CorePCI 5.21 Product
Detailed Design Description

By Joel Landry
September 2000

Table of Contents

Intent

The intent of this document is to provide a more detailed description or the CorePCI 5.21 macro than is provided in the datasheet. The following is described in detail:

- State machine control
- Datapath and parity descriptions
- I/O descriptions of each functional block

General Description

The CorePCI 5.21 macro is designed to provide the following capabilities:

- PCI V2.2 Compliant
- Functions: Target-only, Master-only, Target+DMA, Target/Master
- Bus widths: 32-bit or 64-bit
- Supports any transfer rate including zero wait state bursts
- Provides a "bolt-on" back-end interface
- 66MHz for all Functions in select 54SX devices
- Device families: 54SX, and ProASIC

Functional Overview

The CorePCI macro performs 3 unique functions that are spread out over the various modules described in later sections. The functions are:

- PCI control
  - IRDYn, TRDYn, FRAMEn, STOPn, DEVSELn, REQ64n, ACK64n
    - Single DWORD Reads and Writes
    - Burst Reads and Writes
    - PCI Retry, Disconnect, and Target Abort
  - PCI Tri-state enables
  - Parity timing
  - PCI address decoding
  - Interrupt
- Back-end control
  - Cycle start/end: DP_START and DP_DONE
  - Cycle type: BAR0_MEM_CYC and BAR1_CYC
  - Cycle direction: RD_CYC or WR_CYC
  - RD_BE_NOW, WR_BE_NOW, RD_BE_RDY, WR_BE_RDY, PIPE_FULL_CNT
  - BE Tri-state enables
  - Address incrementing

- Datapath and Datapath Control
  - Transfer data from PCI to Back-end
  - Transfer data from Back-end to PCI
  - IRDY/TRDY deassertion hold register
  - Configuration write/reads
  - Parity generation and checking
  - SERRn, PERRn generation

PCI/Back-end Control

PCI and back-end control are closely coupled and share the following state machines:

- Burst (Target or Master)
- Back-end Latency (Target or Master)
- TRYDN/IRDYN (Target or Master)
- DEVSEL (Target only)
- STOP (Target only)
- FRAME (Master only)
- WR_NOW (Both)
- RD_NOW (Both)

The Burst state machine is the main control system between the PCI bus and the back-end. Because of input setup timing restrictions, the other state machines are all one-bit and use a variety of information from the Burst state machine, back-end control signals, and PCI control signals to correctly perform their functions.

Burst State Machine

The Burst state machine consists of four main blocks: IDLE, TRANSFER, WAITING, and DISCONNECT. The state machine is normally in the IDLE state. The TRANSFER state is entered whenever the back-end is ready or a configuration cycle is taking place. When the back-end is not ready, the state machine enters a sequence of WAIT states which can cause the macro to perform a PCI disconnect without data if the back-end is not ready within defined limits (within 16 cycles of the address phase and within 8 cycles after the first PCI dataphase). The DISCONNECT state is entered whenever a timeout occurs, if a PCI retry cycle is requested (BE_GNT='1'), or if a PCI target abort cycle is requested (ERROR='1').

A PCI transaction begins when the PCI signal FRAMEn is driven low. The address and command are then decoded to determine if the transaction is to the PCI macro. If an address "hit" is detected, then the DP_START signal is generated which initiates the Burst and any back-end state machines. The state machine checks on the status of the BE_RDY signals (a function of cycle type, RD_BE_RDY, WR_BE_RDY, and CONFIG_CYC) to determine the next state. If the transaction is a configuration cycle or the back-end is ready, then the state machine enters the TRANSFER state. If the back-end is not ready, then it enters a waiting sequence. For zero wait state bursts, the state machine will remain in the TRANSFER state.

For bursts with back-end forced wait states, the state machine will toggle between TRANSFER and WAITING states. A transfer ends and returns to the IDLE state when the signal DP_END is asserted. DP_END is a function of FRAMEN='1', IRDY='0', and either TRYDN or STOPN='0'.

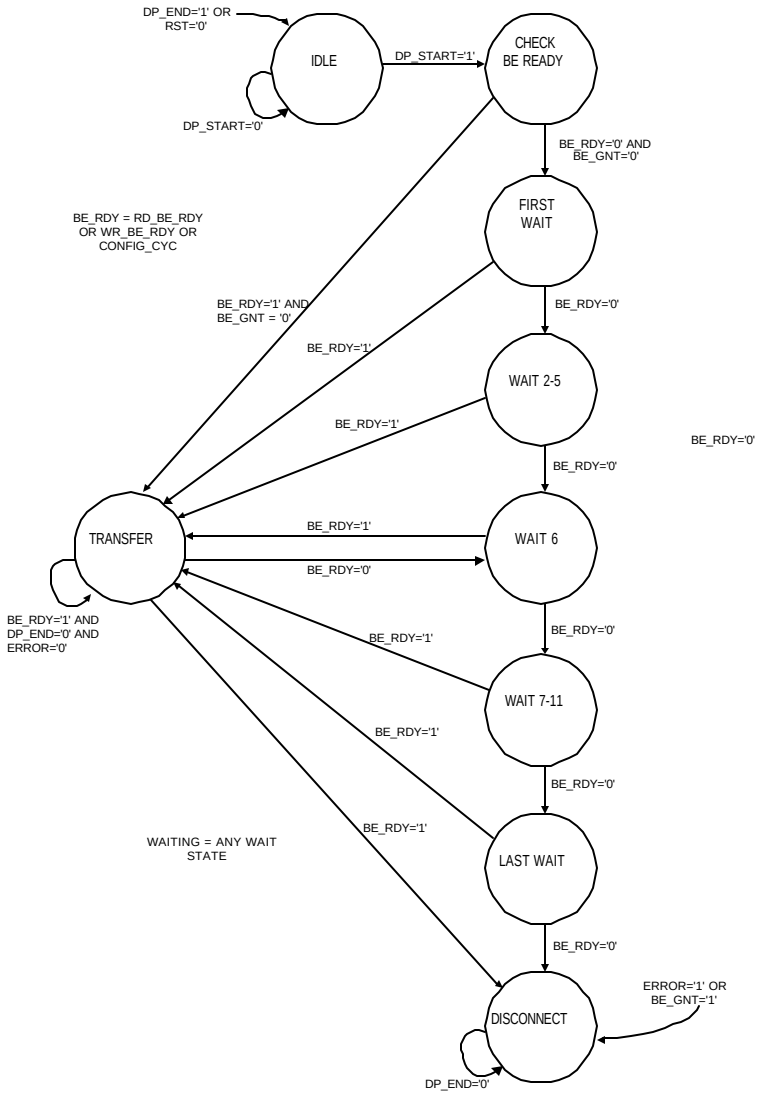

Figure 1. Burst State Machine Diagram

Back-end Latency State Machine

The CorePCI macro is designed to work with a variety of back-ends with different needs. The standard operation for back-end control assumes that the address and data are in phase with one another. That is, when the address is "010", then the data supplied on that cycle corresponds to location "010". For some back-ends like synchronous SRAMs, the address and data are out of phase. The back-end latency state machine was added to provide back-end control for this issue. The back-end is responsible for providing information about the back-end on the RD_BE_RDY, the WR_BE_RDY, and the PIPE_FULL_CNT[ ] signals.

When the PIPE_FULL_CNT[ ] is set to “000”, then the address and data will be in phase and data transfers will begin as soon as either the RD_BE_RDY or WR_BE_RDY is asserted. In the case where the PIPE_FULL_CNT[ ] is set to a non-zero value, the macro will perform a latency “timeout”. During the timeout period, the address will increment and the RD_BE_NOW or WR_BE_NOW signal will be active but no data will be transferred. Figure 2 illustrates the state flow diagram for a read transaction.

