Report Number: WUCS-01-16

2001-01-01

RAD Module Infrastructure of the Field-programmable Port eXtender (FPX) Version 2.0

David E. Taylor, John W. Lockwood, and Naji Naufel

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). In order to facilitate the design and implementation of portable hardware modules for the Reprogrammable Application Device (RAD) on the FPX board, infrastructure components have been developed. These components abstract application module designers from device-specific timing specifications of off-chip memory devices, as well as processing system-level control cells. This document describes the design and internal functionality of the infrastructure components and is intended as a reference for future component revisions and additions. Application module... Read complete abstract on page 2.

RAD Module Infrastructure of the Field-programmable Port eXtender (FPX) Version 2.0

David E. Taylor, John W. Lockwood, and Naji Naufel

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). In order to facilitate the design and implementation of portable hardware modules for the Reprogrammable Application Device (RAD) on the FPX board, infrastructure components have been developed. These components abstract application module designers from device-specific timing specifications of off-chip memory devices, as well as processing system-level control cells. This document describes the design and internal functionality of the infrastructure components and is intended as a reference for future component revisions and additions. Application module designers should refer to the Generalized RAD Module Interface Specification of the Field Programmable Port Extender (FPX), EUCS-TM-01-15.

RAD Module Infrastructure of the Field-programmable Port eXtender (FPX) Version 2.0

David E. Taylor, John W. Lockwood and Naji Naufel

WUCS-01-16

July 2001

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

RAD Module Infrastructure of the Field-programmable Port eXtender (FPX) Version 2.0

David E. Taylor, John W. Lockwood, Naji Naufel

WUCS-TM-01-16

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). In order to facilitate the design and implementation of portable hardware modules for the Reprogrammable Application Device (RAD) on the FPX board, infrastructure components have been developed. These components abstract application module designers from device-specific timing specifications of off-chip memory devices, as well as processing system-level control cells. This document describes the design and internal functionality of the infrastructure components and is intended as a reference for future component revisions and additions. Application module designers should refer to the Gereneralized RAD Module Interface Specification of the Field Programmable Port Extender (FPX), WUCS-TM-01-15.

Supported by: NSF ANI-0096052

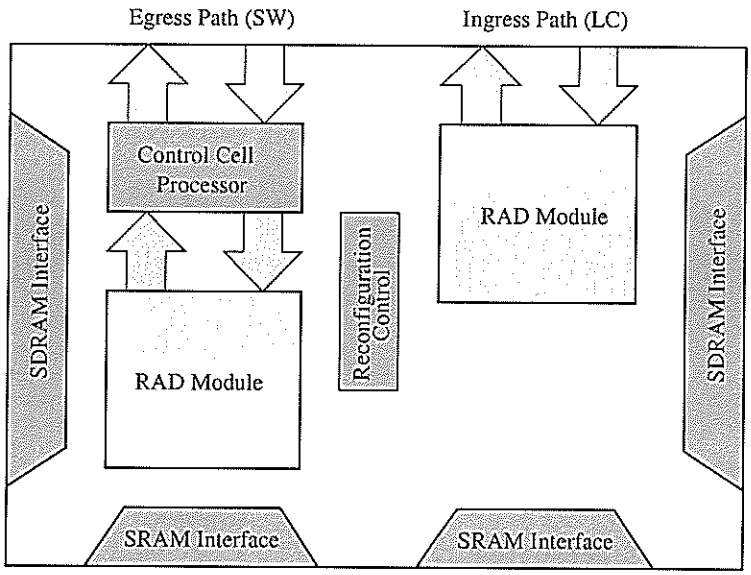


Figure 1: RAD design containing a single ingress/egress modules, SRAM interfaces, and reconfiguration control.

1 Introduction

Designers may use FPX resources in a multitude of ways to perform network data processing. The entities that perform the network data processing will be referred to as modules. Some modules may use all of the available memory and logic resources, while others use only a single memory resource and a fraction of the available logic resources. In the later case, multiple modules could be implemented in the RAD FPGA. In order to support such designs, several pieces of infrastructure logic are necessary.

This document provides the necessary information for creating RAD designs containing two modules. As shown in Fig. 1, double module RAD designs place a module on both the ingress and egress path while providing SRAM interfaces and a reconfiguration control block. SDRAM interfaces are not implemented at this time. ALL MODULES SHOULD CONFORM TO THE RAD MODULE INTERFACE SPECIFICATION. This specification is provided in a supplemental document and ensures that modules remain interoperable and position independent.

