CorePCI Target+DMA 33/66 MHz

Product Summary

Intended Use

- High-Performance PCI Target+DMA Applications
  - 32/64-Bit, 66MHz, 3.3V (66MHz requires the 54SX72A device)
  - 32/64-Bit, 33MHz, 5V or 3.3V
- Back-End Support for Synchronous DRAM, I/O Subsystems, and SRAM

Key Features

- Standard Memory Base Address Register
- Optional 2nd Base Address Register (I/O or Memory)
- User-Selectable Address Space Size
- Interrupt Capability
- Flexible Back-end Data Flow Control

Data Transfer Rates

- Fully Compliant Zero Wait State Burst (32-Bit or 64-Bit Transfer Each Cycle)
- Optional Paced Burst (Wait states inserted between transfers)

Targeted Devices

- 54SX Family: A54SX16P, A54SX32A, A54SX72A
- 42MX Family: A42MX24, A42MX36

Design Source Provided

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

Synthesis and Simulation Support

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

Macro Verification

- Actel-Developed Test Bench

Compliance

- Hardware Tested
- I/O Drive Compliant in Targeted Devices
- Compliant with the PCI 2.2 Specification.

Version

This data sheet defines the functionality of Version 5.1 of the CorePCI macro.

General Description

The CorePCI Target+DMA connects I/O, memory, and processor subsystem resources to the main system via the PCI bus. The CorePCI Target+DMA macro is intended for use with a wide variety of peripherals where high-performance data transactions are required. Figure 1 depicts typical system applications using the baseline macro. The CorePCI Target+DMA macro 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 33/66MHz CorePCI Target+DMA macro has been developed for specific devices in the 54SX and 42MX families from Actel. The CorePCI Target+DMA can handle any transfer rate; however, most applications will operate at zero wait states. When required, wait states can be automatically inserted by a slower peripheral.

The core consists of three basic units: the Target+DMA, the back-end, and the wrapper. The Target+DMA 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 controller through a generic interface. The

wrapper combines the Target+DMA block and the back-end for implementation in a single Actel device.

The CorePCI Target+DMA macro can be customized in two different ways. First, 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.

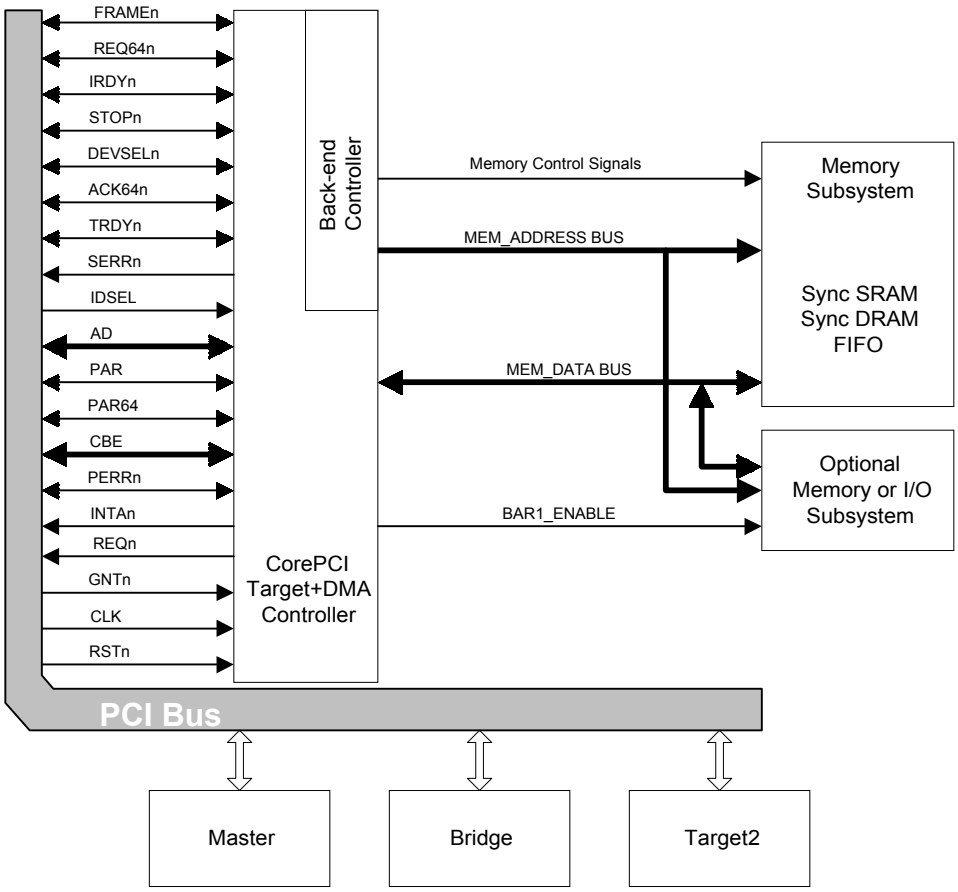


Figure 1 • *System Block Diagram Depicting Target+DMA Macro Usage*

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 address phase state machine, the dataphase state machine, datapath, parity, and the configuration block.

DMA State Machine

The DMA state machine is responsible for obtaining master ownership of the PCI bus and launching a data transfer transaction by asserting FRAMEn. 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+DMA controller. When a hit is detected, the DP_START/DP_START64 signals are activated, setting off the dataphase machine and back-end logic. The address phase state machine also determines the cycle type and provides this on the RD_CYC, WR_CYC, BAR0_MEM_CYC, BAR1_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. When operating as a target, the PCI outputs are TRDYn, DEVSELn, and STOPn. When operating as a master, IRDYn is the primary PCI output.

Data transfers to the back-end are coordinated using the signals RD_BE_RDY, RD_BE_NOW, WR_BE_RDY, and WR_BE_NOW. The two "BE_RDY" inputs indicate that the back-end is ready to transmit or receive data. The "BE_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.

Datapath

The datapath block provides the steering and registers for the data between the PCI bus and the back-end. Additionally, Datapath contains the address counters and increments the value after each data transaction.

Parity

The parity block generates and checks parity on the PCI bus.

Configuration

The configuration block contains the configuration space for the Target+DMA controller. These registers include the ID registers, status and control registers, and the base address registers. This block also includes the address, count, and control registers associated with DMA transfers.

I/O Signal Descriptions

The PCI and back end signals are defined in Tables 1 and 2. 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.

