Report Number: WUCSE-2002-24

2002-08-14

Implementation of a Pipelined Control Cell Processor

Michael Attig and John W. Lockwood

A fast control cell processor (CCP) has been designed and implemented in order to process control cells as they enter the module. This fast CCP is capable of receiving back-to-back control cells, processing them, and sending them out in back-to-back fashion. The fast CCP comes equipped with a SRAM interface and a statistics interface. Currently, the fast CCP uses the Statistics Counter Plus to count the number of control cells on each VCI, the number of SRAM reads on each VCI, the number of SRAM writes on each VCI, and the total number of control cells that pass through... Read complete abstract on page 2.

Implementation of a Pipelined Control Cell Processor

Michael Attig and John W. Lockwood

Complete Abstract:

A fast control cell processor (CCP) has been designed and implemented in order to process control cells as they enter the module. This fast CCP is capable of receiving back-to-back control cells, processing them, and sending them out in back-to-back fashion. The fast CCP comes equipped with a SRAM interface and a statistics interface. Currently, the fast CCP uses the Statistics Counter Plus to count the number of control cells on each VCI, the number of SRAM reads on each VCI, the number of SRAM writes on each VCI, and the total number of control cells that pass through the module.

2002-24

Implementation of a Pipelined Control Cell Processor

Authors: Attig, Michael; Lockwood, J. W.

August 14, 2002

Abstract: A fast control cell processor (CCP) has been designed and implemented in order to process control cells as they enter the module. This fast CCP is capable of receiving back-to-back control cells, processing them, and sending them out in back-to-back fashion. The fast CCP comes equipped with a SRAM interface and a statistics interface. Currently, the fast CCP uses the Statistics Counter Plus to count the number of control cells on each VCI, the number of SRAM reads on each VCI, the number of SRAM writes on each VCI, and the total number of control cells that pass through the module.

Implementation of a Pipelined Control Cell Processor

Michael Attig
John W. Lockwood

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

WUCS-2002-24
http://www.arl.wustl.edu/arl/projects/fpx

August 14, 2002

Abstract

A fast control cell processor (CCP) has been designed and implemented in order to process control cells as they enter the module. This fast CCP is capable of receiving back-to-back control cells, processing them, and sending them out in back-to-back fashion. The fast CCP comes equipped with a SRAM interface and a statistics interface. Currently, the fast CCP uses the Statistics Counter Plus to count the number of control cells on each VCI, the number of SRAM reads on each VCI, the number of SRAM writes on each VCI, and the total number of control cells that pass through the module.

1 Introduction

The Field programmable Port Extender (FPX) [1][2] is a reconfigurable network platform that uses control cells to read and write to memory. These control cells are generally generated in software, but can also be generated in hardware [3]. Many applications in networking hardware send and receive control cells in back-to-back fashion. In such cases, a control cell processor (CCP) capable of sending and receiving back-to-back control cells is a valuable commodity. The fast CCP has been designed in order to process incoming back-to-back control cells and to send them out back-to-back. When control cells exit back-to-back, the separation from start-of-cell to start-of-cell (SOC) is fourteen clock cycles.

The first implementation of the CCP required nearly four times as many clock cycles to process control cells. For example, when back-to-back control cells of four SRAM memory writes were processed, the delay from SOC to SOC was 52 clock cycles. When four SRAM memory reads were performed, the delay rose to 80 clock cycles. The fast CCP has been designed to perform two main functions: allow control cells to read/write to SRAM and maintain statistics of incoming control cells on each virtual circuit identifier (VCI), the number of reads on each VCI, the number of writes on each VCI, and the total number of control cells. The fast CCP is loaded into the reprogrammable application device (RAD) field programmable gate array (FPGA). Refer to Figure 1 for a view of how the fast CCP is arranged in the RAD FPGA.

