DESIGN AND IMPLEMENTATION OF A LIGHT-WEIGHT MICROCONTROLLER ON FPGA FOR HIGH-SPEED DATA ACQUISITION APPLICATIONS

A Thesis in Computer Science and Engineering

by

Wilfred A. Glover-Akpey

© 2008 Wilfred A. Glover-Akpey

Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science

August 2008
The thesis of Wilfred A. Glover-Akpey was reviewed and approved* by the following:

Kyusun Choi  
Assistant Professor Computer Science and Engineering  
Thesis Advisor

Lee D. Coraor  
Associate Professor Computer Science and Engineering  
Committee Member

Raj Acharya  
Professor of Computer Science and Engineering  
Head of the Department of Computer Science and Engineering

*Signatures are on file in the Graduate School
ABSTRACT

This thesis describes the development and implementation of a light-weight microcontroller on FPGA for high-speed data acquisition. It details the design from the conceptualization stage through to its implementation on the FPGA board. Growing application demands and complexity have created a need for additional microcontroller choices. Our goal is to design a high performance microcontroller without the unnecessary features and components which usually adds to performance overhead when implemented for a specific application.

The initial stage of the development process involved the design of an 8-bit RISC microcontroller with a dual port RAM and its description in Very High Speed Integrated Circuit Hardware Description Language (VHDL). The later stages involved the implementation of the design on an FPGA board. The microcontroller was designed, verified, and implemented on a Xilinx Field Programmable Gate Array (FPGA) on a Windows platform. Our experimental results obtained using our light-weight microcontroller, demonstrated its use for high speed data acquisition on an FPGA and how it compares to other existing commercial products on the market.
TABLE OF CONTENTS

LIST OF FIGURES ................................................................. vii
LIST OF TABLES ........................................................................ viii
ACKNOWLEDGEMENTS .............................................................. viii

Chapter 1 Introduction ............................................................. 1
  Background ........................................................................... 1
  Motivation and Goal .............................................................. 5
  Organization of Thesis ............................................................ 6

Chapter 2 Related Works ......................................................... 8
  Overview of FPGA Related Works .......................................... 8
  Overview Microcontroller Related Works ............................... 9

Chapter 3 Architecture and Functional Design ......................... 12
  Read Only Memory (ROM) ..................................................... 16
  Dual-Port RAM .................................................................. 16
  Program Counter (PC) .......................................................... 18
  Interrupt Control ................................................................. 20
  Instruction Decoder .............................................................. 20
  Control Unit ......................................................................... 21
  Instruction Cycle ................................................................. 22
  Arithmetic and Logic Unit (ALU) ........................................... 23
  Instruction Set ....................................................................... 24
  Digital Clock Manager (DCM) .............................................. 27
  Input/Output Port ............................................................... 27
  Analog to Digital Converter (ADC) ........................................ 28

Chapter 4 Implementation ....................................................... 30
  Conversion from Functional Design to VHDL ....................... 30
  Testing on Xilinx Spartan 3 FPGA ........................................ 31
  Setup .................................................................................. 36

Chapter 5 Conclusion .............................................................. 40
  Summary ............................................................................... 40
  Future Works ....................................................................... 42
    A Transmitter-Receiver Implementation .............................. 42
    Additional Features ............................................................ 42

References ............................................................................. 44
Appendix A  VHDL Code of the Sim2 Microcontroller ....................................................48
Appendix B  Simulation Results ..................................................................................206
Appendix C  Schematic of implementation board ................................................……208
LIST OF FIGURES

Figure 1-1: Reconfigurable computing systems – Flexibility vs. Performance[15] ................3

Figure 3-1: A top level view of the Sim2 Microcontroller. ......................................................... 14

Figure 3-2: Diagram showing the connection of I/O serial component to Figure 3-1 ..........15

Figure 3-3: Top level view of Sim2 on the FPGA board with the connection to Computer ... 15

Figure 3-4: A 16-Deep by 1-Wide Dual Port Synchronous RAM .............................................. 17

Figure 3-5: A 16 X 8 Dual Port RAM Edge-Triggered Schematic ........................................... 18

Figure 3-6: Program Counter. ....................................................................................................... 19

Figure 3-7: Interrupt Control Unit. .................................................................................................. 20

Figure 3-8: Instruction Cycle. .......................................................................................................... 23

Figure 3-9: Arithmetic and Logic Unit (ALU). ................................................................................. 24

Figure 3-10: I/O Serial Port. ........................................................................................................... 28

Figure 4-1: The Xilinx Spartan 3 starter kit was used in the initial test of Sim2 ...................... 32

Figure 4-2: First sample output from the oscilloscope of the PWM program. ...................... 32

Figure 4-3: Second sample output from the oscilloscope of the PWM program .................. 33

Figure 4-4: Third sample output from the oscilloscope of the PWM program ..................... 33

Figure 4-5: FPGA Design Flow [20] ............................................................................................ 35

Figure 4-6: FPGA Design Implementation. [20] ........................................................................ 36

Figure 4-7: Setup of our Implementation on the FPGA board ................................................... 37

Figure 4-8: Microcontroller layout in Xilinx showing PIN connections and Sim2’s floor placement. ........................................................................................................................................ 39
LIST OF TABLES

Table 3-1: Instruction Operation Code .................................................................21
Table 3-2: Instruction Set....................................................................................24
Table 4-1: I/O Pin Assignment for Figure 4-1..........................................................34
Table 4-2: Synthesis Results Table for Sim2 in FPGA. .................................................38
Table 4-3: I/O Pin Assignments for Figure 4-7. .........................................................39
Table 5-1: Comparing Sim2 with Other existing commercial brands. .........................41
I would like to thank the following people who have been instrumental to the completion of my thesis:

- I thank the Lord God Almighty for bringing me this far in my education.
- My advisor, Professor Kyusun Choi, for his guidance which has led to the research presented in this thesis.
- Professor Lee D. Coraor for his feedback and suggestions.
- My colleagues, family and friends for their questions, comments, feedback and continued support.
Chapter 1

Introduction

Background

Microcontrollers are hidden inside a large number of devices on the market these days. They are very important components of products such as automobiles and consumer electronics (microwaves, remote controls, etc). Basically, any product or device that interacts with the user has a microcontroller buried inside it. A microcontroller is a generic computing device that can do almost anything with the right programming.

By reducing the size, cost, and power consumption compared to a design using a separate microprocessor, memory, and input/output devices, microcontrollers make it economical to electronically control many more processes. Microcontrollers differ from microprocessors in that pretty much everything they need to operate is right there on one chip. On a typical chip, the microcontroller contains its microprocessor brain, a place to store its program (ROM, EPROM or flash), some storage for temporary data (RAM) and I/O pins. Some microcontrollers also contain timers and counters, non-volatile data storage areas, analog to digital converters, hardware serial ports and many other components on the same chip. A microcontroller is basically a computer-on-a-chip.

Reconfigurable computer system is a broad term that many systems can be classified under. In the broadest sense any computer whose operation can be changed is a reconfigurable computer system. We will constrain this term to reference systems with the built in ability to reconfigure its hardware.
The key feature of reconfigurable computing is its ability to perform computations in the hardware to increase performance, while retaining much of the flexibility of a software solution. Its ability to increase the performance of a wide range of applications has made it the subject of a great deal of research.

The use of an Application Specific Integrated Circuit (ASIC) to perform operations in hardware and the use of software-programmed microprocessors to execute instruction sets are the two conventional methods of computing algorithms. Reconfigurable computing is intended to bridge the gap between these two conventional methods (Figure 1-1.), maintaining a higher level of flexibility than hardware, at the same time achieving potentially much higher performance than software.

Historically, reconfigurable computing dates back as far as 1960 [17]. Since then there has been continuous work in the area [17]. In addition reconfigurable functional units and pipelines have been utilized in some of the earliest systems (e.g. CDC 7600).

The diagram below shows that reconfigurable computing systems offer an increase in performance over microprocessors, with greater flexibility than ASICs [15].
The use of field programmable gate arrays (FPGAs) in high-performance embedded applications has gained in popularity, in recent years. The latest FPGAs have come to provide "platform" solutions that are easily customizable for digital signal processing (DSP), system connectivity, and/or data processing applications, since FPGA devices have progressed both in terms of resources and performance. They are now able to handle a wide range of jobs, from relatively simple control operations to more complex ones. Another benefit of using FPGAs is the fact that different architectural variations of designs can easily be tested and evaluated on real applications. FPGAs are especially popular for prototyping integrated circuit designs. Once the design is set, hardwired chips are produced for faster performance.

A field-programmable gate array (FPGA) is a semiconductor device containing programmable logic components and programmable interconnects. The programmable logic components can be programmed to duplicate the functionality of basic logic gates (such as AND, OR, XOR, INVERT) or more complex combinatorial functions such as decoders, multiplexers or simple math functions. In most FPGAs, these programmable logic components (or logic blocks, in FPGA parlance) also include memory elements, which may be simple flip-flops or more complete blocks of memories.
A hierarchy of programmable interconnects allows the logic blocks of an FPGA to be interconnected as needed by the system designer, somewhat like a one-chip programmable breadboard. These logic blocks and interconnects can be programmed after the manufacturing process by the customer/designer (hence the term "field-programmable") so that the FPGA can perform whatever logical function is needed.

FPGAs are generally slower than their application-specific integrated circuit (ASIC) counterparts, can't handle as complex of a design, have a lower logic density, are inefficient for word operations and draw more power. However, they have several advantages such as a shorter time to market, ability to re-program in the field to fix bugs, and lower non-recurring engineering costs. Vendors may offer less flexible versions of their FPGAs that are cheaper. The development of these designs is made on regular FPGAs and then migrated into a fixed version that more resembles an ASIC due to lack of ability to modify the design once it is committed.

Data acquisition (DAQ) usually involves acquisition of signals and waveforms and processing the signals to obtain some needed information. The components of data acquisition systems include appropriate transducers that convert any measurement parameter to an electrical signal, which is acquired by data acquisition hardware. The DAQ hardware usually serves as the interface between the signal and a PC. They can be connected to the computer's ports such as the parallel, serial, USB port among others. Serial ports was used as they are simple, cheap, allow interoperability between devices and because serial ports have been around for decades. Some common applications of serial ports are dial-up modems, printers, serial mice, bar code scanners and other point of sale devices, computer terminal, satellite phones. A serial port "serializes" data, that is, it takes a byte of data and transmits the 8 bits in the byte one at a time. An advantage is
that a serial port needs only one wire to transmit the 8 bits. A serial port sends a start bit, which is a single bit with a value of 0. After each byte of data, it sends a stop bit to signal that the byte is complete. It may also send a parity bit. Although most standard serial ports have a maximum transfer rate of 115 Kbps (kilobits per second), higher speed serial ports such as Enhanced Serial Port (ESP) and Super Enhanced Serial Port (Super ESP) can reach data transfer rates of 460 Kbps. Reconfigurable computing may deliver high speed for digital signals. Although the RS-232 standard is formally limited to 20,000 bits per second, serial ports on popular personal computers allow for much higher baud rates; the capability to set a bit rate does not imply that a working connection will result. Serial ports, also called communication (COM) ports, are bi-directional and reduces the pin count of microcontrollers.

In the CHIP Research Group at the Pennsylvania State University, current Project such as the CMOS Ultrasonic Transceiver Chip could utilize this microcontroller. Past projects such as the software radio and a wristwatch Personal Digital Assistant (PDA) could also utilize the design described in this paper.

**Motivation and Goal**

Looking at the current trend in technology, there is the demand for products that are smaller, lighter, faster, and customizable. Today's embedded designers look for high-performance microcontrollers with integrated functions and peripherals to help reduce their total cost. A lot of the existing commercial microcontrollers have met these needs. However, increasing application demands and complexity have created a need for additional choices. Having options is important
to a design’s success. And one of the most important choices the design engineer makes is the choice of an appropriate microcontroller. The Sim2 microcontroller offers another level of choice in performance, speed, size, memory among others.

Our Initial Goal was to design a unique Microcontroller to be used in High speed data acquisition applications. With the current microcontroller Trends, most existing commercial microcontroller are general purpose, that is, they are designed to accommodate a broad variety of applications, capable of a long list of features. However, most of these features are not going to be used - they are unnecessary features which add overhead to the overall performance of the application to be implemented.

So what we want is to design a Microcontroller to be used for our implementation. Most of the existing commercial microcontrollers do not give us enough options either because of the price, unnecessary features and components, performance (most of which do not give us a high enough performance). So to achieve our final goal, our solution was to implement our microcontroller design on FPGA as the target platform since this will give us good performance, cost and complexity; yet flexible enough to be adapted for different applications.

**Organization of Thesis**

The remainder of this paper is structured as follows:

- Chapter 2: Discusses some related works and why some of their ideas were used in our design.
- Chapter 3: Describes the microcontroller design components and the FPGA board used to
implement this design. Details are provided in each section.

- Chapter 4: Discusses the development process, methods employed from the VHDL implementation to the post-layout analysis of the design. Discusses the results of our design and implementation.

- Chapter 5: This follows with a conclusion and suggestions for future work.
Chapter 2

Related Works

Discussed below are a number of microcontroller and FPGA related projects.

Overview of FPGA Related Works

FPGA technology has become a suitable target for processor implementations due to the current developments in FPGA technology. There are a number of publications that discuss methods for designing microprocessors and other computer systems; some for optimizing their performance for increased speed and implementation on FPGA.

Field Programmable Gate Arrays (FPGAs) are becoming increasingly popular for implementation of logic circuits. In [11], Ralf Joost and Ralf Salomon shows how FPGAs can be used for implementing a sophisticated multiprocessor architecture. It further argues that due to the excellent tool support and the availability of reusable functional modules (also known as intellectual properties) this approach can be adopted by almost any industrial hardware designer. The benefits of utilizing an FPGA to implement a multiprocessor platform for an industrial application in the field of offset printing were also discussed. [11]

Arne Linde et al, share their experiences from building a highly parallel computer using FGPAs. A major benefit of using FPGAs was the fact that different architectural variations could easily be tested and evaluated on real applications. Their research gave high performance speeds up to 40–50 MHz [12].
Overview of Microcontroller Related Works

The paper written by S. Ozaki, Y. Nishimichi, T. Kakiage, H. Yamamoto, M. Sumita, G. Inoue, M. Urano, H. Yamashita, T. Maeda, and T. Hishiyama describes the development of a 100MHz Embedded RISC Microcontroller which adopts a zero-cycle branching scheme. That is, two instructions are executed in a single cycle if one of them is a branch instruction. To raise microcontroller performance they realized the importance of reducing the clock cycles per instruction and to increase the operating frequency. [2]

Y. Ying, et al. discusses the Design and VLSI implementation of an asynchronous low power microcontroller. It details the design and implementation of a low power MCU with an instruction set compatible with the PIC16C6X and functions similar to the PIC16. Their design resulted in an asynchronous MCU with a power dissipation which is less than 16% of the PIC16, along with the same MIPS value and a smaller area. [1]

Our microcontroller (Sim2) makes use of a number of architectures including pipelining, the Harvard and the RISC architectures. The Harvard architecture is a computer architecture with physically separate storage and signal pathways for instructions and data. The throughput and the efficiency of the Sim2 is increased as a result of this architecture. Compared to the Princeton (Von Neumann) architecture, the Harvard architecture provides an improvement in bandwidth since the program and data are not fetched from the same memory using the same bus.

