Generalized RAD Module Interface Specification of the Field-programmable Port eXtender (FPX) Version 2.0

David E. Taylor, John W. Lockwood, and Sarang Dharmapurikar

The Field-programmable Port eXtender (FPX) provides dynamic, fast, and flexible mechanisms to process data streams at the ports of the Washington University Gigabit Switch (WUGS-20). By performing all computations in FPGA hardware, cells and packets can be processed at the full line speed of the transmission interface, currently 2.4 Gbits/sec. In order to design and implement portable hardware modules for the Reprogrammable Application Devide (RAD) on the FPX board, all modules should conform to a standard interface. This standard interface specifies how modules receive and transmit ATM cells of data flows, prevent data loss during reconfiguration, and access off-chip... **Read complete abstract on page 2.**

Generalized RAD Module Interface Specification of the Field-programmable Port eXtender (FPX) Version 2.0

David E. Taylor, John W. Lockwood, and Sarang Dharmapurikar

Complete Abstract:

The Field-programmable Port eXtender (FPX) provides dynamic, fast, and flexible mechanisms to process data streams at the ports of the Washington University Gigabit Switch (WUGS-20). By performing all computations in FPGA hardware, cells and packets can be processed at the full line speed of the transmission interface, currently 2.4 Gbits/sec. In order to design and implement portable hardware modules for the Reprogrammable Application Devide (RAD) on the FPX board, all modules should conform to a standard interface. This standard interface specifies how modules receive and transmit ATM cells of data flows, prevent data loss during reconfiguration, and access off-chip memory. Module designers should conform to the standard I/O signal names and take special note of timing diagram references.

Generalized RAD Module Interface Specification of the Field-programmable Port eXtender (FPX) Version 2.0

David E. Taylor, John W. Lockwood and
Sarang Dharmapurikar

WUCS-01-15

July 2001

Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
St. Louis MO 63130

Generalized RAD Module Interface Specification of the Field-programmable Port eXtender (FPX) Version 2.0

David E. Taylor, John W. Lockwood, Sarang Dharmapurikar

WUCS-TM-01-15

July 5, 2001

Department of Computer Science
Applied Research Lab
Washington University
Campus Box 1045
One Brookings Drive
Saint Louis, MO 63130

Abstract

The Field-programmable Port eXtender (FPX) provides dynamic, fast, and flexible mechanisms to process data streams at the ports of the Washington University Gigabit Switch (WUGS-20). By performing all computations in FPGA hardware, cells and packets can be processed at the full line speed of the transmission interface, currently 2.4 Gbits/sec. In order to design and implement portable hardware modules for the Reprogrammable Application Device (RAD) on the FPX board, all modules should conform to a standard interface. This standard interface specifies how modules receive and transmit ATM cells of data flows, prevent data loss during reconfiguration, and access off-chip memory. Module designers should conform to the standard I/O signal names and take special note of timing diagram references.