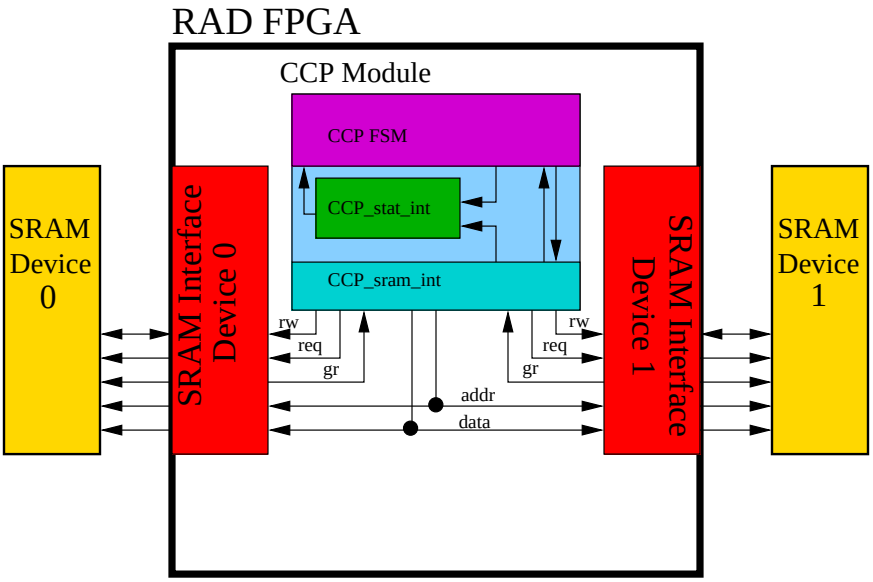

Figure 1: Two modules can be placed within the RAD FPGA. This diagram shows only the fast CCP in the RAD FPGA.

2 Functions

The fast CCP responds to control cells arriving on 17 different VCIs: x0023 and x0040 through x004F. Control cells arriving on other VCIs will be passed through. Currently, the fast CCP uses three different operation codes (opcode): x12, x14, and x18.

Opcode x12 allows the user to change one of the VCIs that the fast CCP responds to. This opcode changes a VCI storage register, which upon restart holds x0023. In order to change the VCI, place x12 in the opcode field and place the new VCI in bits 31 down to 16 of the first payload field.

Opcode x14 is used to perform SRAM reads and writes. The control cell format for SRAM memory operations [4] allows variable burst length reads or writes, and it allows the user to specify which SRAM device to use.

Opcode x18 is used to perform statistics reads. The fast CCP currently tracks four different types of events: the number of cells arriving on each VCI, the number of SRAM reads on each VCI, the number of SRAM writes on each VCI, and the total number of cells that have passed through the module. In order to read the number of events, a control cell format similar to that of a SRAM operation is used. Refer to Figure 2 for the cell format used for statistics operations.

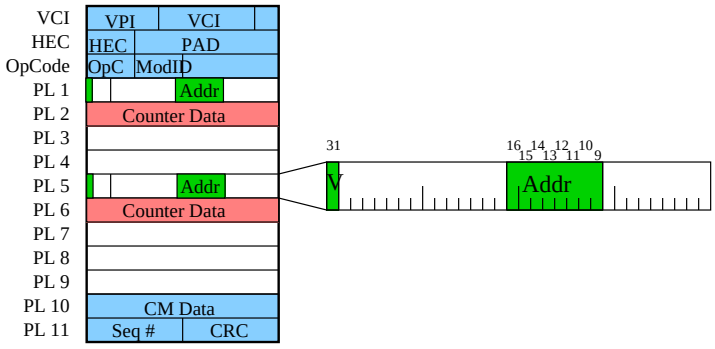

Figure 2: Addresses can be placed in bits 16 down to 9 of payloads 1 and 5. When performing a statistics read, set bit 31 of the corresponding address field. The value of the specified 32-bit counter will be returned in payloads 2 or 6.

2.1 Obtaining the Counter Address

The fast CCP determines the event number in a specific way. First, only the lower 6 bits of the VCI are used. To count an incoming cell on a specific VCI, 00b is appended to the front of these 6 bits. In the case of a SRAM read, 01b is appended, and in the case of a SRAM write, 10b is appended. Refer to Figure 3 for an example. If the user wishes to read the total number of cells that have passed through the module, use 11000000b as the event number.