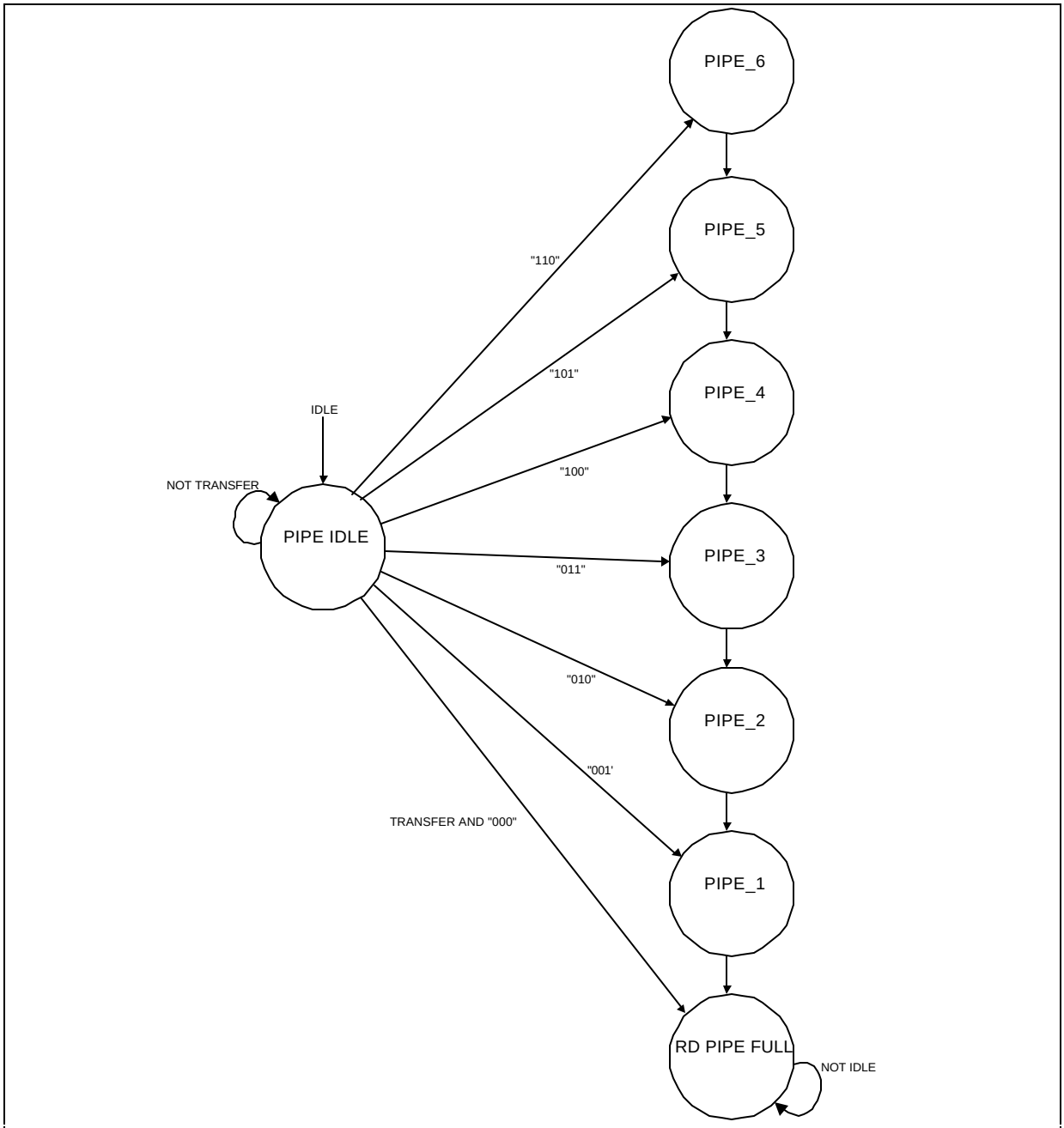

Figure 2. Back-end Latency State Diagram

PCI – DEVSEL State Machine

When the CorePCI macro is behaving as a Target and detects an address "hit", the macro must respond with a DEVSELn. The hit is indicated by DP_START. DEVSELn is normally driven high at the end of a PCI cycle; however, DEVSELn must also be driven high during target abort cycles which occur when the back-end sets its ERROR signal. Figure 3 illustrates this function.

```
DEVSEL_OFF <= (FRAMEN AND !IRDYN AND (!TRDYN OR !STOPN)) OR
              (DISCONNECT AND ERROR);
```

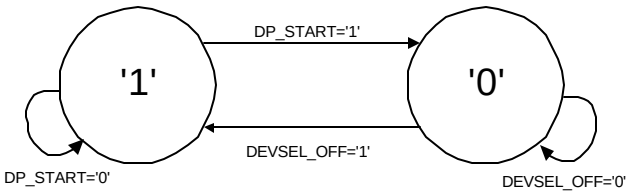

Figure 3. DEVSEL State Machine Equations and Diagram

PCI – TRDYN State Machine

In the CorePCI 5.21 macro, the TRYDN and IRDYN state machines are a separate; however, they are related in many ways. The TRDYN state machine generates the TRDYN signal during slave transfers and generates the IRDYN signal during master transfers. The equations used to generate this signal are defined in Figure 4. TRDYN is asserted whenever an active PCI transfer is in progress (FRAMEN or IRDYN is asserted) and the RDY_TO_XFER signal is active. The RDY_TO_XFER defines whether or not the macro can transmit (read) or receive (write) data for back-end or configuration cycles. The BE_GNT signal is a back-end arbitration signal and when driven high indicates that the PCI macro should respond with a PCI retry cycle (no TRDYN assertion). The PIPE_NOT_FULL signal is a result of the PIPE_FULL state machine which prevents TRDYN assertion until data is available for long latency back-ends.

For back-end writes to the macro, TRDYN is asserted as soon as WR_BE_RDY is active (assuming BE_GNT and PIPE_FULL conditions are satisfied). For reads, the RD_BE_RDY first triggers the RD_BE_NOW signal filling the data pipeline which then in turn causes TRDYN assertion. When the PCI master becomes not ready (IRDYN driven high), then data is stored in the RD_HOLD_REGISTER. To ensure that this data is flushed regardless of the state of the backend ready signals, the RD_HOLD term has been added (new in 5.21). Also, to prevent multiple TRDYn cycles during disconnects, the STOPN signal is now factored into the equation (new in 5.21).

```
RDY_TO_XFER <= (!BE_GNT AND PIPE_FULL AND STOPNS) AND (
               (CONFIG_CYC AND !IDLE_STATE) OR
               (WR_CYC AND WR_BE_RDY AND !IDLE_STATE) OR
               (RD_CYC AND (RD_BE_NOW OR RD_HOLD)));
```

```
TRDY <= (!FRAMEN and !IRDYN AND RDY_TO_XFER) OR
        (!FRAMEN AND IRDYN AND (RDY_TO_XFER OR !TRDYN)) OR
        (FRAMEN AND !IRDYN AND RDY_TO_XFER AND TRDYN);
```