Supported by: NSF ANI-0096052

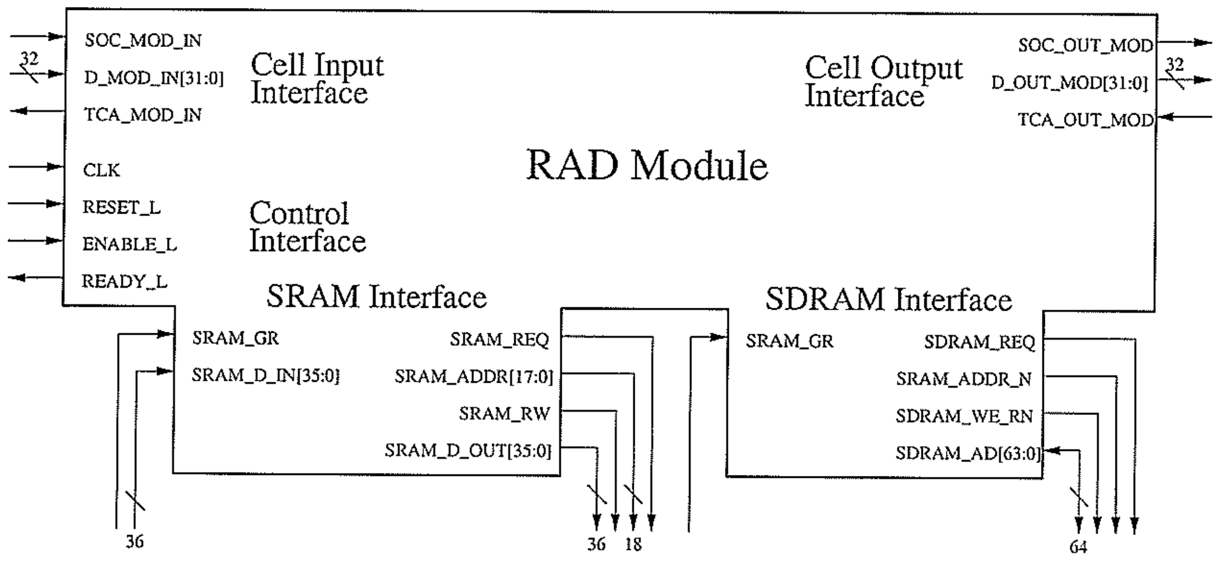

Figure 1: RAD Module Interface

1 Introduction

In order to design and implement portable hardware modules for the Reprogrammable Application Device (RAD) on the FPX board, all modules should conform to a standard interface. In order to process data flows, prevent data loss during reconfiguration, and access off-chip memory, the RAD module includes the following interfaces:

- Control Interface
- Cell Input Interface
- Cell Output Interface
- SRAM Interface
- SDRAM Interface

A diagram of the interface is shown in Figure 1. Cell Input and Output interfaces implement cell-based flow control; therefore, data will be moved in and out of modules as complete ATM cells. The Control Interface implements a two-way handshake to ensure that data is not lost when the RAD FPGA is reconfigured during system operation. The SRAM and SDRAM Interfaces allow the module to interface to off-chip memory. Standard memory interfaces are provided for modules in order to abstract module designs from device specific signaling and timing of the memory chips and reduce redundancy across module designs.

2 Control Interface

The module Control Interface asserts a localized, synchronous reset and global 100MHz clock to the module. It also includes a reconfiguration handshake to prevent data loss during RAD reconfiguration. The timing for these signals is shown in Figure 2.

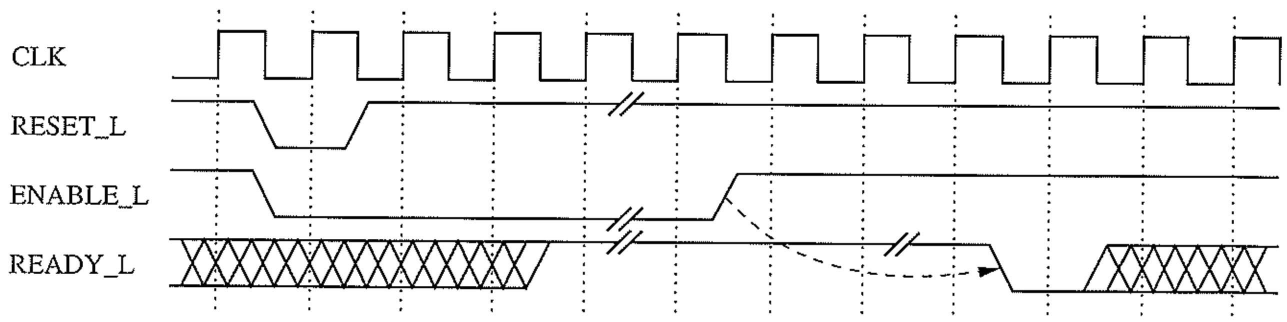

Figure 2: Timing diagram for Control Interface

CLK

This is the 100MHz (T=10ns) global clock signal. All interface signals are asserted synchronous to this clock

RESET_L

This signal is a localized, synchronous reset for the module. It will be asserted low for a single clock cycle. All logic internal to the module should implement a synchronous reset using this signal.

