defines a list of variables which allow the user to optimize the macro for their particular application. Table 14 lists the variables and their meaning.

### Configuration Register Constants

To set the read-only registers in the configuration space, a variety of constants are defined. The constants support the definitions of the device ID register, vendor ID register, class code registers, revision ID register, subsystem ID, and the subsystem vendor ID.

### Other Options

The Target+DMA Master macro offers a variety of memory options to define memory size, error checking, and special needs for back-end FIFOs. The MADDR_WIDTH defines the size of the memory space. The IADDR_WIDTH defines the size of the I/O space. The interrupt capability is defined by the variable ENABLE_EXTERNAL_INTERRUPT.

## System Timing

System timing for the CorePCI Target+DMA Master macro is defined in Tables 15, 16, 18, and 19. These tables should be used in conjunction with the timing waveforms in the following section. The required timing as defined in the PCI specification is included in the PCI timing tables. Input set-up is shown in Figure 3, and output valid delays are shown in Figure 4. All MX timing is defined for the array at 4.75V. However, the I/O can be at 5V or 3.3V.

The PCI signals, MEM_ADD[N:0], and MEM_DATA[31:0] are all external signals. The setup and output valid times for these signals are measured externally to the device. The remaining signals are all internal, and the timing defined does not take into account the internal clock delay. If these signals are to be mapped directly to an output pad, then, for SX, subtract 1.5ns for the external setup, and add 1.5ns for output valid time to account for the delay of the clock buffer. For MX, use 4.3ns as the clock delay.

All of the values presented in this section were achieved using commercially available synthesis tools and timing-driven place and route with fixed pinout for PCI signals. The actual numbers you achieve will vary, but these values should be viewed as expected values.

### PCI I/O Signal Timing

PCI input set-up times for the CorePCI Target+DMA Master are shown in Table 15. PCI output timing is shown in Table 16. All 54SX timing is for the 54SX16P. 66MHz compliant numbers will be provided once the A54SX72AP is characterized.

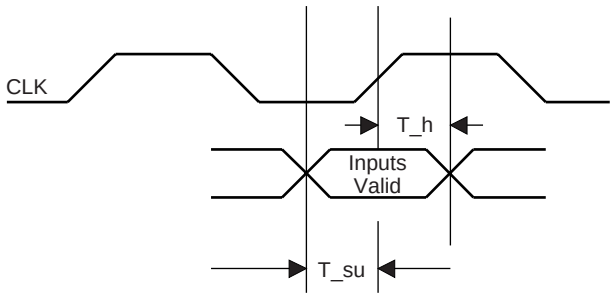



***Figure 3 •*** *Input Timing for PCI Signals*

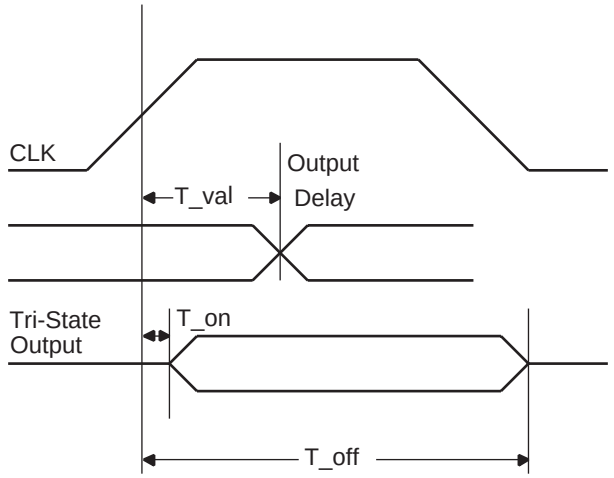



***Figure 4 •*** *Output Timing for PCI Signals*

### Back-End I/O Signal Timing

Table 18 and Table 19 summarize the set-up and hold times for key back-end interface input and output signals, respectively.

***Table 15 •*** *PCI Signals—Input Set-Up Times*

| Signal | 42MX -1 | 54SX -2 | PCI Spec 33/66 |
|---|---|---|---|
| AD[31:0] | 3 | 4 | 7/3 |
| C_BE[3:0] | 4 | 3 | 7/3 |
| PAR | 3 | 3 | 7/3 |
| FRAMEn | 6 | 4 | 7/3 |
| IRDYn | 6 | 7 | 7/3 |
| TRDYn | 6 | 7 | 7/3 |
| STOPn | 0 | 2 | 7/3 |
| DEVSELN | 1 | 2 | 7/3 |

***Notes:***

1. *All timing is for worst-case commercial conditions.*
2. *Expected values from commercially available synthesis tools using standard design practices.*

***Table 16 •*** *PCI Signals—Output Valid Delay Times*

| Signal | 42MX -1 | 54SX-2 | PCI Spec 33/66 |
|---|---|---|---|
| AD[31:0] | 11 | 8 | 6/11 |
| CBE[3:0] | 10 | 6 | 6/11 |
| PAR | 10 | 6 | 6/11 |
| FRAMEn | 11 | 7 | 6/11 |
| IRDYn | 11 | 7 | 6/11 |
| TRDYn | 11 | 6 | 6/11 |
| STOPn | 11 | 6 | 6/11 |
| DEVSELn | 11 | 6 | 6/11 |
| PERRn | 11 | 6 | 6/11 |

***Notes:***

1. *All timing is for worst-case commercial conditions.*
2. *Expected values from commercially available synthesis tools using standard design practices.*

***Table 17 •*** *Internal Propagation Delays*

| Signal | 42MX -1 | 54SX -2 | PCI Spec 33/66 |
|---|---|---|---|
| Reg-Reg | 25 | 17 | 30/15 |

***Notes:***

1. *All timing is for worst-case commercial conditions.*
2. *Expected values from commercially available synthesis tools using standard design practices.*

***Table 18 •*** *Generic Interface Input Set-Up Times (max)*

| Name | 42MX-1 | 54SX-2 |
|---|---|---|
| RD_BE_RDY | 16 | 11 |
| WR_BE_RDY | 17 | 9 |
| BE_REQ | 10 | 1 |
| EXT_INTn | 10 | 3 |
| MEM_DATA[31:0] | 5 | 3 |

***Notes:***

1. *All timing is for worst-case commercial conditions.*
2. *Expected values from commercially available synthesis tools using standard design practices.*
3. *MEM_DATA is an external bus.*

### CorePCI Target+DMA Master Internal Timing

All internal delays meet the required 15ns at worst case commercial environmental conditions.

***Table 19 •*** *Generic Interface Output Valid Times*

| Name | 42MX-1 | 54SX-2 |
|---|---|---|
| IO_CYC | 3 | 4 |
| MEM_CYC | 3 | 4 |
| CONFIG_CYC | 6 | 4 |
| READ_CYC | 6 | 4 |
| WRITE_CYC | 6 | 4 |
| MEM_DATA[31:0] | 14 | 7 |
| MEM_ADD[N:0] | 13 | 6 |
| DP_START | 6 | 6 |
| DP_DONE | 6 | 6 |
| RD_BE_NOW | 2 | 5 |
| WR_BE_NOW[3:0] | 10 | 5 |
| BE_GNT | 6 | 6 |

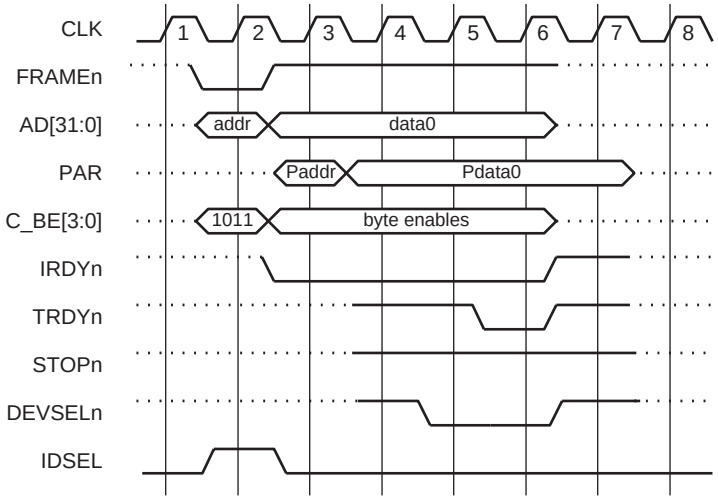
***Notes:***

1. *All timing is for worst-case commercial conditions.*
2. *Expected values from commercially available synthesis tools using standard design practices.*