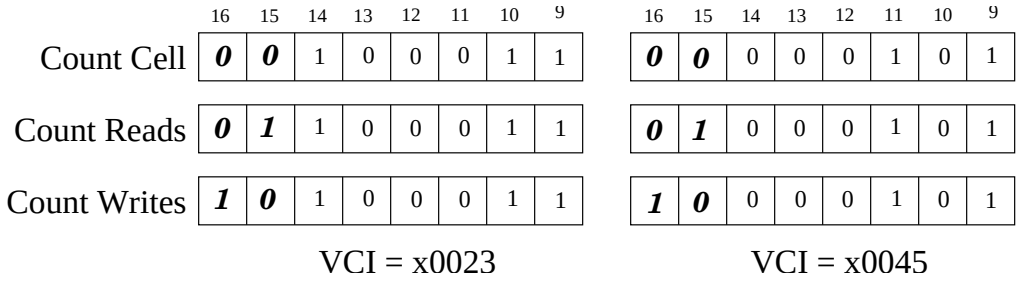

Figure 3: The above figure shows how to determine the counter address for two different VCIs. To read the number of cells arriving on VCI x0023, place 00b in bits 16 down to 15 and 100011b in bits 14 down to 9 of the address field in the control cell of Figure 2. Note that bits 14 down to 9 correspond to the lower 6 bits of the VCI. To read the number of SRAM reads on VCI x0023, place 01b in bits 16 down to 15 and 100011b in bits 14 down to 9. To read the number of SRAM writes on VCI x0023, place 10b in bits 16 down to 15 and 100011b in bits 14 down to 9. The same technique can be used for cells arriving on VCI x0045 except bits 14 down to 9 are 000101b.

3 Design

The main components of the fast CCP are the CCP finite state machine (FSM), CCP SRAM Interface, CCP Stat Interface, Input FIFO FSM, Output FIFO FSM, and two 256 by 32 FIFOs. These components and how they are connected to the fourteen pipeline registers is shown in Figure 4.

The fast CCP consists of fourteen pipeline registers, one for each of the words in a control cell. As the cell enters the pipeline, the CCP FSM begins to process words. The FSM checks the VCI, HEC, Module ID, and Opcode. If these are valid, the FSM will assert the appropriate opcode-dependent signals. If any of these are not valid, the cell is passed through unchanged.

The input and output FIFO FSMs are responsible for interacting with the appropriate FIFO. On the input side, the Input FIFO FSM listens for *soc_in_int*. When this pulses, the FSM asserts a write signal to the input FIFO. This component is also responsible for knowing how many cells are currently in the FIFO. If 18 cells are currently in the FIFO, it will assert *tca_out_int*, which communicates to exterior modules that it can not receive any more cells.

On the output side, the Output FIFO FSM awaits *start_write_fifo_out* from the CCP FSM. When this signal is asserted, it begins writing 32-bit words into the FIFO unless the signal *freeze_pipe* is asserted. The Output FIFO FSM also listens to *tca_in_int* in order to determine if it can send a cell out or not. This component also maintains a counter to determine how many cells are currently in the output FIFO. If 18 cells are awaiting to depart the module, the CCP cannot process cells until at least one cell begins to leave.