A microcontroller utilizing the Harvard architecture is described by Ying, et al [1]. The Reduced Instruction Set Computing (RISC) architecture performs simple instructions relatively faster than a Complex Instruction Set Computer (CISC) architecture. This is due to the fact that the
instruction set of RISC are relatively small and simple to CISC. Pipelining allows the processor to carry out different stages of different instructions simultaneously, thereby completing a set of instructions in a shorter period of time than a similar processor without pipelining. [3] [5].

The watchdog timer is a computer hardware timing device that triggers a system reset or reboot if the main program, due to some fault condition, freezes or hangs. The intention is to bring the system back from the hung state into normal operation. This was inventoried by Janin, Pascal (Tullins, FR), Giovannini, Michael (Grenoble, FR) and Ianigro, Corinne (Grenoble, FR) for the microcontroller. [19]

There are a number of published articles that discusses ways of minimizing power consumption in a microcontroller. One of the methods employed by Piguet, et al for minimizing power consumption is by designing an architecture that uses only one clock cycle per instruction. An architecture with a single cycle instruction execution can achieve a given number of MIPS with a slower clock rate at a lower MIPS/W rating than a comparable architecture with multiple cycles per instruction execution. This means that, by reducing the number of executed instructions through the use of a register file and a powerful instruction set Piguet et al were able to speed up the system by a factor of 2 to 5 (i.e. compared to single accumulator based machines). This resulted in a microcontroller with a rating of more than 2000MIPS/W (delivering a 10 MIPS performance at 3V) [3] [8].

Ian Brynjolfson and Zeljko Zilic introduced an architectural block, that is, the dynamic clock divider that can be added either internally to most modern FPGAs clock managers or as user logic, to allow for dynamic clock management. This is a system level technique that takes
advantage of the clock managers present in most modern FPGAs. As a result, they show that current FPGA clock managers are inadequate for use in dynamically controlled systems. [11]

The microcontroller design described in this thesis draws on some of the ideas used in the above works. It utilizes a Harvard architecture, pipelining, and a RISC architecture for speed optimization. It also makes use of a Digital Clock Manager (DCM). The DCM provides advanced clocking capabilities to Xilinx Spartan-3 FPGA applications. DCMs integrate advanced clocking capabilities into the Spartan-3 global clock distribution network. The Xilinx Spartan-3 DCM was chosen for the microcontroller design because it solves a variety of common clocking issues such as clock skewness, especially in high-performance and high frequency applications similar to ours. [20]

The Xilinx Spartan-3 FPGA was chosen for a number of reasons. The Spartan-3 FPGA offers platform capabilities with a wide range of I/O options and also since it targets logic count designs and applications. It has a DCM for precision clocking and clock synthesis. It has one of the leading embedded and digital signal processing solutions on the market.
Chapter 3

Architecture and Functional Design

The Sim2 microcontroller uses RISC architecture with a nineteen instruction set. This is a RISC architecture because it has a uniform instruction format, very few (and simple) addressing modes and a small instruction set. It implements the Harvard architecture, that is, there are physically separate storage and signal pathways for instructions and data (requiring dedicated buses for each of them). Instructions and operands can therefore be fetched simultaneously. A different program and data bus width was possible, allowing program and data memory to be better optimized to the architectural requirements.

An instruction pipeline is a technique used to increase the throughput of instructions. It is implemented in such a way that while the current instruction is being executed the address of the next instruction is fetched after its address is calculated. Ideally, since Sim2 implements a two stage pipeline the throughput is supposed to increase by a factor of two but this is not the case sometimes. All instructions however, are executed in one instruction cycle, i.e. two clock cycles.

The Sim2 logic is implemented with all storage elements clocked synchronously on the positive edge of the clock. All data and address processing elements are fully asynchronous and have a full clock cycle to reach a stable state prior to being stored and sampled. There are 8-bit input ports that can be implemented as switches and an 8-bit output of which three can be connected to the FPGA’s board LEDs. In our current implementation we make use of them as a serial port. The serial port has a transmitter and a receiver for an RS232 communication. There is also an external interrupt that provides an interface between the operating environment and the program. The key elements of the Sim2 architecture are
- A 256 word by 24 bit Read Only Memory (ROM) which stores up 256 instructions, each consisting of an 8-bit opcode (i.e. operation code), a 16-bit address values (that is, a source and a destination address). The operation code defines the function of the address values.

- A 256 word by 8-bit dual-port Random Access Memory (RAM). It provides 256 words for general purpose storage. It acts as a temporary storage location for program instructions and their results.

- A program counter, this calculates the address of the next instruction to be executed.

- The Arithmetic and Logic Unit (ALU), this unit is responsible for the manipulations of the data.

- The instruction decoder and control unit integrates the interrupt requests with inline instruction processing and generates the control signals for the program counter, the dual-port RAM and the rest of the other components.

- A DCM, this generates the required clock frequency input that is used by the microcontroller. It provides advanced clocking capabilities and solves a variety of clocking problems.

- The instruction register stores the next instruction fetched from the Read Only Memory.

- There are three main addressing modes – memory (direct and base plus offset), implicit and immediate addressing.

The rest of this chapter discusses all other components used in the design of the Sim2 microcontroller. The Figures 3-1, 3-2, and 3-3 give a top level view of the Sim2 microcontroller and the connections to its I/O serial component, the FPGA board and computer.
Figure 3-1: A top level view of the Sim2 Microcontroller.
Figure 3-2: Diagram showing the connection of I/O serial component to Figure 3-1.

Figure 3-3: Top level view of Sim2 on the FPGA board with the connection to Computer.
Read Only Memory (ROM)

The ROM consists of a program matrix which is functionally implemented as 256-by-24 array. It receives an 8-bit input from the program counter which serves as the address of the next instruction to be fetched. The instruction that is fetched from the ROM has a size of 24-bits. The ROMs output consists of three bytes of data consisting of the opcode, source address or immediate value and the destination address. The outputs produced by the array are the 24-bit ROM output.

Dual-Port Random Access Memory (RAM)

The dual-port RAM block consist of 3 address decoders and 16, 16 by 8 Dual Port RAM words. Each 16 by 8 dual port RAM component is made up of 8, 16-Deep by 1-Wide Static Dual Port Synchronous RAM cell. This gives a total memory size of 256 words. The 16X1D RAM cell was the primitive chosen since it is supported on Spartan-3 and because it is an optimized Xilinx library cell. A 16X8D RAM cell was not available so we had to build it from the 16X1D RAM cell.
The RAM16X1D is a 16-word by 1-bit static dual port random access memory with synchronous write capability. It has two separate address ports: the read and write addresses. The dual port RAM has a special cell design that allows for simultaneous accesses from two ports. It has the ability to simultaneously read and write different memory cells at different addresses. This component has separate read and write data paths, meaning that if it operates at the same speed as the RAM in the single-ported, it effectively doubles total bandwidth of the RAM. This was one of the main reasons for choosing a dual port RAM for our design. A single-ported RAM has only one data path that has to be shared between read and write operations. So, if the RAM has a bandwidth of, say, 80 Mbyte/s, then the read bandwidth plus the write bandwidth must be less than 80 Mbytes/s. The read and write operations can share this bandwidth in any ratio, 50-50 or otherwise. In a write transaction, the data written into the specified address comes out of the write output’s port. While in the case of a read transaction, the data stored at the specified address comes out at the read output’s port. These address ports are completely asynchronous.

Figure 3-4: A 16-Deep by 1-Wide Dual Port Synchronous RAM.
Dual-ported RAM (DPRAM) is a type of Random Access Memory that allows multiple reads or writes to occur at the same time, or nearly the same time, unlike single-ported RAM which only allows one access at a time. It serves as a general purpose storage location during program execution.

![A 16 X 8 Dual Port RAM Edge - Triggered Schematic](image)

**Figure 3-5**: A 16 X 8 Dual Port RAM Edge - Triggered Schematic.

**Program Counter**

The program counter controls the program flow by computing the ROM address of each instruction to be executed. It enables Sim2 to compute the address of the next instruction in the
program to be executed. A 24-bit instruction value of opcode, source address/immediate value and destination address is obtained from the ROM output. During a normal inline code execution when no branch is taken the Current address is the address sent through the 8 bit adder resulting in the computation and selection of the next address. When an interrupt is to be serviced, the Interrupt control unit sends an “interrupt request” to the program counter which causes it to save the next instructions address in a register. The program jumps to the interrupt vector indicated in the counter. After the interrupt is serviced, control is given back to the program counter to resume the program execution using the address that was saved in the register.

Figure 3-6: Program Counter.
**Interrupt Control**

Sim2 has an interrupt input which is processed by the Interrupt control unit as shown in Figure 3-5. When an interrupt signal is detected, the Interrupt Control unit synchronizes the signal to Sim2’s internal clock and passes it on to the Instruction Decoder as an interrupt request. The Interrupt controller initiates the interrupt to start on the next clock positive edge. The resulting latency is only 1 clock cycle.

![Diagram of Interrupt Control Unit](image)

**Figure 3-7:** Interrupt Control Unit.

**Instruction Decoder**

The instruction decoder handles the generations of signals that control the flow of addresses and data through the microcontroller. It directs the program counter (PC) in determining the location of the next instruction to be executed in ROM. It generates and coordinates the transfer of control.
signals needed for the microcontroller to enter into an interrupt, execute the interrupt service routine and return successfully from it.

Control Unit

The control unit has a similar function like the instruction decoder. It also handles the generation of signals that control the flow of addresses and data through the dual port RAM and the arithmetic and Logic Unit (ALU). It also controls the input and output of data. It does this by generating the appropriate control signal using a look-up table.

The table below gives the binary operation code for each instruction.

Table 3-1: Instruction Operation Code.

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SUB</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>AND</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>OR</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NOT</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>XOR</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>XNOR</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>SHR</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SHL</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>SHRC</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Instruction Cycle

Instructions are initiated on the rising edge of each clock cycle; this loads the opcode from the ROM into the instruction register (IR). The instruction cycle is implemented as a two stage pipeline system. Each pipeline stage is executed in one clock cycle. While the current one is being executed, the next instruction is fetched. This helps to increase throughput of the entire system. A NOP follows a branch instruction; this ensures that another instruction is not executed during the second stage of the instruction cycle.
Figure 3-8: Instruction Cycle.

Sim2 uses a 2-stage pipeline to minimize the amount of hardware and power consumption. The first stage is either fetch-decode and second stage is execute or the first stage is fetch and the second stage is decode-execute. This depends on the instruction been executed.

**Arithmetic and Logic Unit (ALU)**

The ALU handles all arithmetic and logic functions of Sim2. Operations such as addition, subtraction, logical shift left or right, AND, OR and NOT are examples of some of the operations that are carried out by this unit. The results of the operations are stored back in the dual port RAM to await further instructions.
Instruction Set

Nineteen basic instructions are implemented in Sim2. These instructions are defined by an 8 bit opcode. The microcontroller manipulates data using instructions contained in a look-up table to determine the appropriate signal to generate for executing that particular instruction. All address in the instructions is supplied by the ROM. The ALU make limited use of registers, it obtains the addresses and operand values directly from the dual-port RAM and stores the results back in the dual-port RAM. The MV and NOP instructions use the same opcode. The nineteen instructions and their functionality are shown in the table below.

Table 3-2: Instruction Set.

<table>
<thead>
<tr>
<th>Operation Code (opcode)</th>
<th>Function</th>
</tr>
</thead>
</table>

Figure 3-9: Arithmetic and Logic Unit (ALU).
<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>In the ALU, it adds the source address content to the destination address content and stores the results at the destination address.</td>
</tr>
<tr>
<td>SUB</td>
<td>It subtracts the data at the source address from the data at the destination address. The result is stored back at the destination address.</td>
</tr>
<tr>
<td>AND</td>
<td>Performs a bitwise AND of the data stored in the source and destination address. The result of the operation is stored in the destination address.</td>
</tr>
<tr>
<td>OR</td>
<td>Performs a bitwise OR of the data stored at the source and destination addresses. The resulting value is stored at the destination address.</td>
</tr>
<tr>
<td>NOT</td>
<td>Inverts all bits at the specified address.</td>
</tr>
<tr>
<td>XOR</td>
<td>Performs a bitwise XOR of the value stored at the source address with the value stored at the destination address.</td>
</tr>
<tr>
<td>XNOR</td>
<td>Performs a bitwise XNOR of the value stored at the source address with the value stored at the destination address.</td>
</tr>
<tr>
<td>SHR</td>
<td>Logical shift right. Performs a bitwise shift to the right of all bits at the specified address.</td>
</tr>
<tr>
<td>SHL</td>
<td>Logical shift left – Performs a bitwise shift to</td>
</tr>
<tr>
<td>Instruction</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>SHRC</td>
<td>Logical shift right with carry – performs a bitwise shift of the value 1 bit to the right while using the carry in bit.</td>
</tr>
<tr>
<td>SHLC</td>
<td>Logical shift left with carry – performs a bitwise shift of the data 1 bit to the left while using the carry in bit.</td>
</tr>
<tr>
<td>BC</td>
<td>Branch on carry – Performs a branch operation when the carry bit is set.</td>
</tr>
<tr>
<td>BA</td>
<td>Branch always.</td>
</tr>
<tr>
<td>RTI</td>
<td>Allows you to return from the interrupt service routine.</td>
</tr>
<tr>
<td>NOP</td>
<td>Performs a machine cycle with no operation done.</td>
</tr>
<tr>
<td>MV</td>
<td>Moves the data at the source address to the destination address.</td>
</tr>
<tr>
<td>MV#</td>
<td>Move immediate – this stores the value in the instruction at the destination address.</td>
</tr>
<tr>
<td>IN</td>
<td>Stores data from the input port B to the specified address.</td>
</tr>
<tr>
<td>OUT</td>
<td>Outputs the data at the source address to the 8-bit output port B.</td>
</tr>
</tbody>
</table>
The Digital Clock Manager (DCM)

Clock management units/circuits are relatively new FPGA architectural blocks, critical to solving clock distribution problems associated with high-speed, high-density designs. The lock signal from the DCM was used as the reset signal for the microcontroller. The lock signal is used because it indicates when the output clocks are valid for use within the microcontroller.