2 RAD FPGA Overview

The Reprogrammable Application Device (RAD) in the Field-programmable Port eXtender (FPX) is a Xilinx Virtex 1000-E Field Programmable Gate Array (FPGA). This device contains a 64x96 CLB array, with each CLB containing 4 D-type Flip-Flops and 4 Look-Up-Tables (LUTs), for a total of 24,576 Flops and LUTs. On-chip memory is available in 4096-bit blocks, called BlockRAMs. The device contains 96 BlockRAMs organized in 6 columns of 16 blocks each. The architecture of the Xilinx Virtex 1000-E along with the column addressing scheme is shown in Fig. 2. Coupled with the Network Interface Device (NID), the RAD provides logic and memory resources for network data processing. On-chip memory, two external Zero-Byte-Turnaround (ZBT) SRAMs (Synchronous Random Access Memories), and two external SDRAMs (Synchronous Dynamic Random Access Memories) comprise the memory resources of the RAD FPGA.

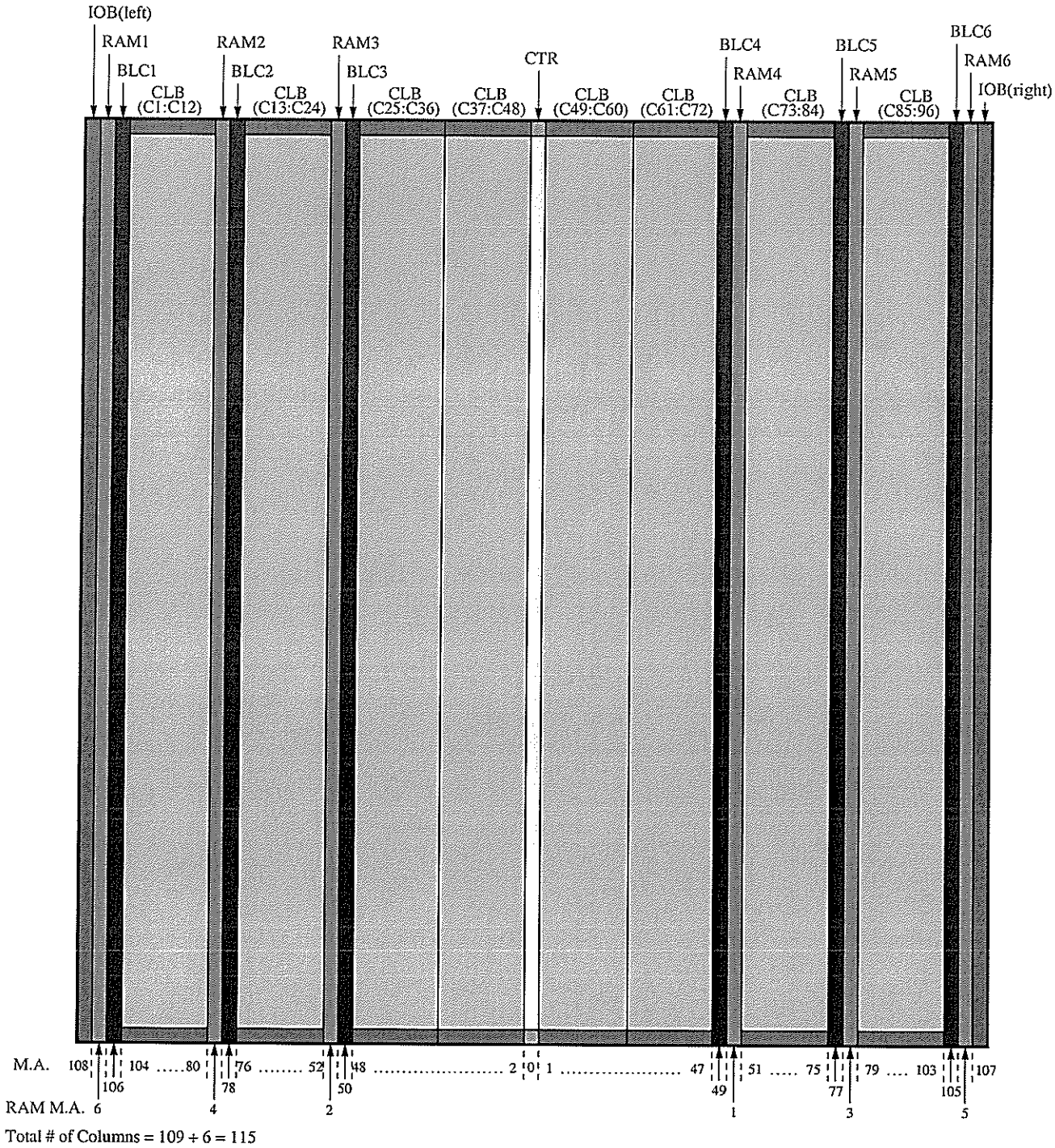


Figure 2: Architecture and column addressing for Xilinx Virtex 1000-E FPGA.