## PCI Target Waveforms

### Configuration Cycles

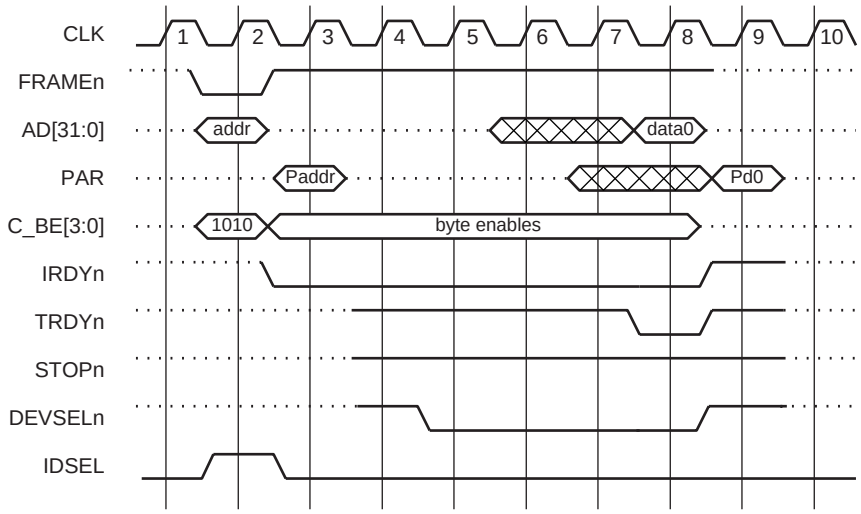
Configuration read and write cycles are used to define and determine the status of the Target+DMA Master's internal configuration registers. Configuration cycles are the only type of transactions that use the IDSEL signal. Register selection is defined by the contents of the address (bits 7 down to 2). A configuration write is shown in Figure 5 and a configuration read is shown in Figure 6.




***Notes:***

1. *If the Target+DMA Master's IDSEL is asserted when FRAMEn is asserted and the command bus is '1011', then a configuration write cycle is indicated.*
2. *The Target claims the bus by asserting DEVSELn in cycle 4.*
3. *Data is registered into the device on the rising edge of cycle 5.*
4. *The single DWORD transfer completes when TRDYn is asserted in cycle 5 and de-asserted in cycle 6.*

***Figure 5 •*** *Configuration Write Cycle*




***Notes:***

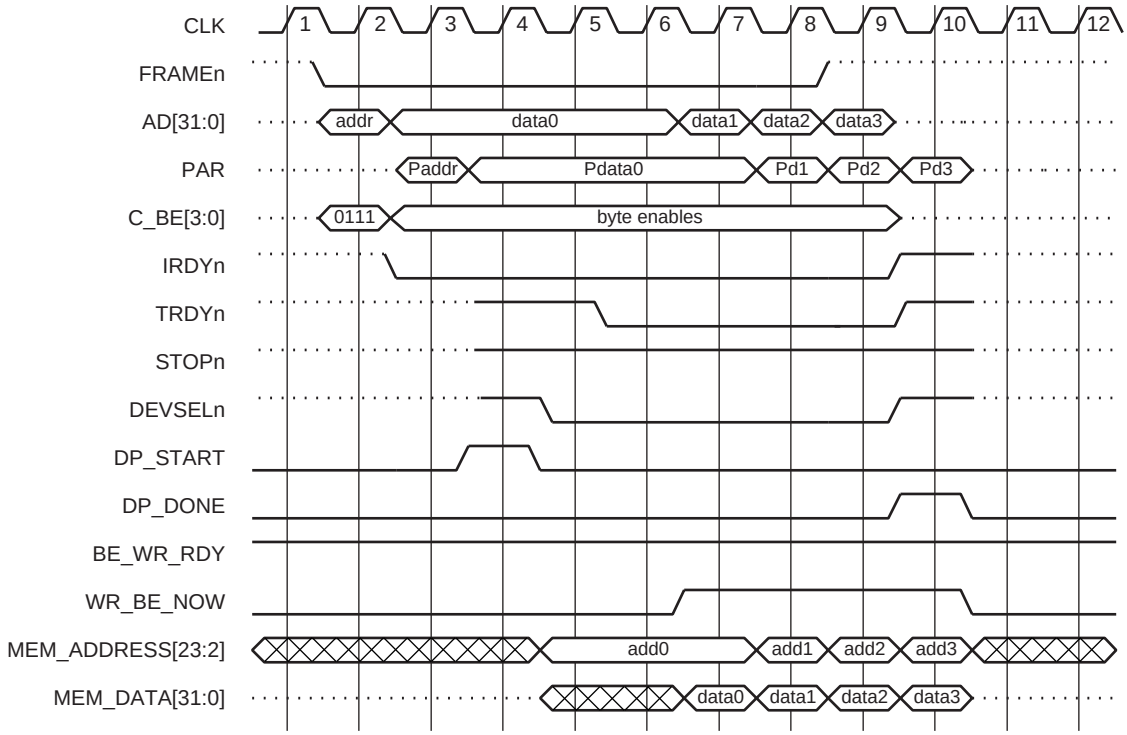
1. *If the Target's IDSEL is asserted when FRAMEn is asserted and the command bus is '1010', then a configuration read cycle is indicated.*
2. *The Target claims the bus by asserting DEVSELn in cycle 4.*
3. *During cycle 7, TRDYn is asserted and valid data is driven onto the PCI bus.*
4. *The single DWORD transfer completes when TRDYn is de-asserted in cycle 8.*

***Figure 6*** • *Configuration Read Cycle*

### Zero Wait State Burst Transactions

Zero wait state burst enables transfer of a DWORD for every clock cycle. All cycles are initiated with a DP_START indicating a hit to the Target. The type of cycle should be qualified with the *_CYC signals (MEM_CYC, IO_CYC, CONFIG_CYC, RD_CYC, and WR_CYC). Valid data to and from the back-end is qualified by the *_BE_NOW signals (WR_BE_NOW and RD_BE_NOW). The end of the cycle is indicated by the DP_DONE signal being asserted.

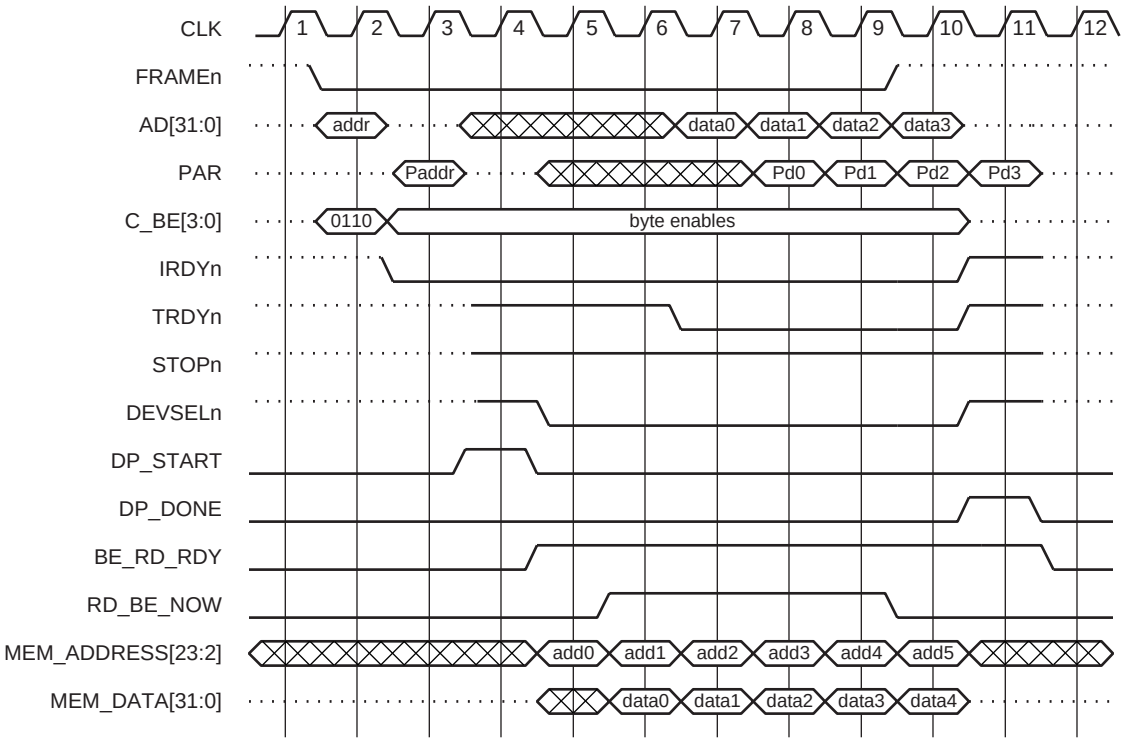
Zero wait state burst transfers are shown in Figure 7 and Figure 8.