Input/Output Port (I/O Port)

The 8-bit input and 8-bit output of Figure 3-1 has been connected to a universal asynchronous receiver/transmitter component (commonly called a UART). This serial port consists of a transmitter and a receiver unit. The transmitter takes the 8 bit output from the microcontroller and transmits it serially (i.e. one bit at a time) using the RS-232 (Recommended Standard 232) standard to the PC. A serial port transfer rate of 115,200 bps was used for the Sim2 implementation. Contents of the transfer can be viewed using a terminal such as the HyperTerminal in Windows. The receiver also receives bits serially from the PC and sends them as an 8-bit input into the microcontroller using the same RS 232 standard. The transmitter consists of input/output buffers, binary counters, shift registers and d flip-flops. The receiver also consists of binary counters, shift registers, input/output buffers and some d flip-flops.
Analog to Digital Converter (ADC)

The analog to digital converter (ADC) is used to convert the incoming data into a form that the Sim2 microcontroller can recognize. Since our microcontroller was built to interpret and process digital data, i.e. 1's and 0's, they won't be able to do anything with the analog signals that may be being sent to it by a device. There is also a digital to analog converter that allows the microcontroller to send data to the receiver from the transmitter transducer.

This ADC was selected because it provides a wide selection of voltage references to match our requirements. The speed, resolution and single-supply operation of the ADC are suited to
applications in set-top-box (STB), video multimedia, imaging, high-speed acquisition (like ours), and communication. The ADC is a CMOS, low-power, 10 bit, 30 MSPS analog-to-digital converter (ADC) that can operate with a supply voltage from 3V to 5.5 V. The ADC has been designed to give circuit developers flexibility. The analog input to the ADC can be either single-ended or differential. We used 5V for powering up the FPGA board and this ADC. [20]
Chapter 4

Implementation

This chapter describes the developmental process of this project from the initial design stage to the implementation stage. We discuss the scientific methods employed in detail, together with our results and evaluations.

Conversion from functional design to VHDL

We have already described the functional components of the microcontroller design in the previous chapter. VHDL (VHSIC Hardware Description Language) was selected as the language to design the microcontroller (Sim2) due to the designer’s familiarity with it. The Xilinx ISE program was used in Windows to develop Sim2 using VHDL.

Sim2 was designed using both structural and behavioral VHDL. Structural VHDL allows the designer to describe the design at the gate level while behavioral VHDL allows the designer to describe the design at a high level. Behavioral VHDL was very helpful in the specification of several fundamental components while structural VHDL was used to connect the major components together. Experiments carried out on Sim2 to optimize it, showed that timing or placement constraints have very little effect on its overall performance.

Designing Sim2 using a modular approach helped in reusing components as and when needed. It also made it easier to design, debug, and modify the microcontroller. The major components of the design are the dual-port RAM, ROM, Control unit, interrupt controller, the program counter,
the control unit and Arithmetic and Logic Unit (ALU). Each of these components has additional sub components.

Testing on a Xilinx Spartan 3 FPGA Starter-Kit

The Sim2 design was downloaded to the Xilinx Spartan 3 XC2S200 FPGA for testing. Simulation of the Sim2 was done using the ModelSim software. A behavioral simulation and a post route simulation were carried out to test the logic and functionality of the microcontroller on FPGA in Xilinx through its integration with ModelSim. Behavioral simulation was used to test the logic of the design, while the post-place-&-route simulation was used to uncover problems that may occur after the logic gates have been mapped to configurable logic blocks (CLBs) within the FPGA and the design has been routed.

The post route simulation was used after the behavioral simulation had completed successfully. The Sim2 microcontroller was moved to the implementation board after successful completion of the above simulations.
Figure 4-1: The Xilinx Spartan 3 starter kit was used in the initial test of Sim2.

Figure 4-2: First sample output from the oscilloscope of the PWM program.
Figure 4-3: Second sample output from the oscilloscope of the PWM program.

Figure 4-4: Third sample output from the oscilloscope of the PWM program.

The above outputs in Figures 4-2, 4-3 and 4-4 were obtained as a results of changes in the switch
inputs.

Table 4-1: I/O Pin Assignments for Figure 4-1.

<table>
<thead>
<tr>
<th>I/O Pin Assignments</th>
</tr>
</thead>
<tbody>
<tr>
<td>NET &quot;Clock&quot; LOC = &quot;T9&quot;;</td>
</tr>
<tr>
<td>NET &quot;output&lt;3&gt;&quot; LOC = &quot;N14&quot;;</td>
</tr>
<tr>
<td>NET &quot;input&lt;0&gt;&quot; LOC = &quot;F12&quot;;</td>
</tr>
<tr>
<td>NET &quot;output&lt;4&gt;&quot; LOC = &quot;P13&quot;;</td>
</tr>
<tr>
<td>NET &quot;input&lt;1&gt;&quot; LOC = &quot;G12&quot;;</td>
</tr>
<tr>
<td>NET &quot;output&lt;5&gt;&quot; LOC = &quot;N12&quot;;</td>
</tr>
<tr>
<td>NET &quot;input&lt;2&gt;&quot; LOC = &quot;H14&quot;;</td>
</tr>
<tr>
<td>NET &quot;output&lt;6&gt;&quot; LOC = &quot;P12&quot;;</td>
</tr>
<tr>
<td>NET &quot;input&lt;3&gt;&quot; LOC = &quot;H13&quot;;</td>
</tr>
<tr>
<td>NET &quot;output&lt;7&gt;&quot; LOC = &quot;P11&quot;;</td>
</tr>
<tr>
<td>NET &quot;input&lt;4&gt;&quot; LOC = &quot;J14&quot;;</td>
</tr>
<tr>
<td>NET &quot;Intrpt_in&quot; LOC = &quot;M14&quot;</td>
</tr>
<tr>
<td>NET &quot;input&lt;5&gt;&quot; LOC = &quot;J13&quot;;</td>
</tr>
<tr>
<td>NET &quot;pc&lt;1&gt;&quot; LOC = &quot;N16&quot;;</td>
</tr>
<tr>
<td>NET &quot;input&lt;6&gt;&quot; LOC = &quot;K14&quot;;</td>
</tr>
<tr>
<td>NET &quot;pc&lt;2&gt;&quot; LOC = &quot;F13&quot;;</td>
</tr>
<tr>
<td>NET &quot;input&lt;7&gt;&quot; LOC = &quot;K13&quot;;</td>
</tr>
<tr>
<td>NET &quot;pc&lt;3&gt;&quot; LOC = &quot;R16&quot;;</td>
</tr>
<tr>
<td>NET &quot;Cout&quot; LOC = &quot;P16&quot;;</td>
</tr>
<tr>
<td>NET &quot;pc&lt;4&gt;&quot; LOC = &quot;P15&quot;;</td>
</tr>
<tr>
<td>NET &quot;output&lt;0&gt;&quot; LOC = &quot;C15&quot;;</td>
</tr>
<tr>
<td>NET &quot;pc&lt;5&gt;&quot; LOC = &quot;N15&quot;;</td>
</tr>
<tr>
<td>NET &quot;output&lt;1&gt;&quot; LOC = &quot;P14&quot;;</td>
</tr>
<tr>
<td>NET &quot;pc&lt;6&gt;&quot; LOC = &quot;G13&quot;;</td>
</tr>
<tr>
<td>NET &quot;output&lt;2&gt;&quot; LOC = &quot;L12&quot;;</td>
</tr>
<tr>
<td>NET &quot;pc&lt;7&gt;&quot; LOC = &quot;E14&quot;;</td>
</tr>
</tbody>
</table>

Figure 4-5, illustrates the steps we went through during our designing process. After creating all the needed files in the Design Entry step we verified the functionality of Sim2 through RTL simulation also known as behavioral simulation and gate-level simulation. Design implementation was the next step after Sim2 was synthesized. The implementation involved Translate, Map and Place and Route. The report generated by synthesizing our design can be seen in Table 4-2. Timing verification of different points of the design could have also be carried out if needed. The Xilinx Spartan 3 device was programmed by creating the programming file (BIT) to program the FPGA. We generated a PROM file to download to our device. iMPACT was then used to program the device with the programming cable as shown in Figure 4-5. Figures 4-5 and 4-6 gives detail steps through the design flow process and implementation on FPGA.
Figure 4-5: FPGA Design Flow [20]
Setup

Once used only for glue logic, FPGAs have progressed to a point where system-on-chip (SoC) designs can be built on a single device. The number of gates and features has increased dramatically to compete with capabilities that have traditionally been offered through Application Specific Integrated Circuits ASIC devices only.
We implemented Sim2 microcontroller on an FPGA because it gives us the flexibility of adapting our design to meet certain specific requirements as and when we need them. Since FPGAs are not specific to a particular application, it is cost effective to migrate quickly to the best available fabrication technologies.

To implement Sim2, the board shown in Figure 4-7 was used. The board had a transmitter and receiver transducers installed on it. It also had an Analog to digital (ADC) converter unit, a parallel port which can be used as switches and an RS232 serial port. Originally, we had wanted to make use of both the transmitter and receiver transducers to implement the Sim2 on our board. However, due to time constraints we will make use of only the receiver transducer component to demonstrate our work.

Figure 4-7: Setup of our Implementation on the FPGA board.
We initially used the Spartan 3 FPGA board starter kit by Xilinx to test Sim2 after which it was transferred to the board above for implementation. Specifications for the Transmitter and Receiver transducers can be seen in [21]. Only 21% of the slices in the XCS200 Spartan device was used. It had a speed grade of -4 and could run up to 75MIPS.

Table 4-2: The table below gives the synthesis results for the Sim2 in FPGA.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Used</th>
<th>Available</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of slices used</td>
<td>411</td>
<td>1920</td>
<td>21%</td>
</tr>
<tr>
<td>Number of Slice Flip Flops</td>
<td>173</td>
<td>3840</td>
<td>4%</td>
</tr>
<tr>
<td>Number of 4 input LUTs</td>
<td>164</td>
<td>3840</td>
<td>15%</td>
</tr>
<tr>
<td>Number of bonded OIBs</td>
<td>34</td>
<td>63</td>
<td>53%</td>
</tr>
<tr>
<td>Number of GCLKs</td>
<td>8</td>
<td>8</td>
<td>100%</td>
</tr>
<tr>
<td>Number of DCMs</td>
<td>1</td>
<td>4</td>
<td>25%</td>
</tr>
</tbody>
</table>

Other results are given below

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum clock frequency</td>
<td>77.750MHz</td>
</tr>
<tr>
<td>Minimum clock period</td>
<td>12.862ns</td>
</tr>
</tbody>
</table>

For the experimental setup used in this study, samples of incoming signals from the receiver transducer are used to verify the functionality of Sim2 (using most of the instructions in the instruction set).

All instructions in the instruction set were used in our sample programs to verify the functionality of the Sim2 microcontroller. Samples of the analog data coming through the receiver is converted into ‘0’ and ‘1’ and read into the microcontroller through the ADC. The data is manipulated and sent out through the transmitter of the serial port of the microcontroller using the RS232 standard. A C++ tool designed to connect to COM port 2(COM2) was made to run on our PC to read the
signals on COM2. This C++ program is a tool that enables us to read data coming through the COM ports on any Windows PC. The digital signals obtained from COM2 are written to a file created by the C++ tool. Appendix C shows the schematic diagram of the board used for the implementation.

Table 4-3: I/O Pin Assignments for Figure 4-7.

<table>
<thead>
<tr>
<th>I/O Pin Assignments</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>NET &quot;Clock&quot;</td>
<td>LOC = &quot;P36&quot;;</td>
</tr>
<tr>
<td>NET &quot;input&lt;0&gt;&quot;</td>
<td>LOC = &quot;P4&quot;;</td>
</tr>
<tr>
<td>NET &quot;input&lt;1&gt;&quot;</td>
<td>LOC = &quot;P5&quot;;</td>
</tr>
<tr>
<td>NET &quot;input&lt;2&gt;&quot;</td>
<td>LOC = &quot;P8&quot;;</td>
</tr>
<tr>
<td>NET &quot;input&lt;3&gt;&quot;</td>
<td>LOC = &quot;P9&quot;;</td>
</tr>
<tr>
<td>NET &quot;input&lt;4&gt;&quot;</td>
<td>LOC = &quot;P11&quot;;</td>
</tr>
<tr>
<td>NET &quot;input&lt;5&gt;&quot;</td>
<td>LOC = &quot;P12&quot;;</td>
</tr>
<tr>
<td>NET &quot;input&lt;6&gt;&quot;</td>
<td>LOC = &quot;P13&quot;;</td>
</tr>
<tr>
<td>NET &quot;input&lt;7&gt;&quot;</td>
<td>LOC = &quot;P14&quot;;</td>
</tr>
<tr>
<td>NET &quot;rxin&quot;</td>
<td>LOC = &quot;P23&quot;;</td>
</tr>
<tr>
<td>NET &quot;txout&quot;</td>
<td>LOC = &quot;P27&quot;;</td>
</tr>
<tr>
<td>NET &quot;oe_adc&quot;</td>
<td>LOC = &quot;P17&quot;;</td>
</tr>
<tr>
<td>NET &quot;ckout20m&quot;</td>
<td>LOC = &quot;P16&quot;;</td>
</tr>
</tbody>
</table>

Figure 4-8: Microcontroller layout in Xilinx showing PIN connections and Sim2’s floor placement.
Chapter 5

Conclusion

Summary

In this thesis we describe the development of a high-performance microcontroller for FPGA implementation. It has also presented our experience and the benefits of utilizing an FPGA to implement a microcontroller for high speed data acquisition. The result of this study provides valuable information about FPGAs as target platforms for data acquisition applications. This project has furthermore shown that FPGA-based solutions lead to the design of flexible, scalable, and easy-to-maintain systems. A major benefit of using FPGAs is the fact that different architectural variations of the design can easily be tested and evaluated on real applications.

Several strengths in the architecture are shown as well areas that could be improved upon in future versions. We also demonstrated some of its strengths against existing commercial microcontrollers on the market. Despite the fact that we made little use of registers in our design, we still showed improved overall performance with the dual-port RAM architecture we used in the design.

The table below shows how our design stacks up against some of the existing commercial microcontrollers on the market using the specifications below.
The microcontrollers mentioned above and Sim2 have a number of similarities and differences.

The PIC18 microcontroller and Sim2 both have interrupt capabilities and make use of the Harvard and RISC architectures. Sim2’s instruction cycle consist of two stages - the fetch and execute cycles, while the PIC18 microcontroller’s instruction cycle consist of four stages. With the exception of Intel’s AD8051 and Motorola’s MC9S12C32 being CISC computers, all the other microcontrollers are use a RISC architecture including Sim2.