3 Reconfiguration Control

In order to prevent data loss during reconfiguration, Reconfiguration Control is needed to facilitate a handshake between the NID FPGA and RAD modules. As shown in the block diagram in Fig. 3, Reconfiguration Control provides independent interfaces for each module and uses the RAD_RECONFIG lines to interface to the NID FPGA. Note that the RAD_RECONFIG lines provide a three-way handshake between with a bit serial module identification line. Each module interface includes a synchronous reset and reconfiguration handshake signals, enable and ready. As shown in Fig. 4, the Reconfiguration Control state machine latches the serial module identification bits sent from the NID FPGA for after a reconfiguration request. Based on the module indentification bits, the module reconfiguration handshake is initiated with the appropriate module. After this handshake is completed, Reconfiguration Control signals back to the NID FPGA that the module is ready for reconfiguration. The NID FPGA signals when reconfiguration is complete and Reconfiguration Control initiates a reset and enable of the module. Timing for this process is shown in Fig. 5.

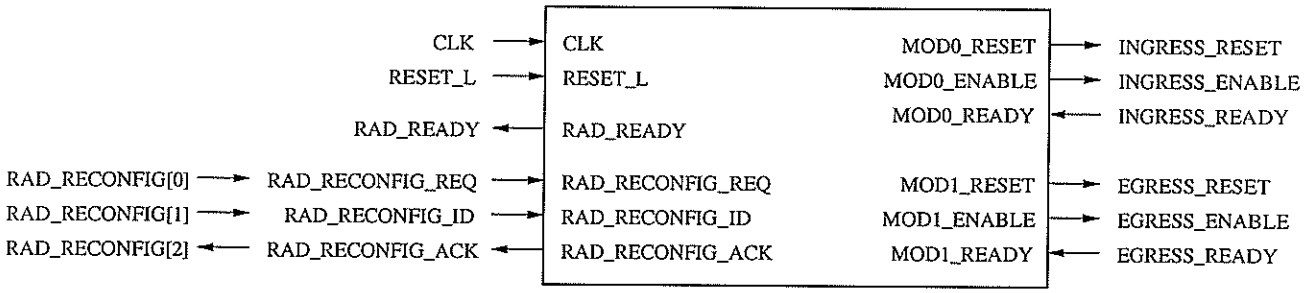


Figure 3: Block diagram of Reconfiguration Control; implements reconfiguration handshake between NID FPGA and RAD modules in order to prevent data loss during reconfiguration.

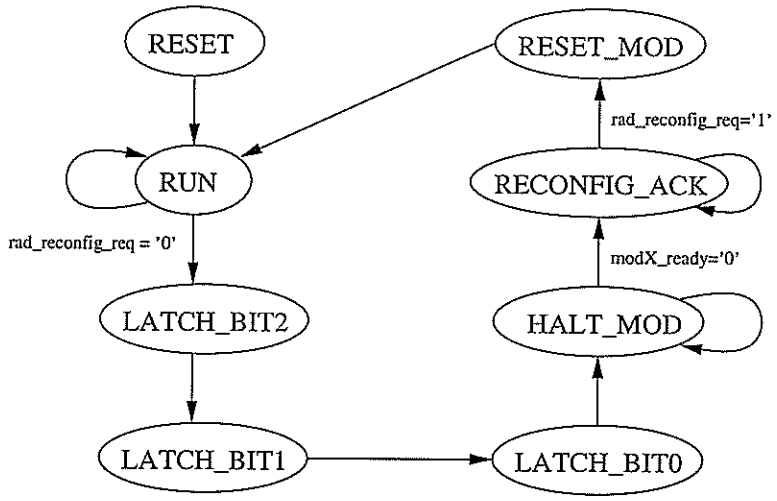


Figure 4: Finite state machine bubble diagram for Reconfiguration Control.

CLK

Global 100MHz clock. All signals are synchronous to this clock.

RESET_L

Global synchronous reset.

RAD_READY

After the RAD is completely configured, this signal is asserted low to the NID.

RAD_RECONFIG_REQ

This signal is driven by the NID to the RAD as RAD_RECONFIG(0). After the RAD is completely configured, the NID will pull this signal high. When a partial reconfiguration command is issued to the NID, it asserts this signal low to notify the RAD. The signal returns high after all reconfiguration data is written.

RAD_RECONFIG_ID

This signal is driven by the NID to the RAD as RAD_RECONFIG(1), and carries a 3 bit module indentification in serial form. This identification specifies which module will be reconfigured and is sent on the 3 clock cycles following the assertion of RAD_RECONFIG_REQ. The bits arrive MSB first, LSB last. After the LSB is received, Reconfiguration Control initiates the reconfiguration handshake with the specified module.