ENABLE_L

This signal is asserted low to the module on the same clock cycle as RESET_L following reconfiguration. When this signal is de-asserted high, the module must stop accepting cells by de-asserting TCA_IN_MOD and flush it's internal pipelines.

READY_L

This signal performs the handshake back to the reconfiguration control. The module must pull this signal high following reset in order to prevent reconfiguration during module operation. After ENABLE_L is de-asserted high to the module and it has stopped accepting cells and flushed it's internal pipelines, the module must assert this signal low in order to signal back to the control interface that it is ready for reconfiguration.

3 Cell Input Interface

The Cell Input Interface takes in ATM cells as a 14, 32-bit words. A 32-bit parallel data path and two flow control signals are used to input cells to a module. If Transmit Cell Available (TCA_MOD_IN) is asserted by a module, it must accept a complete ATM cell. The first 32-bit word of the cell arrives with Start of Cell (SOC_MOD_IN). The timing for these signals is shown in Figure 3.

SOC_MOD_IN

This signal is asserted high for one clock cycle to signal the Start of Cell. This signal may be driven by the NID or an upstream module. It is synchronous to CLK.

D_MOD_IN[31:0]

This 32-bit input data bus carries the 32-bit words of the ATM cell. The first word of the ATM cell arrives on the same clock cycle in which SOC_MOD_IN is asserted. The remaining 13 (32-bit) words arrive

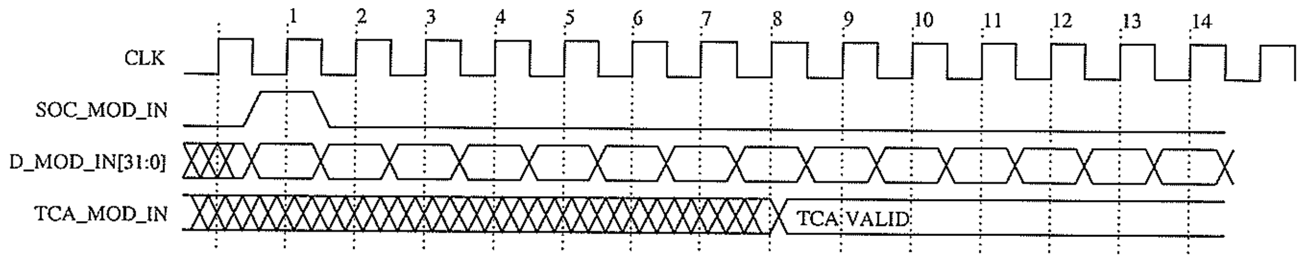

Figure 3: Timing diagram for Cell Input Interface.

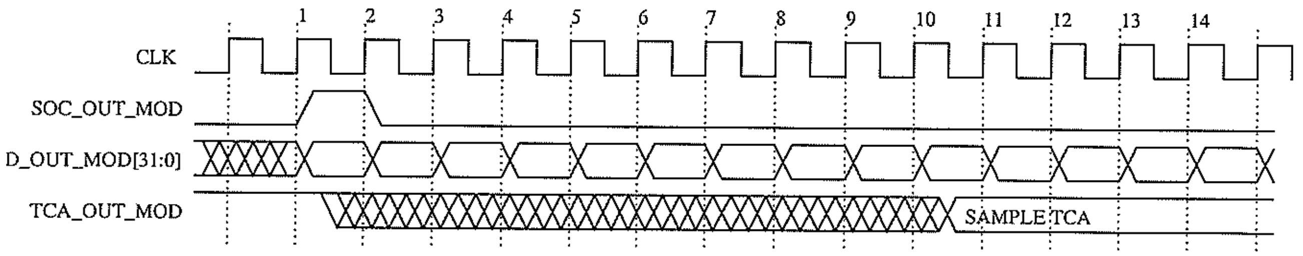

Figure 4: Timing diagram for Cell Output Interface.

on the following 13 clock cycles. Therefore, the entire ATM cell is transferred to the module in 14 clock cycles.

