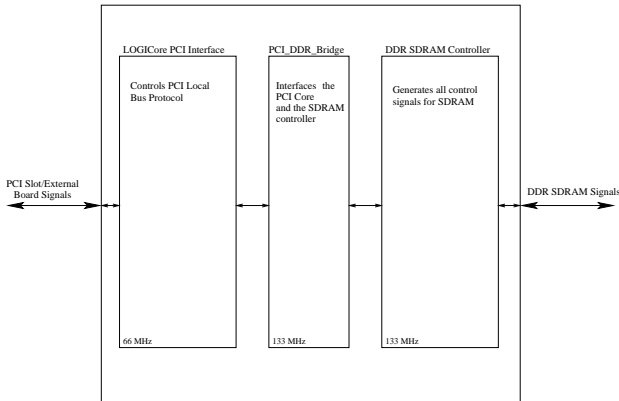


A 64-Bit 66-MHz PCI Local Bus to 133MHz DDR SDRAM Controller

Max Baker, Christopher Case

Fig. 1. Architecture Overview
Xilinx Virtex II FPGA



Abstract—A synthesizable controller that interfaces a 64-bit 66MHz PCI Local Bus to 133MHz DDR SDRAM, implemented on a Xilinx Virtex-ii based board.

I. INTRODUCTION

IN the design we implement a computer expansion board that interfaces between a 66MHz 64-bit PCI Local Bus and 133MHz Dual Data Rate (DDR) SDRAM. We will discuss the implementation details of the design including how we synchronized between the two clock domains while adhering to the strict electrical and timing requirements of the memory and bus standards used.

II. ARCHITECTURE OF DESIGN

In order to reduce the complexity and design-time we use existing solutions. For the PCI bus we are using Xilinx's IP solution "Logicore PCI Interface 3.0" [1]. This PCI Core is a ready to synthesize controller for the bus that operates to exact specifications with zero wait state. The core supported both the 66MHz and 64-bit extensions of the PCI standard that we need in our design. For the Double Data Rate (DDR) controller we start with an example design provided by Xilinx [2].

Our top level design instantiates the PCI core and the DDR controller. The bulk of our design is done in a module we call the *pci_ddr_bridge*. The bridge performs two main tasks : driving the DDR controller's state machine and synchronizing between the two clock domains. In the bridge is the master state machine as well as the buffers used to shuffle data between the two clock domains. Figure 1 shows a basic block diagram of our architecture.

The DDR SDRAM controller is modified to include an autorefresh counter that asserts a signal when the DRAM requires an autorefresh. The signal is reset every time a successful autorefresh is performed by DDR Controller. This signal indicating a refresh is needed is then sent to the *bridge* to allow it take control of the refresh process. When an autorefresh happens the PCI core is put into a wait state by deasserting the S_READY signal in case a memory operation happens during the significant time needed for the precharge and autorefresh operations of the DRAM.

Our state machine has fourteen states as show in Figure 2. For our state bits we use a binary encoding scheme where the upper nibble identifies the type of operation and the lower nibble for the individual state. This facilitates debugging on the logical analyzer; 0x0- states are reset, 0x1- states are init, 0x2- states are read, 0x3- states are write, and 0x4- states are autorefresh.

For the design we convert the all the various shell scripts used in analysis, elaboration, simulation and synthesis stages into a single Makefile. This way when a change is made on a file, all its dependencies are triggered. The other main advantage is that for small changes to a file, only that file will be reanalyzed. To allow both of us to work on the project simultaneously we use the Concurrent Versioning System (CVS) to manage all project files. CVS saves a revision of every file so you can revert changes to an archived version. This feature allows us to back off changes after trying a design modification.

III. IMPLEMENTATION DETAILS

A. RTL Design

The Register Transfer Language (RTL) portion of the design is done in VHDL using the Cadence tools in a command line (CLI) environment. Our design operated in the FPGA clock domain at 133MHz – the same clock used for the DRAM memory. Various signals are synchronized into our datapath using the FPGA clock and some of them are pulsed to be used as control signals to our state machine.

Crucial control signals used to set the operation mode of the PCI core are fed from registers instead of directly from the rails. The synthesis mapper tries to optimize out parts of the PCI core if certain inputs are tied to direct logic values.

B. Simulation

Functional simulation is done in Cadence NC-Sim on a Linux platform using simulation netlists for the PCI core and VHDL simulation models for the DDR SDRAM. Since our

REFERENCES

- [1] Xilinx Corporation, LogiCore PCI Interface v3.0, http://www.xilinx.com/partinfo/pci/xcvpci64_32ds.pdf
- [2] Tran, Jennifer, Synthesizable DDR SDRAM Controller, Xilinx Corporation, <http://www.xilinx.com/bvdocs/appnotes/xapp200.pdf>

Max Baker is currently pursuing a M.S. degree in Electrical Engineering from Columbia University in New York City.