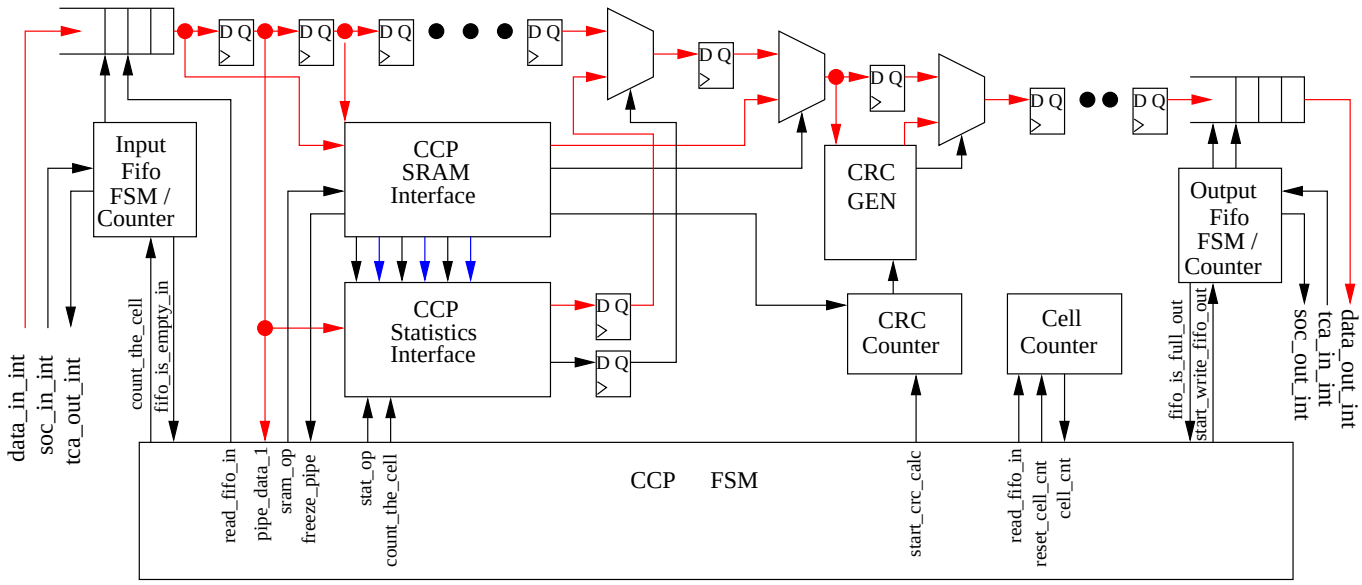

Figure 4: This block diagram shows the main components of the fast CCP. Not all signals are shown in this diagram.

4 Timing

The fast CCP implementation is capable of processing back-to-back cells. The separation from SOC to SOC is 14 clock cycles, as shown in Figure 5. There is an initial delay from incoming SOC to outgoing SOC of 24 clock cycles.

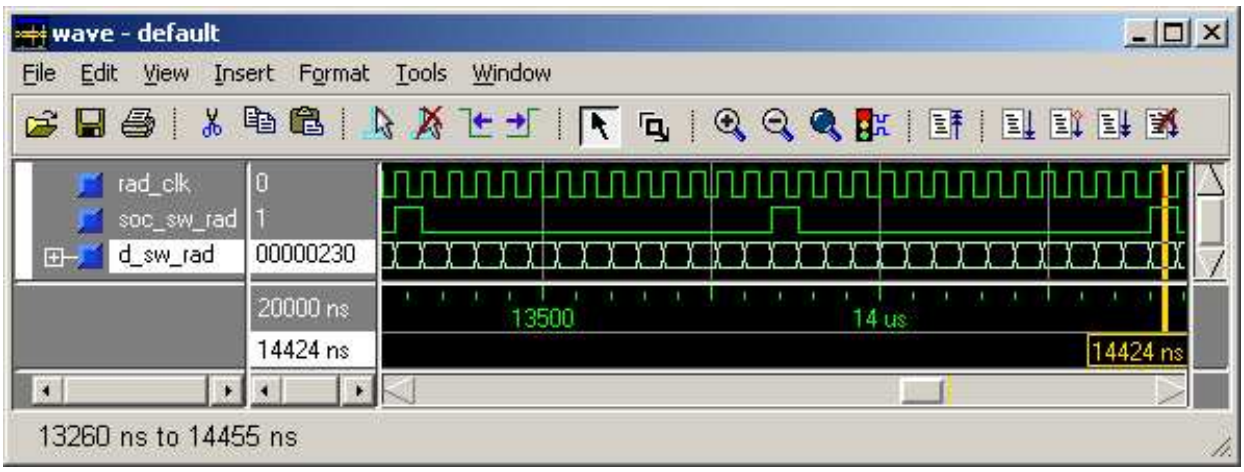

Figure 5: When back-to-back cells are sent to the fast CCP, the response cells will leave back-to-back.

The waveform in Figure 5 assumes that SRAM grant is given immediately. This is not always the case since a contending module may have been granted access to SRAM. The fast CCP cannot perform memory

operations until the other module has completed its transaction. In such cases, the pipeline is frozen in place to wait for grant. Once a grant is given, the pipeline is allowed to advance again. The signal *freeze_pipe* is used to prevent latching the data pipeline flip-flops. In the following example, a memory operation for SRAM device 0 is to occur. However, *dev_0_gr_int* has not been asserted when expected. This causes *freeze_pipe* to be asserted, which freezes the current FSM state and holds the current value of the pipeline registers. Refer to Figure 6 for a waveform showing the pipeline freeze.