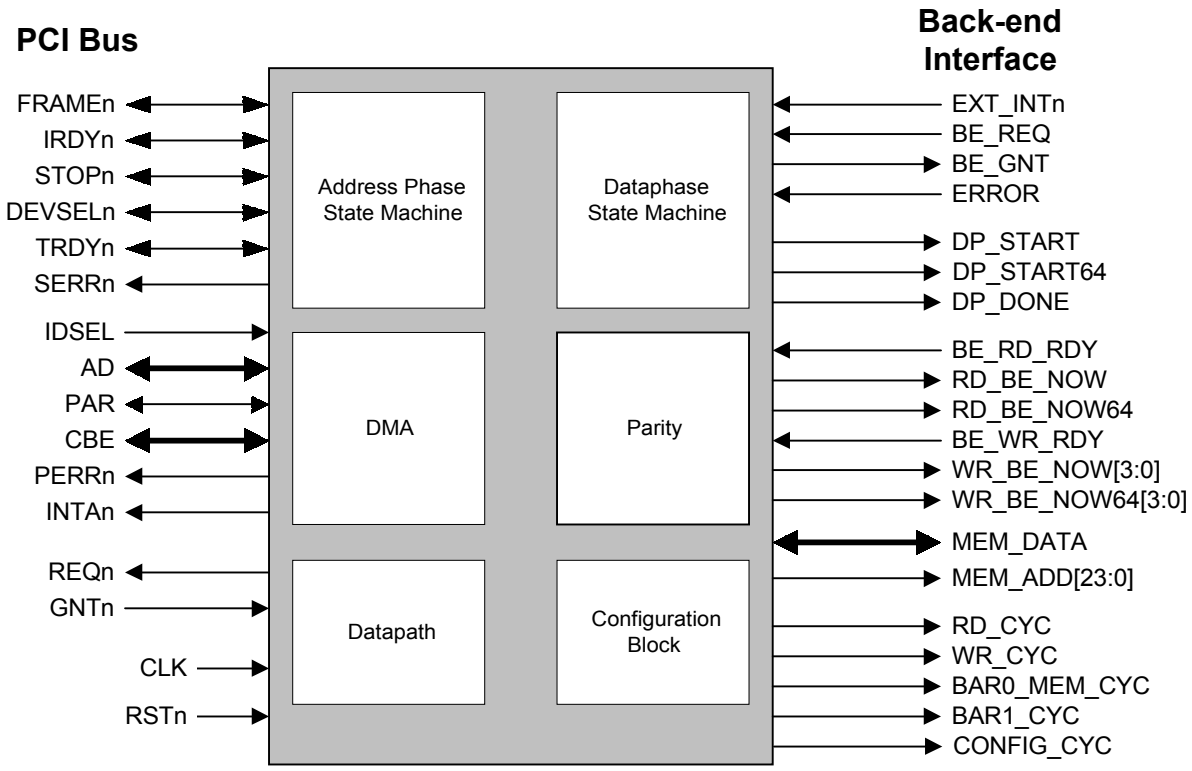


***Figure 2** • Block Diagram of the CorePCI Target+DMA Macro*

Table 1 • CorePCI Target+DMA Signals

Name[1]	Type	Description
CLK	Input	33MHz or 66MHz clock input for the PCI macro.
RSTn	Input	Active LOW asynchronous reset.
AD	Bi-Directional	Multiplexed 32-bit or 64-bit address and data bus. Valid address is indicated by FRAMEn assertion.
CBE	Bi-Directional	Bus command and byte enable information. During the address phase, the lower 4 bits define the bus command. During the data phase, they define the byte enables. This bus is 4 bits for 32-bit PCI systems and 8 bits in 64-bit systems.
PAR	Bi-Directional	Parity signal. Parity is even across AD[31:0] and CBE[3:0]
PAR64	Bi-Directional	Upper parity signal. Parity is even across AD[63:32] and CBE[7:4]. This signal is not required for 32-bit PCI systems.
FRAMEn	Bi-Directional (STS)	Active LOW signal indicating the beginning and duration of an access. While FRAMEn is asserted, data transfers continue.
REQ64n	Bi-Directional (STS)	Active LOW signal with the same timing as FRAMEn indicating that the Master requests a data transfer over the full 64-bit bus. This signal is not required for 32-bit PCI systems.
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.
ACK64n	Bi-Directional (STS)	Active LOW output from the target indicating that it is capable of transferring data on the full 64-bit PCI bus. This signal is driven in response to the REQ64n signal and has the same timing as DEVSELn. This signal is not required in 32-bit PCI systems.
REQn	Output	Active LOW output used to request bus ownership. This signal is asserted by the PCI Target+DMA Controller whenever DMA mode is enabled.
GNTn	Input	Active LOW input from the system arbiter indicating that the Target+DMA controller has mastership of the PCI bus.
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.
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[1,2]	Type	Description
BAR0_MEM_CYC	Output	Active high signal indicating a transaction to the memory space defined in the base address register zero (BAR0) located at 10H in Configuration Header Space.
BAR1_CYC	Output	Active high signal indicating a transaction to the optional memory or I/O space defined in base address register one (BAR1) located at 14H in Configuration Header Space.
CONFIG_CYC	Output	Active high signal indicating a transaction to configuration space.
READ_CYC	Output	Active high signal indicating a read transaction from the back end.
WRITE_CYC	Output	Active high signal indicating a write transaction to the back end.
MEM_DATA	Bi-Directional	32-bit or 64-bit bi-directional data bus.
MEM_ADD[N:0]	Output	Memory address bus. N is defined by the variable MADDR_WIDTH-1.
DP_START DP_START64	Output	DP_START is an active high pulse indicating that a PCI transaction to the back end is beginning. If the transfer is 64-bit, then DP_START64 will be asserted at the same time as DP_START.
DP_DONE	Output	Active high pulse indicating that a PCI transaction to the back end has finished.
RD_BE_NOW RD_BE_NOW64	Output	When active high, these signals indicate that the PCI controller will read data on the MEM_DATA bus on the next rising clock edge. These signals are active whenever both the back-end (as indicated by RD_BE_RDY) and the PCI bus (as indicated by IRDYn) are ready to transmit data. The RD_BE_NOW indicates a read from the lower 32-bits of data and the RD_BE_NOW64 indicates a read from the upper 32-bits of data. Function of these signals are also impacted by the PIPE_FULL_CNT bus.
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] WR_BE_NOW64[3:0]	Output	When active high, these signals indicate that the PCI controller is providing valid write data on the MEM_DATA bus. These signals are active whenever both the back-end (as indicated by WR_BE_RDY) and the PCI bus (as indicated by IRDYn) are ready to transmit data. The WR_BE_NOW indicates a read from the lower 32-bits of data and the WR_BE_NOW64 indicates a read from the upper 32-bits of data. For WR_BE_NOW, each bit represents a byte enable with bit 0 corresponding to the least significant byte (byte 0) on the MEM_DATA bus. Similarly, for WR_BE_NOW64, each bit represents a byte enable with bit 0 corresponding to byte 4 on the MEM_DATA bus. Function of these signals are also impacted by the PIPE_FULL_CNT bus.
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 time limits defined by the PCI bus, then a disconnect without data will be initiated.

Notes:

1. *Active LOW signals are designated with a trailing lower-case n instead of #.*
2. *Signals ending in "CYC" become valid the same cycle DP_START is active and will remain valid throughout the current cycle (until DP_DONE is asserted).*

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

Name[1,2]	Type	Description
PIPE_FULL_CNT[2:0]	Input	Normally, the address on MEM_ADDRESS and the data on MEM_DATA are coincident. In some back-ends, like synchronous SRAMs, the data lags the address by one or more cycles. The PIPE_FULL_CNT bus feeds a latency timer in the PCI controller to help in these cases. When the PIPE_FULL_CNT is non-zero, the PCI controller will increment the address the number of counts defined and will not expect data until the count expires. The RD_BE_NOW and WR_BE_NOW signals need to be ignored during the time-out. For example, if PIPE_FULL_CNT is set to “010”, then the *_NOW signals should be ignored the first two cycles they are active while the address is initially incremented.
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 giving control to the back-end. 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.
ERROR	Input	Active high signal which will force the PCI controller to terminate the current transfer with a target abort cycle.
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 #.*
2. *Signals ending in “CYC” become valid the same cycle DP_START is active and will remain valid throughout the current cycle (until DP_DONE is asserted).*

Supported Target 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 default memory size is 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 configuration write command is used to write data into configuration space. 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

Supported DMA Master Commands

When the Target+DMA controller is acting as a PCI bus master, it only supports memory read (0110) and memory write (0111) commands.

Configuration Space

The PCI specification defines a 64-byte space (configuration header) to define various attributes of the

Data Transactions

The 66MHz CorePCI Target+DMA macro is designed to be fully compliant for all transfer types, including single DWORD and burst transactions. Burst transactions can operate with either zero, one, or more wait states. Either the master or the back end can pace the rate of transactions. If both can operate at zero wait states, then no wait states will be inserted. However, if either the master or the back end requires more than one cycle to complete the transaction, then wait states can be added automatically.

The back-end can inject wait states on the PCI bus by de-asserting the “BE_RDY” signals during a transfer. When the macro is behaving as a target, then the Target+DMA controller will de-assert the TRDYn signal. When the macro is a master, then it will de-assert the IRDYn signal.

Device Utilization

Utilization statistics for targeted devices are listed in Table 4. Each back-end requires different amounts of logic depending on the complexity of the controller is shown as an example..

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

Function	42MX[1]	54SX[2]
32-bit Target+DMA Controller	270/690	270/530
64-bit Target+DMA Controller	390/940	390/700
32-bit SDRAM Controller	60/130	60/90
64-bit SDRAM Controller	70/140	70/110

Notes:

1. *The first number represents the number of S-modules and the second number is the total number of 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 significant 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.

PCI Target+DMA, as shown in Table 5. All registers shown in bold are implemented in this Target+DMA, 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 registers, 40h, 44h, and 48h, are used to define DMA and interrupt specific status and controls. These registers can be addressed in configuration space or can optionally be mapped to an I/O space using base address register #2.

Read-Only Configuration Registers

The read-only registers listed in Table 5 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 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 registers 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 10)
- Base Address Register for I/O (Table 9 on page 10)
- Interrupt Register (Table 10 on page 10)
- User Configuration Registers (Tables 11, 12, and 13)

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 #0 (Memory Location for Baseline Target+DMA)				10h
Base Address #1 (Optional Memory or I/O)				14h
Base Address #2 (Optional I/O for DMA Register Mapping)				18h
Base Address #3				1Ch
Base Address #4				20h
Base Address #5				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 A value of '0' disables the device's response to I/O space addresses. Should be '0' after reset.
1	R/W	Memory Space A value of '0' disables the device's response to memory space addresses. Should be '0' after reset.
2	R/W	Bus Master When set to a '1', this bit indicates that the controller can act as a PCI master. Set to '0' after reset.
3	R/O	Special Cycles No response to special cycles. It is set to '0'.
4	R/O	Memory Write and Invalidate Enable Memory write and invalidate not supported. It is set to '0'.
5	R/O	VGA Palette Snoop Assumes non-VGA peripheral. It is set to '0'.
6	R/W	Parity Error Response When '0', the device ignores parity errors. When '1', normal parity checking is performed. '0' after reset.
7	R/O	Wait Cycle Control No data-stepping supported. It is set to '0'.
8	R/W	SERRn Enable When '0', the SERRn driver is disabled. It is set to '0' after reset.
9	R/O	Fast Back-to-Back Enable 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
3-0	R/O	Reserved—set to '0000'b.
4	R/O	Capabilities List. Currently no New Capabilities linked list is available. This bit is set to '0'.
5	R/O	66 MHz Capable Should be set to '1' to indicate a 66 MHz target, or '0' to indicate a 33MHz target.
6	R/O	UDF Supported Set to '0'—no user definable features.
7	R/O	Fast Back-to-Back Capable Set to '0'—fast back-to-back to same agent only.
8	R/O	Data Parity Error Detected Set to '0'—only used by Masters.
10-9	R/O	DEVSELn Timing Set to '10'—slow DEVSELn response.

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
11	R/W	Signaled Target Abort Set to '0' at system reset. This bit is set to a '1' by internal logic whenever a target abort cycle is executed.
12	R/W	Received Target Abort 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 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 Set to '0' at system reset. This bit is set to '1' by internal logic whenever the SERRn signal is asserted by the Target+DMA.
15	R/W	Detected Parity Error Set to '0' at system reset. This bit is set to '1' by internal logic whenever a parity error, address or data, is detected regardless of the value of bit 6 in the command register.

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 • Memory Base Address Register Bit Definition— (Locations 10h or 14h)

Bit	Type	Description
0	R/O	Indicates memory space. It is set to '0'.
2-1	R/O	Set to '00' to indicate mapping into any 32-bit address space.
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 13 for more information.*

Table 9 • I/O Base Address Register Bit Definitions—(Location 14h and 18h)

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 all '0's.
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 13 for more information. For location 18h, the register definition is as shown.*

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 13 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. 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. 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 13 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 Bit is set when DMA is terminated because of a master abort. Writing a '1' clears this bit.
1	R/W	DMA Abort Writing a '1' to this bit aborts the current DMA transfer. This bit always reads as a '0'.
2	R/O	DMA Done A '1' indicates that the DMA transfer is done. The bit is cleared by issuing a new DMA request.
3	R/W	DMA Direction 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 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 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 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 A '1' enables the DMA interrupt to drive the PCI INTAn signal.
8	R/W	DMA Interrupt Status 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 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 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 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').

Customization Options

The PCI Target+DMA has a variety of options for user customization. A special package is included with the source design files that defines a list of variables that allow the user to optimize the macro for his or her 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

In addition to the read only configuration definitions, the CorePCI Target+DMA macro offers a variety of customization options summarized as follows:

- 32-bit or 64-bit data size (BIT64)
- 33 or 66MHz operation (MHZ_66)
- BAR0 address size (MADDR_WIDTH)
- Optional BAR1 definitions (BAR1_ENABLE, BAR1_IO_MEMORY, BAR1_ADDR_WIDTH, and BAR1_PREFETCH)
- Option to have the DMA registers mapped into I/O space (DMA_IN_IO)

Table 14 • CorePCI Target+DMA 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.
BIT64	Binary	Defines whether the macro should behave as a 32-bit (‘0’) or a 64-bit (‘1’) PCI controller.
MHZ_66	Binary	Defines the value of bit 5 in the status register.
DMA_IN_IO	Binary	When this constant is set to a ‘0’, the DMA registers are mapped into configuration space at addresses 40h, 44h, and 48h. If the constant is set to a ‘1’, then the DMA registers are mapped into I/O space defined by base address register 2. The I/O port addresses for the DMA registers is at 40h, 44h, and 48h.
MADDR_WIDTH	Integer	Defines memory space size for base address register zero. Allowable range is 4-31 where 4 represents 16 bytes and 24 represents 16 Mbytes of memory space.
BAR1_ENABLE	Binary	This constant enables (‘1’) base address register 1 (14h) to be either a memory or an I/O space.
BAR1_IO_MEMORY	Binary	Defines the type of base address register one. A memory is defined by a ‘1’ and an I/O is defined by a ‘0’.
BAR1_ADDR_WIDTH	Integer	Defines memory or I/O space size for base address register one. An integer setting of N in this field corresponds to 2**N bytes. Allowable range for memory is 4-31 where 4 represents 16 bytes and 24 represents 16 Mbytes of memory space. Valid range for I/O is 2 to 8.
BAR1_PREFETCH	Binary	When BAR1 is a memory BAR, this constant defines whether or not it is prefetchable.