TCA_MOD_IN

This signal performs the handshake back to the input for cell transfer. When this signal is asserted high, the input is free to send cells. Holding this signal low prevents the input from sending cells. TCA_MOD_IN must be valid at least 6 clock cycles before the last cycle of the current cell transfer.

4 Cell Output Interface

The Cell Output Interface must send out ATM cells as 14, 32-bit words. A 32-bit parallel data path and two flow control signals are used to output cells from a module. If Transmit Cell Available (TCA_OUT_MOD) is asserted by the output, the module must not send a cell. When sending a cell, the module must send a complete ATM cell. The first 32-bit word of the cell must arrive with Start of Cell (SOC_OUT_MOD) and the remaining 13 words of the cell must be sent on the subsequent clock cycles. The timing for these signals is shown in Figure 4.

SOC_OUT_MOD

This signal is asserted high for one clock cycle to signal the Start of Cell. This signal may be driven to the NID or a downstream module. It must be asserted synchronous to CLK.

D_OUT_MOD[31:0]

This 32-bit output data bus carries the 32-bit words of the ATM cell. The first word of the ATM cell must be sent on the same clock cycle in which SOC_OUT_MOD is asserted. The remaining 13 (32-bit) words must be sent on the following 13 clock cycles. Therefore, the entire ATM cell is transferred to the output in 14 clock cycles.

TCA_OUT_MOD

This signal performs the handshake back to the module for cell transfer. When this signal is asserted high, the module is free to send cells. When this signal is held low, the module must not send cells. Modules must sample TCA_OUT_MOD no sooner than 3 clock cycles prior to asserting SOC_OUT_MOD.

5 SRAM Interface

Both of the SRAMs on the FPX board are accessible from the RAD FPGA. RAD Modules may access SRAM by interfacing to one of the two identical SRAM Interfaces. These interfaces simplify memory transactions by abstracting RAD modules from the signal timing constraints of the Micron ZBT SRAM. Each memory interface arbitrates requests for memory access using a request (SRAM_REQ) and grant (SRAM_GR) signal handshake. Once a module is granted access to memory, it may read from memory by holding the read/write signal (SRAM_RW) high and issuing the read address on the address signals (SRAM_ADDR[17:0]). The read data will appear on the data input signals (SRAM_D_IN[35:0]) after 4 clock cycles. A module may write to memory by asserting the read/write signal low, issuing the write address on the address signals, and issuing the write data on the data output signals (SRAM_D_OUT[35:0]). A module may issue a read or write on every clock cycle.

All modules must flop all IO signals at the module boundary to guarantee that designs will meet post layout timing. Thus, data from read transactions will be available inside the module after six (6) clock-cycles. This will be the reference point for subsequent timing diagrams and is illustrated in Figure 5.

Figure 6 shows an example memory transaction. The module issues a request and receives a grant on the next clock cycle (assuming no other devices are currently using memory). The module then issues sequential transactions: write 0, write 1, write 2, read 0, write 3, read 1, read 2, read 3. The data and addresses are the same in this example for simplicity. Note that the grant signal is lowered before the memory transactions are complete. After receiving the data from the last read, the module de-asserts its request signal to release the memory.

SRAM_REQ

This signal is asserted high to the SRAM interface to request and hold access to memory. The module must hold this signal high until the memory transaction is complete.

SRAM_GR

This signal is asserted high to the module when the SRAM interface grants the module access to memory. When a contending module requests access to memory, this signal is de-asserted low. The module must complete its current transactions and release the memory by de-asserting its request signal. Note that a module may hold memory for several cycles after the grant signal is removed in order to complete a memory transaction. The module designer is responsible for ensuring that starvation of a contending module does not occur.