Table 5-1: Comparing Sim2 with some other existing commercial brands.

<table>
<thead>
<tr>
<th></th>
<th>Sim2</th>
<th>PicoBlaze</th>
<th>PIC18F1230</th>
<th>Intel AD8051</th>
<th>MC9S12C32</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max. CPU Speed MHz</td>
<td>75 MHz</td>
<td>87 MHz</td>
<td>40 MHz</td>
<td>12 MHz</td>
<td>25 MHz</td>
</tr>
<tr>
<td>Oscillator Cycles / Instruction</td>
<td>1²</td>
<td>2¹</td>
<td>4¹</td>
<td>12¹</td>
<td>2¹</td>
</tr>
<tr>
<td>MIPS (calculated values)</td>
<td>75 Million instruction per second</td>
<td>43.5 Million instruction per second</td>
<td>10 Million instructions per second</td>
<td>1 Million instruction per second</td>
<td>12.5 Million instruction per second</td>
</tr>
<tr>
<td>Interrupts</td>
<td>Single Interrupt</td>
<td>Single Interrupt</td>
<td>Multi interrupts</td>
<td>Multi interrupts</td>
<td>Multi Interrupts</td>
</tr>
<tr>
<td>RISC / CISC</td>
<td>RISC</td>
<td>RISC</td>
<td>RISC</td>
<td>CISC</td>
<td>CISC</td>
</tr>
<tr>
<td>Data Memory</td>
<td>256 byte RAM</td>
<td>64 byte</td>
<td>128 byte EEPROM</td>
<td>128 byte RAM</td>
<td>2K bytes RAM</td>
</tr>
<tr>
<td>Program Memory</td>
<td>256 word by 24-bit ROM</td>
<td>1Kbyte by 18-bit PROM</td>
<td>256 word by 16-bit EEPROM</td>
<td>4KB by 8 ROM</td>
<td>32K bytes EEPROM</td>
</tr>
<tr>
<td>Instruction size</td>
<td>24-bit</td>
<td>18-bit</td>
<td>16-bit</td>
<td>16-bit</td>
<td>16-bit</td>
</tr>
<tr>
<td>Data Word Width</td>
<td>8-bits</td>
<td>8-bits</td>
<td>8-bits</td>
<td>8-bits</td>
<td>16-bits</td>
</tr>
<tr>
<td>Number of Instructions</td>
<td>19</td>
<td>57 (21 types of instructions)</td>
<td>75</td>
<td>44</td>
<td>Powerful instruction set</td>
</tr>
<tr>
<td>Operating Voltage</td>
<td>5V</td>
<td>5V</td>
<td>2V to 5.5V</td>
<td>+5V ±10%</td>
<td>3.3V to 5V</td>
</tr>
<tr>
<td>Operating Temperature</td>
<td>0 to 70°C</td>
<td>0 to 70°C</td>
<td>-40 to +125°C</td>
<td>0°C to +70°C</td>
<td>-40 to +85°C</td>
</tr>
<tr>
<td>Timers</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>I/O pins</td>
<td>16</td>
<td>512</td>
<td>16</td>
<td>32</td>
<td>31</td>
</tr>
</tbody>
</table>

The microcontrollers mentioned above and Sim2 have a number of similarities and differences.

The PIC18 microcontroller and Sim2 both have interrupt capabilities and make use of the Harvard and RISC architectures. Sim2’s instruction cycle consist of two stages - the fetch and execute cycles, while the PIC18 microcontroller’s instruction cycle consist of four stages. With the exception of Intel’s AD8051 and Motorola’s MC9S12C32 being CISC computers, all the other microcontrollers are use a RISC architecture including Sim2.
Future Works

Future research can be devoted to the development of a more complete system making use of both the transmitter and receiver transducers. An implementation where the transmitter transducer can be used to send pulses generated from the microcontroller and captured for storage in an external RAM through the receiver transducer. This data can be processed by the Sim2 microcontroller and send out to the PC through the serial port.

There are several areas we could further investigate and improve upon; additional features can be added to enhance the capabilities of the Sim2 microcontroller. The rest of this section focuses on these.

A Transmitter-Receiver Implementation

The transmitter could be used to generate pulse signals at given intervals for the receiver transducer. An external RAM (E.g. 3KB RAM) can be used to store the incoming signals from the ADC. The external RAM needs to be used since there is a difference in speed for communication between the PC and microcontroller and the communication speed between the microcontroller and the ADC. This stored data can then be read and processed by the Sim2 microcontroller.

Additional Features

As discussed in chapter 1, a microcontroller can have a lot of features depending on its usage. Multi level interrupts could also be added to the Sim2 microcontroller to handle multiple interrupts from the program and external sources. Other features like circuit programming and debugging support can also be added. The entire design could also be sped up by increasing the
pipeline stages or/and reducing the fetch-execute stages into a clock cycle, instead of two as shown in our design.

Timers could be added to our design. A watchdog: triggers a program restart if it hangs or freezes due to some faulty condition. In addition to the watchdog, a timer like the programmable interval timer (PIT) can be included. A PIT just counts down from some value to zero. Once it reaches zero, it sends an interrupt to the processor indicating that it has finished counting. This is however useful for systems such as thermostats, which periodically test the temperature around them to see if they need to turn the air conditioner on, the heater on, etc. Another feature is the time processing unit or TPU for short. It is essentially just another timer, but more sophisticated. In addition to counting down, the TPU can detect input events; generate output events, and other useful operations. Special purpose interface modules could also be added to extend the application of the Sim2 microcontroller. The efficiency of the Instruction set could be increase by adding more instructions for bit manipulation and some other branch instructions if required.

The Sim2 microcontroller provides designers with a high performance, light-weight, cost-effective system solution for high-speed data acquisition applications that have strict budgetary requirements and very limited space.
References


9. "Using Digital Clock Managers (DCMs) in Spatan-3 FPGAs", *Xilinx, XAPP462 (v1.1) January 5, 2006*


12. Arne Linde, Tomas Nordstrom and Mikael Taveniku, "Using FPGAs to implement a reconfigurable highly parallel computer".


22. www.cse.psu.edu/~kyusun

Appendix A

VHDL Code of the Sim2 microcontroller

--Designer: Wilfred A. Glover-Akpey
--Date: August 2008
--Course: CSE600
--Affiliation: The Pennsylvania State University
--Description: Top Level Module, behavioral VHDL
--  I/O signals for the implementation board

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

--- Uncomment the following library declaration if instantiating
--- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity Sim2 is
  Port ( Clock : in  std_logic;
         Input : in  std_logic_vector(7 downto 0);
         Intrpt_in : in  std_logic;
         rxin : in  std_logic;
         Cout : out std_logic;
         rxo : out std_logic_vector(7 downto 0);
         txout : out std_logic;
         oe_adc : out std_logic;
         ckout20m : out std_logic;
         pc : out std_logic_vector(7 downto 0);
         output : out std_logic_vector(7 downto 0));
end Sim2;

architecture Behavioral of Sim2 is

component wag8bitpc is
  Port ( Reset : in  std_logic;
         PCClock : in  std_logic;
         Intrpt : in  std_logic;
         B : in std_logic_vector(7 downto 0);
         ...);
Sel : in std_logic_vector(1 downto 0);
C_out : out std_logic;
PC : out std_logic_vector(7 downto 0));
end component;

COMPONENT wagDCM is
  port ( CLKIN_IN : in std_logic;
          RST_IN : in std_logic;
          CLKFX_OUT : out std_logic;
          CLKFX180_OUT : out std_logic;
          CLKin_IBUF : out std_logic;
          CLK0_OUT : out std_logic;
          CLK180_OUT : out std_logic;
          LOCKED_OUT : out std_logic);
END COMPONENT;

COMPONENT InstROM is
  Port (
    InstAddress : in std_logic_vector(7 downto 0);
    dat2Out : out std_logic_vector(7 downto 0);
    dat1Out : out std_logic_vector(7 downto 0);
    dat0Out : out std_logic_vector(7 downto 0));
END COMPONENT;

COMPONENT wagIntControl is
  Port ( int : in STD_LOGIC;
          clk : in STD_LOGIC;
          reset : in std_logic;
          z : out STD_LOGIC);
END COMPONENT;

COMPONENT wagid is
  Port ( CSM : in std_logic;
          RSM : in std_logic;
          CLK_INV : in std_logic:
          CF : in std_logic:=‘0’;
          INTreq : in std_logic;
          IR : in std_logic_vector(7 downto 0);
          sel : out std_logic_vector(1 downto 0);
          pres : out std_logic;
          pck : out std_logic;
          bx : out std_logic;
          rir : out std_logic;
          wir : out std_logic);
END COMPONENT;

COMPONENT wagir is
  Port ( IR : in std_logic_vector(7 downto 0);
          WIR : in std_logic;
          RIR : in std_logic;
COMPONENT wagpmux is
Port (a : in std_logic_vector(7 downto 0);
b : in std_logic_vector(7 downto 0);
i : in std_logic;
    pmuxout : out std_logic_vector(7 downto 0));
END COMPONENT;

COMPONENT wag8bitreg is
Port (D : in std_logic_vector(7 downto 0);
    W : in std_logic;
    Y : out std_logic_vector(7 downto 0));
END COMPONENT;

COMPONENT wag1inv is
Port(a:in std_logic;
    z:out std_logic);
END COMPONENT;

COMPONENT wagdpmux is
    port (a,b,c,d : in std_logic_vector (7 downto 0);
        sel : in std_logic_vector (1 downto 0);
        pmuxout : out std_logic_vector(7 downto 0));
END COMPONENT;

COMPONENT wagDualPortRam is
    Port (w0 : in STD_LOGIC;
        dat0 : in STD_LOGIC_VECTOR(7 downto 0);
        adr0 : in STD_LOGIC_VECTOR(7 downto 0);
        dat2 : out STD_LOGIC_VECTOR(7 downto 0);
        adr1 : in STD_LOGIC_VECTOR(7 downto 0);
        dat1 : out STD_LOGIC_VECTOR(7 downto 0));
END COMPONENT;

COMPONENT wag8bitRegister is
    port (IR : in std_logic_vector(7 downto 0);
        WIR : in std_logic; --CLK
        LOAD : in std_logic;
        RIR : in std_logic; --CLR (RESET)
        IROUT : out std_logic_vector(7 downto 0));
END COMPONENT;

COMPONENT wagControlUnit is
    Port (Clock2 : in STD_LOGIC;
        IR : in std_logic_vector(7 downto 0);
        mdw : out std_logic; --
COMPONENT wag8bitalu
Port (A : in std_logic_vector(7 downto 0);
B : in std_logic_vector(7 downto 0);
opcode2 : in std_logic_vector(7 downto 0);
SUB : in std_logic;
Ans : out std_logic_vector(7 downto 0);
COUT : out std_logic);
END COMPONENT;

COMPONENT wagdff is
port( Cl, D : in std_logic;
Q : out std_logic);
END COMPONENT;

COMPONENT wag8bitff is
Port ( D : in std_logic_vector(7 downto 0);
W : in std_logic;
Y : out std_logic_vector(7 downto 0));
END COMPONENT;

COMPONENT adc2tx01 is
port ( ckin : in std_logic;
Logic1 : in std_logic;
rxin : in std_logic;
txp : in std_logic_vector (7 downto 0);
ckout1p8m : out std_logic;
ckout8k : out std_logic;
ckout20m : out std_logic;
rxo : out std_logic_vector (7 downto 0);
txout : out std_logic);
END COMPONENT;

COMPONENT wag8bitOr is
Port ( x : in STD_LOGIC_VECTOR(7 downto 0);
y : in STD_LOGIC_VECTOR(7 downto 0);
z : out STD_LOGIC_VECTOR(7 downto 0));
END COMPONENT;

SIGNAL CLK0_SIG : std_logic;
SIGNAL LOCK_SIG : std_logic;
SIGNAL dat0_sig : std_logic_vector (7 downto 0);
SIGNAL dat1_sig : std_logic_vector (7 downto 0);
SIGNAL  dat2_sig : std_logic_vector (7 downto 0);
SIGNAL  pc_sig  : std_logic_vector (7 downto 0);
SIGNAL  intreq_sig : std_logic;
SIGNAL  sel_sig : std_logic_vector (1 downto 0);
SIGNAL  pres_sig : std_logic;
SIGNAL  pck_sig : std_logic;
SIGNAL  bx_sig : std_logic;
SIGNAL  rir_sig : std_logic;
SIGNAL  wir_sig : std_logic;
SIGNAL  pmux_sig : std_logic_vector (7 downto 0);
SIGNAL  sdat1_sig : std_logic_vector (7 downto 0);
SIGNAL  dmux_out : std_logic_vector (7 downto 0);
SIGNAL  ddat1_sig : std_logic_vector (7 downto 0);
SIGNAL  au_out : std_logic_vector (7 downto 0);
SIGNAL  dat0_sig_0 : std_logic_vector (7 downto 0);
SIGNAL  dat0_sig_00 : std_logic_vector (7 downto 0);
SIGNAL  dat1_sig_1 : std_logic_vector (7 downto 0);
SIGNAL  dat1_sig_11 : std_logic_vector (7 downto 0);
SIGNAL  dat2_sig_2 : std_logic_vector (7 downto 0);
SIGNAL  dat2_sig_22 : std_logic_vector (7 downto 0);
SIGNAL  mdw_sig : std_logic;
SIGNAL  subSig : std_logic;
SIGNAL  wc_sig : std_logic;
SIGNAL  dx1_sig : std_logic;
SIGNAL  dx0_sig : std_logic;
SIGNAL  wot_sig : std_logic;
SIGNAL  cout_sig : std_logic;
SIGNAL  CLK_INV_SIG : std_logic;
SIGNAL  CLK2X_SIG : std_logic;
SIGNAL  CLK2X_SIG_INV : std_logic;
SIGNAL  CLKIN_IBUFG_SIG : std_logic;
SIGNAL  ckout1p8m_sig : std_logic;
SIGNAL  ckout8k_sig : std_logic;
SIGNAL  ckout20m_sig : std_logic;
SIGNAL  output_sig : std_logic_vector (7 downto 0);
SIGNAL  rxo_sig, IROUT_sig : std_logic_vector (7 downto 0);

begin

input_mux : wagpmux Port map ( a=>input , b=>rxo_sig , i=>'0' , pmuxout =>
input_sig); --sel_in

-----------------------------------------

-- output
output : wag8bitff Port map ( D => sdat1_sig, W => wot_sig, Y => output_sig);

-- D ff for ALU

dff : wagdff port map (Cl => wc_sig, D => cout_sig, Q => cf_sig);