RAD_RECONFIG_ACK

This signal is driven by the RAD to the NID as RAD_RECONFIG(2). After the RAD is completely configured, this signal will be pulled high. After the partial reconfiguration request, module identification, and module reconfiguration handshake is completed, this signal is asserted low to signal that partial reconfiguration may take place without data loss. This signal returns high after reconfiguration is complete and the module has been reset and enabled.

The following signals are duplicated for each module in the RAD FPGA. The module number (#) is specified by the module identification.

MOD#_RESET

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

MOD#_ENABLE

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 and flush it's internal pipelines.

MOD#_READY

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

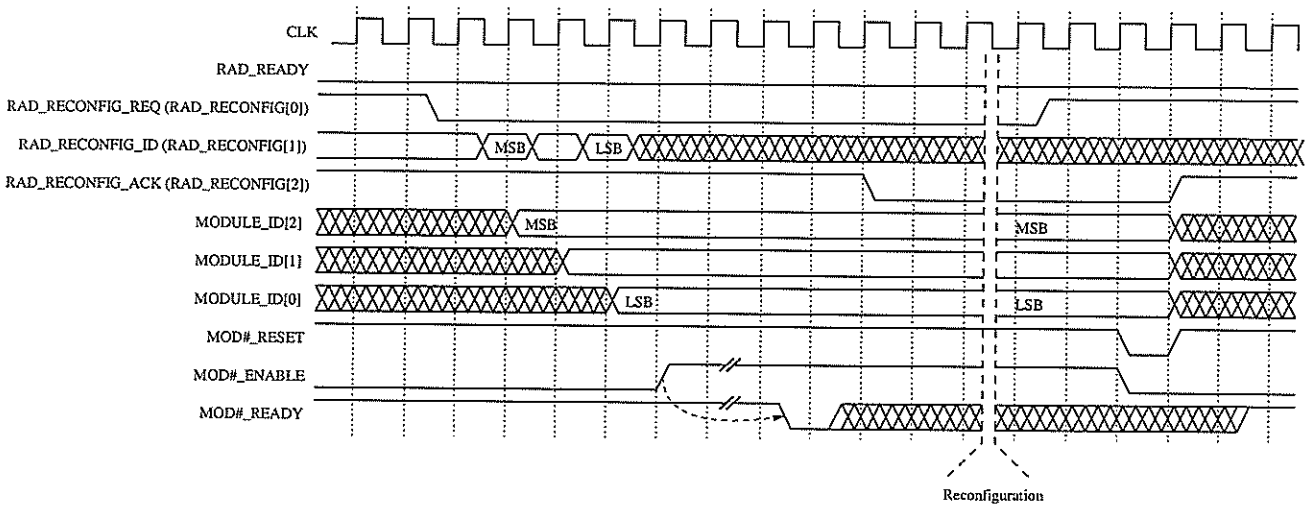


Figure 5: Timing diagram for Reconfiguration Control.

4 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. A block diagram of the SRAM Interface is shown in Figure 6. 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 (mod#_req) and grant (mod#_gr) signal handshake with each module. The finite state machine diagram for this handshake is shown in Figure 7. Once a module is granted access to memory, it may read from memory by holding its read/write signal (mod#_rw) high and issuing the read address on its address signals (mod#_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.

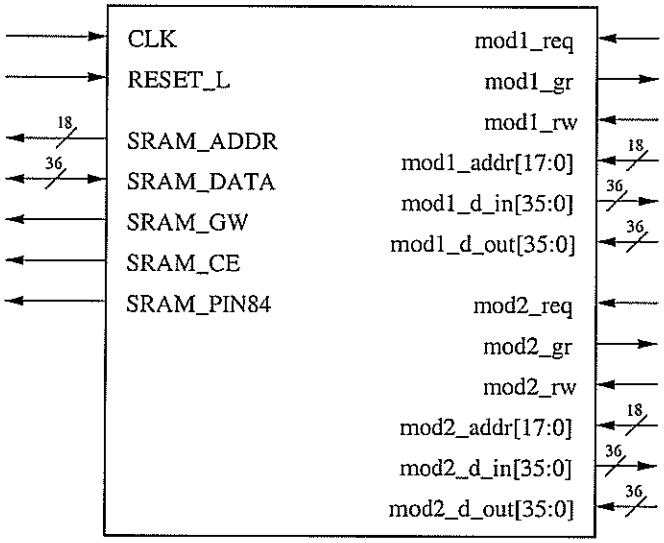


Figure 6: Block diagram for SRAM Interface.