***Notes:***

1. *When FRAMEn is asserted and the command bus is '0111', then a write to memory space is indicated.*
2. *The Target will compare the address to the programmed space set in the memory base address register.*
3. *If an address hit occurs, then the Target asserts DP_START in cycle 3 and claims the PCI bus by asserting DEVSELn in cycle 4.*
4. *Data transfer to the back-end begins on the rising edge of cycle 7 and continues for each subsequent cycle until the PCI bus ends the data transfer.*
5. *The PCI transaction completes when TRDYn is de-asserted in cycle 9 and completes on the back-end in cycle 10.*

***Figure 7 •*** *Burst Write with Zero Wait States*



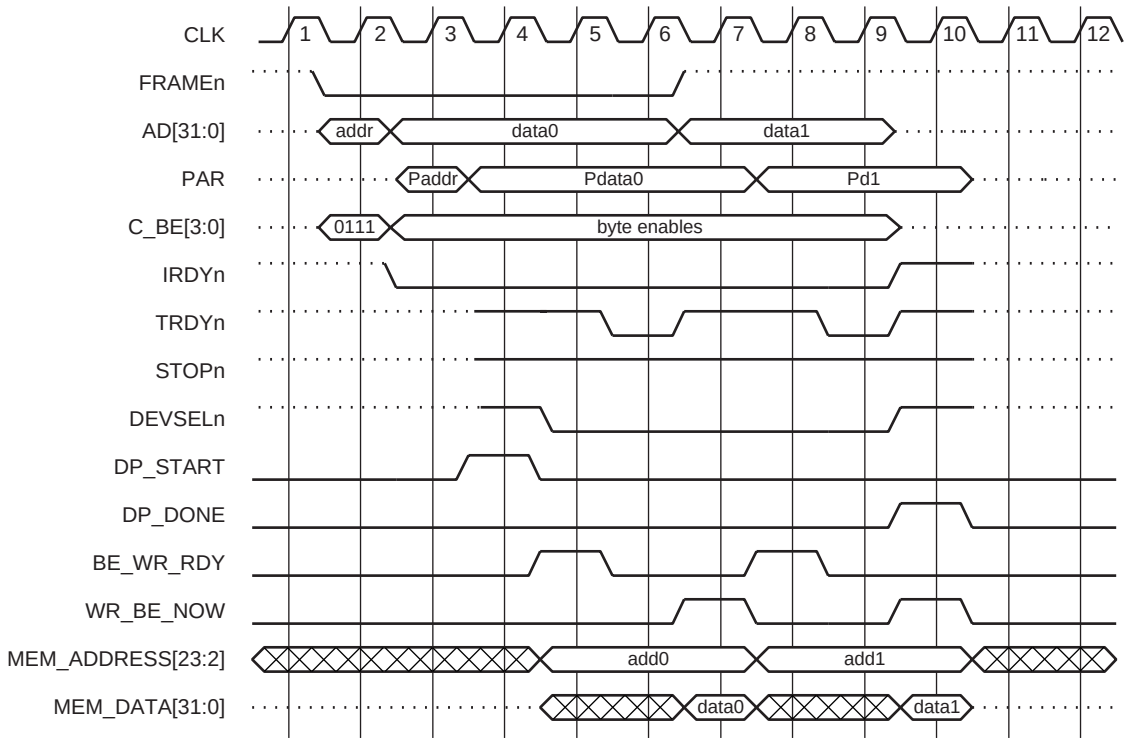

***Notes:***

1. *When FRAMEn is asserted and the command bus is '0110', then a read from memory space is indicated.*
2. *The Target will compare the address to the programmed space set in the memory base address register.*
3. *If an address hit occurs, then the Target asserts DP_START in cycle 3 and claims the PCI bus by asserting DEVSELn in cycle 4.*
4. *Data transfer from the back-end begins on the rising edge of cycle 7 and continues for each subsequent cycle until the PCI bus ends the data transfer. The back-end prefetches three DWORDs during zero wait state bursts.*
5. *The PCI transaction completes when TRDYn is de-asserted in cycle 10.*

***Figure 8 •*** *Burst Read with Zero Wait States*

### Paced Transactions

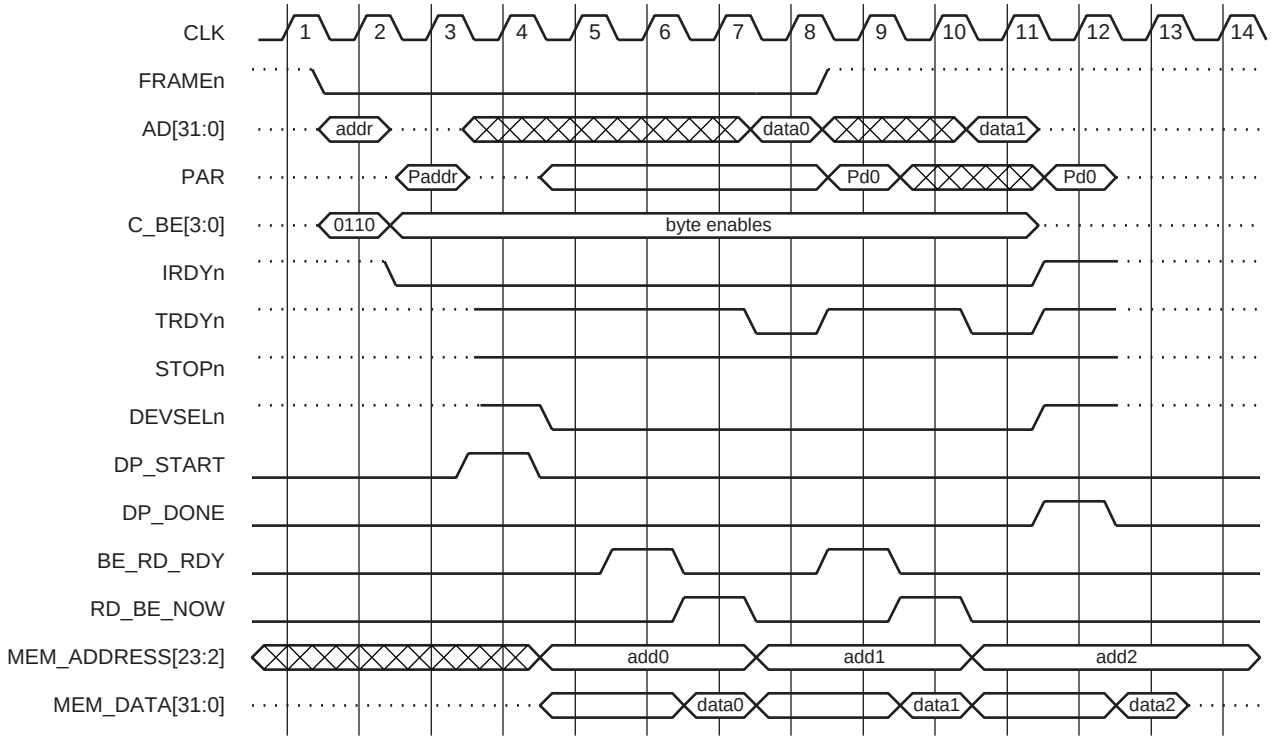
Back-end throttle transfers provide a handshake mechanism for supporting slow response devices. The back-end transactions are paced using the BE_RD_RDY and BE_WR_RDY signals. These signals can be used to pace either single DWORD or burst transactions. Figure 9 and Figure 10 illustrate this mechanism for a back-end that requires three cycles to respond to a read or write command from the PCI bus.