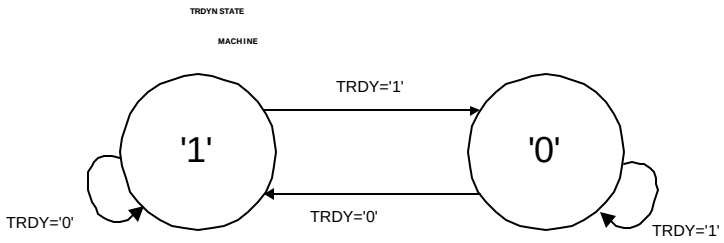

Figure 4. TRDYN/IRDYN State Machine Equations and Diagram

PCI - IRDYN State Machine

The IRDYN is very similar to the TRYDN state machine except that it must meet several additional conditions. The PCI specification requires IRDYN to assert the same cycle that FRAMEN deasserts. Therefore, there is not a normal bus condition of FRAMEN AND IRDYN as there is with FRAMEN AND TRYDN. A second condition occurs when the Target being accessed drives its STOPN signal requesting a disconnect. This condition forces FRAMEN inactive which in turn drives IRDYN high on the following cycle. The third condition is when a PCI master timeout occurs. A timeout again forces FRAMEN high, IRDYN low, and IRDYN low one cycle later

```
IRDY_TO_XFER <= (RDY_TO_XFER AND MASTER) OR MASTER_TIMEOUT;

IRDY <= (!FRAMEN and !TRDYN AND IRDY_TO_XFER) OR
        (!FRAMEN AND TRDYN AND (IRDY_TO_XFER OR !IRDYN)) OR
        (FRAMEN AND TRDYN AND STOPN AND !IRDYN);
```

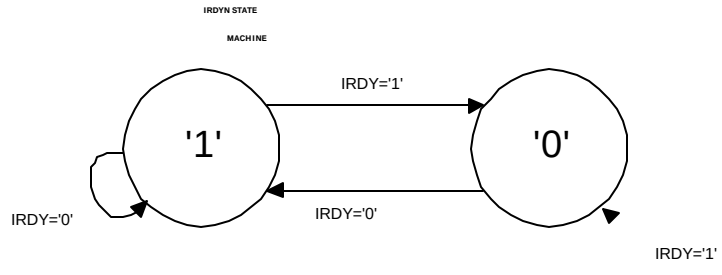

Figure 5. IRDYN State Machine Equations and Diagram

PCI – STOPN State Machine

The PCI STOPN signal is activated whenever the target is requesting termination of a transfer. The three conditions where this occurs is a PCI retry, disconnect, and target abort. A retry or disconnect is caused by a timeout of the burst state machine, by BE_GNT being active, or by BUSY being high. A target abort is initiated by the ERROR signal from the back-end. STOPN will remain asserted until FRAMEN is driven high indicating the end of the PCI cycle.

```
DISCONNECT <= (WAITSTATE_LAST) OR                  -- timeout
              (CHECK_BE_RDY AND BE_GNT) OR         -- back-end arbitration
              (ERROR AND NOT IDLE) OR              -- target abort
              (BUSY AND NOT IDLE);                 -- back-end disconnect req.
```

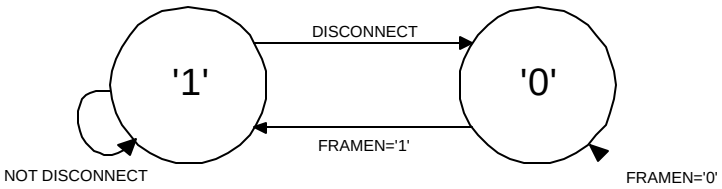

Figure 6. PCI STOPN State Machine

PCI – FRAMEN State Machine

When the DMA_REQ bit in the DMA Control Register is set to a '1', a simple state machine is initiated which performs PCI bus arbitration and address phase sequencing (internal to the CorePCI macro). Once the initial setup is complete, the FRAMEN state machine becomes the primary DMA controller.

Asserting FRAMEN is a fairly simple operation and does not have any stringent timing requirements. Basically, FRAMEN is asserted once the macro has received a GNTN signal and the drivers have been enabled (state DRIVE_TURNON from the DMA controller).

FRAMEN remains asserted until either a Target termination is requested (STOPN='0') or the transfer count approaches zero. Two signals, TERM_CNT1 and TERM_CNT2, were developed to handle this case and are functions of the initial count value, the decremented count value, IRDYN, and TRDYN. In addition, to comply to PCI standards FRAMEN can only be driven inactive whenever IRDYN is being or remains asserted. To meet this requirement, an intermediate signal in the IRDYN/TRDYN state machine is used to inform FRAMEN when IRDYN will be asserted (ASSERT_IRDY).

```
FRAME_ON <=  DRIVE_TURNON AND !GNTN;

FRAME_OFF <= ASSERT_IRDY AND (
             ((!TRDYN AND !IRDYN) AND (!STOPN OR TERM_CNT1)) OR
```

```
((TRDYN OR IRDYN) AND (!STOPN OR TERM_CNT2)));
```

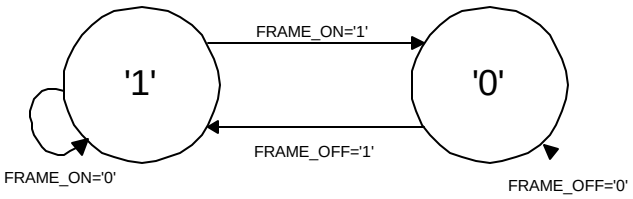

Figure 7. FRAMEN State Diagram and Equations

Back-end – RD_BE_NOW

The RD_BE_NOW controls the flow of read data from the back-end. The RD_BE_NOW is designed to take data from the back-end and pass it to the PCI bus. For the RE_BE_NOW to be asserted, it must be a read cycle and the back-end is ready (RD_BE_RDY asserted). If these conditions are met and the PCI bus is ready (IRDYN and TRDYN low) then RD_BE_NOW will be asserted. If the PCI bus is not ready (IRDYN or TRYDN high), then the RD_BE_NOW will assert only if the AD_REG register does not currently have any data stored in it.

```
RDY_TO_RD <=  !CONFIG_CYC AND RD_CYC AND RD_BE_RDY AND !BE_GNT AND !IDLE;

RD_NOW <= (!TRDYN AND !IRDYN AND RDY_TO_RD) OR
          ((TRDYN OR IRDYN) AND RDY_TO_RD AND !AD_REG_FULL);
```

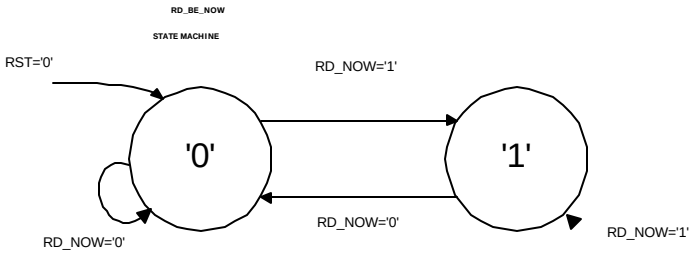

Figure 8. RD_BE_NOW State Machine

Back-end – WR_BE_NOW

The WR_BE_NOW signal is asserted whenever a PCI dataphase occurs (TRDYN=’0’ and IRDYN=’0’) and the cycle is not to configuration space.