Device Selection

PCI performance requirements and bus size both drive device selection. For 66MHz systems, only 54SX devices are able to meet all PCI requirements. For 33MHz systems, either the 54SX or 42MX family devices may be used. A typical 64-bit PCI system requires at least 200 I/O's. This I/O count mandates the use of the A42MX36 in the 42MX family and either the A54SX32A or A54SX72A devices in the 54SX family. Table 15 is a summary of the minimum device requirements for various PCI size/performance options. In order to meet the PCI

Table 15 • Minimum Device Requirements

Function	42MX	54SX[1]
32-bit, 33MHz	A42MX24-2	A54SX16P-2
64-bit, 33MHz	N/A	A54SX32A-3
32-bit, 66MHz	N/A	A54SX72A-2
64-bit, 66MHz	N/A	A54SX72A-3

Notes:

1. All references to the A54SX72A are preliminary.

timing requirements for output valid (6ns for 66MHz, 11ns for 33MHz) and input setup (3ns for 66MHz, 7ns for 33MHz) times, the speed grades shown in Table 15 must be used.

System Timing

System timing for the CorePCI Target+DMA macro is defined in Tables 16 and 17 should be used in conjunction with the timing waveforms in the next section. The required timing as defined in the PCI specification is included in the PCI timing tables. Input set-up is defined in Figure 3, and output valid delays are described in Figure 4. To meet the PCI timing specifications in MX devices, the core must be operated at 5V; however, the I/O can be either 3.3V or 5V.

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, 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.

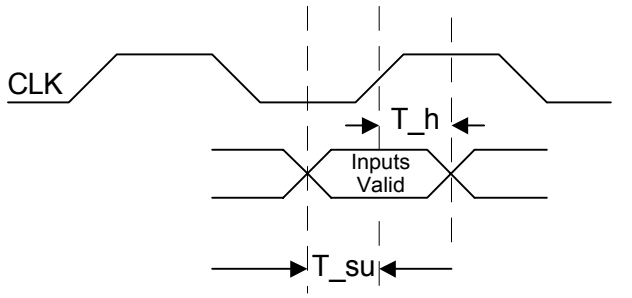


Figure 3 • Input Timing for PCI Signals

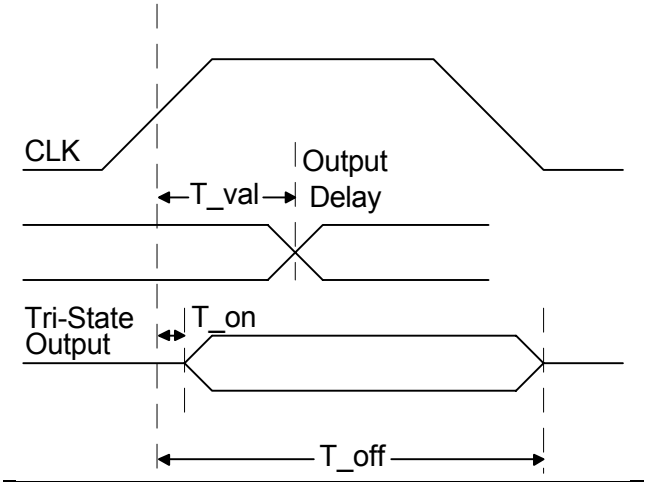


Figure 4 • Output Timing for PCI Signals

Back-End I/O Signal Timing

Table 16 and Table 17 summarize the set-up and hold times for key back-end interface input and output signals, respectively.

Table 16 • Generic Interface Input Set-Up Times (ns max)

Name	42MX-2	54SX-2
RD_BE_RDY	13	9
WR_BE_RDY	16	9
BE_REQ	4	3
EXT_INTn	7	5
MEM_DATA	5	4

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.

Table 17 • *Generic Interface Output Valid Times (ns max)*

Name	42MX-2	54SX-2
MEM_BAR0_CYC	6	4
BAR1_CYC	6	4
CONFIG_CYC	7	7
READ_CYC	7	6
WRITE_CYC	9	7
MEM_DATA	14	8
MEM_ADD	11	7
DP_START	8	6
DP_START64	8	6
DP_DONE	8	7
RD_BE_NOW	11	7
RD_BE_NOW64	11	7
WR_BE_NOW	7	7
WR_BE_NOW64	7	7

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.*

PCI Waveforms

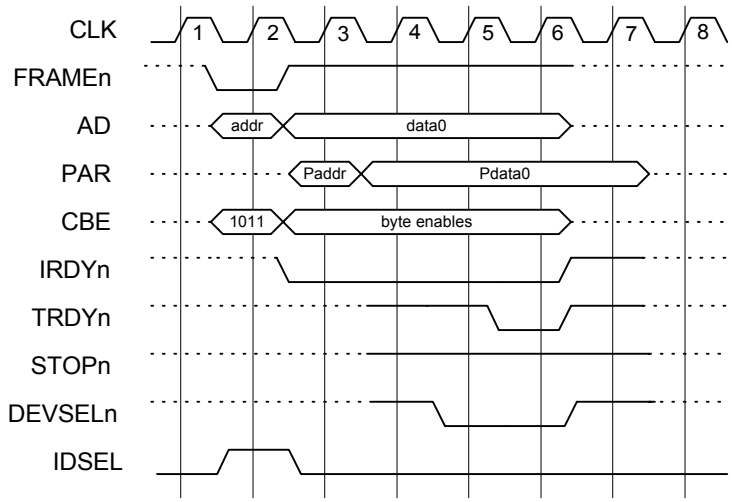
Revision 5.1 of the CorePCI macro supports both 32-bit and 64-bit data transfers. Configuration and I/O cycles are limited to 32-bit transfers; however, memory transactions can be either. For simplicity's sake, most of the waveforms are shown for 32-bit transfers. To move from 32-bit to 64-bit, a set of 64-bit control signals are supplied by both the back-end as well as the PCI bus. These signals mirror their 32-bit counterparts with identical function and timing and are as follows:

- REQ64n is the same as FRAMEn
- ACK64n is the same as DEVSELn
- PAR64 is the same as PAR
- DP_START64 is the same as DP_START
- RD_BE_NOW64 is the same as RD_BE_NOW
- WR_BE_NOW64 is the same as WR_BE_NOW

In addition to these control signals, the AD and MEM_DATA buses are 64-bit rather than 32-bit. Also, the CBE bus is expanded from 4-bits to 8-bits. A complete example of a 64-bit read and write is illustrated in Figure 9 and Figure 10.

Configuration Cycles

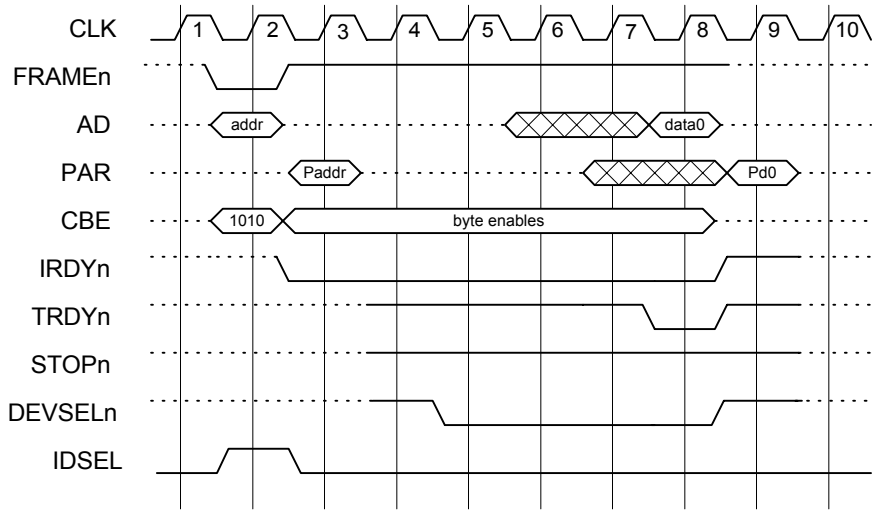
Configuration read and write cycles are used to define and determine the status of the Target+DMA's internal configuration registers. Configuration cycles are the only type of transactions that use the IDSEL signal. Register selection is defined by the contents of the address (bits 7 down to 2). A configuration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



Notes:

1. *If the Target+DMA's IDSEL is asserted when FRAMEn is asserted and the command bus is '1011', then a configuration write cycle is indicated.*
2. *The Target+DMA 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+DMA's IDSEL is asserted when FRAMEn is asserted and the command bus is '1010', then a configuration read cycle is indicated.*
2. *The Target+DMA 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*

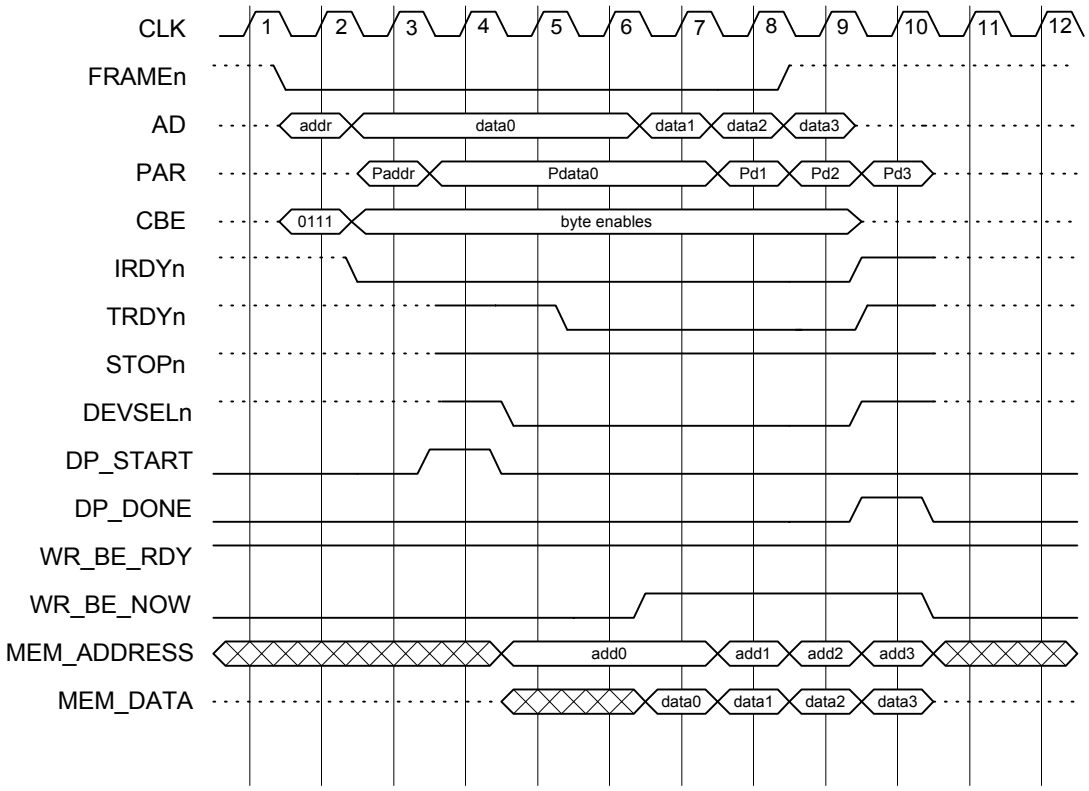
Zero Wait State Burst Transactions

Zero wait state bursting enables transfer of a DWORD (32-bit PCI) or two DWORDs (64-bit PCI) for every clock cycle. All cycles are initiated with DP_START/DP_START64 indicating a hit to the Target+DMA. The back-end should then look at the BAR0_MEM_CYC and BAR1_CYC to determine which space is being addressed. The RD_CYC and WR_CYC signals define the direction of the transfer. All of the *_CYC signals become valid during the DP_START pulse cycle and will remain in the this state until the next DP_START occurs. If a DP_START64 is coincident with a DP_START, then the transaction is expected to be 64-bits wide.