***Notes:***

1. *The BE_WR_RDY should be asserted two cycles prior to when the back-end is ready to receive data.*
2. *The BE_WR_RDY signal will initiate assertion of TRYDYn, completing the PCI write cycle. One cycle later, the data is available on the back-end and is qualified by the WR_BE_NOW[3:0] bus.*
3. *The WR_BE_NOW[3:0] should not be assumed to happen at this time because it is also dependent on the state of IRDYn.*

***Figure 9 •*** *Write Using Back-End Throttling*




***Notes:***

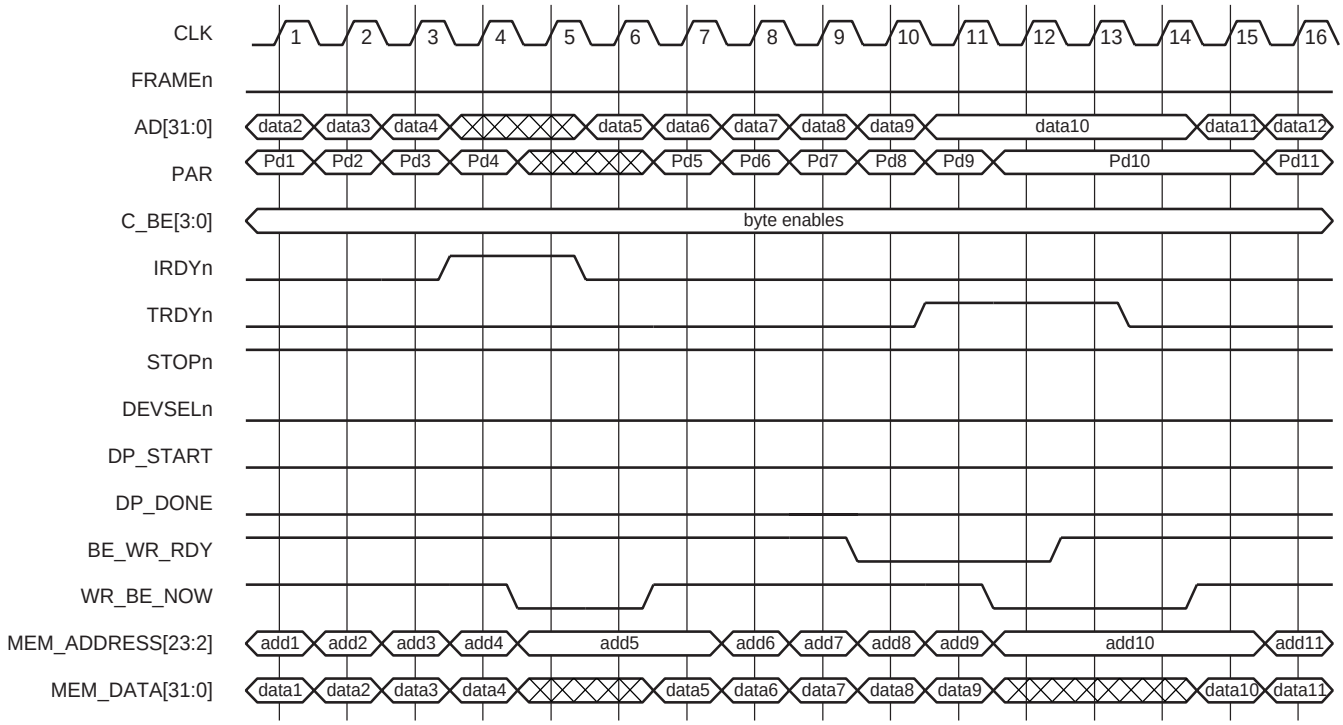
1. *The BE_RD_RDY should be asserted one cycle prior to when the back-end is ready to transmit data.*
2. *The BE_RD_RDY signal will initiate assertion of RD_BE_NOW, latching the data into the controller. The data transfer will complete when TRDYn is asserted on the following cycle.*
3. *The RD_BE_NOW should not be assumed to happen at this time, because it is also dependent on the state of IRDYn and may not respond to the back-end, requiring the back-end to hold the data until the data transfer completes.*

***Figure 10 •*** *Read Using Back-End Throttling*

### Paused Transactions

During long bursts, either the back-end controller or the PCI Master may insert wait states to accommodate some functional requirement. The PCI Master inserts wait states by de-asserting the IRDYn signal. The wait state is indicated to the back-end by de-assertion of the WR_BE_NOW bus or the RD_BE_NOW signal.

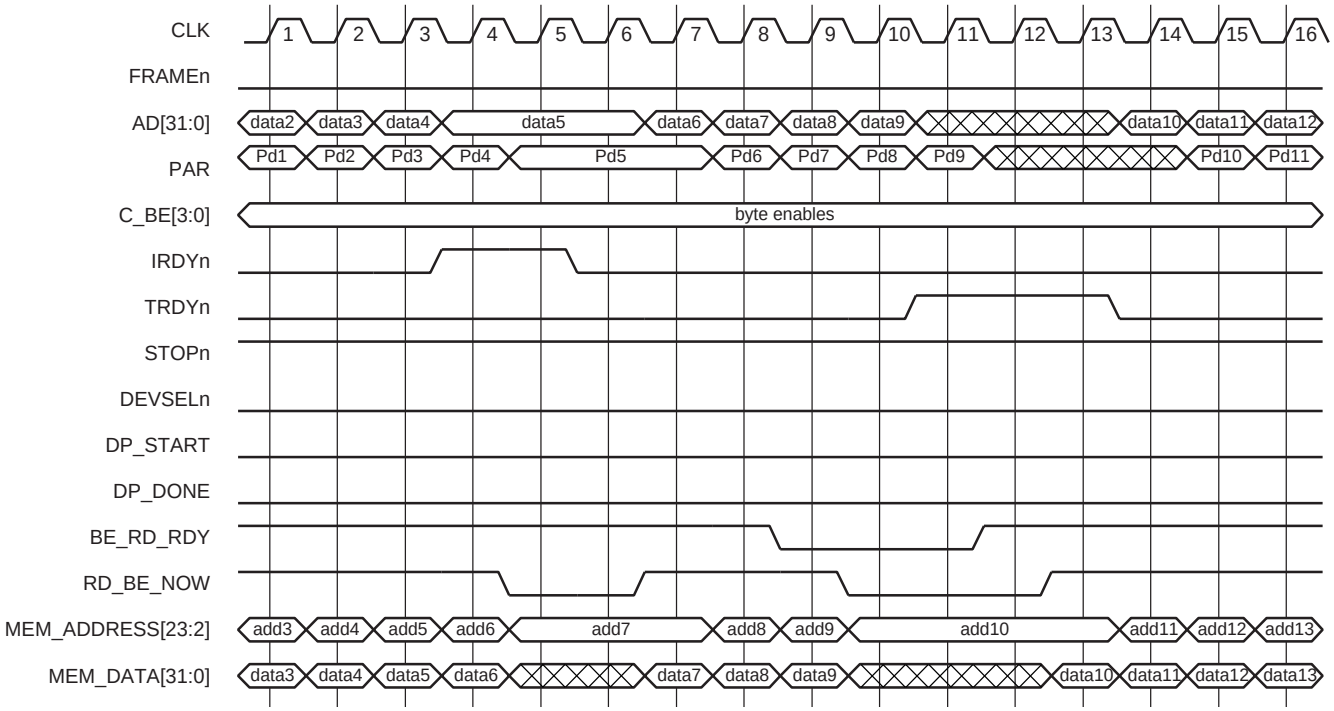
The back-end can insert wait states by de-assertion of the *_BE_RDY signals. These signals cause the Target+DMA Master controller to de-assert TRDYn, and, therefore, insert wait states on the PCI bus. For writes, the back-end must be prepared to accept up to two DWORDs of data prior to data transfer termination. For reads, the back-end must be prepared to transmit one DWORD of data prior to data transfer termination.