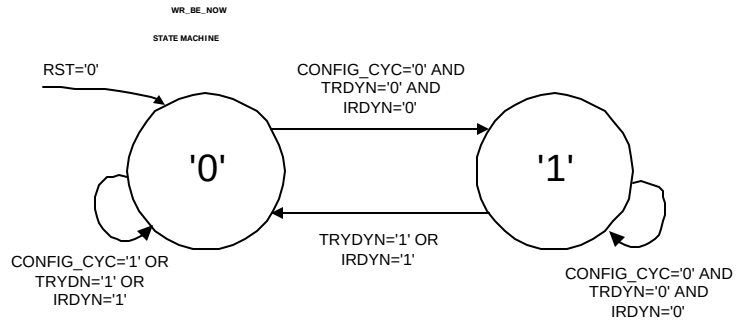

Figure 9. WR_BE_NOW State Machine

Datapath Flow and Control

The datapath function in the CorePCI macro is fairly simple. There are six possible sources and destinations for data: PCI AD bus, the back-end MEM_DATA bus, the Target configuration registers, the DMA configuration registers, the PCI Start Address register, and the RD_HOLD register. The RD_HOLD register is used to store data in the event the PCI bus IRDYN signal becomes inactive to prevent loss of data in the pipe-line. The source data is always mapped to both the AD_REG bus and MEM_DATA_REG bus. The multiplexer is used to steer data from the correct source for any give bus transaction. The selection is dependent on the state of the signals P_BUS_EN, RD_HOLD, MAST_ADDR_PH, DMA_SELECT, and CONFIG_RD. The selection is as follows:

- PCI_START_ADD - The PCI Start Address register is the source MAST_ADDR_PH = '1';
- DMA_DATA_RD - The read path for the DMA configuration registers are the source when DMA_SELECT = '1' and RD_CONFIG='1'.
- CONFIG_DATA_RD - The Target configuration registers are the source when DMA_SELECT = '0' and RD_CONFIG='1'.
- AD – The AD bus is the source whenever none of the above applies and P_BUS_EN='0'. That is, the macro is not driving the AD bus.
- RD_HOLD – The source whenever P_BUS_EN='1' and CONFIG_RD='0' and RD_HOLD='1'.
- MEM_DATA – This bus is the source when none of the above conditions apply.

The AD_REG[ ] bus is also subject to enable control and is a function of IRDYN and LOAD_AD_REGN. The IRDYN input disables the updating of the AD_REG whenever the system master inserts wait states (IRDYN='1'). The LOAD_AD_REGN is responsible for enabling the AD_REG during master address phases and when the AD_REG is empty during back-end read transactions.

In addition to basic data steering, the datapath logic is also responsible for the following:

- Providing data for parity generation and checking
- Providing address information to the address counters. The source for this is the MEM_DATA_REG[ ] when MASTER='0' and RAM_START_ADD[ ] from the DMA registers when MASTER='1'

For Target+Master and Master functions, the DMA configuration registers are accessed through the back-end logic and MEM_DATA is the read/write bus. In this case, the DMA_DATA_RD input into the dataflow MUX is tied to all zeroes.

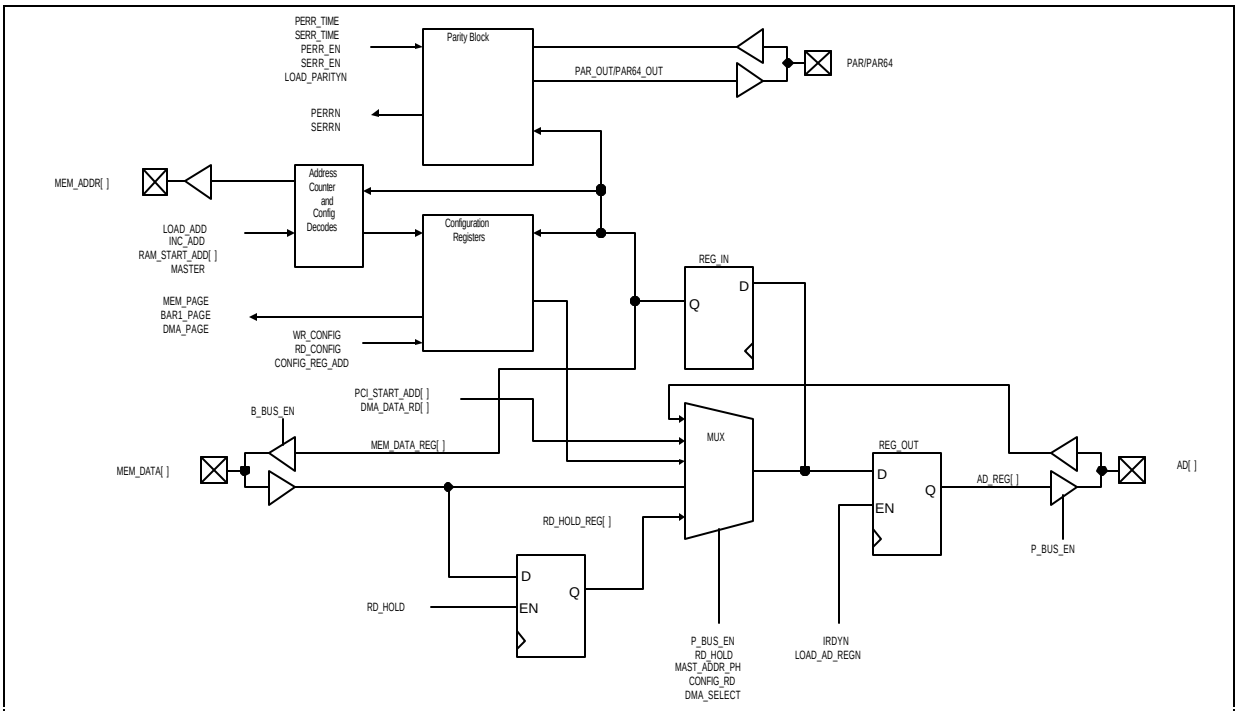

Figure 10. CorePCI Datapath Diagram

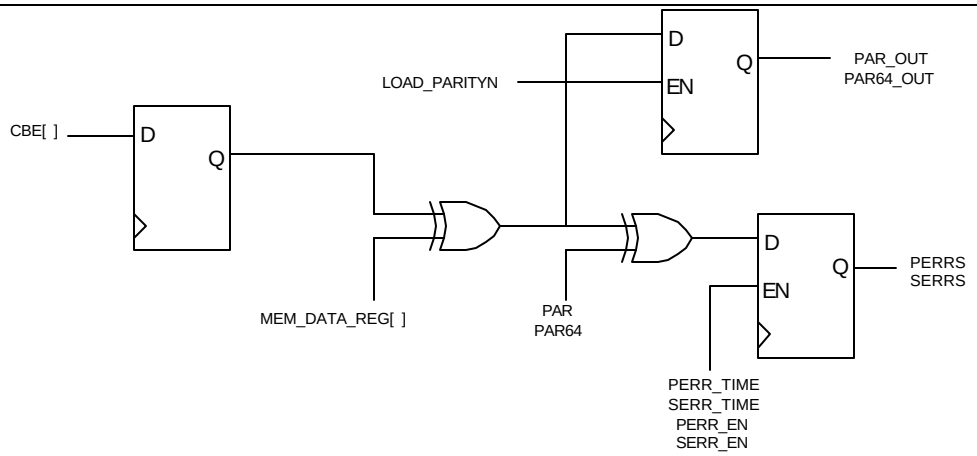

Figure 11. Parity Details

When a Target+DMA

- DMA_DATA_WR <= MEM_DATA_REG
- AD <= DMA_DATA_RD

When a Target/Master or Master

- DMA_DATA_WR <= MEM_DATA
- MEM_DATA <= DMA_DATA_RD

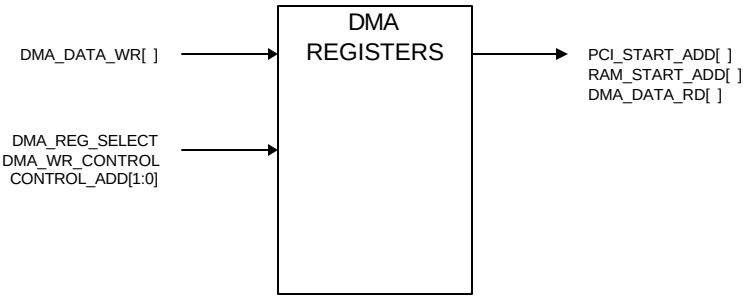

Figure 12. DMA Configuration Registers

Design Organization

The CorePCI 5.21 macro is organized as defined in the following sections. For each design, there is a "targpack" file with constants that impact how the design is compiled. The targpack constants are thoroughly defined in the CorePCI datasheet. Each indentation represents a lower level of hierarchy.