For PCI writes, the back-end indicates that it is prepared to receive data by setting the WR_BE_RDY signal high. Valid data to the back-end is qualified by the WR_BE_NOW bus. For PCI reads, the back-end indicates that it is prepared to provide read data by setting the RD_BE_RDY signal. The PCI controller will respond on a following cycle with a RD_BE_NOW signal which qualifies the read data. The data is then transferred to the PCI bus on the following cycle. In either the read or write case, the macro will automatically increment the address.

In the case of a PCI read, the back-end must prefetch the memory data in order to ensure continuity on long bursts. If prefetching causes a problem, for example in a FIFO, then the back-end logic should shadow the last two data transactions.

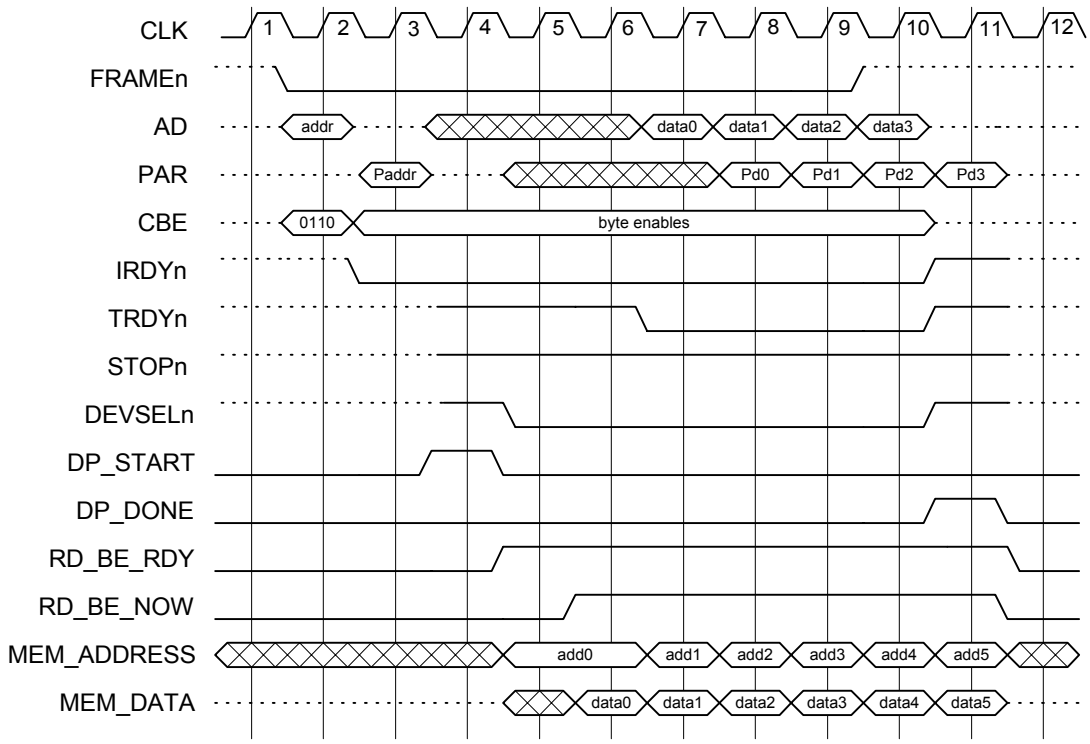
32-bit zero wait state burst transfers are shown in Figure 7 and Figure 8. 64-bit zero wait state burst transfers are illustrated in Figure 9 and Figure 10.



Notes:

1. *When FRAMEn is asserted and the command bus is '0111', then a write to memory space is indicated.*
2. *The Target+DMA will compare the address to the programmed space set in the memory base address register.*
3. *If an address hit occurs, then the Target+DMA 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 address will increment each cycle following an active RD_BE_NOW.*
6. *The PCI transaction completes when TRDYn is de-asserted in cycle 9 and completes on the back-end in cycle 10.*
7. *For this case, the PIPE_FULL_CNT is set to "000" (See "Back-end Latency Control" on page 26. for more information)*

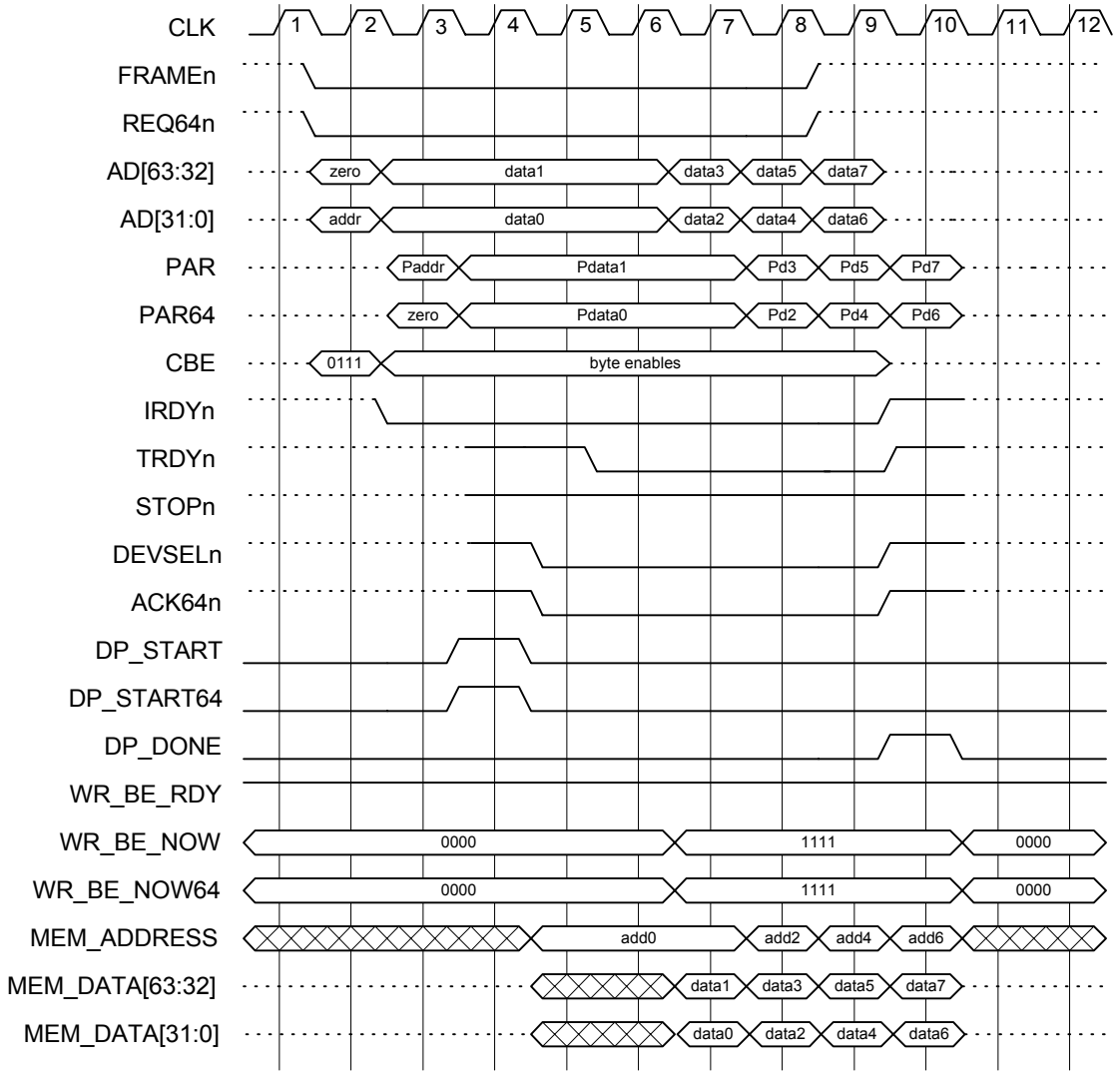
Figure 7 • *Burst Write with Zero Wait States*



Notes:

1. *When FRAMEn is asserted and the command bus is '0110', then a read from memory space is indicated.*
2. *The Target+DMA will compare the address to the programmed space set in the memory base address register.*
3. *If an address hit occurs, then the Target+DMA 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 address will increment each cycle following an active RD_BE_NOW.*
6. *The PCI transaction completes when TRDYn is de-asserted in cycle 10.*
7. *For this case, the PIPE_FULL_CNT is set to "000" (See "Back-end Latency Control" on page 26. for more information)*

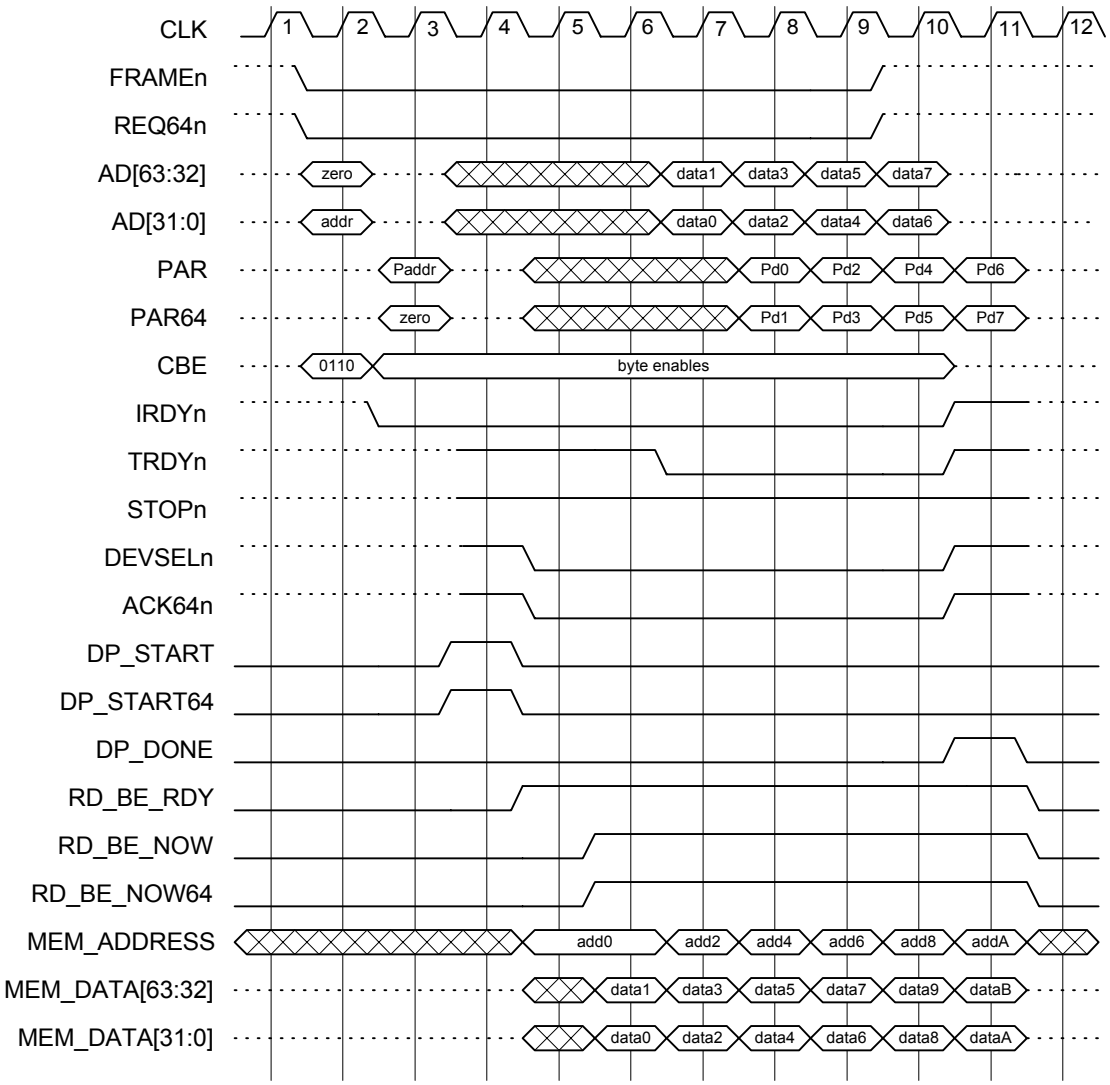
Figure 8 • *32-bit Burst Read with Zero Wait States*