-- Arithmetic and Logic Unit

alu : wag8bitalu Port map (A => ddat1_sig, B => sdat1_sig, opcode2 => dat2_sig_22, SUB => sub_sig, Ans => au_out, COUT => cout_sig);

-- Control Unit

cu : wagControlUnit Port map (Clock2 => CLK0_SIG, IR => dat2_sig_22, mdw => mdw_sig, sub => sub_sig, dx1 => dx1_sig, dx0 => dx0_sig, wc => wc_sig, wot => wot_sig);

dual port RAM

dpRAM : wagDualPortRam Port map (w0 => mdw_sig, dat0 => dmux_out, adr0 => dat0_sig_00, dat2 => ddat1_sig, adr1 => dat1_sig_11, dat1 => sdat1_sig);

-- (dual port R) mux

dmux : wagdpmux Port map (a => sdat1_sig, b => au_out, c => dat1_sig_11, d => input_sig, sel(1) => dx1_sig, sel(0) => dx0_sig, pmuxout => dmux_out);

-- Pipeline Register 1

Pipeline0 : wag8bitRegister Port map (IR => dat0_sig, WIR => CLK0_SIG, LOAD => '1', RIR => rir_sig, IROUT => dat0_sig_00);

-- Pipeline Register 2

Pipeline1 : wag8bitRegister Port map (IR => dat1_sig, WIR => CLK0_SIG, LOAD => '1', RIR => rir_sig, IROUT => dat1_sig_11);
-- Pipeline Register 3
Pipeline2 : wag8bitRegister Port map (IR=>dat2_sig, WIR=>CLK0_SIG,
LOAD=>'1', RIR=>rir_sig, IROUT =>dat2_sig_2);

----------------------------------
-- pc mux
----------------------------------

pmux : wagpmux Port map (a =>"00000001", b =>dat1_sig , i =>bx_sig ,
pmuxout => pmux_sig);

-- Interrupt Control

IntCntrl : wagIntControl Port Map ( int=>Intrpt_in, clk=>CLK0_SIG,
reset=>LOCK_SIG, z=>intreq_sig);

-- Instruction Register

ir : wag8bitRegister Port map (IR=>dat2_sig, WIR=>wir_sig, LOAD=>'1',
RIR=>rir_sig, IROUT =>IROUT_sig);

-- Instruction Decoder

id : wagid Port map (CSM =>CLK0_SIG, RSM=>LOCK_SIG,
CLK_INV=>CLK_INV_SIG, CLK_INV=>CLK_INV_SIG, CF=>cf_sig,INTreq=>intreq_sig ,IR=>IROUT_sig,
pres=>pres_sig ,pck=>pck_sig ,bx=>bx_sig ,rir=>rir_sig ,wir =>wir_sig);

-- ROM

rom : InstROM Port map (InstAddress=>pc_sig, dat0Out=>dat0_sig ,
dat1Out =>dat1_sig , dat2Out =>dat2_sig);

-- PC

prg_counter : wag8bitpc Port map(Reset =>pres_sig, PCClock
=>pck_sig , Intrpt =>Intrpt_in , B =>pmux_sig , sel =>sel_sig , C_out =>Cout , PC =>pc_sig);

-- DCM
dcm_sim : wagDCM port map ( CLKIN_IN => Clock, RST_IN => '0', CLKFX_OUT => CLKO_SIG, CLKF180_VOL => CLK_INV_SIG, CLKIN_IBUFG_OUT => CLK0_OUT => CLK2X_SIG, CLK180_OUT => CLK2X_SIG_INV, LOCKED_OUT => LOCK_SIG);

IO_Com : adc2tx01 port map ( ckin => CLK0_SIG, Logic1 => '1', rxin => rxin, txp => output_sig, ckout1p8m => ckout1p8m_sig, ckout8k => ckout8k_sig, ckout20m => ckout20m_sig, rxo => rxo_sig, txout => txout);

ckout20m <= ckout20m_sig;
pc <= pc_sig;
output <= output_sig;
oe_adc <= '0';
rxo <= rxo_sig;

end Behavioral;

--Designer: Wilfred A. Glover-Akpey
--Date: September 11, 2007
--Course: CSE600
--Affiliation: The Pennsylvania State University
--Description: Top level Module (Sim2) for PMW program

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

--  rxin : in std_logic;
--  rxo : out std_logic_vector(7 downto 0);
--  txout : out std_logic;
--  oe_adc : out std_logic;
--  ckout20m : out std_logic;

entity Sim2 is
Port ( Clock : in std_logic;
       input : in std_logic_vector(7 downto 0);
       Intrpt_in : in std_logic;
       Cout : out std_logic;
       AN : out std_logic_vector(2 downto 0);
       pc : out std_logic_vector(7 downto 0);
   );
architecture Behavioral of Sim2 is

component wag8bitpc is
    Port ( Reset : in std_logic;
          PCClock : in std_logic;
          Intrpt : in std_logic;
          B : in std_logic_vector(7 downto 0);
          sel : in std_logic_vector(1 downto 0);
          C_out : out std_logic;
          PC : out std_logic_vector(7 downto 0));
end component;

COMPONENT wagDCM is
    port ( CLKIN_IN : in std_logic;
          RST_IN : in std_logic;
          CLKFX_OUT : out std_logic;
          CLKFX180_OUT : out std_logic;
          CLKIN_IBUFG_OUT : out std_logic;
          CLK0_OUT : out std_logic;
          CLK180_OUT : out std_logic;
          LOCKED_OUT : out std_logic);
end COMPONENT;

COMPONENT InstROM is
    Port ( InstAddress : in std_logic_vector(7 downto 0);
          dat2Out : out std_logic_vector(7 downto 0);
          dat1Out : out std_logic_vector(7 downto 0);
          dat0Out : out std_logic_vector(7 downto 0));
end COMPONENT;

COMPONENT wagIntControl is
    Port ( int : in STD_LOGIC;
          clk : in STD_LOGIC;
          reset : in std_logic;
          z : out STD_LOGIC);
end COMPONENT;

COMPONENT wagid is
    Port ( CSM : in std_logic;
           RSM : in std_logic;
           CLK_INV : in std_logic;
           CF : in std_logic:=’0’;
           INTreq : in std_logic;
           IR : in std_logic_vector(7 downto 0);
           sel : out std_logic_vector(1 downto 0);
           pres : out std_logic;
          output : out std_logic_vector(7 downto 0));
end component;
COMPONENT wagir is
    Port ( IR : in std_logic_vector(7 downto 0);
          WIR : in std_logic;
          RIR : in std_logic;
          IROUT : out std_logic_vector(7 downto 0));
END COMPONENT;

COMPONENT wagpmux is
    Port ( a : in std_logic_vector(7 downto 0);
           b : in std_logic_vector(7 downto 0);
           i : in std_logic;
           pmuxout : out std_logic_vector(7 downto 0));
END COMPONENT;

COMPONENT wag8bitreg is
    Port ( D : in std_logic_vector(7 downto 0);
           W : in std_logic;
           Y : out std_logic_vector(7 downto 0));
END COMPONENT;

COMPONENT wag1inv is
    Port( a:in std_logic;
          z:out std_logic);
END COMPONENT;

COMPONENT wagdpmux is
    port ( a,b,c,d : in std_logic_vector(7 downto 0);
           sel : in std_logic_vector(1 downto 0);
           pmuxout : out std_logic_vector(7 downto 0));
END COMPONENT;

COMPONENT wagDualPortRam is
    Port ( w0 : in STD_LOGIC;
           dat0 : in STD_LOGIC_VECTOR(7 downto 0);
           adr0 : in STD_LOGIC_VECTOR(7 downto 0);
           dat2 : out STD_LOGIC_VECTOR(7 downto 0);
           adr1 : in STD_LOGIC_VECTOR(7 downto 0);
           dat1 : out STD_LOGIC_VECTOR(7 downto 0));
END COMPONENT;

COMPONENT wag8bitRegister is
    port(   IR : in std_logic_vector(7 downto 0);
WIR : in std_logic; --CLK
LOAD : in std_logic;
RIR : in std_logic; --CLR (RESET)
IROUT : out std_logic_vector(7 downto 0));
END COMPONENT;

COMPONENT wagControlUnit is
Port ( Clock2 : in STD_LOGIC;
IR : in std_logic_vector(7 downto 0);
mdw : out std_logic; --
sub : out std_logic; --
dx1 : out std_logic; --
dx0 : out std_logic; --
wco : out std_logic; --
wot : out std_logic);
END COMPONENT;

COMPONENT wag8bitalu
Port ( A : in std_logic_vector(7 downto 0);
B : in std_logic_vector(7 downto 0);
opcode2 : in std_logic_vector(7 downto 0);
SUB : in std_logic;
Ans : out std_logic_vector(7 downto 0);
COUT : out std_logic);
END COMPONENT;

COMPONENT wagdff
port (
Cl, D : in std_logic;
Q : out std_logic);
END COMPONENT;

COMPONENT wag8bitff
Port ( D : in std_logic_vector(7 downto 0);
W : in std_logic;
Y : out std_logic_vector(7 downto 0));
END COMPONENT;

--COMPONENT adc2tx01
-- port
-- ckin : in std_logic;
-- Logic1 : in std_logic;
-- rinx : in std_logic;
-- tsp : in std_logic_vector(7 downto 0);
-- ckout1p8m : out std_logic;
-- ckout8k : out std_logic;
-- ckout20m : out std_logic;
-- rxo : out std_logic_vector(7 downto 0);
-- txout : out std_logic);
--END COMPONENT;

--END COMPONENT;
COMPONENT wag8bitOr is
  Port ( x : in  STD_LOGIC_VECTOR(7 downto 0);
        y : in  STD_LOGIC_VECTOR(7 downto 0);
        z : out  STD_LOGIC_VECTOR(7 downto 0));
END COMPONENT;

SIGNAL  CLK          : std_logic;
SIGNAL  LOCK_SIG     : std_logic;
SIGNAL  dat0_sig     : std_logic_vector (7 downto 0);
SIGNAL  dat1_sig     : std_logic_vector (7 downto 0);
SIGNAL  dat2_sig     : std_logic_vector (7 downto 0);
SIGNAL  pc_sig       : std_logic_vector (7 downto 0);
SIGNAL  intreq_sig   : std_logic;
SIGNAL  IROUT_sig    : std_logic_vector (7 downto 0);
SIGNAL  sel_sig      : std_logic_vector (1 downto 0);
SIGNAL  pres_sig     : std_logic;
SIGNAL  pck_sig      : std_logic;
SIGNAL  bx_sig       : std_logic;
SIGNAL  rir_sig      : std_logic;
SIGNAL  wir_sig      : std_logic;
SIGNAL  cf_sig       : std_logic;
SIGNAL  pmux_sig     : std_logic_vector (7 downto 0);
SIGNAL  sdat1_sig    : std_logic_vector (7 downto 0);
SIGNAL  dmux_out     : std_logic_vector (7 downto 0);
SIGNAL  ddat1_sig    : std_logic_vector (7 downto 0);
SIGNAL  au_out       : std_logic_vector (7 downto 0);
SIGNAL  dat0_sig_0   : std_logic_vector (7 downto 0);
SIGNAL  dat0_sig_00  : std_logic_vector (7 downto 0);
SIGNAL  dat1_sig_1   : std_logic_vector (7 downto 0);
SIGNAL  dat1_sig_11  : std_logic_vector (7 downto 0);
SIGNAL  dat2_sig_2   : std_logic_vector (7 downto 0);
SIGNAL  dat2_sig_22  : std_logic_vector (7 downto 0);
SIGNAL  mdw_sig      : std_logic;
SIGNAL  sub_sig      : std_logic;
SIGNAL  wc_sig       : std_logic;
SIGNAL  dx1_sigs     : std_logic;
SIGNAL  dx0_sigs     : std_logic;
SIGNAL  wot_sigs     : std_logic;
SIGNAL  cout_sigs    : std_logic;
SIGNAL  CLK_INV_SIG  : std_logic;
SIGNAL  CLK2X_SIG    : std_logic;
SIGNAL  CLK2X_SIG_INV: std_logic;
SIGNAL  CLKIN_IBUFG_SIG : std_logic;
SIGNAL  ckout1p8m_sig: std_logic;
SIGNAL  ckout8k_sig  : std_logic;
SIGNAL  ckout20m_sig : std_logic;
SIGNAL output_sig : std_logic_vector (7 downto 0);
SIGNAL input_sig : std_logic_vector (7 downto 0);
SIGNAL rxo_sig, IROUT_sig : std_logic_vector (7 downto 0);

begin

input_mux : wagpmux Port map ( a=>input , b=>rxo_sig , i =>'0' , pmuxout => input_sig); -- sel_in

---------------
-- output
---------------

outputt : wag8bitff Port map ( D => sdat1_sig, W =>wot_sig , Y =>output_sig);

---------------
-- D ff for ALU
---------------

dff : wagdff port map( Cl=>wc_sig, D =>cout_sig , Q => cf_sig);

---------------
-- Arithmetic and Logic Unit
---------------

alu : wag8bitalu Port map ( A => ddat1_sig , B => sdat1_sig, opcode2=>dat2_sig_22 , SUB => sub_sig, Ans => au_out , COUT =>cout_sig);

---------------
-- Control Unit
---------------

cu : wagControlUnit Port map( Clock2 => CLK0_SIG , IR => dat2_sig_22, mdw =>mdw_sig , sub => sub_sig , dx1 => dx1_sig, dx0 =>dx0_sig , wc => wc_sig, wot =>wot_sig);

---------------
-- dual port RAM
---------------

dpRAM : wagDualPortRam Port map (w0 =>mdw_sig,dat0 =>dmux_out ,adr0 =>dat0_sig_00, dat2 =>ddat1_sig , adrl =>dat1_sig_11, dat1=> sdat1_sig);

---------------
-- (dual port R) mux
---------------

dmux : wagdpmux Port map ( a => sdat1_sig, b => au_out, c => dat1_sig_11, d => input_sig , sel(1) =>dx1_sig, sel(0) =>dx0_sig, pmuxout => dmux_out);

---------------
-- Pipeline Register 1
---------------
Pipeline0 : wag8bitRegister Port map  
(IR=>dat0_sig, WIR=>CLK0_SIG, LOAD=>'1',
RIR=>rir_sig, IROUT =>dat0_sig_00);

-- Pipeline Register 2

Pipeline1 : wag8bitRegister Port map  
(IR=>dat1_sig, WIR=>CLK0_SIG, LOAD=>'1',
RIR=>rir_sig, IROUT =>dat1_sig_11);

-- Pipeline Register 3

Pipeline2 : wag8bitRegister Port map  
(IR=>dat2_sig, WIR=>CLK0_SIG, LOAD=>'1',
RIR=>rir_sig, IROUT =>dat2_sig_22);