Target-Only Organization

The Target-only function is comprised of 6 primary building blocks: Add_cntr64, Add_phase64, Burst64, Config, Datapath, and Parity64. For 64-bit designs, a second Datapath block is included to handle the upper 32-bits of data. All of the other blocks are dynamically 32-bit or 64-bit depending on the value of Bit64 constant in the Targpack. For 32-bit designs, 64-bit logic is eliminated during compile time to improve utilization.

Target64
- Add_cntr64
- Add_phase64
- Burst64
  - CM8D
  - CM8DP
  - CM8DX
- Config
- Datapath (lower 32-bit)
  - Datapath_registers
  - Mux4_8
- Datapath (optional 64-bit)
  - Datapath_registers
  - Mux4_8
- Parity64
  - Cbe_par

Target+DMA Organization

The Target+DMA is built on a wrapper that combines all of the function of the Target-only macro with the DMA module. The DMA module provides the functionality required to handle master transfers on the bus and is limited to memory reads and writes. The Config block in the Target+DMA differs from the Target-only block by adding 3 additional registers that drive DMA transfers.

- Target+DMA Wrapper
  - DMA
    - CM8D
  - DMA_REG
  - Target64
    - Add_cntr64
    - Add_phase64
    - Burst64
      - CM8D
      - CM8DP
      - CM8DX
    - Config
    - Datapath (lower 32-bit)
      - Datapath_registers
      - Mux4_8
    - Datapath (optional 64-bit)
      - Datapath_registers
      - Mux4_8
    - Parity64
      - Cbe_par

Master-only Organization

The Master-only function is similar to the Target+DMA function. The major difference is that configuration space has been eliminated and replaced with the three DMA registers that are now accessible from the back-end. To handle these differences, a separate Target64 block is required and is referred to as the Master_Target module.

- Master Wrapper
  - DMA
    - CM8D
  - DMA_REG
  - Master_Target
    - Add_cntr64
    - Add_phase64
    - Burst64

CM8D
CM8DP
Datapath (lower 32-bit)
Datapath_registers
Mux4_8
Datapath (optional 64-bit)
Datapath_registers
Mux4_8
Parity64
Cbe_par

Detailed Module and Function Description

Most major PCI functions are handled by a variety of blocks. The key functions are:

- PCI Bus Control Operations
  - IRDYn, TRDYn, FRAMEn, STOPn, DEVSELn, REQ64n, ACK64n (Burst64/DMA)
  - PCI Tri-state enables (Burst64)
  - Parity timing (Burst64)
  - PCI address decoding (Add_phase64/Config)
- Back-end Bus Control Operations
  - Cycle start/end: DP_START (Add_phase64) and DP_DONE (Burst64)
  - Cycle type: BAR0_MEM_CYC and BAR1_CYC (Add_phase64)
  - Cycle direction: RD_CYC or WR_CYC (Add_phase64)
  - RD_BE_NOW, WR_BE_NOW, RD_BE_RDY, WR_BE_RDY, PIPE_FULL_CNT (Burst64)
  - Error function (Burst64)
  - Back-end Tri-state enables (Burst64)
  - Address incrementing (Burst64, Addr_cntr64)
- Parity
  - Parity generation and checking (Parity64, Cbe_par)
  - SERRn, PERRn generation (Parity64)
- Dataflow
  - Transfer data from PCI to Back-end (Burst64/Datapath)
  - Transfer data from Back-end to PCI (Burst64/Datapath)
  - IRDY/TRYD deassertion hold register (Burst64/Datapath)
- Configuration Space
  - Configuration write/reads (Config, Addr_cntr64, Burst64, Datapath)
- Interrupt (Config)

Addr_cntr64 Module

The Addr_cntr64 block is responsible for the following:

- Load the PCI address
- Provide the address to the back-end
- Provide decode for configuration registers when a configuration cycle
- Increment by 1 for 32-bit transactions, 2 for 64-bit transactions

Signal Name	Type	Description
CLK	Input	PCI clock
CYC64	Input	Output of the Add_phase64 module. Input that defines count increments of 1 (CYC64='0') or 2 (CYC64='1') depending on whether the transfer is 32-bit or 64-bit.
INC_ADD	Input	Output of the Burst64 module. Causes the counter to increment when asserted.
LOAD_ADD	Input	Output of the Add_phase64 module. When asserted, causes the counter to be loaded. The load value will be either the RAM_START_ADD from the Config module (Master='1') or the MEM_DATA_REG bus (Master='0').
MASTER	Input	Output of the DMA module. Is active ('1') when the macro is behaving as a master on the PCI bus. This signal is used to force the address decode to be "PCI_START_REG" decode. This function maps the PCI start address from the Config module to the Datapath module for loading when MAST_ADDR_PH is active.
MEM_DATA_REG	Input	Output of the Dataphase module. This bus contains the PCI address information during Target transfers.
RAM_START_ADD	Input	Output of the Config module. This bus defines the back-end RAM address to be used during DMA or Master transfers.
CONFIG_REG_ADD	Output	14-bit bus used to define address decodes for configuration space. The actual decodes are defined in the Targpack module.
MEM_ADD_INT	Output	Memory address to the back-end.

Add_phase64 Module

The Add_phase64 module is responsible for the following functions:

- Monitors FRAMEn to detect a PCI address cycle
- Determines if the PCI cycle is to the macro by comparing the address coupled with the CBE command
  - Configuration space cycle (CBE="101X" and IDSEL='1')
  - Memory cycle to BAR0 (CBE="011X" and MEM_PAGE=PCI Address)
  - Optional BAR1 Cycle
    - Memory (CBE="011X" and BAR1_PAGE=PCI Address)
    - I/O (CBE="001X" and BAR1_PAGE= PCI Address)
  - Optional DMA in I/O Cycle (CBE="001X" and DMA_PAGE= PCI Address)
- Determines direction of transfer (PCI read or PCI Write)
- Determines size of transfer (32-bit or 64-bit)

- Provides arbitration control between the back-end, the DMA engine, and the PCI bus