***Notes:***

1. *In the example, the flow of data is interrupted from the PCI master indicated by de-assertion of IRDYn in cycle 3. The PCI master injects two wait states. This state of the PCI bus is defined to the back-end by de-asserting the WR_BE_NOW[3:0] bus one cycle later.*
2. *The back-end can also interrupt the flow of data by de-asserting the BE_WR_RDY signal. One cycle later, TRDYn is de-asserted halting the flow of data on the PCI bus. The back-end must accept two DWORDs of data following de-assertion of the WR_BE_RDY signal.*

***Figure 11 •*** *PCI Write Illustrating both IRDYn and TRDYn De-Assertion.*



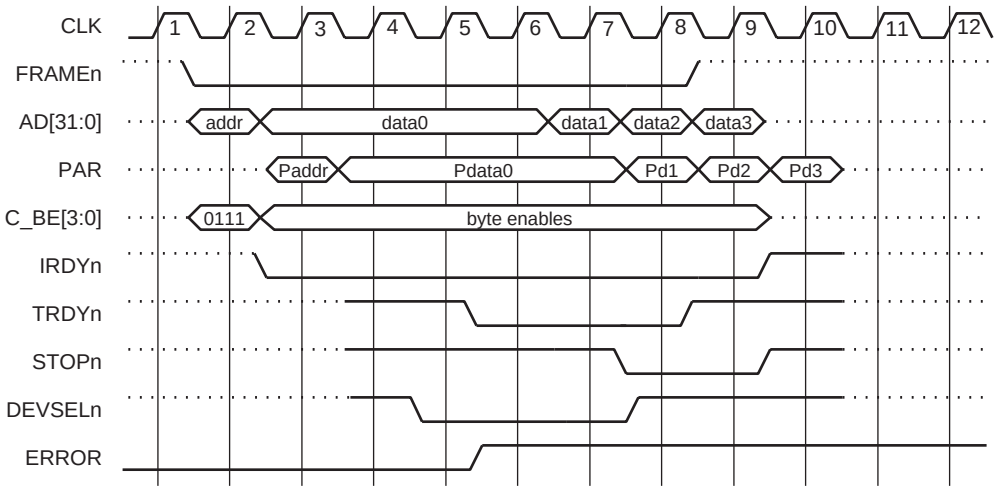

***Notes:***

1. *In the example, the PCI master interrupts the flow of data by de-asserting the IRDYn sign in cycle 4. Once cycle later the, RD_BE_NOW signal becomes inactive, indicating that the back-end should stop supplying data.*
2. *The back-end can also interrupt the flow of data by de-asserting the BE_RD_RDY signal. The back-end should be prepared to provide one additional DWORD of data to the PCI bus prior to halting the data flow. One cycle after BE_RD_RDY is de-asserted, the RD_BE_NOW signal is driven inactive, which is then followed by the de-assertion of TRYDn.*

***Figure 12 •*** *PCI Read Illustrating IRDYn De-Assertion*

### Target Abort

A target abort occurs when some type of error condition occurs as shown in Figure 13. When an error occurs on the back-end, this condition is reported on the ERROR signal. The ERROR signal will cause a target abort, which is defined by the target simultaneously asserting the STOPn signal and de-asserting the DEVSELn signal.




***Notes:***

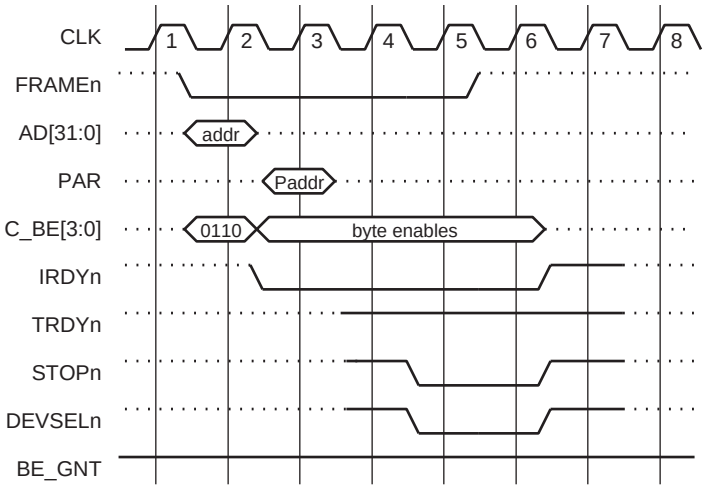
1. *During a PCI cycle, the back-end ERROR signal indicates that a problem occurred on the back-end such that the transfer cannot be completed.*
2. *The Target initiates a target abort by asserting STOPn and de-asserting DEVSELn in the same cycle.*
3. *The Master will begin cycle termination by de-asserting FRAMEn first, and then IRDYn on a subsequent cycle.*
4. *The transaction completes when STOPn is de-asserted in cycle 9.*

***Figure 13 •*** *Target Abort Cycle*

### Target Re-Try and Disconnect

When the back-end is busy or unable to provide the data requested, then the Target controller can respond with either a retry cycle or a disconnect cycle. When the back-end has arbitrated for control and the BE_GNT signal is active, then the controller will respond with a retry cycle as shown in Figure 14. The Target indicates that it is unable to respond by asserting the STOPn and DEVSELn simultaneously.

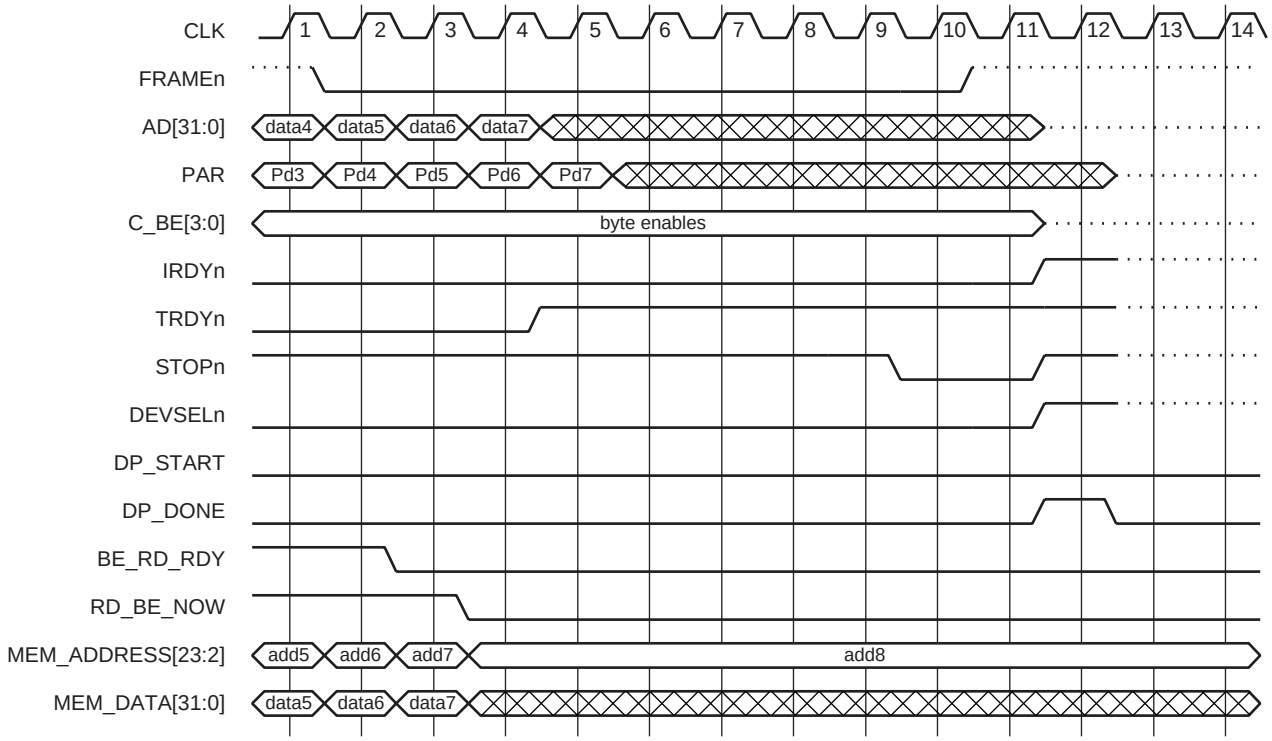
During a regular PCI transfer, the BE_RD_RDY and BE_WR_RDY indicate that data is available to be received from or transmitted to the back-end. If, during a PCI cycle, the back-end becomes unable to read or write data, then the *_RD_RDY signals are de-asserted. After several cycles, a PCI time-out will occur and the Target controller will initiate a Target disconnect without data cycle, as shown in Figure 15.