-- pc mux

pmux : wagpmux Port map  
(a =>"00000001", b =>dat1_sig, i =>bx_sig,
pmuxout => pmux_sig);

-- Interrupt Control

IntCntrl : wagIntControl Port Map ( int=>Intrpt_in, clk=>CLK0_SIG,
reset=>LOCK_SIG, z=>intreq_sig);

-- Instruction Register

ir : wag8bitRegister Port map  
(IR=>dat2_sig, WIR=>wir_sig, LOAD=>'1',
RIR=>rir_sig, IROUT =>IROUT_sig);

-- Instruction Decoder

id : wagid Port map  
(CSM =>CLK0_SIG, RSM=>LOCK_SIG,
CLK_INV=>CLK_INV_SIG, CF=>cf_sig,INTreq=>intreq_sig ,IR=>IROUT_sig,sel=>sel_sig,
pres=>pres_sig,pck=>pck_sig,bx=>bx_sig,rir=>rir_sig,wir =>wir_sig);

-- ROM

rom : InstROM Port map  
(InstAddress=>pc_sig, dat0Out=>dat0_sig,
dat1Out => dat1_sig, dat2Out => dat2_sig);

-----------------------------------------

-- PC

-----------------------------------------

CLK0_SIG

prg_counter : wag8bitpc Port map(Reset => pres_sig, PCClock => pck_sig, Intrpt => Intrpt_in, B => pmux_sig, sel => sel_sig, C_out => Cout, PC => pc_sig);

-----------------------------------------

-- DCM

-----------------------------------------

dcm_sim : wagDCM port map (CLKIN => Clock, RST_IN => '0', CLKFX_OUT => CLK0_SIG, CLKFX180_OUT => CLK_INV_SIG, CLKIN_IBUFG_OUT => CLKIN_IBUFG_SIG, CLK0_OUT => CLK2X_SIG, CLK180_OUT => CLK2X_SIG_INV, LOCKED_OUT => LOCK_SIG);

-- IO_Com : adc2tx01 port map (ckin => CLK0_SIG, Logic1 => '1', rxin => rxin, txp => output_sig, ckout1p8m => ckout1p8m_sig, ckout8k => ckout8k_sig, ckout20m => ckout20m_sig, rxo => rxo_sig, txout => txout);

--

   pc <= pc_sig;
   output <= output_sig;
end Behavioral;

--

   oe_adc <= '0';
--

   rxo <= rxo_sig;
--

   ckout20m <= ckout20m_sig;

------------------------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity wag8bitpc is

   Port (Reset : in std_logic;
PCClock : in std_logic;
Intrpt : in std_logic;

------------------------------------------------------------------------------------------------

-- Designer: Wilfred A. Glover-Akpey
-- Date: August 2008
-- Course: CSE600
-- Affiliation: The Pennsylvania State University
-- Description: 8-bit Program Counter (PC)

------------------------------------------------------------------------------------------------
architecture Behavioral of wag8bitpc is

COMPONENT wag8bit2_1mux
    Port ( I0 : in std_logic_vector(7 downto 0);
          I1 : in std_logic_vector(7 downto 0);
          R  : in std_logic;
          Z  : out std_logic_vector(7 downto 0));
END COMPONENT;

COMPONENT wag8bitff
    Port ( D : in std_logic_vector(7 downto 0);
           W : in std_logic;
           Y : out std_logic_vector(7 downto 0));
END COMPONENT;

COMPONENT wag8bitreg
    Port ( D  : in std_logic_vector(7 downto 0);
           W  : in std_logic;
           Y  : out std_logic_vector(7 downto 0));
END COMPONENT;

COMPONENT wag1add8bit
    Port(  ci : in std_logic;
           a  : in std_logic_vector(7 downto 0);
           b  : in std_logic_vector(7 downto 0);
           s  : out std_logic_vector(7 downto 0);
           co : out std_logic);
END COMPONENT;

COMPONENT wag8bit4_1mux
    Port ( d0 : in std_logic_vector(7 downto 0);
           d1 : in std_logic_vector(7 downto 0);
           d2 : in std_logic_vector(7 downto 0);
           d3 : in std_logic_vector(7 downto 0);
           s  : in std_logic_vector(1 downto 0);
           z  : out std_logic_vector(7 downto 0));
END COMPONENT;

COMPONENT wagdff
    Port ( D : in std_logic;
           Cl : in std_logic;
           Q  : out std_logic);
BEGIN
  SIGNAL ground0 : std_logic_vector(7 downto 0);
  SIGNAL ground1 : std_logic;
  SIGNAL CarryOut : std_logic;
  SIGNAL MuxToFF : std_logic_vector(7 downto 0);
  SIGNAL A : std_logic_vector(7 downto 0);
  SIGNAL sum : std_logic_vector(7 downto 0);
  SIGNAL varsum : std_logic_vector(7 downto 0);
  SIGNAL savedPC : std_logic_vector(7 downto 0);
  SIGNAL Int_Vector : std_logic_vector(7 downto 0);

  begin
  ground0 <= "00000000";
  Int_Vector <= "00110010"; --Interrupt Vector @ 50;
  ground1 <= '0';
  mux8bit : wag8bit2_1mux PORT MAP (I0=>ground0, I1=>varsum, R=>Reset,
  Z=>MuxToFF);
  ff8bit : wag8bitff PORT MAP (D=>MuxToFF, W=>PCClock, Y=>A);
  adder8bit : wag1add8bit PORT MAP (a=>A, b=>B, ci=>ground1, s=>sum,
  co=>CarryOut);
  bitmux : wag8bit4_1mux PORT MAP (d0=>savedPC, d1=>ground0, d2=>sum,
  d3=>Int_Vector, s=>sel, z=>varsum);
  reg : wag8bitreg PORT MAP (D=>sum, W=>Intrpt, Y=>savedPC); -- Serves as a
  Register to save the ROM Address
  PC <= A;
  C_out <= CarryOut;

  end Behavioral;

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
Uncomment the following library declaration if instantiating any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity wagIntControl is
  Port (int : in STD_LOGIC;
        clk : in STD_LOGIC;
        reset : in std_logic;
        z   : out STD_LOGIC);
end wagIntControl;

architecture Behavioral of wagIntControl is

COMPONENT wagfcd is
  port(C, D, CLR : in std_logic;
       Q       : out std_logic);
end COMPONENT;

COMPONENT wag1or2 is
  port(a,b : in std_logic;
       z : out std_logic);
end COMPONENT;

COMPONENT wag1inv is
  Port(a : in std_logic;
       Z : out std_logic);
end COMPONENT;

SIGNAL D_sig,reset_inv : std_logic;
SIGNAL D2_sig,or_sig1 : std_logic;

begin
  inv_gate  : wag1inv port map (reset,reset_inv);
  D1_gate   : wagfcd port map (int, '1', or_sig1, D_sig);
  D2_gate   : wagfcd port map (clk, D_sig, reset_inv, D2_sig);
  or_gate1  : wag1or2 port map (reset_inv,D2_sig,or_sig1);

  z<= D2_sig;
end Behavioral;

.Designer: Wilfred A. Glover-Akpey
.Date: September 11,2007
.Course: CSE600
.Affiliation: The Pennsylvania State University
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity wagControlUnit is
   Port (Clock2 : in  STD_LOGIC;
          IR  : in std_logic_vector(7 downto 0);
          mdw : out std_logic; --
          sub : out std_logic; --
          dx1 : out std_logic; --
          dx0 : out std_logic; --
          wc  : out std_logic; --
          wot : out std_logic);
end wagControlUnit;

architecture Behavioral of wagControlUnit is

COMPONENT wag1and2
   Port ( a : in std_logic;
          b : in std_logic;
          z : out std_logic);
END COMPONENT;

COMPONENT wag1and3
   Port ( a : in std_logic;
          b : in std_logic;
          c : in std_logic;
          z : out std_logic);
END COMPONENT;

COMPONENT wagand5
   Port ( a : in std_logic;
          b : in std_logic;
          c : in std_logic;
          d : in std_logic;
          e : in std_logic;
          z : out std_logic);
END COMPONENT;

COMPONENT wagor3
Port ( a : in std_logic;
   b : in std_logic;
   c : in std_logic;
   z : out std_logic);
END COMPONENT;

COMPONENT wag1inv
Port ( a : in std_logic;
       z : out std_logic);
END COMPONENT;

COMPONENT wagdelay
Port ( a : in std_logic;
       z : out std_logic);
END COMPONENT;

COMPONENT wagLookupTable
Port ( CLK2 : in std_logic;
      IPCODE : in std_logic_vector(7 downto 0);
      MDW : out std_logic;
      DX0 : out std_logic; --
      DX1 : out std_logic); --
END COMPONENT;

-- Bits to decode from the instruction
SIGNAL IR7 : std_logic;
SIGNAL IR6 : std_logic;
SIGNAL IR5 : std_logic;
SIGNAL IR4 : std_logic;
SIGNAL IR3 : std_logic;
SIGNAL IR2 : std_logic;
SIGNAL IR1 : std_logic;
SIGNAL mdw_sig : std_logic;
SIGNAL dx1_sig : std_logic;
SIGNAL dx0_sig : std_logic;

begin

----------------------------------------------------------------------------------
--- Look up table
----------------------------------------------------------------------------------
  -- T => T,   CL2X=>CL2X ,
lookupTable : wagLookupTable PORT MAP ( CLK2 =>Clock2 , IPCODE => IR ,MDW
=> mdw_sig , DX0 => dx0_sig , DX1 =>dx1_sig );

----------------------------------------------------------------------------------
-- Setup the opcode lines IR7, IR6, IR5, IR4, IR3, IR2, IR1
IR7 <= IR(7);
IR6 <= IR(6);
IR5 <= IR(5);
IR4 <= IR(4);
IR3 <= IR(3);
IR2 <= IR(2);
IR1 <= IR(1);

-- This block produces the WOT output

wot <= '1' when (Clock2='0' and IR6='1' and IR3='1') else '0';

-- This section produces the wc output

wc <= '1' when (Clock2='0' and IR5='0' and IR4='1' and IR2='0' and IR1='0') else '0';

-- This section produces the SUB output

SUB <= '1' when (IR4='1' and IR3='1' and IR5='0' and IR6='0' and IR7='0' and IR2='0' and IR1='0') else '0';

--- DP RAM control signals

dx1 <= dx1_sig;
dx0 <= dx0_sig;
mdw <= mdw_sig;

end Behavioral;

-- Designer:  Wilfred A. Glover-Akpey
-- Date:     August 2008
-- Course: CSE600
-- Affiliation: The Pennsylvania State University
-- Description: Arithmetic and Logic Unit (ALU)
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity wag8bitalu IS
  Port (A : in std_logic_vector(7 downto 0);
        B : in std_logic_vector(7 downto 0);
        opcode2 : in std_logic_vector(7 downto 0);
        SUB : in std_logic;
        Ans : out std_logic_vector(7 downto 0);
        COUT : out std_logic);
end wag8bitalu;

architecture Behavioral of wag8bitalu is

COMPONENT wagxor2
  Port (a : in std_logic;
        b : in std_logic;
        z : out std_logic);
END COMPONENT;

COMPONENT wag1add8bit
  Port (ci : in std_logic;
        a : in std_logic_vector(7 downto 0);
        b : in std_logic_vector(7 downto 0);
        s : out std_logic_vector(7 downto 0);
        co : out std_logic);
END COMPONENT;

COMPONENT wag8bitAnd
  Port (x : in STD_LOGIC_VECTOR(7 downto 0);
        y : in STD_LOGIC_VECTOR(7 downto 0);
        z : out STD_LOGIC_VECTOR(7 downto 0));
END COMPONENT;

COMPONENT wag8bitOr
  Port (x : in STD_LOGIC_VECTOR(7 downto 0);
        y : in STD_LOGIC_VECTOR(7 downto 0);
        z : out STD_LOGIC_VECTOR(7 downto 0));
END COMPONENT;

COMPONENT wag8bitXor
  Port (x : in STD_LOGIC_VECTOR(7 downto 0);
        y : in STD_LOGIC_VECTOR(7 downto 0);
        z : out STD_LOGIC_VECTOR(7 downto 0));
END COMPONENT;

COMPONENT wag8bitlsl
  is
    Port (x : in STD_LOGIC_VECTOR(7 downto 0);
          Cin : in STD_LOGIC;
Cout : out STD_LOGIC;
z : out  STD_LOGIC_VECTOR(7 downto 0));
END COMPONENT;

COMPONENT wag8bitlsr is
 Port ( x : in  STD_LOGIC_VECTOR(7 downto 0);
 Cin : in STD_LOGIC;
 Cout : out STD_LOGIC;
 z : out  STD_LOGIC_VECTOR(7 downto 0));
END COMPONENT;

COMPONENT wagInvAll
 Port ( x : in  STD_LOGIC_VECTOR(7 downto 0);
 y : out  STD_LOGIC_VECTOR(7 downto 0));
END COMPONENT;

COMPONENT wag8InputMux
 port ( a,b,c,d,e,f,g,h     : in std_logic_vector(7 downto 0);
 opcode_2           : in std_logic_vector(7 downto 0);
 pmuxout             : out std_logic_vector(7 downto 0));
END COMPONENT;

COMPONENT wagCarry4Mux is
 port ( a,b,c,d : in std_logic;
 opcode_2 : in std_logic_vector(7 downto 0);
 pmuxout : out std_logic);
END COMPONENT;

SIGNAL B_tmp : std_logic_vector(7 downto 0);
SIGNAL Cin_tmp : std_logic;
SIGNAL Cout_tmp : std_logic;
SIGNAL lsl_Cout_sig : std_logic;
SIGNAL lsr_Cout_sig : std_logic;
SIGNAL lsl_sig : std_logic_vector(7 downto 0);
SIGNAL lsr_sig : std_logic_vector(7 downto 0);
SIGNAL not_sig : std_logic_vector(7 downto 0);
SIGNAL or_sig  : std_logic_vector(7 downto 0);
SIGNAL xor_sig : std_logic_vector(7 downto 0);
SIGNAL and_sig : std_logic_vector(7 downto 0);
SIGNAL addsub_sig : std_logic_vector(7 downto 0);
SIGNAL Cout_sig : std_logic;
SIGNAL CarrySel_sig : std_logic:='0';

begin

------------------------------------------------------
-- 4 Input Multiplexer ALU
------------------------------------------------------
CarryMux : wagCarry4Mux port map (a=>Cout_sig, b=>lsr_Cout_sig, c=>lsr_Cout_sig, d=>'0', opcode_2=>opcode2, pmuxout =>CarrySel_sig);

-- 8 Input Multiplexer ALU

InputMuxAlu : wag8InputMux port map (a=>addsub_sig, b=>lsl_sig, c=>and_sig, d=>lsr_sig, e=>or_sig, f=>not_sig, g=>xor_sig, h=>'00000000', opcode_2=>opcode2, pmuxout =>Ans);