Signal Name	Type	Description
CLK	Input	PCI system clock.
BAR1_PAGE	Input	Output of the Config module. Bus defining the PCI plug'n'play address associated with BAR1.
BE_REQ	Input	Input from the back-end. Control signal from the back-end requesting ownership of the back-end bus.
CBE	Input	PCI input. Defines the PCI command.
DMA_DIRECTION	Input	Input from the DMA control block indicating the direction (read/write) of the transfer.
DMA_PAGE	Input	Output of the Config module. Bus defining the PCI plug'n'play address associated with BAR2. This bus only has meaning if the DMA_IN_IO constant in the Targpack is set to a '1'.
DMA_REQUEST	Input	Output of the DMA module. This input is a request from the DMA engine to take over control of the PCI macro.
FRAMEn	Input	PCI input. When this signal asserts, it indicates an address phase.
GNTn	Input	PCI input. Signal from the PCI bus indicating that the PCI macro has been granted control of the bus. Assertion of this signal will in turn cause DMA_GNT to be asserted.
IDSEL	Input	PCI input . Special select for configuration commands.
IO_EN	Input	Output of the Config module. Control signal from configuration space enabling response to I/O cycles.
MASTER	Input	Output of the DMA module. When the macro is behaving as a Target (MASTER='0'), then address checking is performed normally. When the macro is a DMA/Master (Master='1'), then an address hit to MEM_BAR0 is forced. The Master signal also reverses the polarity of the RD_CYC/WR_CYC signals during DMA/Master transfers.
MEM_DATA_REG	Input	Output of the Datapath module. This bus contains PCI address information.
MEM_EN	Input	Output of the Config module. Control signal from configuration space enabling response to memory cycles.
MEM_PAGE	Input	Output of the Config module. Bus defining the PCI plug'n'play address associated with BAR0.
REQ64n	Input	PCI input. Indicates a 64-bit data request.
BAR0_MEM_CYC	Output	Active high signal decoding a hit to BAR0 address space.
BAR1_CYC	Output	Active high signal decoding a hit to BAR1 address space.
BE_GNT	Output	Active high signal granting control of the back-end system.
CONFIG_CYC	Output	Active high signal decoding a hit to configuration space.
CYC64	Output	Active high signal indicating that the current transaction is 64 bits wide.
DMA_GNT	Output	Active high signal indicating the PCI macro is currently under control of the DMA engine.
DP_START	Output	Active high pulse indicating that the macro has decoded a transaction to its configuration space or back-end address space. This signal also indicates the beginning of a DMA initiated transaction.
DP_START64	Output	Active high pulse indicating that the macro has decoded a transaction to a 64-bit back-end address space.
LOAD_ADD	Output	Active high pulse indicating that the PCI address should be loaded into the address counter (Add_cntr64).

RD_CYC	Output	Active high signal indicating a transfer form the back-end to the PCI bus.
SERR_TIME	Output	Active high pulse defining the cycle when address parity should be checked.
TARGET_START	Output	Active high pulse similar to DP_START that indicates the beginning of a PCI initiated transaction.
WR_CYC	Output	Active high signal indicating a transfer from the PCI bus to the back-end.

Burst64 Module

The Burst64 module is the main control block in the CorePCI macro. The complexity of this block is a result of the wide range of functions it performs as well as the need to maintain PCI setup requirements. Anytime a PCI input is required to perform a logical function, a CM8D macro is used. The CM8D macro is essentially a 4:1 mux feeding a flip-flop. PCI signals are attached to the select signals on the 4:1 mux providing excellent functionality while maintaining minimum logic between the input pad and the register. An example is included. Functions the Burst64 module perform are as follow:

- Generates the handshake signals to the back-end that controls data flow
- Generates DEVSELn, ACK64n, TRDYn, and STOPn when acting as a Target
- Generates IRDYn when acting as a Master
- Generate tri-state enables for PCI Bus and the back-end

Signal Name	**Type**	**Description**
CLK	Input	PCI system clock.
BE_GNT	Input	Output of the Add_phase64 module indicating that the back-end is currently unavailable for transfers. A PCI retry will be forced when this signal is active.
BUSY	Input	Primary back-end input requesting an immediate disconnect.
CBE	Input	Primary PCI input defining with bytes are enabled.
CONFIG_CYC	Input	Output of the Add_phase64 module indicating a transaction to configuration space.
CYC64	Input	Output of the Add_phase64 module indicating a 64-bit transaction.
DP_START	Input	Output of the Add_phase64 module. Active high pulse indicating a PCI transaction to the macro is beginning. This signal initiates the main state machine.
DP_START64	Input	Output of the Add_phase64 module. Active high pulse indicating a 64-bit PCI transaction to the macro is beginning.
ERROR	Input	Primary back-end input indicating that a back-end error has occurred and the macro should respond with Target abort command.
FRAMEN	Input	Primary PCI input.
FRAMEN_INT	Input	Internal version of FRAMEn used during Master cycles.
GNTn	Input	Primary PCI input. Used to determine if bus parking is required.
IRDYN	Input	Primary PCI input from the PCI Master indicating it is ready to transfer data with the Target.
MAST_ADDR_PH	Input	Output of the DMA module. Indicates that a master address phase is in progress.
MASTER	Input	Output of the DMA module. Indicates whether the macro is behaving as a master (MASTER='1') or a target (MASTER='0').

MASTER_TIMEOUT	Input	Output of the DMA module indicating that a Target has failed to respond and the Burst state machine should terminate the current operation.
PIPE_FULL_CNT	Input	Primary back-end input. This 3-bit bus defines startup latency between the initial *_NOW signal and when data is actually valid.
RD_BE_RDY	Input	Primary back-end input indicating that the back-end is ready to transmit data.
RD_CYC	Input	Output of the Add_Phase64 module. Active high signal indicating a transaction from the back-end to the PCI bus.
STOPn	Input	Primary PCI input from a Target requesting termination of the current transfer.
TARGET_START	Input	Input from the Add_phase64 module. Active high pulse indicating the beginning of a transfer requested by the PCI bus (target mode only).
TRDYN	Input	Primary PCI input from the PCI Target indicating it is ready to transfer data with the Master.
WR_BE_RDY	Input	Primary back-end input indicating that the back-end is ready to receive data.
WR_CYC	Input	Output of the Add_Phase64 module. Active high signal indicating a transaction from the PCI bus to the back-end.
ACK64N	Output	Primary PCI output. Only has meaning when the macro is a Target (MASTER='0') and is operating in a 64-bit mode.
ASSERT_IRDY	Output	Output to the DMA module indicating that IRDYN will be asserted on the following clock cycle.
B_BUS_EN	Output	Tri-state enable for the back-end data bus.
B_BUS_EN64	Output	Tri-state enable for the back-end data bus.
DEVSELN	Output	Primary PCI output. Only has meaning when the macro is a Target (MASTER='0').
DP_DONE	Output	Output pulse indicating that the current transaction has completed.
INC_ADD	Output	Output to the Add_cntr64 module indicating that the address should be incremented.
IRDYN_OUT	Output	Primary PCI output during Master transfers.
IRDYN_PAD	Output	Single load IRDYN that goes directly to the PAD.
LOAD_AD_REGN	Output	Active low signal indicating that the AD register is empty and needs to load data when available from the back-end.
LOAD_PCI_ADD	Output	Forces the loading of the AD registers when the burst state machine is idle.
P_BUS_EN	Output	Tri-state enable for the PCI data bus.
P_BUS_EN64	Output	Tri-state enable for the PCI data bus.
PERRN_TIME	Output	Active high output indicating that data parity should be checked.
PIPE_FULL	Output	Active high output indicating that the initial latency has completed and the data pipeline now has data.
RD_BE_NOW	Output	Primary output to the back-end. This active high signal indicates that the data currently on the back-end data bus will be transmitted to the PCI bus on the following cycle.
RD_BE_NOW64	Output	Primary output to the back-end. This active high signal indicates that the data currently on the upper 32-bits of the back-end data bus will be transmitted to the PCI bus on the following cycle.
RD_CONFIG	Output	Output to the Datapath module indicating that configuration data should be steered to the PCI bus.
RD_HOLD	Output	Active high signal to the Datapath module indicating that a ready (TRDYN or IRDYN) deassertion has occurred and the back-end data needs to be held until the PCI cycle can complete.
RD_HOLD64	Output	Active high signal to the Datapath module indicating that a ready (TRDYN or IRDYN) deassertion has occurred and the upper 32-bits of back-end data needs to be held until the PCI cycle can complete.