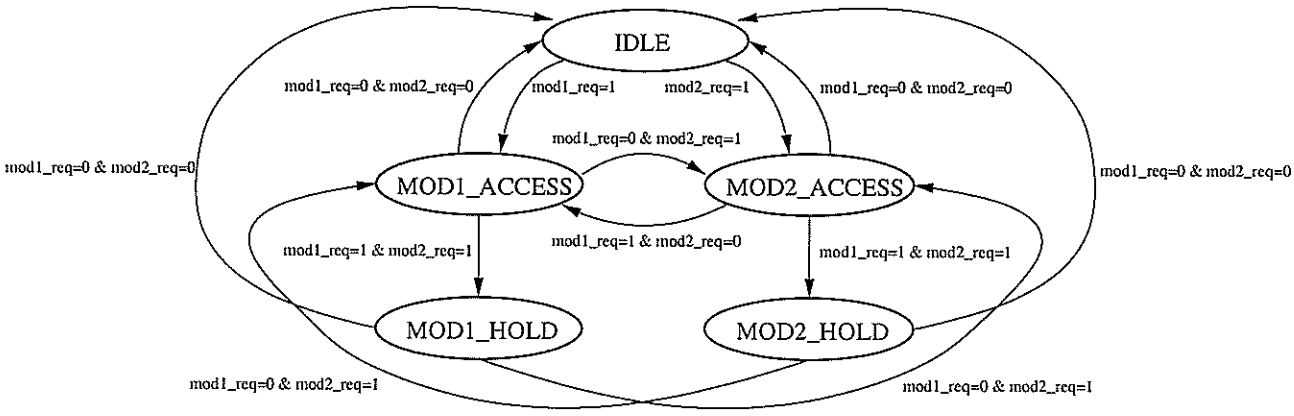


Figure 7: Finite state machine diagram for request/grant handshake with SRAM Interface.

Figure 8 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.

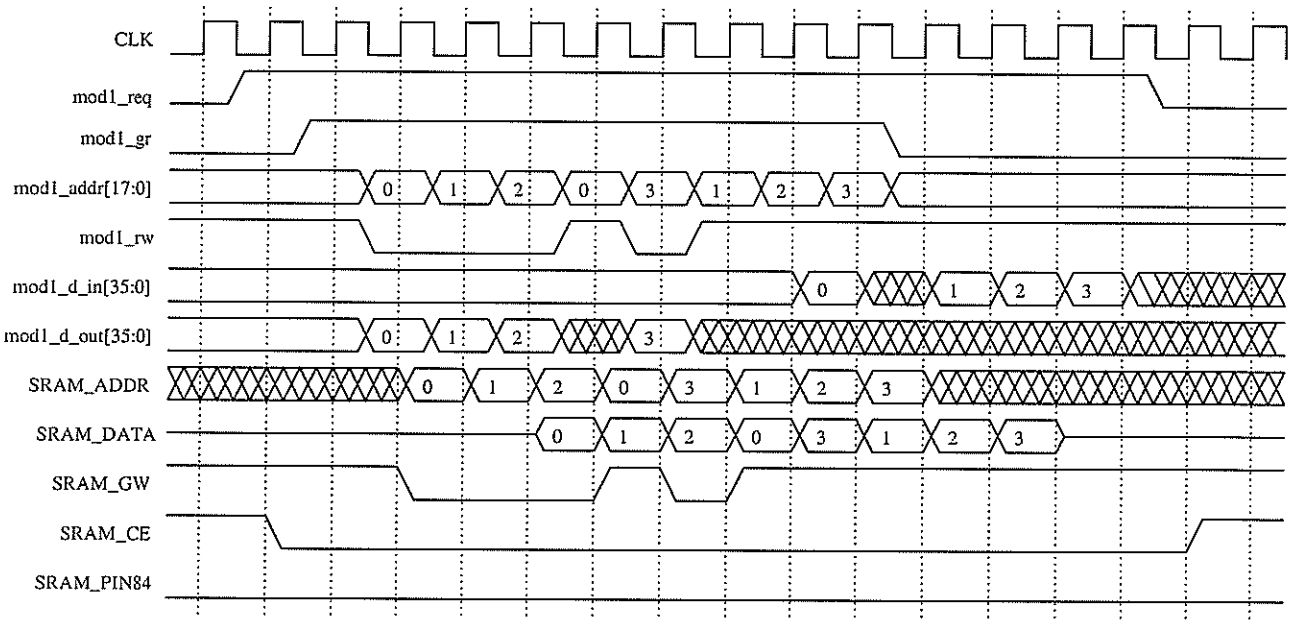


Figure 8: Timing diagram for an example memory transaction with SRAM Interface. Note that the grant signal is deasserted prior to the end of the memory transactions. This occurs when other modules request memory access during transactions. The module holds its request signal high until its transactions are complete, then releases the memory.

CLK

Global 100MHz clock. All signals are synchronous to this clock.

RESET_L

Global synchronous reset. This signal is used to reset the finite state machine only.