SRAM_ADDR[17:0]

This 18-bit address bus carries the memory address for reads and writes.

SRAM_RW

This signal specifies the type of memory access. High assertion specifies a read, while a low assertion specifies a write. The module should hold this signal high (READ) except when asserting it low (WRITE) with the address and data for a write transaction to prevent overwriting valid memory contents.

SRAM_D_IN[35:0]

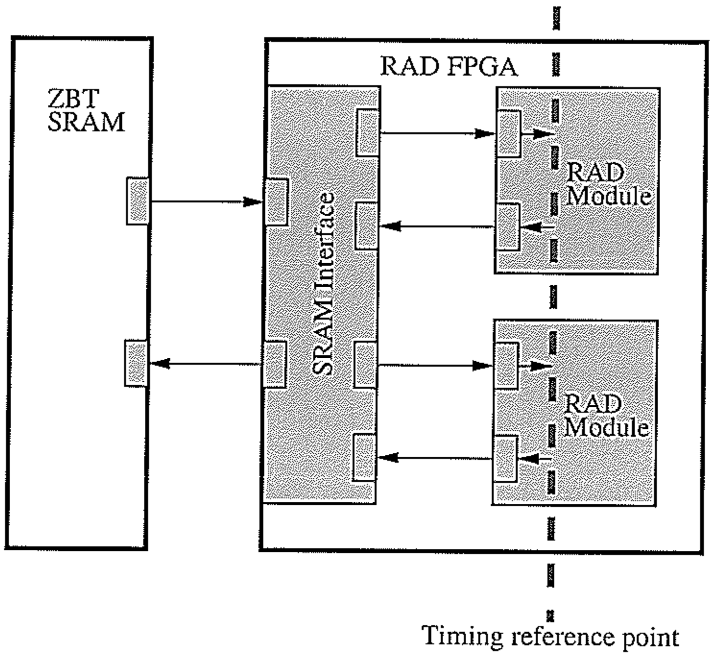

Figure 5: Point of reference for module SRAM timing diagram. This assumes that all signals are flopped at the module boundary, providing a six (6) clock-cycle latency to external SRAM.

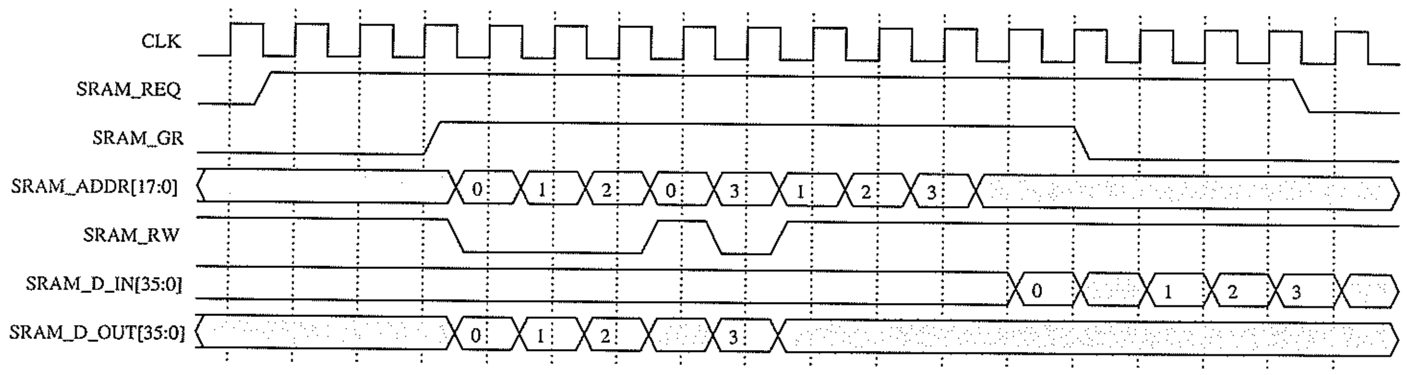

Figure 6: Module timing diagram for SRAM Interface. Note that this diagram uses a point of reference from inside the module and assumes that all signals are flopped at the module boundary.