Notes:

1. *When FRAMEn and REQ64n is asserted and the command bus is '0111', then a 64-bit write to memory space is indicated.*
2. *The Target+DMA will compare the address to the programmed space set in the memory base address register.*
3. *If an address hit occurs, then the Target+DMA asserts DP_START and DP_START64 in cycle 3 and claims the PCI bus by asserting DEVSELn and ACK64n 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. *For 64-bit transfers the MEM_ADDRESS will increment by 2 each cycle.*
6. *The PCI transaction completes when TRDYn is de-asserted in cycle 9 and completes on the back-end in cycle 10.*
7. *For this case, the PIPE_FULL_CNT is set to "000" (See "Back-end Latency Control" on page 26. for more information).*

Figure 9 • *64-bit Burst Write with Zero Wait States*



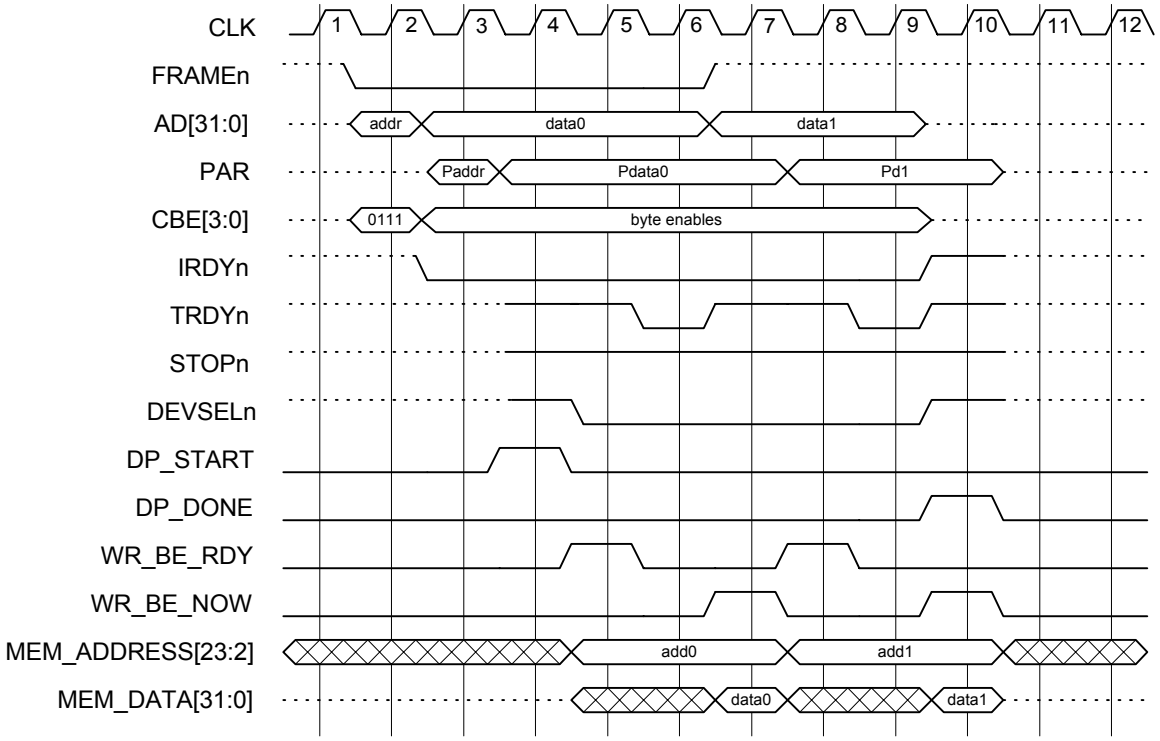
Notes:

1. *When FRAMEn and REQ64n is asserted and the command bus is '0110', then a 64-bit read from memory space is indicated.*
2. *The Target+DMA will compare the address to the programmed space set in the memory base address register.*
3. *If an address hit occurs, then the Target+DMA asserts DP_START and DP_START64 in cycle 3 and claims the PCI bus by asserting DEVSELn and ACK64n 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. *For 64-bit transfers, the MEM_ADDRESS will increment by 2 each cycle.*
6. *The PCI transaction completes when TRDYn is de-asserted in cycle 10.*
7. *For this case, the PIPE_FULL_CNT is set to "000" (See "Back-end Latency Control" on page 26. for more information)*

Figure 10 • *64-bit Burst Read with Zero Wait States*

Paced Transactions

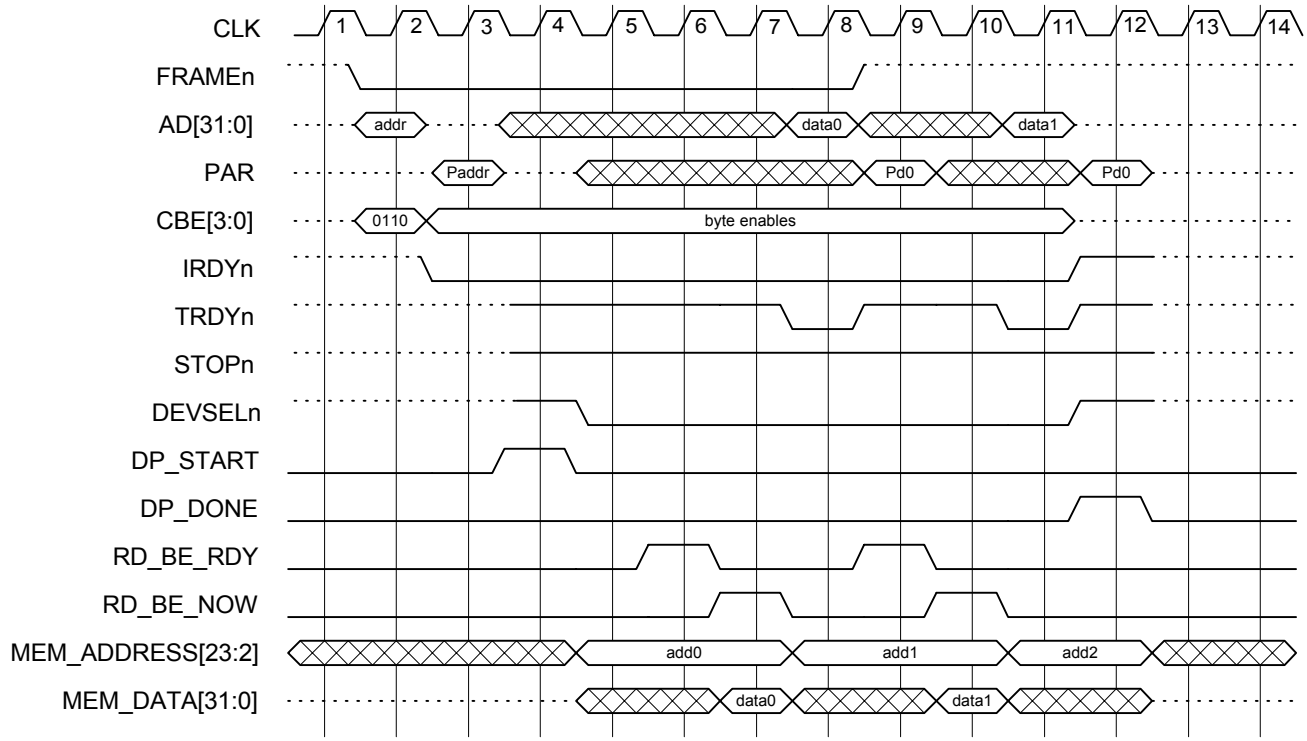
Back-end throttle transfers provide a handshake mechanism for supporting slow response devices. The back-end transactions are paced using the RD_BE_RDY and WR_BE_RDY signals. These signals can be used to pace either single DWORD or burst transactions. Figure 11 and Figure 12 illustrate this mechanism for a back-end that requires three cycles to respond to a read or write command from the PCI bus.