-- Shift right (logical)

Rightshift : wag8bitlsr Port map (x =>A, Cin =>CarrySel_sig, Cout =>lsr_Cout_sig, z =>lsr_sig);

-- Shift left (logical)

Leftshift : wag8bitlsl Port map (x =>A, Cin =>CarrySel_sig, Cout =>lsl_Cout_sig, z =>lsl_sig);

-- "and" operation

And_op : wag8bitAnd Port map (x =>A, y =>B, z =>and_sig);

-- "or" operation

Or_op : wag8bitOr Port map (x =>A, y =>B, z =>or_sig);

-- "xor" operation

Xor_op : wag8bitXor Port map (x =>A, y =>B, z =>xor_sig);

-- "not" operation
NotAll : wagInvAll Port map ( x => A , y => not_sig );

-- "addition" or "subtraction" operation

xor_0 : wagxor2 PORT MAP(a=> SUB, b=> B(0), z=> B_tmp(0));
xor_1 : wagxor2 PORT MAP(a=> SUB, b=> B(1), z=> B_tmp(1));
xor_2 : wagxor2 PORT MAP(a=> SUB, b=> B(2), z=> B_tmp(2));
xor_3 : wagxor2 PORT MAP(a=> SUB, b=> B(3), z=> B_tmp(3));
xor_4 : wagxor2 PORT MAP(a=> SUB, b=> B(4), z=> B_tmp(4));
xor_5 : wagxor2 PORT MAP(a=> SUB, b=> B(5), z=> B_tmp(5));
xor_6 : wagxor2 PORT MAP(a=> SUB, b=> B(6), z=> B_tmp(6));
xor_7 : wagxor2 PORT MAP(a=> SUB, b=> B(7), z=> B_tmp(7));
xor_8 : wagxor2 PORT MAP(a=> SUB, b=> '0', z=> Cin_tmp);

au : wag1add8bit PORT MAP(ci=> Cin_tmp, a=> A, b=> B_tmp, s=> addsub_sig, co=> Cout_tmp);

xor_9 : wagxor2 PORT MAP(a=> SUB, b=> Cout_tmp, z=> Cout_sig);
end wagdff;

architecture archi of wagdff is

begin

process (Cl)
begin
if (rising_edge(Cl)) then
Q <= D after 200 ps;
end if;
end process;
end archi;

-------------------------------------------------------------------------------
--Designer: Wilfred A. Glover-Akpey
--Date: August 2008
--Course: CSE600
--Affiliation: The Pennsylvania State University
--Description: 8 bit flip flop, behavioral VHDL
-------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity wag8bitff is
Port ( D : in std_logic_vector(7 downto 0);
W : in std_logic;
Y : out std_logic_vector(7 downto 0));
end wag8bitff;

architecture Behavioral of wag8bitff is

begin

process (W)
begin
if (rising_edge(W)) then
Y <= D after 200 ps;
end if;
end process;
end Behavioral;

----------------------------------------------------------------------------------------------
--Designer: Wilfred A. Glover-Akpey
--Date: August 2008
--Course: CSE 600
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity wagpmux is
    port (
        a, b : in std_logic_vector(7 downto 0);
        i    : in std_logic;
        pmuxout : out std_logic_vector(7 downto 0));
end wagpmux;

architecture archi of wagpmux is
begin
    process (a, b, i)
    begin
        case i is
        when '0' => pmuxout <= a after 100 ps;
        when others => pmuxout <= b after 100 ps;
        end case;
    end process;
end archi;

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity wagdpmux is
    port (a, b : in std_logic_vector(7 downto 0);
          i : in std_logic;
          pmuxout : out std_logic_vector(7 downto 0));
end wagdpmux;
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;
--adr2 : out STD_LOGIC_VECTOR(7 downto 0);

entity wagDualPortRam is
    Port (
        w0 : in STD_LOGIC;
        dat0 : in STD_LOGIC_VECTOR(7 downto 0);
        adr0 : in STD_LOGIC_VECTOR(7 downto 0);
        dat2 : out STD_LOGIC_VECTOR(7 downto 0);
        adr1 : in STD_LOGIC_VECTOR(7 downto 0);
        dat1 : out STD_LOGIC_VECTOR(7 downto 0))
end wagDualPortRam;

architecture Behavioral of wagdpmux is
begin
    process (sel,a,b,c,d)
    begin
        case sel is
            when "00" => pmuxout <= a;
            when "01" => pmuxout <= b;
            when "10" => pmuxout <= c;
            when "11" => pmuxout <= d;
            when others => pmuxout <= a;
        end case;
    end process;
end Behavioral;

----------------------------------------------------------------------------------
-- Designer : Wilfred A. Glover-Akpey
-- Date : August 2008
-- Course : CSE 600
-- Affiliation : The Pennsylvania State University
-- Description : Dual-port RAM, structural VHDL
----------------------------------------------------------------------------------
end wagDualPortRam;

architecture Behavioral of wagDualPortRam is

COMPONENT wag4to16AddDec
    Port ( a : in std_logic_vector(3 downto 0);
          t : out std_logic_vector(15 downto 0));
END COMPONENT;

COMPONENT wagRam16X8D
    Port ( clk2    : in std_logic;
          we      : in std_logic;
          din     : in STD_LOGIC_VECTOR(7 downto 0);
          adr     : in  STD_LOGIC_VECTOR(3 downto 0);
          ra      : in  STD_LOGIC_VECTOR(3 downto 0);
          dpo     : out  STD_LOGIC_VECTOR(7 downto 0);
          spo     : out  STD_LOGIC_VECTOR(7 downto 0));
END COMPONENT;

COMPONENT wagoutputbuffer
    Port ( B : in std_logic_vector(7 downto 0);
          oe : in std_logic;
          D : out std_logic_vector(7 downto 0));
END COMPONENT;

COMPONENT wag16to1Mux is
    Port ( a : in std_logic_vector(7 downto 0);
          b : in std_logic_vector(7 downto 0);
          c : in std_logic_vector(7 downto 0);
          d : in std_logic_vector(7 downto 0);
          e : in std_logic_vector(7 downto 0);
          f : in std_logic_vector(7 downto 0);
          g : in std_logic_vector(7 downto 0);
          h : in std_logic_vector(7 downto 0);
          i : in std_logic_vector(7 downto 0);
          j : in std_logic_vector(7 downto 0);
          k : in std_logic_vector(7 downto 0);
          l : in std_logic_vector(7 downto 0);
          m : in std_logic_vector(7 downto 0);
          n : in std_logic_vector(7 downto 0);
          o : in std_logic_vector(7 downto 0);
          p : in std_logic_vector(7 downto 0);
          sel : in STD_LOGIC_VECTOR(3 downto 0);
          z   : out std_logic_vector(7 downto 0));
END COMPONENT;

SIGNAL Adr0SelSig1 : std_logic_vector(15 downto 0);
SIGNAL ddppoo_sig0,ddppoo_sig1,ddppoo_sig2,ddppoo_sig3, ddppoo_sig4,ddppoo_sig5
ddppoo_sig6 ,ddppoo_sig7 : std_logic_vector(7 downto 0);
SIGNAL ddppoo_sig8,ddppoo_sig9,ddppoo_sig10,ddppoo_sig11,ddppoo_sig12, ddppoo_sig13 ,ddppoo_sig14 ,ddppoo_sig15 : std_logic_vector(7 downto 0);

SIGNAL ssppoo_sig0, ssppoo_sig1 ,ssppoo_sig2 ,ssppoo_sig3, ssppoo_sig4 ,ssppoo_sig5 ,ssppoo_sig6 ,ssppoo_sig7 : std_logic_vector(7 downto 0);
SIGNAL ssppoo_sig8, ssppoo_sig9, ssppoo_sig10 ,ssppoo_sig11,ssppoo_sig12, ssppoo_sig13 ,ssppoo_sig14 ,ssppoo_sig15 : std_logic_vector(7 downto 0);

begin

AddDec1 : wag4to16AddDec Port map ( a =>adr0(7 downto 4) , t => Adr0SelSig1);

ram16X8D_0 : wagRam16X8D Port map( clk2 => w0,we => Adr0SelSig1(0), din =>dat0 ,adr => adr0(3 downto 0) ,ra =>adr1(3 downto 0), dpo => ddppoo_sig0, spo => ssppoo_sig0);
ram16X8D_1 : wagRam16X8D Port map( clk2 => w0,we => Adr0SelSig1(1), din =>dat0 ,adr => adr0(3 downto 0),ra => adr1(3 downto 0), dpo => ddppoo_sig1, spo => ssppoo_sig1);
ram16X8D_2 : wagRam16X8D Port map( clk2 => w0,we => Adr0SelSig1(2), din =>dat0 ,adr => adr0(3 downto 0),ra =>adr1(3 downto 0), dpo => ddppoo_sig2, spo => ssppoo_sig2);
ram16X8D_3 : wagRam16X8D Port map( clk2 => w0,we => Adr0SelSig1(3), din =>dat0 ,adr => adr0(3 downto 0),ra =>adr1(3 downto 0), dpo => ddppoo_sig3, spo => ssppoo_sig3);
ram16X8D_4 : wagRam16X8D Port map( clk2 => w0,we => Adr0SelSig1(4), din =>dat0 ,adr => adr0(3 downto 0),ra =>adr1(3 downto 0), dpo => ddppoo_sig4, spo => ssppoo_sig4);
ram16X8D_5 : wagRam16X8D Port map( clk2 => w0,we => Adr0SelSig1(5), din =>dat0 ,adr => adr0(3 downto 0),ra =>adr1(3 downto 0), dpo => ddppoo_sig5, spo => ssppoo_sig5);
ram16X8D_6 : wagRam16X8D Port map( clk2 => w0,we => Adr0SelSig1(6), din =>dat0 ,adr => adr0(3 downto 0),ra =>adr1(3 downto 0), dpo => ddppoo_sig6, spo => ssppoo_sig6);
ram16X8D_7 : wagRam16X8D Port map( clk2 => w0,we => Adr0SelSig1(7), din =>dat0 ,adr => adr0(3 downto 0),ra =>adr1(3 downto 0), dpo => ddppoo_sig7, spo => ssppoo_sig7);
ram16X8D_8 : wagRam16X8D Port map( clk2 => w0,we => Adr0SelSig1(8), din =>dat0 ,adr => adr0(3 downto 0),ra =>adr1(3 downto 0), dpo => ddppoo_sig8, spo => ssppoo_sig8);
ram16X8D_9 : wagRam16X8D Port map( clk2 => w0,we => Adr0SelSig1(9), din =>dat0 ,adr => adr0(3 downto 0),ra =>adr1(3 downto 0), dpo => ddppoo_sig9, spo => ssppoo_sig9);
ram16X8D_10 : wagRam16X8D Port map( clk2 => w0,we => Adr0SelSig1(10), din =>dat0 ,adr => adr0(3 downto 0),ra =>adr1(3 downto 0), dpo => ddppoo_sig10, spo => ssppoo_sig10);
ram16X8D_11 : wagRam16X8D Port map( clk2 => w0,we => Adr0SelSig1(11), din =>dat0 ,adr => adr0(3 downto 0),ra =>adr1(3 downto 0), dpo => ddppoo_sig11, spo => ssppoo_sig11);
Ram16X8D_12 : wagRam16X8D Port map(clk2 => w0, we => Adr0SelSig1(12), din => dat0, adr => adro(3 downto 0), ra => adro(3 downto 0), dpo => ddppoo_sig12, spo => ssppoo_sig12);
ram16X8D_13 : wagRam16X8D Port map(clk2 => w0, we => Adr0SelSig1(13), din => dat0, adr => adro(3 downto 0), ra => adro(3 downto 0), dpo => ddppoo_sig13, spo => ssppoo_sig13);
ram16X8D_14 : wagRam16X8D Port map(clk2 => w0, we => Adr0SelSig1(14), din => dat0, adr => adro(3 downto 0), ra => adro(3 downto 0), dpo => ddppoo_sig14, spo => ssppoo_sig14);
ram16X8D_15 : wagRam16X8D Port map(clk2 => w0, we => Adr0SelSig1(15), din => dat0, adr => adro(3 downto 0), ra => adro(3 downto 0), dpo => ddppoo_sig15, spo => ssppoo_sig15);

A16to1Mux_1 : wag16to1Mux Port map(a => ssppoo_sig0, b => ssppoo_sig1, c => ssppoo_sig2, d => ssppoo_sig3, e => ssppoo_sig4, f => ssppoo_sig5, g => ssppoo_sig6, h => ssppoo_sig7, i => ssppoo_sig8, j => ssppoo_sig9, k => ssppoo_sig10, l => ssppoo_sig11, m => ssppoo_sig12, n => ssppoo_sig13, o => ssppoo_sig14, p => ssppoo_sig15, sel => adro(7 downto 4), z => dat2);
A16to1Mux_2 : wag16to1Mux Port map(a => ddppoo_sig0, b => ddppoo_sig1, c => ddppoo_sig2, d => ddppoo_sig3, e => ddppoo_sig4, f => ddppoo_sig5, g => ddppoo_sig6, h => ddppoo_sig7, i => ddppoo_sig8, j => ddppoo_sig9, k => ddppoo_sig10, l => ddppoo_sig11, m => ddppoo_sig12, n => ddppoo_sig13, o => ddppoo_sig14, p => ddppoo_sig15, sel => adro(7 downto 4), z => dat1);

end Behavioral;

----------------------------------------------------------------------------------------
-- Designer : Wilfred A. Glover-Akpey
-- Date : August 2008
-- Course : CSE 600
-- Affiliation : The Pennsylvania State University
-- Description : 8-bit Register
----------------------------------------------------------------------------------------
-- n-bit Register (ESD book figure 2.6)
-- by Weijun Zhang, 04/2001
--
-- KEY WORD: concurrent, generic and range
----------------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
----------------------------------------------------------------------------------------
entity wag8bitRegister is
port(
  IR : in std_logic_vector(7 downto 0);
  WIR : in std_logic;
  LOAD : in std_logic;
  RIR : in std_logic;
  IROUT : out std_logic_vector(7 downto 0));
end wag8bitRegister;

architecture Behavioral of wag8bitRegister is

signal IROUT_tmp: std_logic_vector(7 downto 0);

begin

  process(IR, WIR, LOAD, RIR)
  begin
    if RIR = '1' then
      -- use 'range in signal assignment
      IROUT_tmp <= (IROUT_tmp'range => '0');
    elsif (WIR='1' and WIR'event) then
      if LOAD = '1' then
        IROUT_tmp <= IR;
      end if;
    end if;
  end process;

  IROUT <= IROUT_tmp;

end Behavioral;