This 36-bit data bus carries read data from memory. Read data is available 4 clock cycles after the read signal and address are asserted.

SRAM_D_OUT[35:0]

This 36-bit data bus carries write data to memory. Write data must be issued during the same clock cycle that address and write (SRAM_RW = 0) are asserted. Data will be written to memory 4 clock cycles after the read signal, address, and data are asserted.

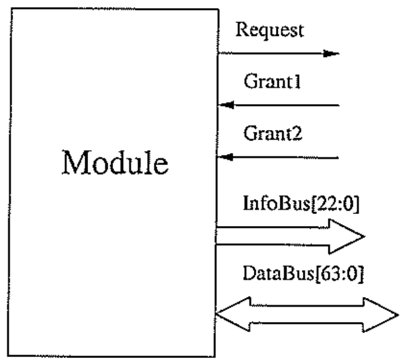

Figure 7: Module Interface for the SDRAM access

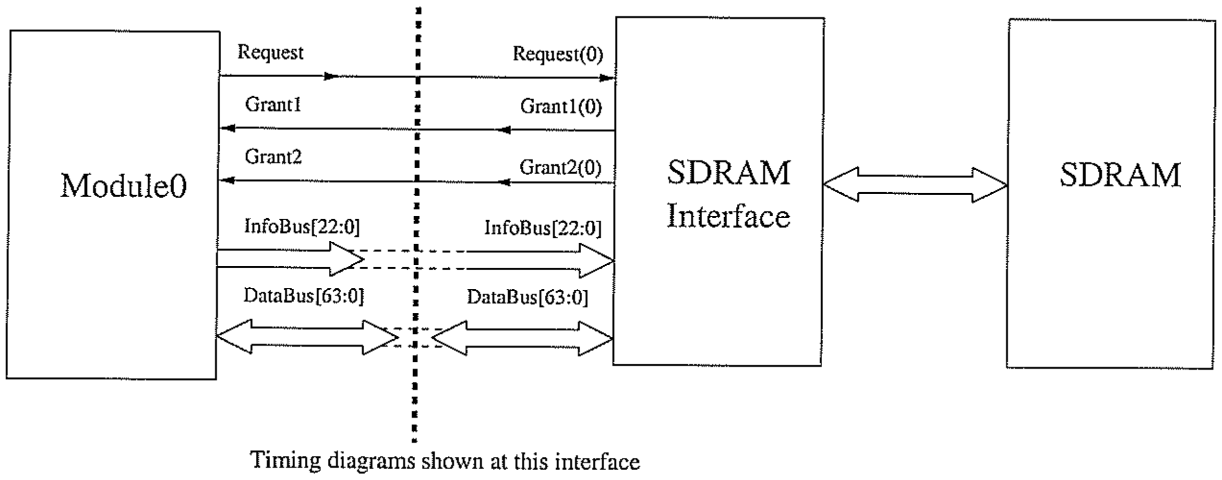

Figure 8: Connecting a module to the SDRAM interface

6 SDRAM Interface

Figure 7 shows the module interface for the SDRAM. This interface has two types of signals : one which participate in the arbitration for the SDRAM access and other which actually access the SDRAM. The signals Request, Grant1, Grant2 shown in this figure are used for the arbitration while others are used to access the SDRAM. When a module wants to make a SDRAM operation, it requests the SDRAM interface for the SDRAM access by asserting the Request signal high. A module can make multiple requests for multiple SDRAM operations by holding the Request signal high for multiple clock cycles. Each request corresponds to one SDRAM operation. After making the request(s) then module waits for the Grant1 signal. When the Grant1 signal is asserted high, the module gives all the information regarding its SDRAM transaction like the address, the operation type and the burst length to the SDRAM interface on the InfoBus. The SDRAM interface analyzes this information and schedules the actual SDRAM access for this module. At the scheduled time, the SDRAM interface asserts Grant2 high which means that the request (first request in the queue in case of multiple requests) has been granted access to the SDRAM and module can read the data from the DataBus in case of read request or put data on the DataBus in case of a write request. All the requests made my a module are serviced sequentially which means if information regarding request A was sent before the information regarding the request B then A will be serviced before B.