***Notes:***

1. *If BE_GNT is asserted at the beginning of a cycle, then a re-try is initiated.*
2. *The Target simultaneously asserts the STOPn and DEVSELn signals without asserting the TRDYn signal.*
3. *The Master will begin cycle termination by de-asserting FRAMEn first and then IRDYn on a subsequent cycle.*

***Figure 14*** • *Target Re-Try*




***Notes:***

1. *During a normal PCI transaction, the back-end reaches a point where it is unable to deliver data and de-asserts BE_RD_RDY.*
2. *The Target initiates a disconnect by asserting the STOPn signal once the controller is sure a PCI time-out will occur.*
3. *The Master will begin cycle termination by de-asserting FRAMEn first, and then IRDYn on a subsequent cycle.*

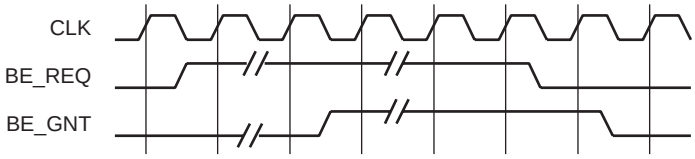
***Figure 15 •*** *Target Disconnect*

### Back-end Arbitration

When the back-end needs to take control of the back-end bus, the back-end should arbitrate for control using the BE_REQ and BE_GNT handshake signals, as shown in Figure 16.

### Interrupt

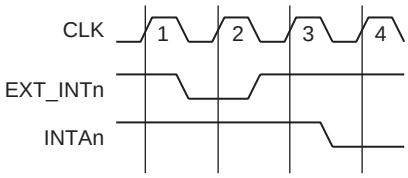
To initiate an interrupt, the back-end needs to assert the EXT_INTn input as shown in Figure 17. Two cycles later the PCI INTAn interrupt signal will assert.




***Notes:***

1. *Arbitration begins by the back-end asserting the BE_REQ signal. The Target Controller will grant control as soon as the PCI controller goes into an IDLE state.*
2. *The back-end will maintain control as long as the BE_REQ signal remains active.*
3. *To relinquish control, the back-end will de-assert the BE_REQ and the BE_GNT will de-assert on the following cycle.*

***Figure 16*** • *Back-end arbitration cycle.*




***Notes:***

1. *The EXT_INTn signal is sampled on the rising edge of each clock.*
2. *If the EXT_INTn signal is asserted and sampled in cycle 2, then the INTAn PCI signal will be asserted in cycle 3.*

***Figure 17*** • *Interrupt*

## PCI DMA Master Transactions

To perform DMA master transfers, the CorePCI Target+DMA has three configuration registers used to set addresses, transfer length, control, and check status of the transfer. These registers are located at 40h, 44h, and 48h. A basic sequence of events for executing DMA is as follows.

1. Write the location of the desired PCI address into the PCI Start Address configuration register at address 40h.
2. Write the location of the back-end memory location into the RAM Start Address register (44h)
3. Enable the completion interrupt (optional) by setting bit 7 at 48h to a '1'.
4. Set the direction of the transfer using bit 3 of 48h.
5. Define the transfer length using bits 27-16 in register 48h. The length must be at least 4 DWORDs and can be up to 1024 DWORDs. The transfer length value should be all zeros for 1024 DWORDs.
6. Initiate the transfer by setting bit 4 in 48h to a '1'.
7. At completion, bit 1 in 48h is set to a '1'.

### PCI DMA Read

DMA reads begin with arbitration for control of the PCI bus. The request and grant signals are used to arbit master access to the bus. Once control is granted to the macro, the macro begins by asserting FRAMEn, the address on the AD bus, and the command '0110' on the C_BE bus. A DMA read fetches data from the PCI bus and writes data to the back-end memory. The macro asserts IRDYn and then waits for the addressed target to provide the data indicated by TRDYn assertion. The transfer continues until the DMA transfer length is reached. The waveforms, shown in figure 18, depict the action of the macro when it operates as a DMA master in a zero wait state read transfer.

For DMA transactions, either zero wait state transfers or paced transfers can be used. During DMA transactions these two transfer modes function identically, as they do in a target only transaction.

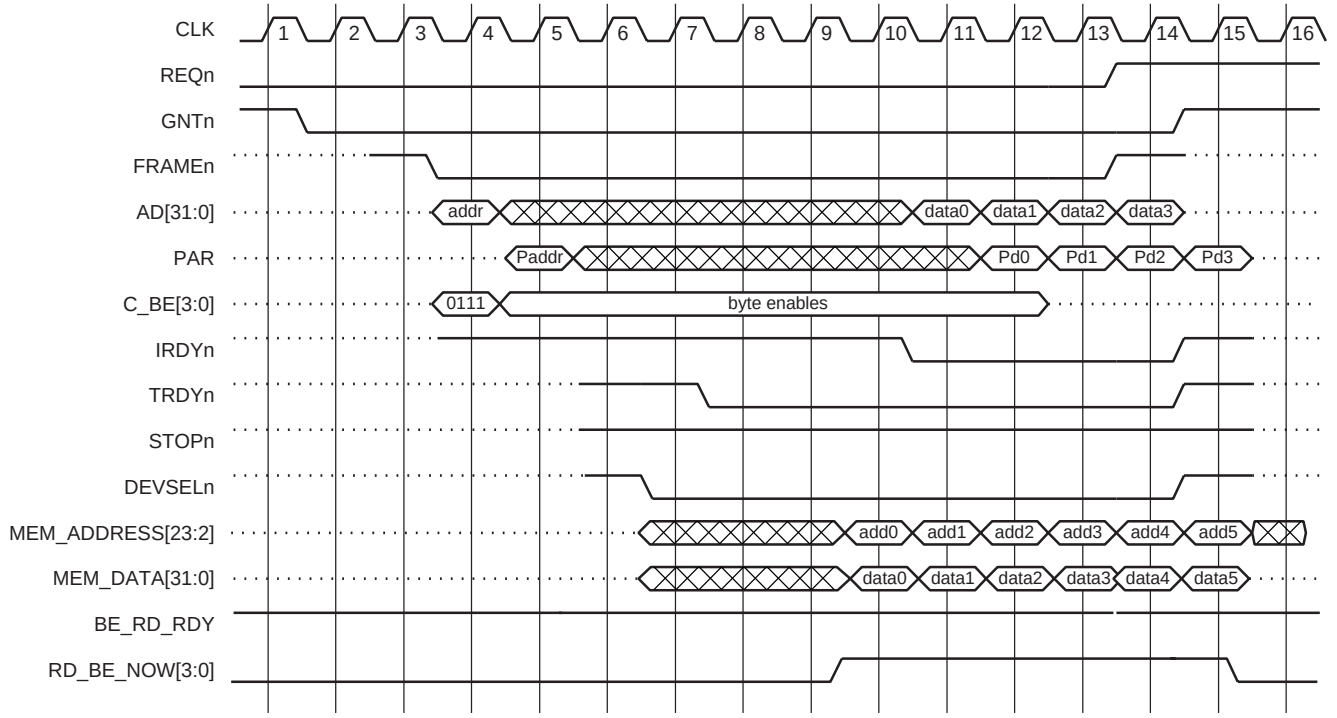



***Figure 18 •*** *Zero Wait State DMA Master Read (Read from the PCI bus)*

### PCI DMA Write

A DMA write begins by the macro requesting control of the bus. Once the bus is granted (GNTn asserted), the macro will initiate the transfer by asserting FRAMEn. A DMA write reads information from RAM and writes information onto the PCI bus. The back-end begins fetching data and when data is available, the datapath pipe fills and data flows onto the PCI bus. At that point, IRDYn is asserted and the burst transfer begins. The transfer is terminated once the DMA transfer length is reached. Figure 19 shows the zero wait state burst write transfer.