Max received a B.S. degree in Computer Engineering with highest honors from the University of California, Santa Cruz in 2002.

Christopher Case was born in Anaheim, California in 1979. He received the B.S. degree in electrical engineering and computer science from the University of California, Riverside in 2002.

He is currently pursuing the M.S. degree in electrical engineering from Columbia University, New York, New York with an interest in micro-electronic circuits.

TIMING DIAGRAMS

Fig. 3. Auto Refresh Cycle

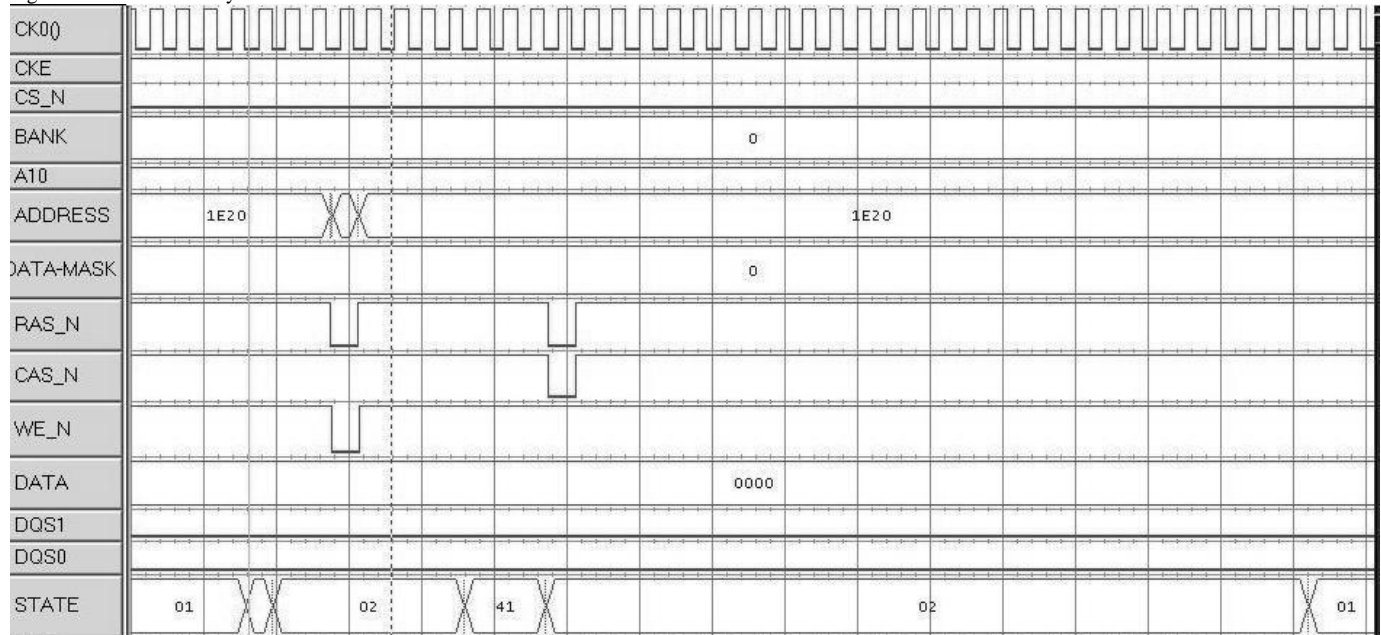


Fig. 4. DDR Memory Init

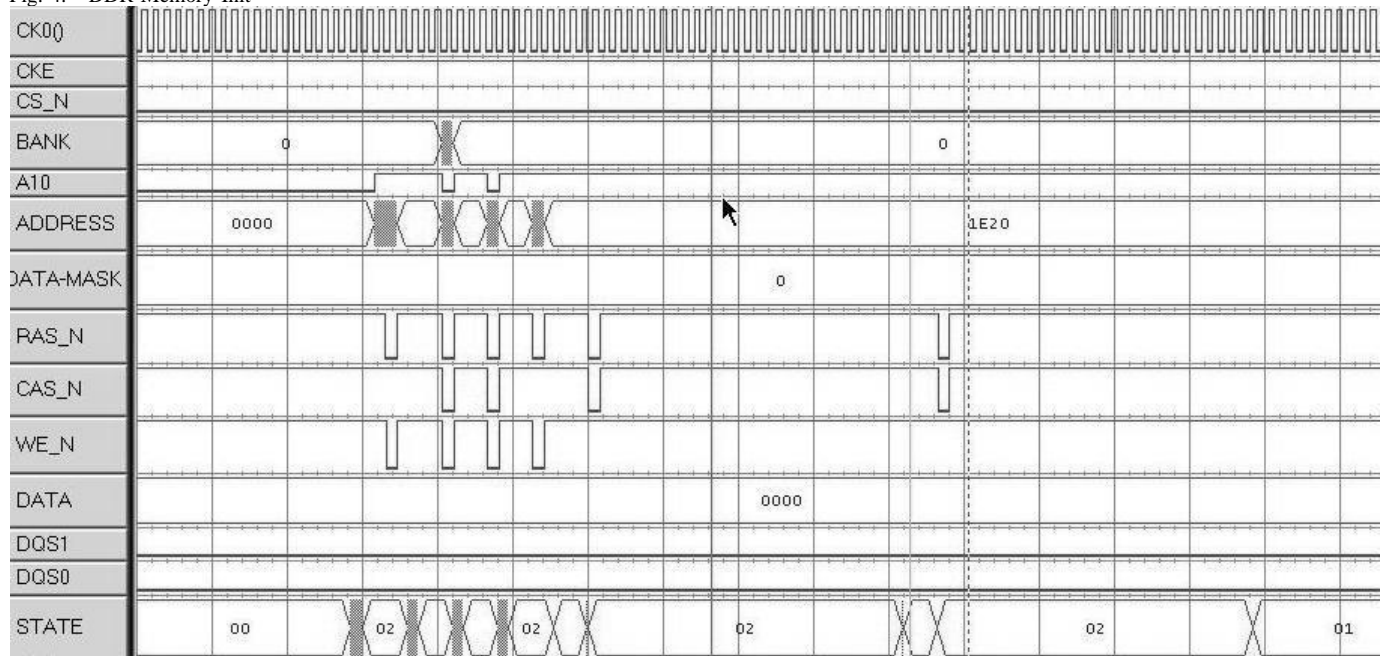


Fig. 5. 64-bit Read Cycle

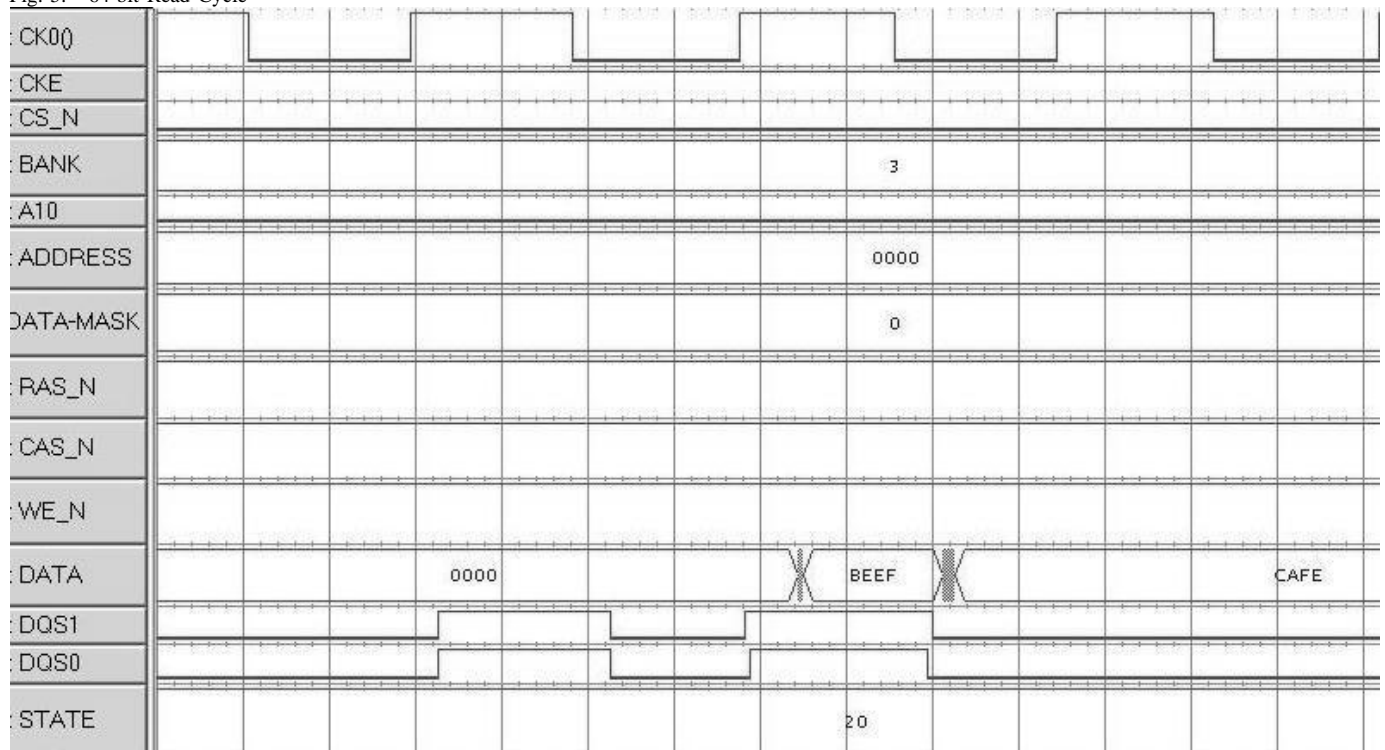


Fig. 6. 64-bit Write Cycle