SRAM_ADDR[17:0]

This 18-bit address bus carries the memory address for reads and writes from the SRAM Interface to the memory. The SRAM Interface latches the address lines in the output D-flip-flop of the chip IOB.

SRAM_DATA[35:0]

This 36-bit tri-state data bus carries the data between the SRAM Interface and memory for reads and writes. Write data is delayed in the SRAM Interface for two clock cycles to satisfy the timing specifications of the Micron ZBT SRAM. The SRAM Inteface implements the necessary tri-state buffers and control.

SRAM_GW

This signal is connected to the read/write pin of the memory and is asserted low for writes, high for reads. When a module has access to the memory, this signal is controlled by the module read/write signal.

SRAM_CE

This signal is connected to the chip enable pin of the memory. When a module is granted access to memory, the SRAM Interface asserts this signal low, otherwise it is pulled high.

SRAM_PIN84

This signal is currently an unused address pin that is pulled low.

The following signals are duplicated for each module connected to the interface. The module number (#) is specified by the module identification.

mod#_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.

mod#_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.

mod#_addr[17:0]

This 18-bit address bus carries the memory address for reads and writes from the module to the SRAM Interface.

mod#_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.

mod#_d_in[35:0]

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

mod#_d_out[35:0]

This 36-bit data bus carries write data from the module to the SRAM Interface. Write data must be issued during the same clock cycle that address and write (mod#_rw = 0) are asserted. Data will reside in memory 4 clock cycles after the write signal, address, and data are issued to the SRAM Interface.

5 Control Cell Processor

The Control Cell Processor (CCP) allows data to be written to FPX memory devices connected to the RAD FPGA via ATM control cells. Read and write commands may be issued for either of the two ZBT SRAM devices or SODIMM SDRAM devices; however, only the SRAM interface is implemented at this time. A block diagram of the CCP is shown in Figure 9.

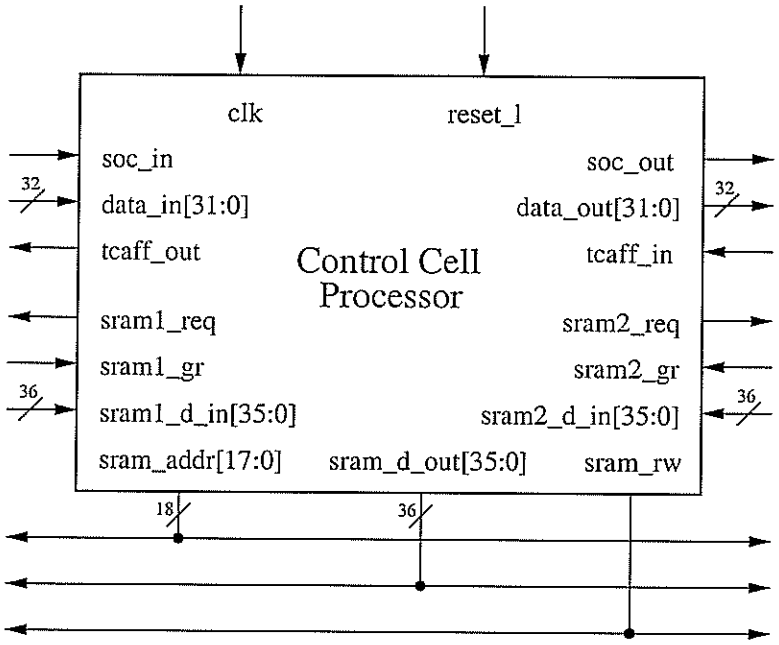


Figure 9: Block diagram of the Control Cell Processor (CCP).

The CCP examines all cell headers and captures properly formatted control cells destined for the CCP. All cells with an incorrect HEC field are dropped. CCP control cells must arrive on the FPX Control VCI for the RAD FPGA; hence, the ATM header fields must be VPI=0x000, VCI=0x0032, and a correct HEC. RAD control cells are addressed to RAD modules using the ModuleID field. As the CCP will most likely be included in most designs, the CCP is identified by ModuleID=0x00.

The control cell format used for memory transactions depends on the destination device. The control cell format for SRAM transactions is shown in 10. A control cell may issue read andor write commands to either SRAM devices on the FPX. Transactions may consist of one 36-bit read or write, two 36-bit reads or writes tofrom consecutive addresses, or one to eight 32-bit read or writes tofrom consecutive addresses. The control cell is parsed starting with the first payload word. Following is a definition of each field in the command word.

V (Valid command bit) identifies valid command words. 1 - valid command, 0 - invalid command. Upon reaching the end of the control cell or the first invalid command word, the CCP prepares to send a response cell.