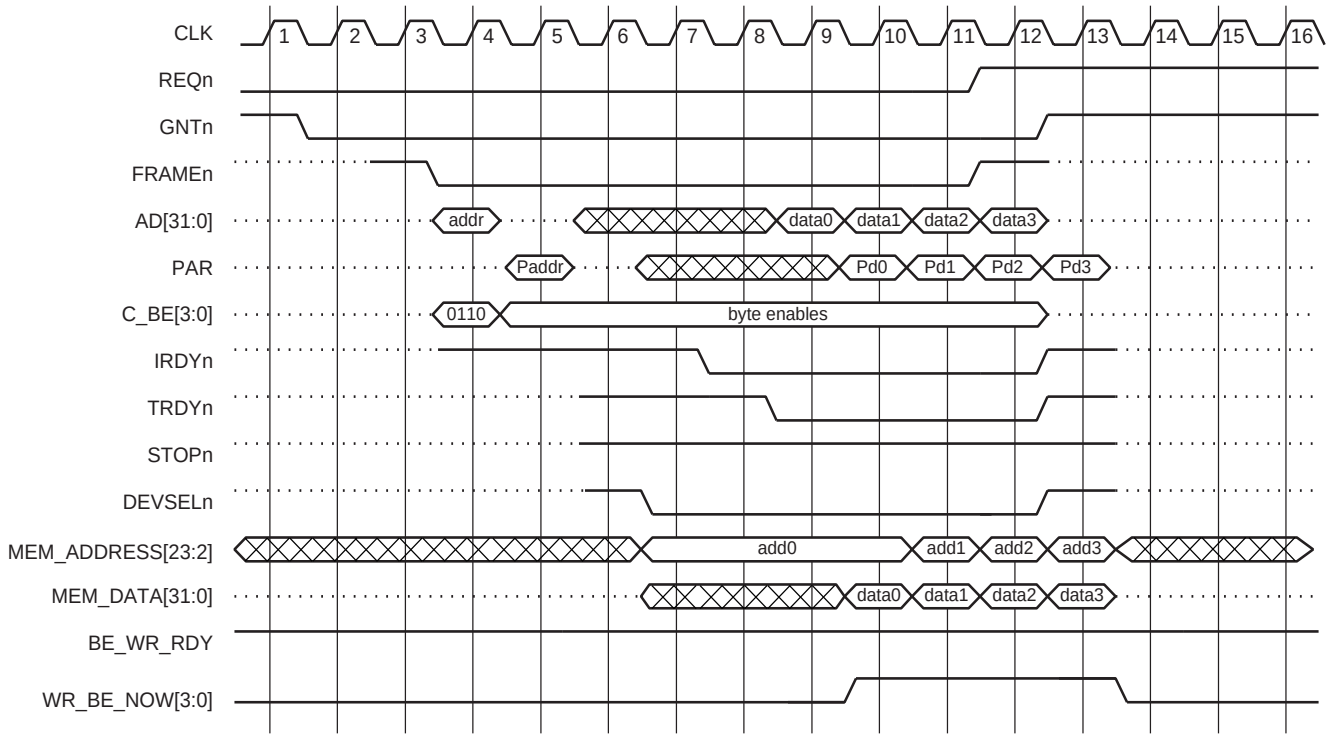



***Figure 19 •*** *Zero Wait State DMA Master Write (write to the PCI bus)*

## SDRAM Back-End

The SDRAM controller was designed to control a Micron MT48LC1M16A1TG S 1 MEG x 16 (512k x 16 x 2 banks) SDRAM operating at rates up to 66 MHz. By providing the major functional blocks of an SDRAM memory controller, the design can be customized to a specific controller design with very little effort. The SDRAM I/Os are defined in Table 20. The control interface is generic, except the IRDYn signal, which is specific to a PCI design. In this specific exception, the IRDYn signal controls the clock enable signal (CKE) during read/write burst sequences.The SDRAM performs three basic functions:

- SDRAM initialization
- Refresh
- Read/write transfers to the SDRAM

***Table 20 • SDRAM Controller I/O Signal Description***

| Name[1] | Type | Description |
|---|---|---|
| CSn | Output | Active low chip select. |
| RASn | Output | Row address strobe |
| CASn | Output | Column address strobe. |
| WEn | Output | Write enable strobe. |
| DQM | Output | DQ mask. |
| MAD(11:0) | Output | Multiplexed SDRAM address signals. |
| BA | Output | Bank select. |
| CKE | Output | Clock enable. |
| CLK | Input | System clock. |
| RESET_n | Input | Active low asynchronous reset signal. |
| IRDYn[2] | Input | Special PCI input used to control the CKE signal during read and write burst sequences. This is required to give the SDRAM controller time to disable the SDRAM when the PCI master suspends data transfers. |
| MEM_CYC | Input | Active high signal indicating a transaction to memory space. |
| READ_CYC | Input | Active high signal indicating a read transaction. |
| WRITE_CYC | Input | Active high signal indicating a write transaction. |
| AD[19:0] | Input | Memory address bus. The size of the address bus is defined by the variable MADDR_WIDTH. |
| DP_START | Input | Active high pulse indicating that a transaction to the SDRAM is beginning. |
| DP_DONE | Input | Active high pulse indicating that a transaction to the SDRAM has finished. |
| RD_BE_RDY | Output | This active high signal indicates that data is available two clock cycles prior to actual availability during a read burst cycle. |
| WR_BE_RDY | Output | This active high signal indicates that the controller is capable of writing data to the device in two clock cycles. |
| RD_BE_NOW | Input | Active high signal that starts a read transaction and indicates when read data is expected. |
| WR_BE_NOW | Input | Active high signal that starts a write transaction and indicates when write data is valid. |

***Notes:***

1. *Active LOW signals are designated with a trailing lower-case n instead of #.*
2. *This signal is a PCI intra-device signal.*

### SDRAM Controller System Timing

SDRAM timing consists of both internal and external timing delays. The PCI Target+DMA Master and the SDRAM Controller have been designed to work together at a clock frequency of up to 66MHz. External timing delays to the SDRAM are shown in Table 21 and Table 22. The values shown indicate that the 54SX devices will readily work with high performance SDRAM devices.

***Table 21 •*** *Output valid times (max)*

| Name | 54SX-2 |
|---|---|
| CSn | 6 |
| RASn | 6 |
| CASn | 6 |
| WEn | 6 |
| DQM | 6 |
| MAD(11:0) | 7 |
| BA | 6 |
| CKE | 6 |
| MEM_DATA[31:0] | 9 |

***Notes:***

1. *All timing is for worst-case commercial conditions.*
2. *Expected values from commercially available synthesis tools using standard design practices.*

***Table 22 •*** *Input setup times (max)*

| Name | 54SX-2 |
|---|---|
| MEM_DATA[31:0] | 5 |

***Notes:***

1. *All timing is for worst-case commercial conditions.*
2. *Expected values from commercially available synthesis tools using standard design practices.*