STOPN_OUT	Output	Primary PCI output. Signal used to force target disconnects and only has meaning when the macro is a Target (MASTER='0').
TABORT	Output	Active high output to the Config module indicating that a Target abort cycle has occurred.
TARG_CTL_EN	Output	Tri-state enable for PCI control signals when the macro is behaving as a target.
TARG_CTL_EN64	Output	Tri-state enable for PCI control signals when the macro is behaving as a target in a 64-bit mode.
TRDYN_OUT	Output	Primary PCI output during Target transfers.
TRDYN_PAD	Output	Single load TRDYN that goes directly to the PAD.
WR_BE_NOW	Output	Primary output to the back-end. . This active high signal indicates that the data currently on the back-end data bus is valid write data.
WR_BE_NOW64	Output	Primary output to the back-end. . This active high signal indicates that the upper 32-bits of data currently on the back-end data bus is valid write data.
WR_CONFIG	Output	Output to the Config module indicating that the data should be written to the configuration registers.

The Burst64 module can be broken up into the following functional sections:

- Start-up latency counter – monitors the PIPE_FULL_CNT bus and the *_BE_RDY signals. The latency counter generates the "PIPE_FULL" flags once the latency has been satisfied.
- Main state machine – the main function of the state machine is to indicate when the back-end is ready, to provide time-out logic when the back-end is not ready, and to initiate disconnects (retry, disconnect without data, and target abort).
- PCI and back-end control signal generation – The input logic controlling these functions are a mix of PCI signals (predominantly IRDYN and FRAMEN), back-end inputs, state machine signals, and "PIPE_FULL" flags.
- Internal bus control signal generation – Ditto as previous.

The burst state machine is elaborated in the figure.

Config Module

The Config module implements the Target Configuration space for the CorePCI macro. The Config module is mostly just a register file whose outputs feed a wide variety of functions in the CorePCI macro.

Signal Name	**Type**	**Description**
CLK	Input	PCI system clock.
CONFIG_REG_ADD	Input	Output of the Add_cntr64 block. Defines the register decode for both read from and writes to configuration space.
DMA_CAPABLE	Input	Active high input indicating that the CONFIG module is being used in a master-capable macro.
EXT_INTN	Input	Primary back-end interrupt input. This signal cause the INT_OUT signal to be driven active if the interrupts are enabled.
MASTER_ERROR	Input	3-bit Input bus that reports target abort, master abort, and parity errors during master transfers.
MEM_DATA_REG	Input	Output from the Dataphase module. Write data.

PERRS	Input	Output from the Parity64 module indicating a data or address parity error has occurred causing a status bit to be set.
SERRS	Input	Output from the Parity64 module indicating an address parity error has occurred causing a status bit to be set.
TABORT	Input	Output from the Burst64 module indicating a Target Abort cycle has been executed. This also causes a status bit to be set.
WR_CONFIG	Input	Output of the Burst64 block. This signal is a synchronous write strobe to the addressed register.
BAR1_PAGE	Output	Plug'n'play address location for BAR1.
CONFIG_DATA_RD	Output	Multiplexed read data bus for configuration space.
DMA_PAGE	Output	Plug'n'play I/O address location for BAR2. This bus only has meaning when the DMA_IN_IO constant (in the Targpack) is set to a '1'.
INT_OUT	Output	Primary PCI Output INTAn.
IO_EN	Output	Active high enable indicating that the macro can respond to I/O commands.
LATENCY_TIMER	Output	Output to the DMA controller indicating the maximum bus latency for the Master controller.
MEM_EN	Output	Active high enable indicating that the macro can respond to memory commands.
MEM_PAGE	Output	Plug'n'play address location for BAR0 (memory).
PAR_EN	Output	Active high drive enable for PERRn.
SERR_EN	Output	Active high enable allowing address parity errors to be reported on the SERRn signal.

Datapath Module

The datapath module is responsible for correctly steering data between the PCI bus and the back-end data bus. In addition, the dataphase block contains the RD_HOLD register which prevents overflow conditions from occurring when the PCI bus ready signals are driven inactive. There are 4 basic sources for data in the macro: the AD bus, the MEM_DATA bus, the internal configuration registers, and the RD_HOLD register. Depending on the state of the input control signals, the datapath module steers the data to the appropriate location. The conditions for each source is as follows:

- AD – The AD bus is the source whenever P_BUS_EN='0'. That is, the macro is not driving the AD bus.
- Configuration – The configuration registers are the source whenever P_BUS_EN='1' and (CONFIG_RD='1' or MAST_ADDR_PH='1').
- RD_HOLD – The source whenever P_BUS_EN='1' and CONFIG_RD='0' and RD_HOLD='1'.
- MEM_DATA – This bus is the source when none of the above conditions apply.

Signal Name	**Type**	**Description**
CLK	Input	PCI system clock.
AD	Input	Primary PCI input.

CONFIG_DATA_RD	Input	Output of the Config module. Read data from configuration space.
DMA_DATA_RD	Input	Output of the DMA module. Read data from the DMA configuration registers.
DMA_SELECT	Input	Wrapper level input selecting DMA configuration space for reads.
IRDYN	Input	Primary PCI input. Used to hold the value in the AD_REG bus whenever IRDYn is inactive during PCI data transfers.
MAST_ADDR_PH	Input	Output of the DMA module indicating that the macro is performing a Master address phase cycle.
MEM_DATA	Input	Primary input from the back-end.
P_BUS_EN	Input	Output of Burst64. Tri-state enable for the PCI AD bus.
PCI_START_ADD	Input	Output of the DMA module. PCI Start Address value.
PIPE_FULL	Input	Output of Burst64 (called LOAD_AD_REGN). This signal indicates that no data is currently loaded in the AD_REG. Combined with IRDYN, this signal is an enable for loading data onto the AD bus.
RD_CONFIG	Input	Output of the Burst64 module. Indicates a read from configuration space.
RD_HOLD	Input	Output of the Burst64 module. Indicates that the PCI bus TRDY/IRDY has deasserted and the datapath modules needs to temporarily hold the data.
TRDYN	Input	Primary PCI input. Used to hold the value in the AD_REG bus whenever TRDYn is inactive during PCI data transfers.
AD_REG	Output	Bus which supplies data to the PCI AD bus.
LOAD_PARITYN	Output	Active high signal indicating when the PAR bit should be updated. This signal is essentially a one cycle delayed version of the AD_REG enables.
MEM_DATA_REG	Output	Registered AD data that supplies data to address detect logic, configuration write data, and back-end write data.
PAR_DATA	Output	Bus which supplies data to the Parity check/generate logic.

DMA and DMA_REG Modules

The DMA module is responsible for coordinating master transfers. The functions performed are:

- Handshake with the DMA Control Register for DMA request and done
- FRAMEN and REQ64N generation
- Internal control for the address phase
- Generation of the CBE command and byte signals
- Latency timeout counter
- Master abort timeout counter
- Implements DMA configuration registers and provides read/write logic for them

DMA Module

Signal Name	**Type**	**Description**
CLK	Input	PCI system clock.
ASSERT_IRDY	Input	Output of Burst64 indicating when IRDYN will be driven active.
BE_GNT	Input	Output of the Add_phase64 module defining ownership of the back-end.
DEVSELN	Input	Primary PCI input.
DMA_CTL_REG	Input	Control register from the DMA_REG module indicating type and