D (Device bit) selects the destination device for the memory transaction. 1 - SRAM2, 0 - SRAM1

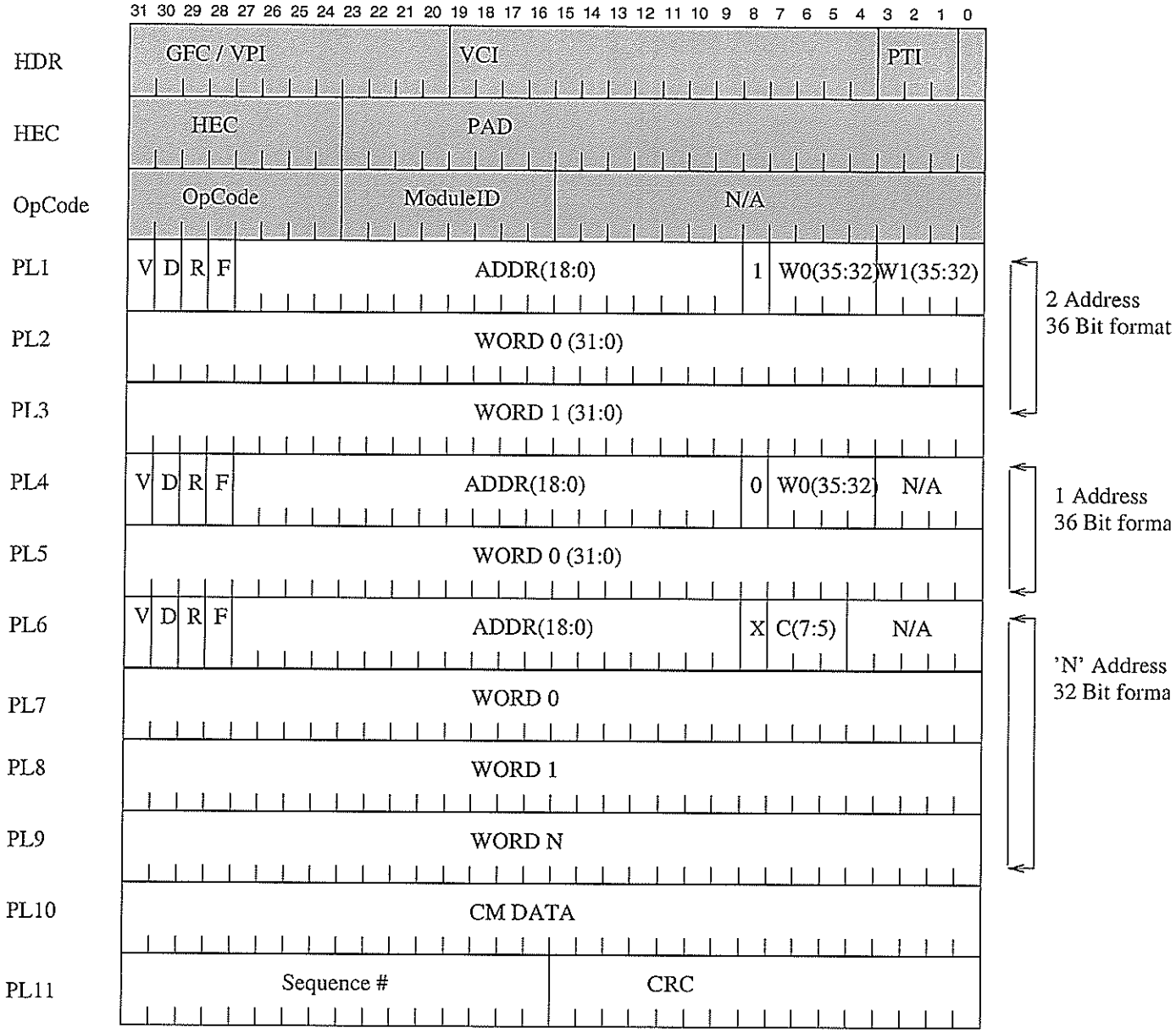
R (ReadWrite bit) designates the type memory transaction. 1 - read, 0 - write. Read response cells will contain the data read from memory, while write response cells will simply echo the data written to memory.

F (32- or 36-bit transaction) designates the width of the data words used in the memory transaction. 1 - 36-bit,

0 ¯ 32-bit.

Count Fields For 36-bit transactions, bit eight (8) of the command word is used to select a single- or double-word transaction. 1 ¯ double-word, 0 ¯ single-word.