### Operation

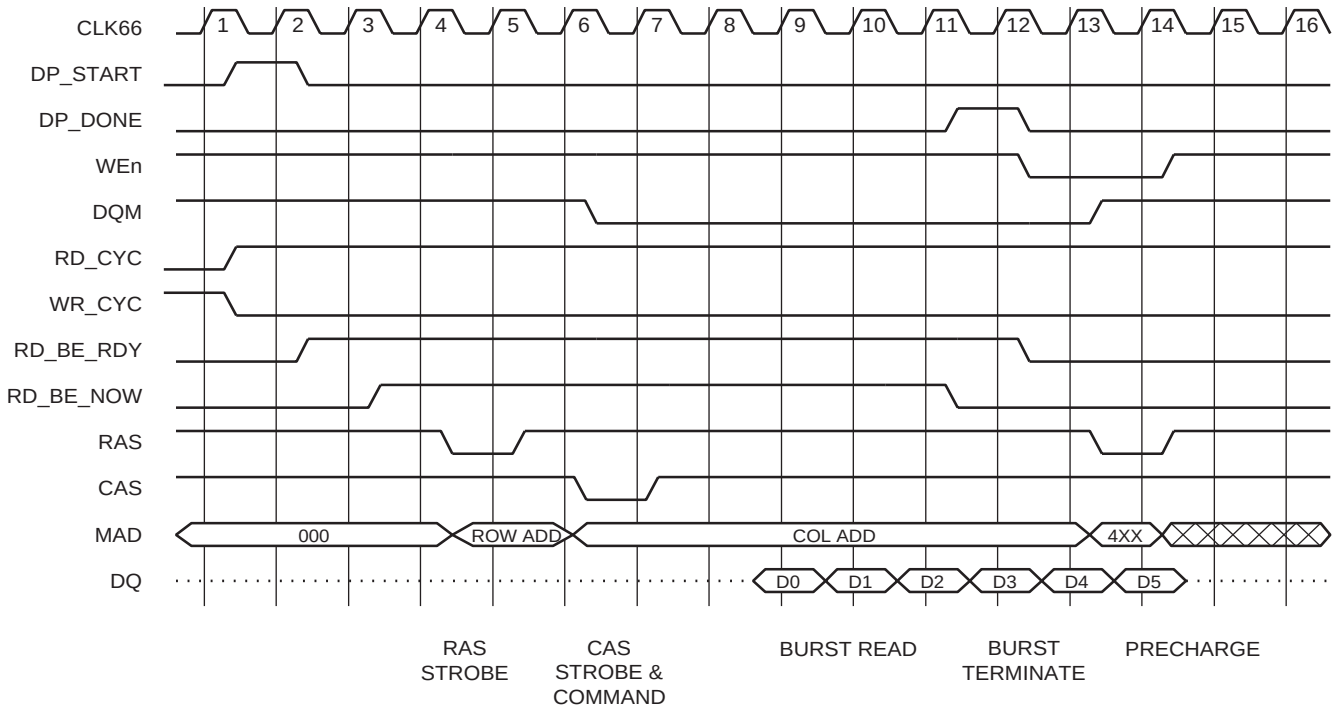
SDRAMs are required to be initialized in a predefined manner. Once power has been applied and the clock is stable, the SDRAM requires a 100 micro second delay prior to applying an executable command. The controller accommodates this requirement and the requirement of applying NOP commands during this period. After the 100 micro second delay, the controller initiates a PRECHARGE command that places the device in ALL BANKS IDLE state. Once in the IDLE state two AUTO REFRESH cycles are performed. After the AUTO REFRESH cycles are performed the Mode Register is written by the controller. The Mode Register is written with the following values (See Micron SDRAM data sheet for more details):

- Write burst mode – is set to '0' for programmable burst length.
- CAS latency – is set to '010' for a CAS latency of 2.
- Burst type – is set to '0' for sequential burst type.
- Burst length – is set to '111' for full page burst length.

A read cycle is initiated when DP_START and RD_CYC are active with the rising edge of the clock. The controller commences the burst read terminating only when the DP_DONE signal is activated. A write cycle is initiated when DP_START and WR_CYC are active with the rising edge of the clock. The controller commences the burst write terminating only when the DP_DONE signal is active. SDRAM writes and reads are shown in Figure 20 and Figure 21.

***CAUTION: Full page bursts are 256 locations, so the DP_DONE signal must occur before the 257 location is written or read otherwise the SDRAM device will not behave properly.***

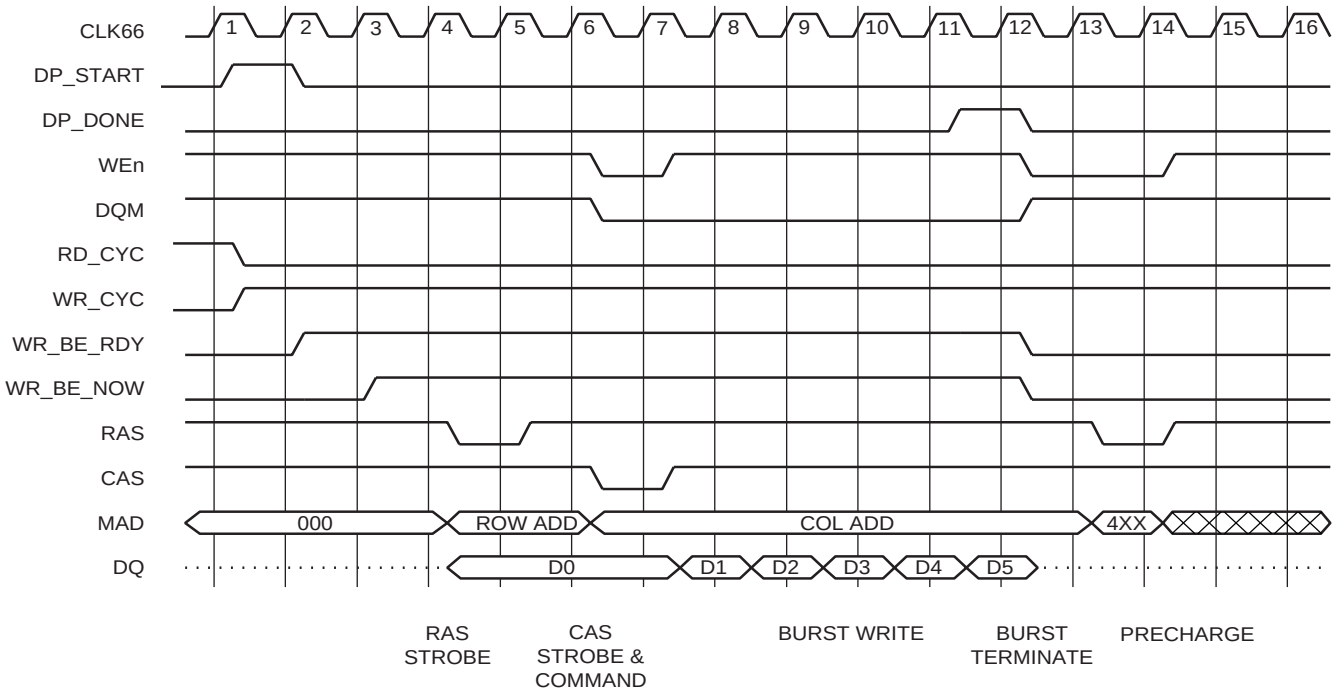
For refresh sequences, the controller provides an auto refresh sequence to the device every 15.6 micro seconds. The refresh sequence is show in Figure 22.




***Notes:***

1. *The SDRAM controller asserts RD_BE_RDY immediately following DP_START.*
2. *When the RD_BE_NOW signal becomes active, the PCI controller will expect data six cycles later. Once the initialization is complete, then data is expected every cycle where RD_BE_NOW is active.*
3. *At the completion of a cycle, DP_DONE active, the SDRAM controller terminates the transfer and precharges all banks.*

***Figure 20 •*** *SDRAM Burst Read*




***Notes:***

1. *The SDRAM controller asserts WR_BE_RDY immediately following DP_START.*
2. *When the WR_BE_NOW signal becomes active, the PCI controller will provide data four cycles later. Once the initialization is complete, then data will be provided every cycle where WR_BE_NOW is active.*
3. *At the completion of a cycle, DP_DONE active, the SDRAM controller terminates the transfer and precharges all banks.*

***Figure 21 •*** *Burst Write to SDRAM*

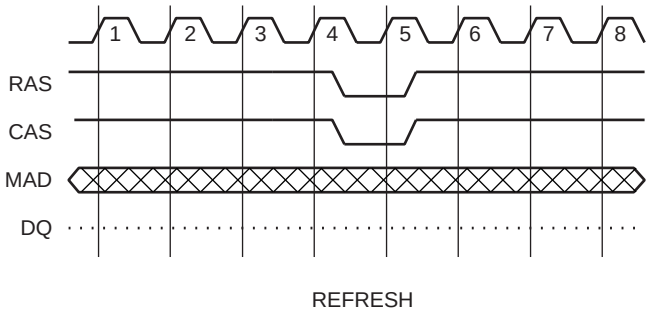



***Figure 22 •*** *SDRAM Refresh.*

http://www.actel.com

**Actel Europe Ltd.**
Daneshill House, Lutyens Close
Basingstoke, Hampshire RG24 8AG
United Kingdom
**Tel:** +44.(0)1256.305600
**Fax:** +44.(0)1256.355420

**Actel Corporation**
955 East Arques Avenue
Sunnyvale, California 94086
USA
**Tel:** 408.739.1010
**Fax:** 408.739.1540

**Actel Japan**
EXOS Ebisu Bldg. 4F
1-24-14 Ebisu Shibuya-ka
Tokyo 150 Japan
**Tel:** +81.(0)3.3445.7671
**Fax:** +81.(0)3.3445.7668

5172145-0/1.99