Notes:

1. *The WR_BE_RDY should be asserted two cycles before the back-end is ready to receive data.*
2. *The WR_BE_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 11 • *Write Using Back-End Throttling*



Notes:

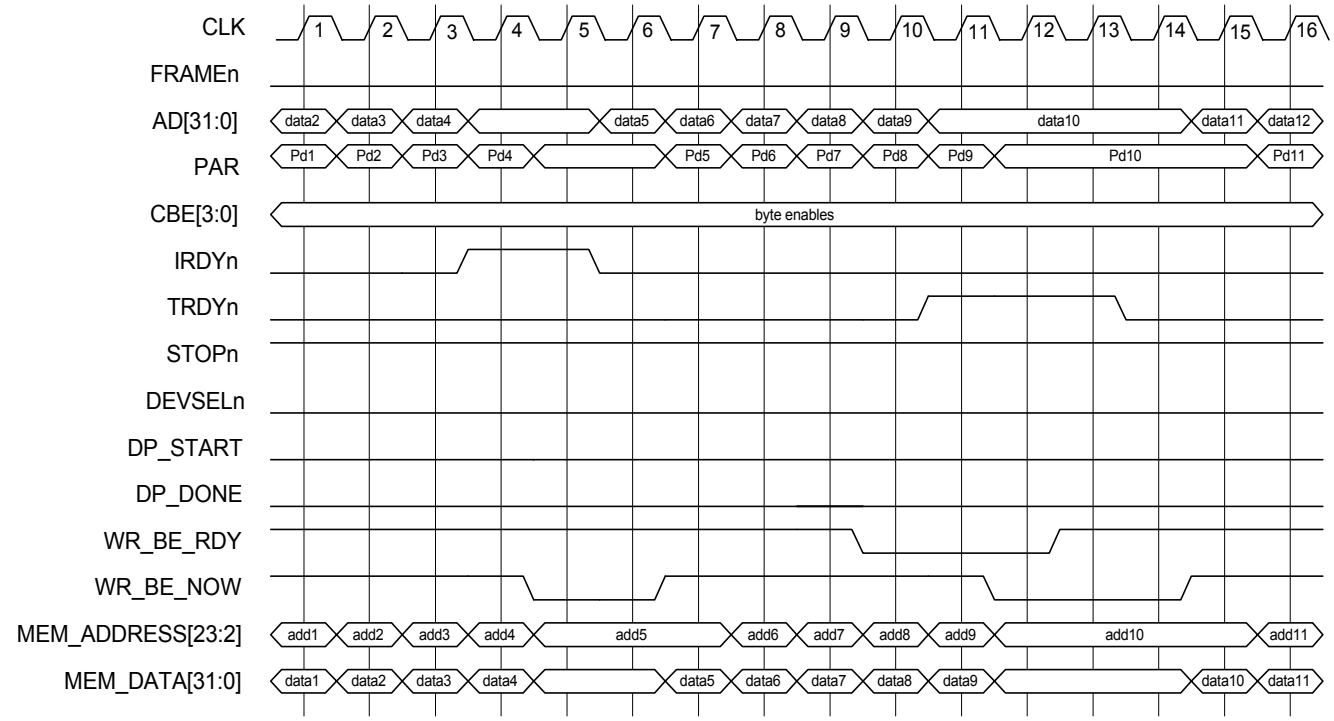
1. *The RD_BE_RDY should be asserted one cycle before the back-end is ready to transmit data.*
2. *The RD_BE_RDY signal will initiate assertion of RD_BE_NOW latching the data into the controller. The data transfer will complete when TRDYn is asserted on the following cycle.*
3. *The RD_BE_NOW should not be assumed to happen at this time because it is also dependent on the state of IRDYn.*

Figure 12 • *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.

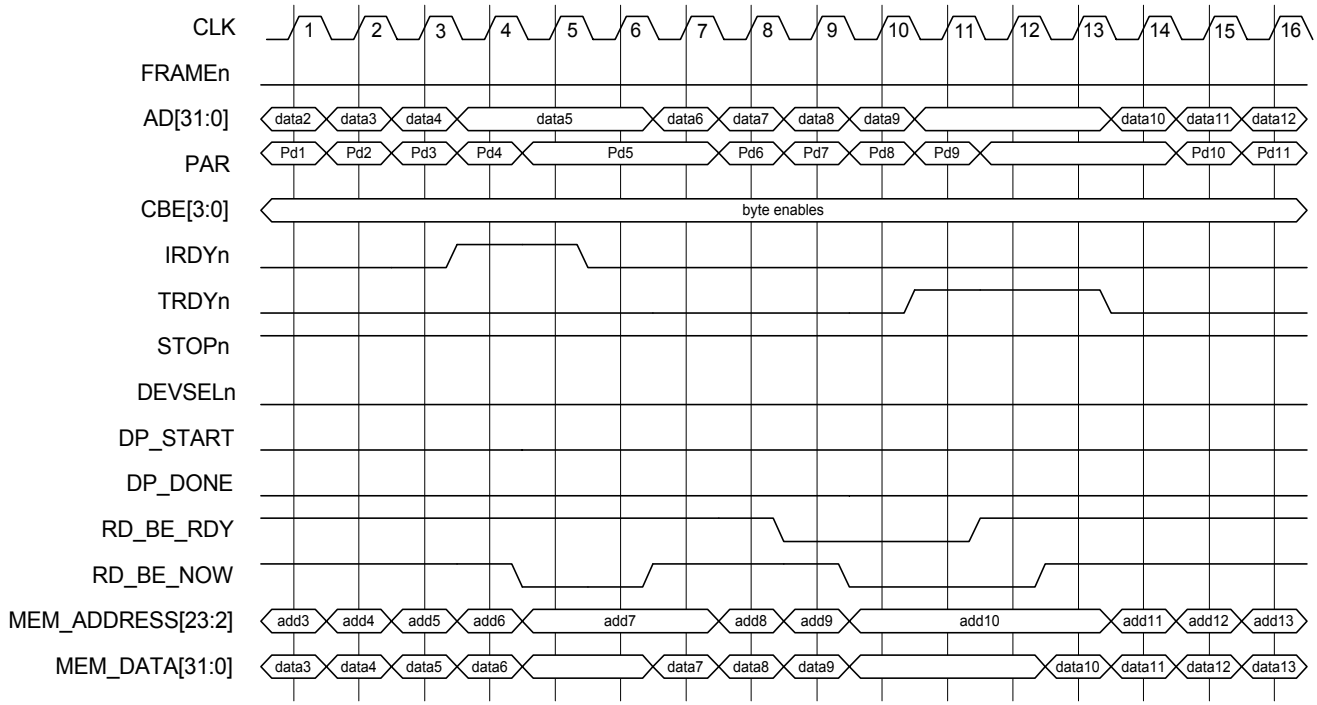
The back-end can insert wait states by de-assertion of the *_BE_RDY signals. These signals cause the Target+DMA controller to de-assert TRDYn and insert wait states on the PCI bus. For writes, the back-end must be prepared to accept up to two DWORDs of data prior to data transfer termination. For reads, the back-end must be prepared to transmit one DWORD of data prior to data transfer termination. Example paused transactions are shown in Figures 13 and Figures 14.



Notes:

1. *In the example, the flow of data is interrupted from the PCI master de-assertion of IRDYn in cycle 3. The PCI master inserts 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 WR_BE_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 13 • *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, 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 RD_BE_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 RD_BE_RDY is de-asserted, the RD_BE_NOW signal is driven inactive which is then followed by the de-assertion of TRDYn.*

Figure 14 • *PCI Read Illustrating IRDYn De-assertion*

Back-end Latency Control

Some back-ends require that the address must be available at least one cycle prior to data being valid. This is true for most synchronous back-ends. In order to support this need, the CorePCI macro provides the PIPE_FULL_CNT control bus to the back-end. This bus can be used to define the relative delay between address and data. When the PIPE_FULL_CNT is set to "000", then the data will expected to be coincident with the data and the data should be valid whenever the *NOW lines are asserted.

When PIPE_FULL_CNT is set to a non-zero value, then the operation of the back-end is as follows:

- The back-end asserts the *RDY signal
- The *NOW signal will assert, and the address will begin incrementing; however, the data will not be expected to be valid until N cycles after the value defined on the PIPE_FULL_CNT bus.
- Once the initial time-out occurs, valid data must be available whenever the *NOW signal is asserted.

Figure 15 is an example of this function for a read cycle with the PIPE_FULL_CNT set to "001".

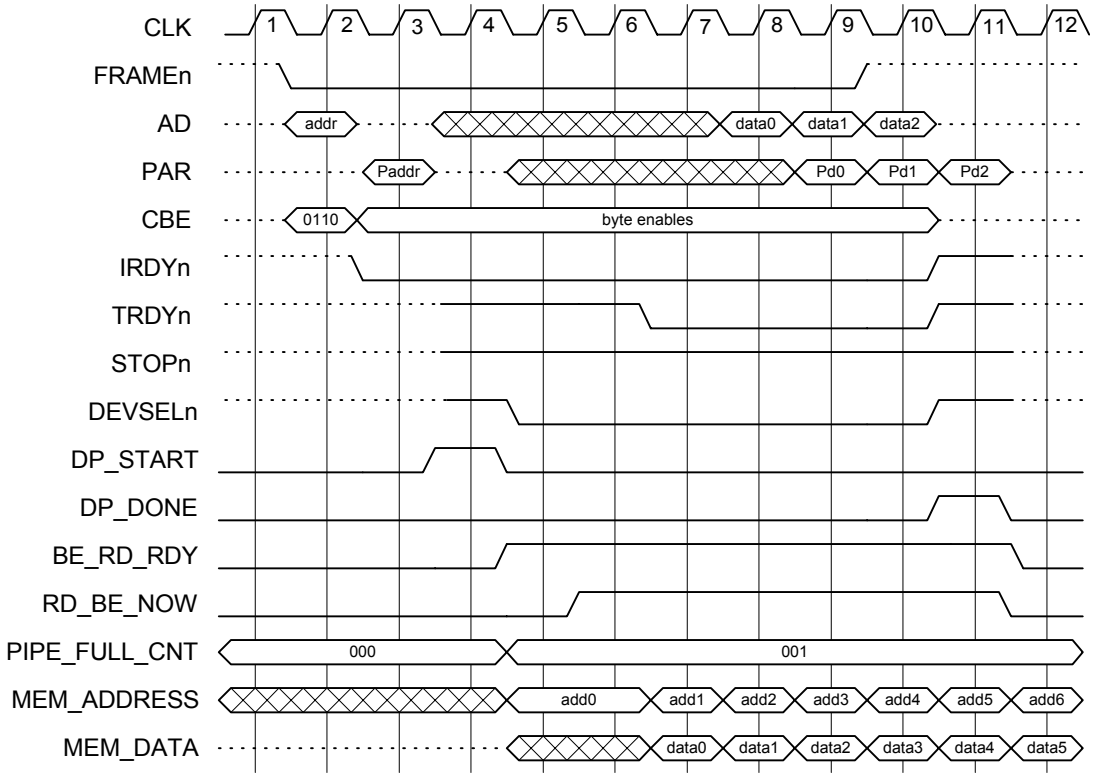
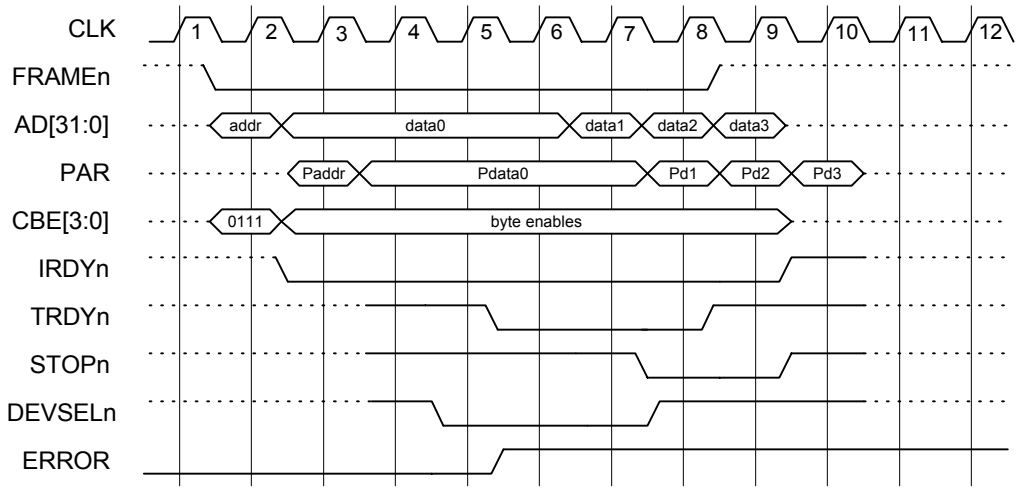