		length of DMA transfer.
DMA_GNT	Input	Input from the ADD_PHASE64 module granting ownership of the macro to the DMA engine.
GNTN	Input	Primary PCI input.
IRDYN	Input	Primary PCI input.
LATENCY_TIMER	Input	Maximum bus latency value from the Config module.
MASTER_BE	Input	4-bit bus from the back-end defining byte enables.
MASTER_BE64	Input	4-bit bus from the back-end defining byte enables on the upper 32-bits of data.
PERRn	Input	Primary PCI input.
STOPN	Input	Primary PCI input.
TRDYN	Input	Primary PCI input.
CBE_OUT	Output	Primary PCI output for command and byte enables.
CBE_PAR*	Output	CBE parity value for the CBE bus during master cycles.
DMA_DIRECTION	Output	Output indicating the direction, read or write, of the DMA transfer.
DMA_DONE	Output	Output pulse indicating the end of the DMA transfer.
DMA_INCREMENT	Output	Output indicating a successful dataphase on the PCI bus.
DMA_REQUEST	Output	Active high request from the DMA engine for control of the PCI macro.
FRAMEN_OUT	Output	Primary PCI output.
FRAMEN_PAD	Output	Single load FRAMEN signal that directly drives the output PAD.
MAST_ADDR_PH	Output	Active high output indicating that an address phase is in progress.
MASTER	Output	Active high input indicating that the macro is currently the bus master.
MASTER_ACTIVE	Output	Active high output to the back-end indicating that the macro is currently busy running a master transaction.
MASTER_EN	Output	Active high tri-state enable during Master cycles.
MASTER_ERROR	Output	3-bit bus that reports parity errors, master aborts, and target aborts during DMA transfers.
MASTER_TIMEOUT	Output	Active high output indicating that a Target failed to respond to the PCI transfer request within the time specified by the PCI specification.
RAM_START_ADD	Output	Back-end address source for the Add_cntr64 module during master cycles.
PCI_START_ADD	Output	PCI Start Address to the Datapath module.
REQ64N_PAD	Output	Primary PCI output indicating a 64-bit transaction.
REQN	Output	Primary PCI output.

DMA_REG Module

Signal Name	**Type**	**Description**
CLK	Input	PCI system clock.
ACK64N	Input	Primary PCI input.
DMA_DONE	Input	Input pulse from the DMA engine indicating the end of a DMA transfer.
DMA_INCREMENT	Input	Input form the DMA engine indicating successful transfer of data. This input is used to increment the address counters.
EXT_INTN	Input	Active low interrupt signal from the back-end.
MASTER_ERROR	Input	3-bit bus that reports parity error, master abort, and target aborts during DMA transfers.
MEM_ADD	Input	Address bus for the DMA registers.
GNTN	Input	Primary PCI input.
PARITY_ERROR	Input	Reports parity errors in master-only implementations.
WR_CONTROL	Input	Active high input from the wrapper level indicating that valid data should be registered into the selected DMA configuration register.
WR_DATA	Input	Data bus containing the write data for the DMA configuration registers.
DMA_CTL_REG	Output	DMA control register which defines type, direction, and length of PCI

		transfers.
DMA_DATA_RD	Output	DMA configuration read data.
INT_OUT	Output	Active high interrupt output.
PCI_START_ADD	Output	PCI Start Address to the Datapath module.
RAM_START_ADD	Output	Back-end address source for the Add_cntr64 module during master cycles.

Parity64 Module

The Parity64 module is responsible for checking and generating parity for both address and data cycles.

Signal Name	Type	Description
CLK	Input	PCI system clock.
CBE	Input	Primary PCI input used in parity calculations.
CBE_PAR32	Input	Parity generation for the CBE signals during DMA transfers.
CBE_PAR64	Input	Parity generation for the upper CBE signals during DMA transfers.
CYC64	Input	Active high output of the Add_phase64 module indicating a 64-bit transfer.
LOAD_PARITYN	Input	Active low signal from the Datapath module causing the PAR_OUT and PAR64_OUT signals to update.
MASTER	Input	Active high signal indicating that the DMA engine is currently in control of the PCI bus. For this case, parity for CBE is sourced from the CBE_PAR32 and CBE_PAR64 signals.
PAR	Input	Primary PCI parity input for the lower 32-bits of data and the lower 4-bits of CBE.
PAR_DATA	Input	Output of the Datapath module. Data used to generate and check parity.
PAR64	Input	Primary PCI parity input for the upper 32-bits of data and the upper 4-bits of CBE.
PERR_EN	Input	Active high output of the Config module enabling data parity checking.
PERRN_TIME	Input	Timing signal output of the Burst64 module indicating what cycles parity should be checked.
SERR_EN	Input	Active high output of the Config module enable address parity checking.
SERR_TIME	Input	Timing signal output of the Burst64 module indicating what cycles parity should be checked for address parity errors.
DRIVE_PERRN	Output	PCI PERRN tri-state enable.
PAR_OUT	Output	PCI PAR output.
PAR64_OUT	Output	PCI PAR64 output.
PERRN_SIG	Output	PCI PERRN output.
PERRS	Output	Output to the Config module indicating a parity error has occurred.
SERRN_OE	Output	PCI SERRN output.
SERRS	Output	Output to the Config module indicating a system level error has occurred (address parity error).

A Typical Target Cycle

A typical target cycle operation is as follows:

1. The macro continuously monitors the PCI bus for activity. Whenever FRAMEn asserts low, the macro registers the command and address and determines if it is being addressed.
2. If the macro is the addressed device, then a DP_START pulse is generated and the address is registered into the address counter which drives the MEM_ADDR bus. If the address is to some other device, then the macro returns to step #1.
3. The DP_START pulse initiates the state machine in the Burst64 module and should initiate back-end control logic to either begin fetching data (read) or prepare to receive data (write). The burst state machine waits for the back-end to go ready (RD_BE_RDY or WR_BE_RDY).
4. For reads from the back-end, the macro will assert the RD_BE_NOW signal and register the data on MEM_DATA bus and put it on the AD bus. On the following cycle, TRDYn will be asserted. Dataflow will continue so long as RD_BE_RDY and IRDYn are both active. If the RD_BE_RDY signal deactivates, then the macro will cease providing data to the PCI bus and deassert TRDYn. If IRDYn deasserts, then the macro will cease fetching data indicated by the deassertion of RD_BE_NOW signal. The back-end address increments whenever the RD_BE_NOW signal is active.
5. For writes to the back-end, once the WR_BE_RDY signal is active, then the macro will assert TRDYn. Whenever a PCI dataphase occurs (TRYDN and IRDYN are both low), the macro will register the data and pass it to the MEM_DATA bus on the following cycle and qualify it with a WR_BE_NOW signal. Transfers will continue so long as both IRDYn and WR_BE_RDY are active. If IRDYn deasserts, then data will stop flowing to the back-end indicated by the WR_BE_NOW signal. If the back-end deasserts WR_BE_RDY, then the macro will drive TRDYn inactive cutting off the flow of data from the PCI bus. The back-end address increments whenever the WR_BE_NOW signal is active.
6. Either step #4 or #5 will continue until the bus master drives FRAMEn inactive. When this occurs, the next dataphase will be the last transfer. The next dataphase asserts the DP_DONE signal which notifies the burst state machine, the address phase state machine, and the back-end that the transfer is over and should return to their idle states.

A Typical Master Cycle

A typical master cycle is as follows:

1. A master cycle begins when the DMA Request bit is set in the DMA Control Register. This bit causes the DMA state machine to move out of its idle state and begins requesting the PCI bus (REQn low). When the system arbiter grants the bus (GNTn low), then the macro begins driving the bus (AD, CBE, and FRAMEn) and the PCI Start Address register value is loaded onto AD. FRAMEn is then asserted beginning the cycle.
2. The transmit count value is loaded into a down counter which will decrement whenever a PCI dataphase occurs.
3. A master transfer causes the target logic to ignore its normal PCI decode logic and forces a DP_START. The DP_START initiates a transfer with the back-end just like in the

target except that the RAM Start Address is loaded into the MEM_ADDR counter. Steps #3 or #4 from the target operation then control the transfer. One difference with the Target is that the macro is now driving IRDYn instead of TRDYn.

4. The transmit counter continues to decrement until the count value approaches zero. On the second to last transfer, FRAMEn is driven inactive. The next dataphase causes the DP_DONE and DMA_DONE to pulse indicating completion of the transfer.