Control Cell Format for 32/36 bit RAD SRAM Memory Operations

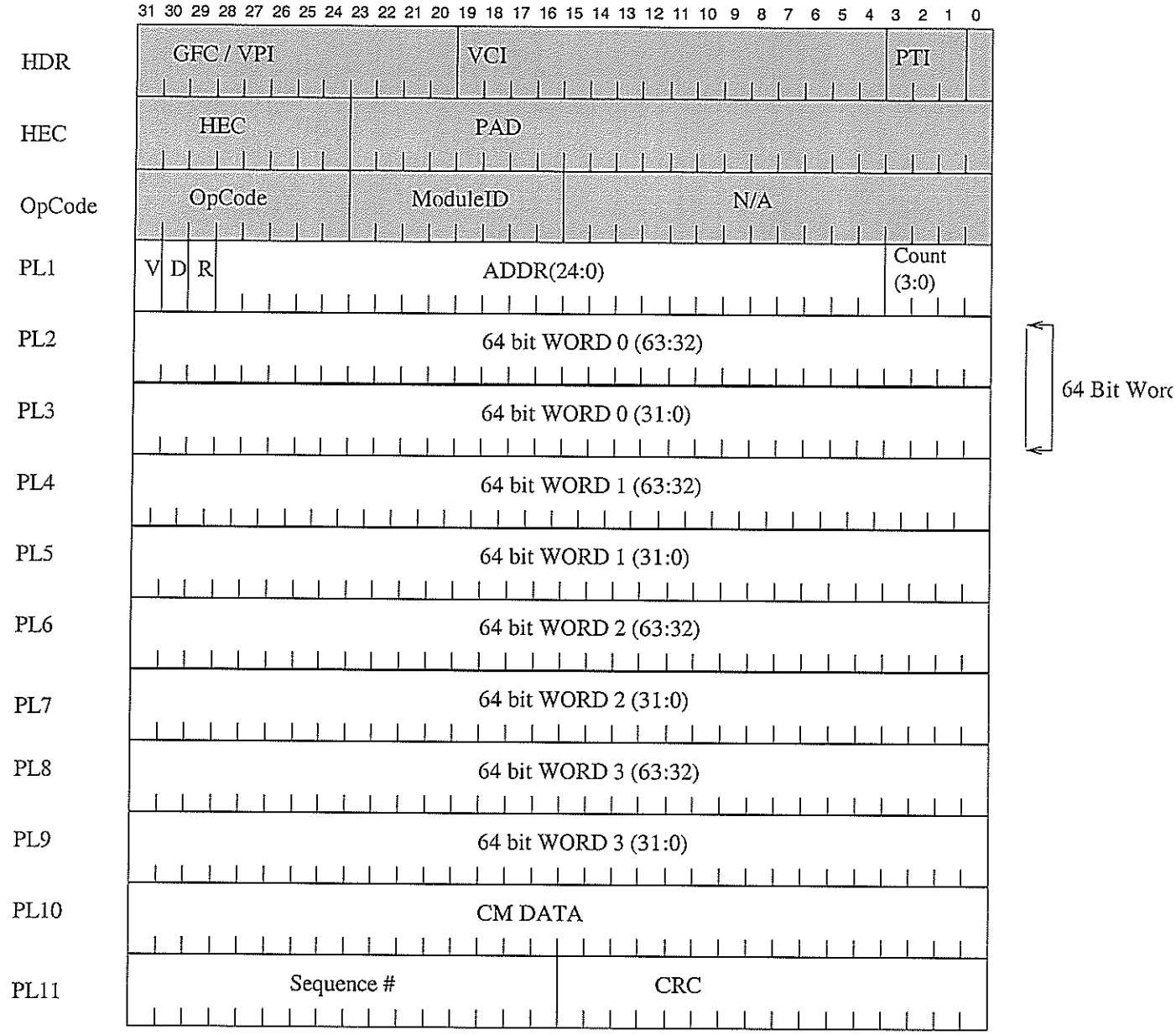


V – Valid Command: 1 = Valid command, 0 = Invalid, EOC
D – Device: 1 = Device 1, 0 = Device 0
R – Read or Write: 1 = Read, 0 = Write
F – 32 or 36 bit: 1 = 36 bit, 0 = 32 bit

VPI = 0x000, VCI = 0x0023 (35) RAD Control Cell
OpCode = 0x14 SRAM Memory Operation
OpCode = 0x15 SRAM Memory Operation Response
ModuleID = 0x00 RAD Control Cell Processor

Figure 10: Control cell format for SRAM transactions.

Control Cell Format for 64 bit RAD SDRAM Memory Operations



V – Valid Command: 1 = Valid command, 0 = Invalid, EOC
D – Device: 1 = Device 1, 0 = Device 0
R – Read or Write: 1 = Read, 0 = Write

VPI = 0x000, VCI = 0x0023 (35) RAD Control Cell
OpCode = 0x16 SDRAM Memory Operation
OpCode = 0x17 SDRAM Memory Operation Response
ModuleID = 0x00 RAD Control Cell Processor

Figure 11: Control cell format for SDRAM transactions.