Figure 15 • *Back-end latency read transaction.*

Target Abort

A target abort occurs when an error condition occurs, as shown in Figure 16. When an error occurs on the back-end, this condition is reported with 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:

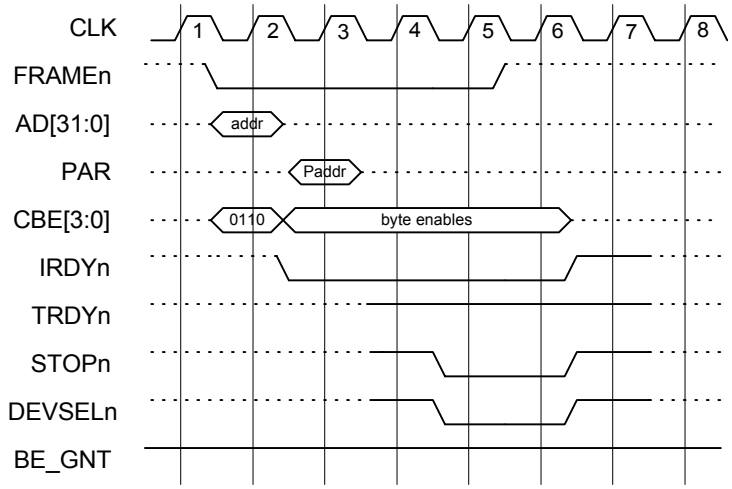
1. *During a PCI cycle, the back-end ERROR signal indicates that a problem occurred on the back-end such that the transfer cannot be completed.*
2. *The Target+DMA initiates a target abort by asserting STOPn and de-asserting DEVSELn in the same cycle.*
3. *The Master will begin cycle termination by de-asserting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

Figure 16 • *Target Abort Cycle*

Target Retry and Disconnect

When the back-end is busy or unable to provide the data requested, then the Target+DMA 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 17. The Target+DMA indicates that it is unable to respond by asserting STOPn and DEVSELn simultaneously.

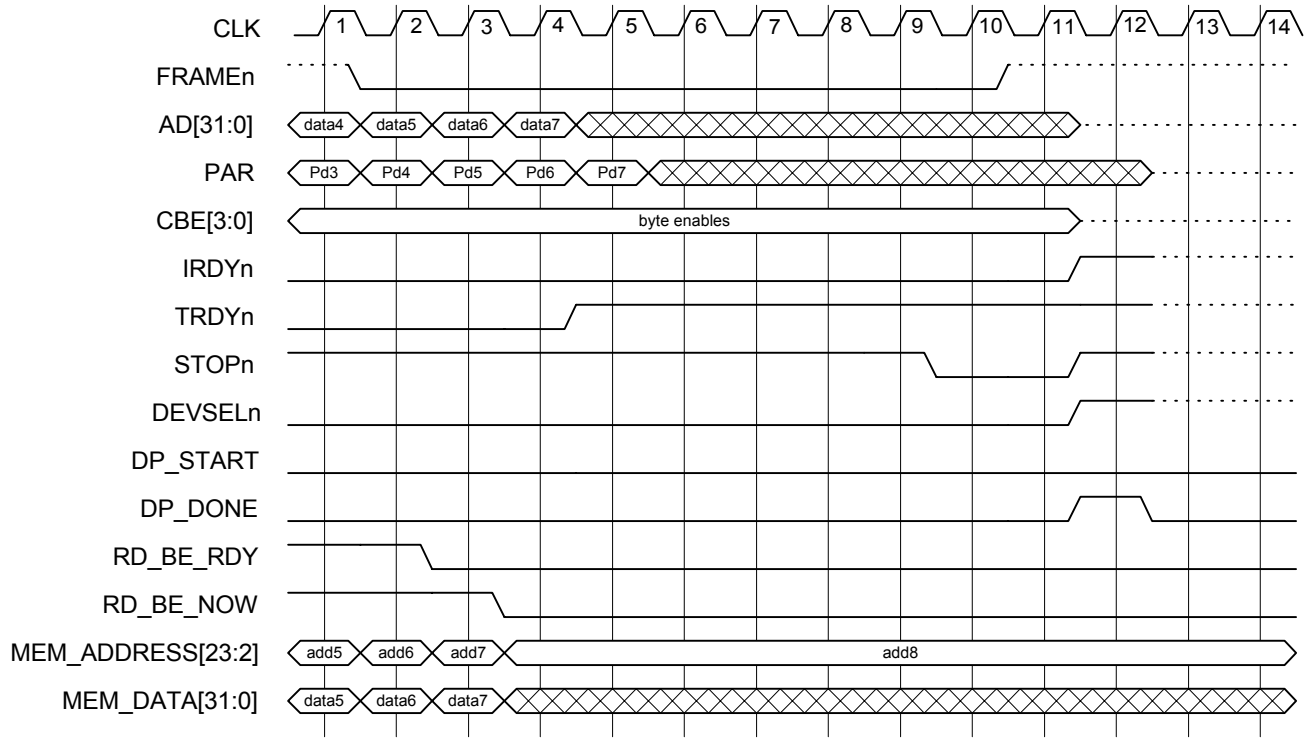
During a regular PCI transfer, the RD_BE_RDY and WR_BE_RDY indicate that data is available to be received from or transmitted to the back-end. If, during a PCI cycle, the back-end becomes unable to read or write data, then the *_RD_RDY signals are de-asserted. After several cycles, a PCI time-out will occur and the Target+DMA controller will initiate a Target disconnect without data cycle, as shown in Figure 18.



Notes:

1. *If BE_GNT is asserted at the beginning of a cycle, then a re-try is initiated.*
2. *The Target+DMA 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 17 • *Target Retry*



Notes:

1. *During a normal PCI transaction, the back-end reaches a point where it is unable to deliver data and de-asserts RD_BE_RDY.*
2. *The Target+DMA 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.*

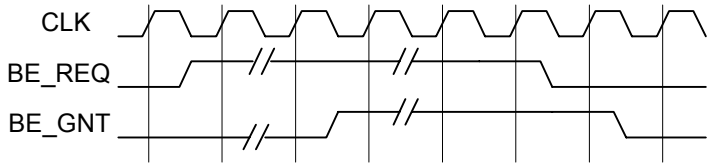
Figure 18 • *Target Disconnect*

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 Figures 19.

Interrupt

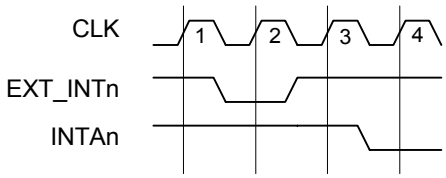
To initiate an interrupt, the back-end needs to assert the EXT_INTn input, as shown in Figure 20. Two cycles later the PCI INTAn interrupt signal will assert.



Notes:

1. *Arbitration begins by the back-end asserting the BE_REQ signal. The Target+DMA 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 BE_GNT will de-assert on the following cycle.*

Figure 19 • *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 20 • *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 arbitrate 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.

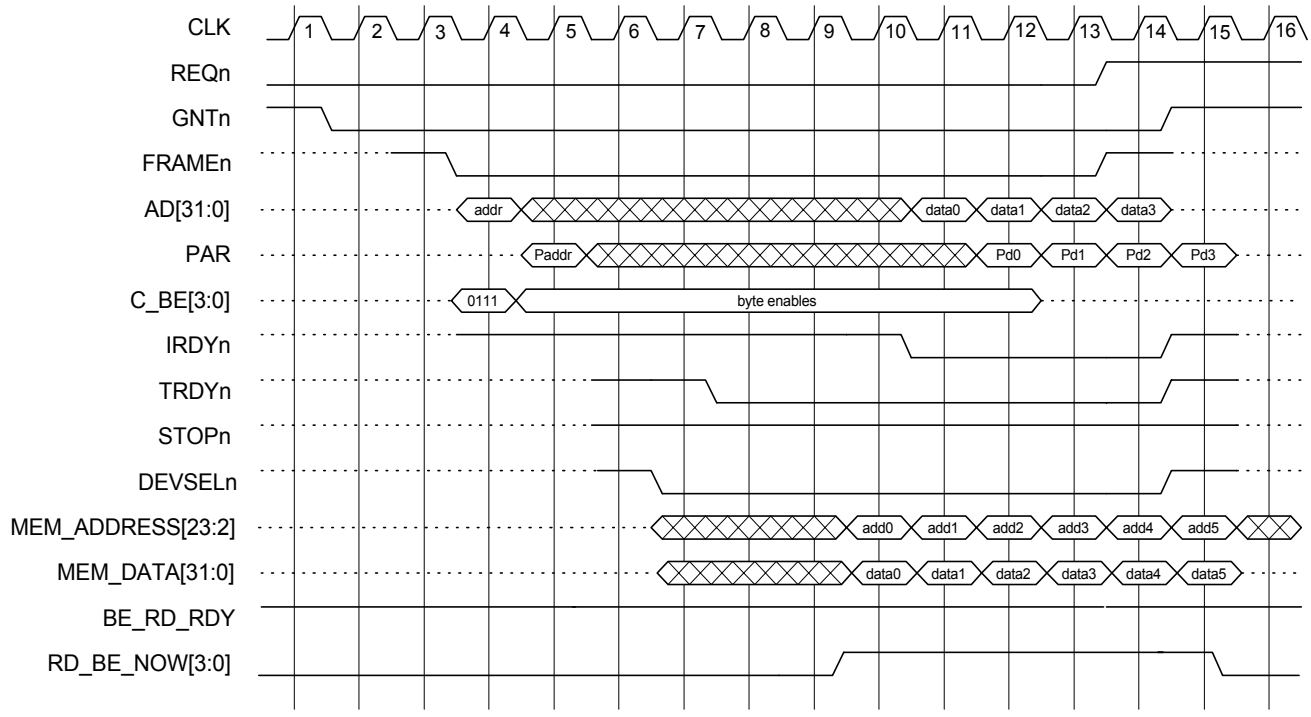


***Figure 21** • Zero Wait State DMA Master Read (Read from the PCI bus)*

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.