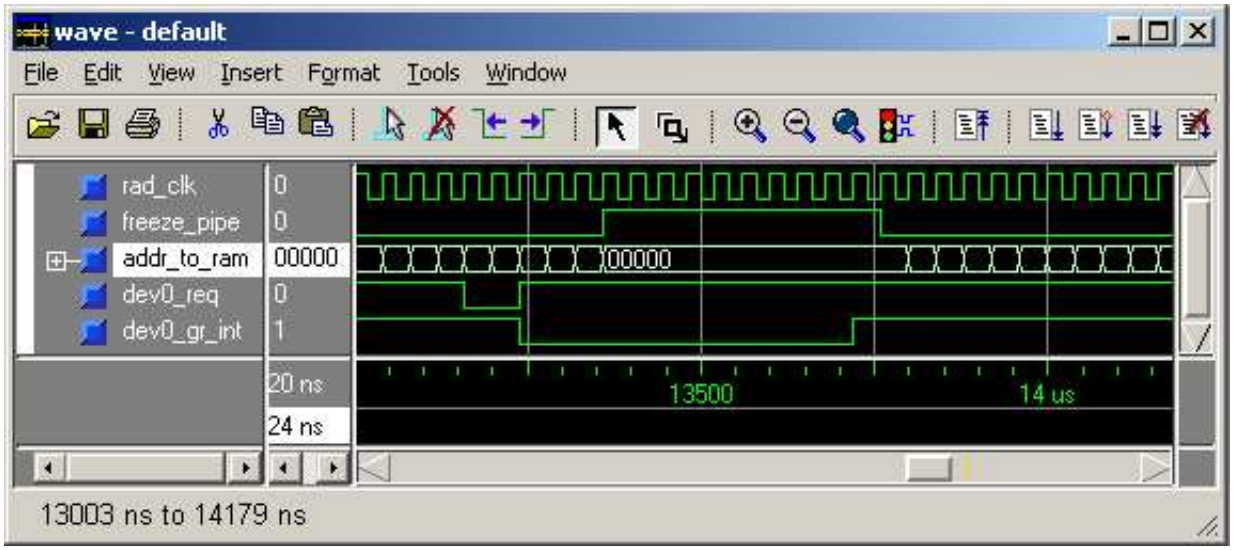

Figure 6: When SRAM grant is not given immediately, the pipeline will remain in its current state The above waveform shows the address presented to RAM being held.

5 Resources

The fast CCP is relatively small, utilizing only 1092 of 19,200 (5%) slices of the Xilinx Virtex XCV2000E FPGA. The design uses 757 of 38,400 (1%) 4-input LUTs and 6 of 160 (3%) on-chip block RAMs. This design has been placed and routed on the chip, and it is capable of running at a clock frequency of 100.7 MHz.

6 Code Location

The source code, edf files, test cells, and documentation for this module can be found in the CVS tree at: /project/arl/fpx/cvsroot/FPX_ROOT/RAD/TOP/FAST_RAD_CCP

7 Conclusion

The fast CCP is capable of processing back-to-back cells on the fly, with a separation as they leave of 14 clock cycles. This is the best case, however, in that if a grant is not received from SRAM immediately, the fast CCP pipeline will be frozen in place until grant is asserted. In addition, the fast CCP instantiates a Statistics Counter Plus [5] component to count the number of cells arriving on each VCI, the number of SRAM reads on each VCI, the number of SRAM writes on each VCI, and the total number of control cells passed through the module.

References

[1] J. W. Lockwood, J. S. Turner, and D. E. Taylor, "Field programmable port extender (FPX) for distributed routing and queuing," in *ACM International Symposium on Field Programmable Gate Arrays (FPGA'2000)*, (Monterey, CA, USA), pp. 137–144, Feb. 2000.

[2] J. W. Lockwood, N. Naufel, J. S. Turner, and D. E. Taylor, "Reprogrammable Network Packet Processing on the Field Programmable Port Extender (FPX)," in *ACM International Symposium on Field Programmable Gate Arrays (FPGA'2001)*, (Monterey, CA, USA), pp. 87–93, Feb. 2001.

[3] D. E. T. Todd Sproull, John W. Lockwood, "Control and configuration software for a reconfigurable networking hardware platform," in *IEEE Symposium on Field-Programmable Custom Computing Machines, (FCCM)*, (Napa, CA), Apr. 2002.

[4] D. E. Taylor, J. W. Lockwood, and N. Naufel, "Rad module infrastructure of the field-programmable port extender (fpx)," Tech. Rep. WUCS-TM-01-16, Applied Research Laboratory, Department of Computer Science, Washington University in Saint Louis, July 2001.

[5] M. E. Attig and J. W. Lockwood, "Usage of the Statistics Counter Plus Component in Networking Hardware Modules," tech. rep., WUCS-02-25, Washington University, Department of Computer Science, Aug. 2002.