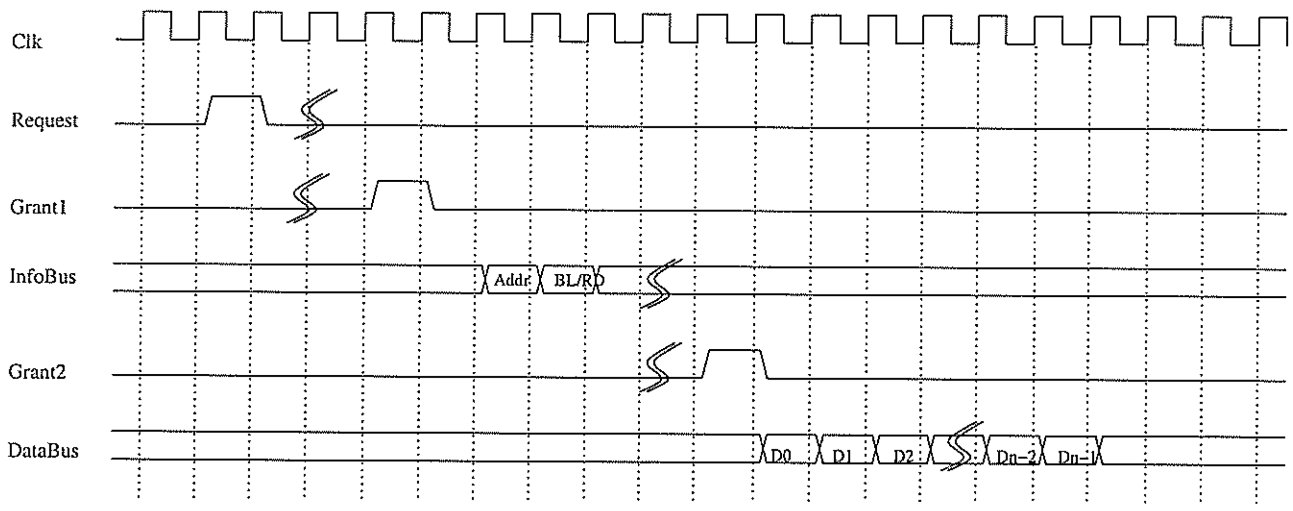

Figure 9: Timing diagram for a burst read operation

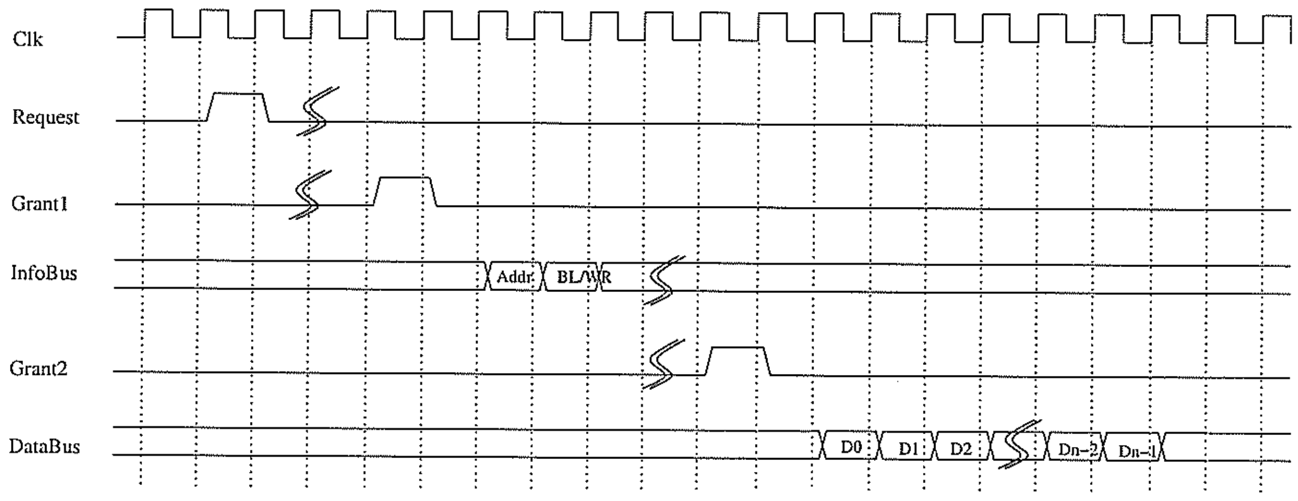

Figure 10: Timing diagram for a burst write operation

Figure 9 shows the exact protocol and timing for a burst read operation of burst length n and Figure 10 shows the timing of a burst write operation of length n . The discontinuation sign on the signals indicates that there is a variable clock cycles of latency between the Request and Grant1 as well as Grant1 and Grant2.

Figure 8 shows how the module is connected to the SDRAM interface. Request signal of module 'x' is connected to the Request(x) of the SDRAM interface. Similarly Grant1 signal of module is conneccted to Grant1(x) and Grant2 to Grant2(x) of the SDRAM interface.

Following is the list of interface signals and their purpose.

- Request : This signal is used to make a request to the SDRAM interface for SDRAM access. Each request corresponds to one SDRAM transaction. The number of requests made is equal to the number of clock cycles for which the Request is asserted high. Every request gets corresponding grant. *The number of outstanding requests which haven't got the grant can not exceed 3* . Thus, if a module asserts the request signal high for 3 clock cycles then it means it has made 3 requests and if the first request gets a grant then the module can make only one more request.

- Grant1 : This signal is the input to the module from the SDRAM interface. When this signal is asserted high, it means that the module has been given grant to access the InfoBus(described below), onwhich it can put the information regarding *only one* the SDRAM transaction it wants to do. This information includes the SDRAM address, the type of operation (read or write) and the burst length of this operation. First the 23 bit address should appear on the InfoBus at the time shown in the Figure 10 and Figure 9 then burst length and operation type should appear together on the bus in the next clock cycle.*The number of clock cycles between the Request and Grant1 is variable.*

- InfoBus[22:0] : This is a unidirectional tristate bus which is used to supply information regarding a SDRAM transaction to the SDRAM interface. After Grant1 is given to the module, it should put the 23 bit address on the InfoBus such that InfoBus[22:0] = Address[22:0]. In the next clock cycle it should put the 8 bit burst length and the operation type on the InfoBus such that InfoBus(8) = OperationType and InfoBus[7:0] = BurstLength[7:0]. The convention for the operation type is OperationType = '0' for a *read* operation and '1' for a *write* operation. *Other than these two clock cycles, the module should always drive HiZ on the InfoBus.*

- Grant2 : This signal is a input to the module from the SDRAM interface. For a read operation as shown in Figure 9, when Grant2 signal is asserted high the module should start taking in the data from the next clock cycle. This data corresponds to the read request which was at the head of the queue. Since all the requests made by the module are serviced in the same order in which they appear the module should keep track of the which data corresponds to which request. For a write operation as shown in Figure 10, when Grant2 signal is asserted high, then module should start putting data words on the data bus two clock cycles thereafter.

- DataBus[63:0] : This is a bidirectional tri-state data bus. The module should drive HiZ when it is not used by the module.