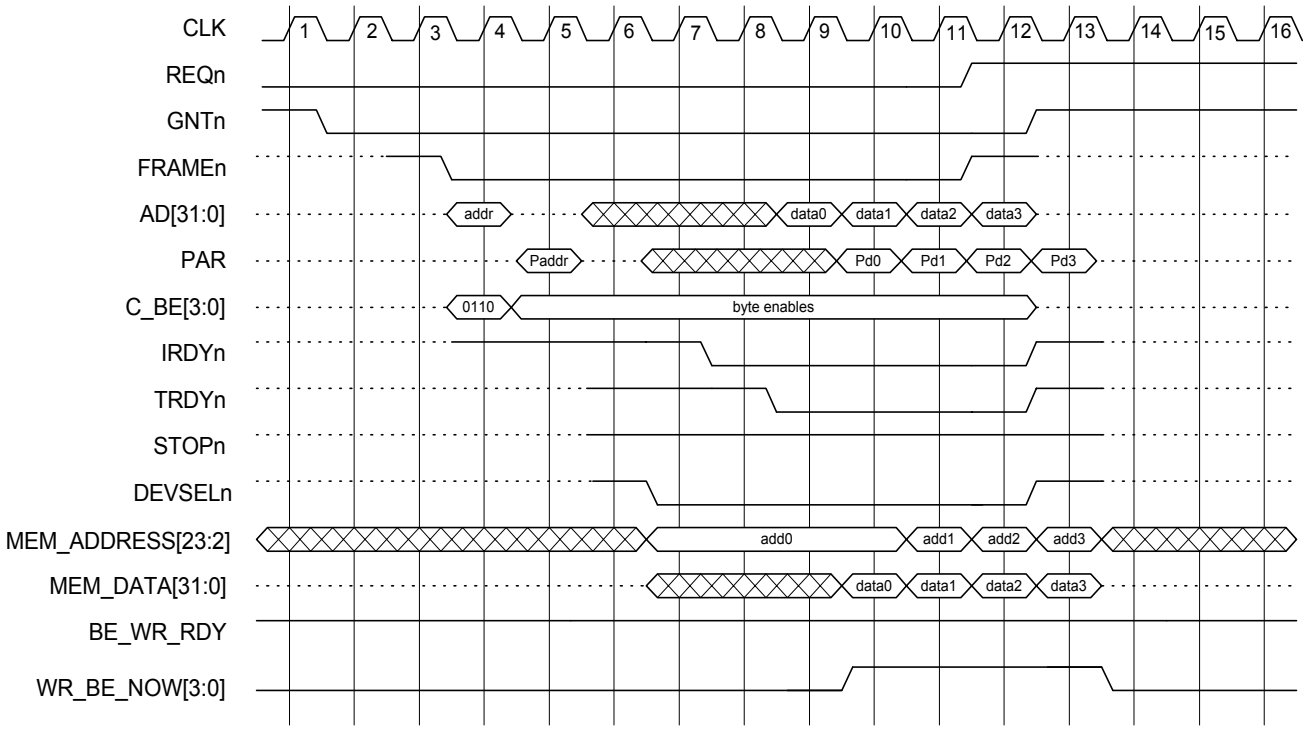


Figure 22 • Zero Wait State DMA Master Write (write to the PCI bus)

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 66 Mhz. By providing the major functional blocks of an SDRAM memory controller, the design can be customized with very little effort. The SDRAM I/Os are shown in Table 18. 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 controller performs three basic functions:

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

Table 18 • 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.
CLK66	Input	System clock.
RESETn	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 gives 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 the controller is ready to start a read access.
WR_BE_RDY	Output	This active high signal indicates that the controller is ready to start a write access.
RD_BE_NOW	Input	This active high signal starts a read transaction, and indicates when read data is expected.
WR_BE_NOW	input	This active high signal 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 and the SDRAM Controller have been designed to work together at a clock frequency of 66MHz. External timing delays to the SDRAM are shown in Table 19 and Table 20. The values shown indicate that the 54SX devices will readily work with high performance SDRAM devices.

Table 19 • Output valid times (ns max)

Name	54SX-2
CSn	6
RASn	7
CASn	9
WEn	9
DQM	7
MAD(11:0)	7
BA	7
CKE	10
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 20 • Input setup times (ns 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

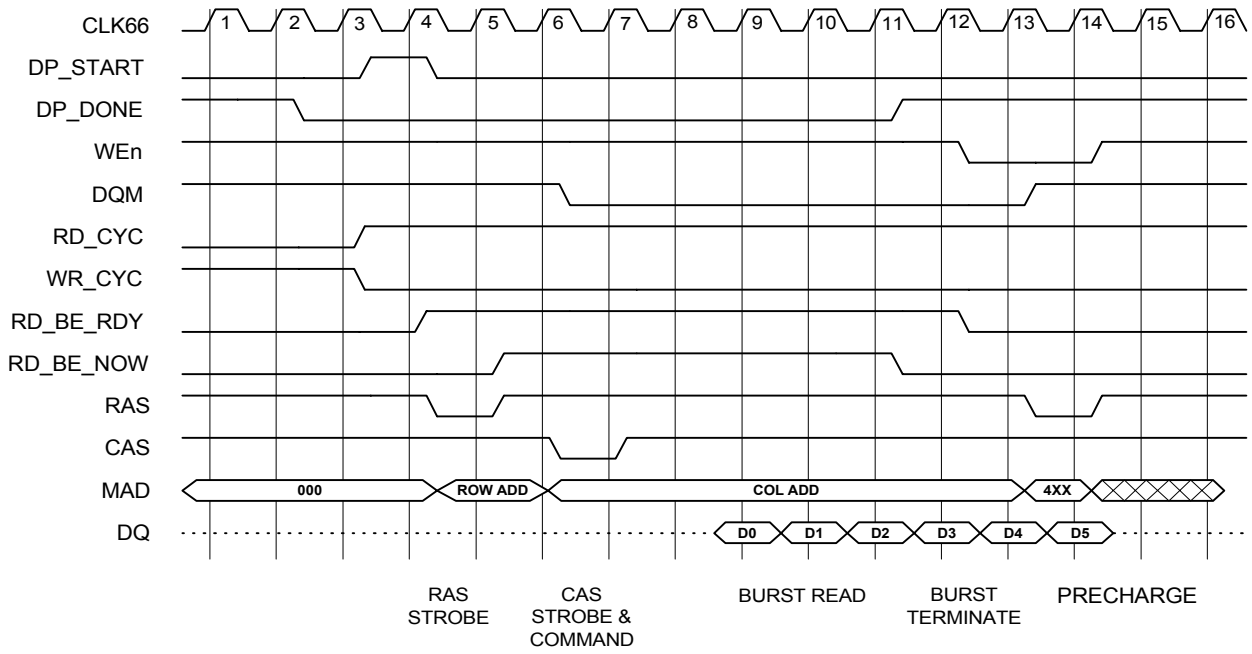
SDRAMs are required to be initialized in a predefined manner. Once power has been applied and the clock is stable, the SDRAM requires a 100 microsecond delay prior to applying an executable command. The controller accommodates this requirement and the requirement of applying NOP commands during this period. After the 100 microsecond 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 at the rising edge of the clock. The controller begins 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 23 and Figure 24.

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

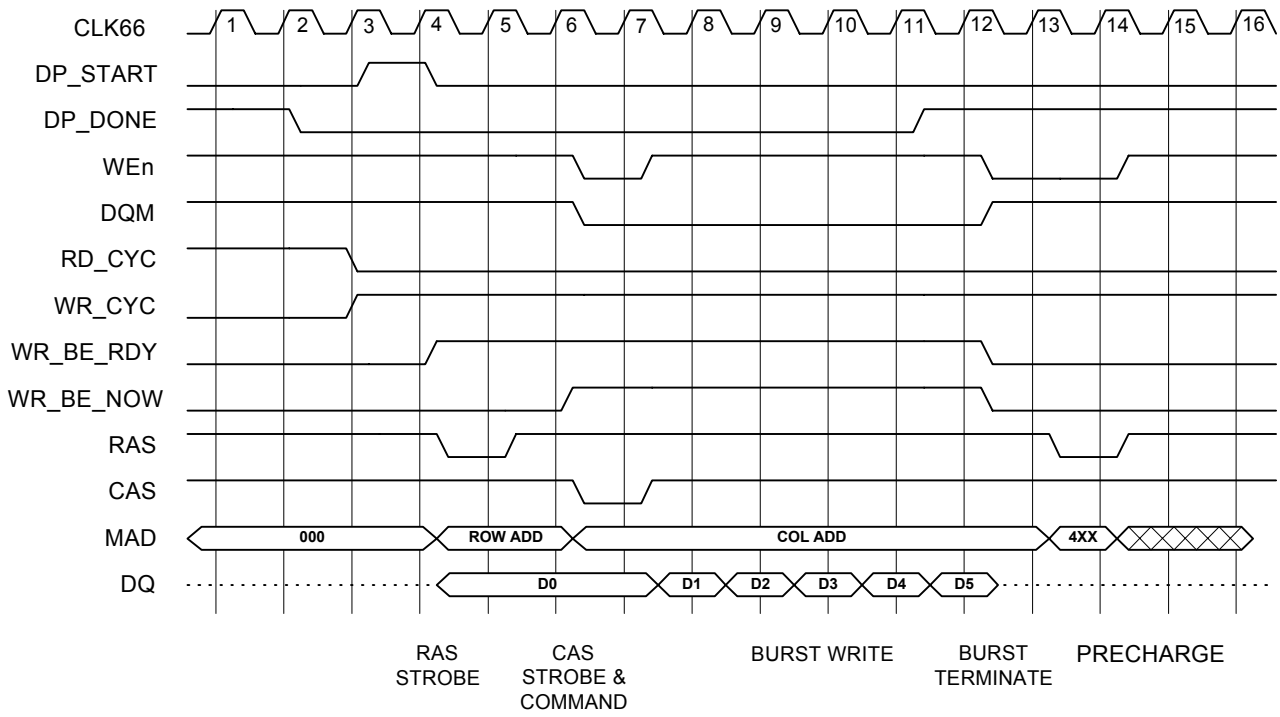
For refresh control, the controller provides an auto refresh sequence to the device every 15.6 microseconds. The refresh sequence is shown in Figure 25.



Notes:

1. *The SDRAM controller asserts RD_BE_RDY immediately following DP_START.*
2. *The PIPE_FULL_CNT bus is set to "011" for SDRAM reads; therefore, the PCI ontroller will expect data four cycles after "RD_BE_NOW" is assrted.*
3. *At the completion of a cycle, DP_DONE active, the SDRAM controller terminates the transfer and precharges all banks.*

Figure 23 • *SDRAM Burst Read*



Notes:

1. *The SDRAM controller asserts WR_BE_RDY immediately following DP_START.*
2. *For writes, the PIPE_FULL_CNT bus is set to "000".*
3. *AT the completion of a cycle, DP_DONE active, the SDRAM controller terminates the transfer and precharges all banks.*

Figure 24 • *Burst Write to SDRAM*

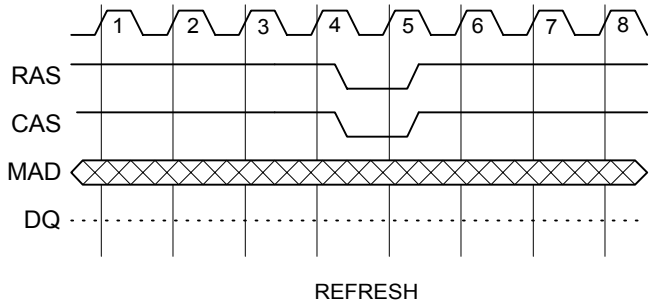


Figure 25 • *SDRAM Refresh.*

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

5172145-1/5.99