--Designer: Wilfred A. Glover-Akpey
--Date: August 2008
--Course: CSE600
--Affiliation: The Pennsylvania State University
--Description: Micro controller instruction decoder (ID)

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity wagid is
  Port ( CSM : in std_logic;
          RSM : in std_logic;
          CLK_INV : in std_logic;
          CF : in std_logic:='0';
        );
end wagid;
INTreq : in std_logic;
IR : in std_logic_vector(7 downto 0);
sel : out std_logic_vector(1 downto 0);
pres : out std_logic; --
pck : out std_logic; --
bx : out std_logic; --
rir : out std_logic; --
wir : out std_logic); --
end wagid;

architecture Behavioral of wagid is

COMPONENT wag1inv
    Port(a:in std_logic;
z:out std_logic);
END COMPONENT;

COMPONENT wag1and2
    Port ( a : in std_logic;
b : in std_logic;
z : out std_logic);
END COMPONENT;

COMPONENT wag1and3
    Port ( a : in std_logic;
b : in std_logic;
c : in std_logic;
z : out std_logic);
END COMPONENT;

COMPONENT wag1or2
    Port ( a : in std_logic;
b : in std_logic;
z : out std_logic);
END COMPONENT;

COMPONENT wagdelay2
    Port ( a : in std_logic;
z : out std_logic);
END COMPONENT;

COMPONENT wagor3
    Port ( a : in std_logic;
b : in std_logic;
c : in std_logic;
z : out std_logic);
END COMPONENT;
SIGNAL RSM_n,bx_sig1,bx_sig2 : std_logic;
SIGNAL IR7,IR3,IR2,IR1,IR0 : std_logic; -- Bits to decode from the instruction
SIGNAL CFsig : std_logic;

begin

process (INTreq, IR(0)) is
begin
    if INTreq = '1' and IR(0) = '0' then
        sel <= "11"; -- go to interrupt service routine
    elsif (INTreq = '0' and IR(0) = '1' and IR(7) = '0' and
           IR(1) = '0' and IR(1) = '0') then
        sel <= "00"; -- RTI
    else
        sel <= "10"; -- normal mode
    end if;
end process;

-- Setup the opcode lines IR7
IR7 <= IR(7);
IR3 <= IR(3);
IR0 <= IR(0);
IR2 <= IR(2);
IR1 <= IR(1);

-- This section produces the CPC output
pck <= CSM;

gate1 : wag1inv PORT MAP(a=>RSM, z=>RSM_n);
pres <= RSM_n;

-- This block produces the WIR output
wir <= CLK_INV;

-- This block produces the RIR output
RIR <= RSM_n;
-- This block produces the BX output for BC
-----------------------------------------

gate16_BC : wag1and2 PORT MAP(a=>IR7, b=>CFsig, z=>bx_sig1);

-----------------------------------------
-- This block produces the BX output for BA
-----------------------------------------

CFsig <= '0' when CF = 'U' else CF;

-----------------------------------------
-- This block produces the BX output for BA
-----------------------------------------

JSR_gate : wag1and2 PORT MAP(a=>IR7, b=>IR3,z=>bx_sig2);
or_gate  : wag1or2 port map (bx_sig1,bx_sig2,bx);

end Behavioral;

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity InstROM is
  Port ( InstAddress : in std_logic_vector(7 downto 0);
         dat2Out    : out std_logic_vector(7 downto 0);
         dat1Out    : out std_logic_vector(7 downto 0);
         dat0Out    : out std_logic_vector(7 downto 0));
end InstROM;

architecture Behavioral of InstROM is

SIGNAL InstOut : std_logic_vector(23 downto 0);

type ROM_Array is array (0 to 255) of std_logic_vector(23 downto 0);
constant Content: ROM_Array := ( 
B"0000000000010100001010", -- NOP 10, 10
B"001000000001000110010", -- MV #33, 50 -- Testing Logic instruction
B"0010000000101100001110", -- MV #44, 60
B"0010000001000001001100", -- MV #5, 52
B"00000110001100011001001", -- AND 50, 52
B"0000010000111000111110", -- OR 60, 62
B"0010000000101000101001", -- MV #22, 40
B"001000000001011000101000", -- MV #66, 80
B"000001110010100000101000", -- XOR #80, 80
B"001000000011110101000110", -- MV #55, 70
B"001000010010001001100011", -- XNOR 70, 70
B"001000000010100010100110", -- MV #21, 30
B"001000000001010100010110", -- MV #6, 13
B"01000000000001010000000", -- SUB 90, 91
B"01000000000000110000001", -- ADD 5, 70
B"01000000000001100000000", -- OUT 70
B"01000000000000110000000", -- SUB 90, 91
B"10000000000000000000000", -- BC -7 -- end of sample program

begin

InstOut <= Content(conv_integer(InstAddress)) after 100 ps;

dat2Out <= InstOut(23 downto 16) after 100 ps;
dat1Out <= InstOut(15 downto 8) after 100 ps;
dat0Out <= InstOut(7 downto 0) after 100 ps;

end Behavioral;
library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;

entity wagDCM is
  port ( CLKIN_IN    : in    std_logic;
          RST_IN     : in    std_logic;
          CLKFX_OUT : out   std_logic;
          CLKFX180_OUT : out   std_logic;
          CLkin_IBUFG_OUT : out   std_logic;
          CLK0_OUT : out   std_logic;
          CLK180_OUT : out   std_logic;
          LOCKED_OUT : out   std_logic);
end wagDCM;

architecture BEHAVIORAL of wagDCM is
  signal CLKFB_IN    : std_logic;
  signal CLKFX_BUF   : std_logic;
  signal CLKFX180_BUF : std_logic;
  signal CLkin_IBUFG : std_logic;
  signal CLK0_BUF    : std_logic;
  signal CLK180_BUF  : std_logic;
  signal GND_BIT     : std_logic;
  begin
    GND_BIT <= '0';
    CLkin_IBUFG_OUT <= CLkin_IBUFG;
    CLK0_OUT <= CLKFB_IN;
    CLKFX_BUF_INST : BUFG
      port map (I=>CLKFX_BUF,
                O=>CLKFX_OUT);
    CLKFX180_BUF_INST : BUFG
      port map (I=>CLKFX180_BUF,
                O=>CLKFX180_OUT);
    CLkin_IBUFG_INST : IBUF
      port map (I=>CLkin_IN,
                O=>CLkin_IBUFG);
    CLK0_BUF_INST : BUFG
      port map (I=>CLK0_BUF,
O=>CLKFB_IN);

CLK180_BUFG_INST : BUFG
port map (I=>CLK180_BUF,
O=>CLK180_OUT);

DCM_INST : DCM
generic map( CLK_FEEDBACK => "1X",
CLKDIV_DIVIDE => 2.0,
CLKFX_DIVIDE => 20,
CLKFX_MULTIPLY => 20,
CLKIN_DIVIDE_BY_2 => FALSE,
CLKIN_PERIOD => 50.000,
CLKOUT_PHASE_SHIFT => "NONE",
DESKEW_ADJUST => "SYSTEM_SYNCHRONOUS",
DFS_FREQUENCY_MODE => "LOW",
DLL_FREQUENCY_MODE => "LOW",
DUTY_CYCLE_CORRECTION => TRUE,
FACTORY_JF => x"8080",
PHASE_SHIFT => 0,
STARTUP_WAIT => FALSE)
port map (CLKFB=>CLKFB_IN,
CLKIN=>CLKIN_IBUFG,
DSSEN=>GND_BIT,
PSCLK=>GND_BIT,
PSEN=>GND_BIT,
PSINCDEC=>GND_BIT,
RST=>RST_IN,
CLKDV=>open,
CLKFX=>CLKFX_BUF,
CLKFX180=>CLKFX180_BUF,
CLK0=>CLK0_BUF,
CLK2X=>open,
CLK2X180=>open,
CLK90=>open,
CLK180=>CLK180_BUF,
CLK270=>open,
LOCKED=>LOCKED_OUT,
PSDONE=>open,
STATUS=>open);

end BEHAVIORAL;

-- Copyright (c) 1995-2007 Xilinx, Inc.  All rights reserved.
---------------------------------------------------------------------
-- ___ ___
library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;

entity adc2tx01 is
  port ( ckin : in std_logic;
          Logic1 : in std_logic;
          rxin : in std_logic;
          txp : in std_logic_vector (7 downto 0);
          ckout1p8m : out std_logic;
          ckout8k : out std_logic;
          ckout20m : out std_logic;
          rxo : out std_logic_vector (7 downto 0);
          txout : out std_logic);
end adc2tx01;

architecture BEHAVIORAL of adc2tx01 is
  attribute IOSTANDARD : string := "33k";
  attribute SLEW : string := "medium";
  attribute DRIVE : string := "medium";
  attribute BOX_TYPE : string := "minimal";
  signal XLXN_219 : std_logic;
  signal XLXN_221 : std_logic;
  signal XLXN_223 : std_logic;
  signal XLXN_243 : std_logic;
  signal XLXN_245 : std_logic;
  signal XLXN_272 : std_logic;
component OBUF
  port ( I : in std_logic;
O : out std_logic);
end component;
attribute IOSTANDARD of OBUF : component is "DEFAULT";
attribute SLEW of OBUF : component is "SLOW";
attribute DRIVE of OBUF : component is "12";
attribute BOX_TYPE of OBUF : component is "BLACK_BOX";

component fdiv11v1
  port ( cki : in std_logic;
        dvcko : out std_logic);
end component;

component fdiv1250v1
  port ( ckin : in std_logic;
        dvcko : out std_logic);
end component;

component tx
  port ( txpdi : in std_logic_vector (7 downto 0);
        txck16i : in std_logic;
        txwi : in std_logic;
        txsdo : out std_logic;
        txfo : out std_logic;
        txfwro : out std_logic);
end component;

component rx
  port ( rxsdi : in std_logic;
        rxck16i : in std_logic;
        rxwi : in std_logic;
        rxpdo : out std_logic_vector (7 downto 0);
        rxfo : out std_logic);
end component;

component IBUF
  port ( I : in std_logic;
         O : out std_logic);
end component;
attribute IOSTANDARD of IBUF : component is "DEFAULT";
attribute BOX_TYPE of IBUF : component is "BLACK_BOX";

component AND2
  port ( I0 : in std_logic;
        I1 : in std_logic;
        O : out std_logic);
end component;
attribute BOX_TYPE of AND2 : component is "BLACK_BOX";

begin
XLXI_3 : OBUF
    port map (I=>ckin,
              O=>ckout20m);

XLXI_4 : OBUF
    port map (I=>XLXN_272,
              O=>txout);

XLXI_5 : OBUF
    port map (I=>XLXN_245,
              O=>ckout8k);

XLXI_6 : OBUF
    port map (I=>XLXN_223,
              O=>ckout1p8m);

XLXI_105 : fdiv11v1
    port map (cki=>ckin,
              dvcko=>XLXN_223);

XLXI_106 : fdiv1250v1
    port map (ckin=>ckin,
              dvcko=>XLXN_245);

XLXI_169 : tx
    port map (txck16i=>XLXN_223,
              txpdi(7 downto 0)=>txp(7 downto 0),
              txwi=>XLXN_243,
              txfo=>open,
              txfwro=>open,
              txsdo=>XLXN_272);

XLXI_197 : rx
    port map (rxck16i=>XLXN_223,
              rxsdi=>XLXN_219,
              rxwi=>XLXN_221,
              rxfo=>XLXN_221,
              rxdpdo(7 downto 0)=>rxo(7 downto 0));

XLXI_198 : IBUF
    port map (I=>rxin,
              O=>XLXN_219);

XLXI_202 : AND2
    port map (I0=>XLXN_245,
              I1=>Logic1,
              O=>XLXN_243);

end BEHAVIORAL;
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity wag1add8bit is
  Port(
      ci: in std_logic;
      a : in std_logic_vector(7 downto 0);
      b : in std_logic_vector(7 downto 0);
      s : out std_logic_vector(7 downto 0);
      co : out std_logic);
end wag1add8bit;

architecture struc of wag1add8bit is

component wag2fa
  Port(
      a : in std_logic;
      b : in std_logic;
      ci : in std_logic;
      s : out std_logic;
      co : out std_logic);
end component;

SIGNAL X0:STD_LOGIC;
SIGNAL X1:STD_LOGIC;
SIGNAL X2:STD_LOGIC;
SIGNAL X3:STD_LOGIC;
SIGNAL X4:STD_LOGIC;
SIGNAL X5:STD_LOGIC;
SIGNAL X6:STD_LOGIC;

begin

fa7 : wag2fa PORT MAP(a=>a(7), b=>b(7), ci=>X0, s=>s(7), co=>co);
fa6 : wag2fa PORT MAP(a=>a(6), b=>b(6), ci=>X1, s=>s(6), co=>X0);
fa5 : wag2fa PORT MAP(a=>a(5), b=>b(5), ci=>X2, s=>s(5), co=>X1);

end;
fa4 : wag2fa PORT MAP(a=>a(4), b=>b(4), ci=>X3, s=>s(4), co=>X2);
fa3 : wag2fa PORT MAP(a=>a(3), b=>b(3), ci=>X4, s=>s(3), co=>X3);
fa2 : wag2fa PORT MAP(a=>a(2), b=>b(2), ci=>X5, s=>s(2), co=>X4);
fa1 : wag2fa PORT MAP(a=>a(1), b=>b(1), ci=>X6, s=>s(1), co=>X5);
fa0 : wag2fa PORT MAP(a=>a(0), b=>b(0), ci=>ci, s=>s(0), co=>X6);
end struc;

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity wagxor2 is
  Port ( a : in std_logic;
         b : in std_logic;
         z : out std_logic);
end wagxor2;

architecture Behavioral of wagxor2 is
begin
  z <= (a xor b) after 171 ps;
end Behavioral;
entity wagCarry4Mux is
  port ( a,b,c,d : in std_logic;
        opcode_2 : in std_logic_vector(7 downto 0);
        pmuxout : out std_logic);
end wagCarry4Mux;
architecture Behavioral of wagCarry4Mux is
begin
  process (opcode_2,a,b,c,d)
  begin
    if (opcode_2="00010000" or opcode_2 ="00011000") then -- add/sub
      pmuxout<=a after 100 ps;
    elsif opcode_2="00011010" then -- logical shift left shl
      pmuxout<=b after 100 ps;
    elsif opcode_2="00011100" then -- logical shift right - shr
      pmuxout<=c after 100 ps;
    else
      pmuxout<=d after 100 ps;
    end if;
  end process;
end Behavioral;

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

end Behavioral;

--Designer: Wilfred A. Glover-Akpey
--Date: August 2008
--Course: CSE600
--Affiliation: The Pennsylvania State University
--Description: 8-bit inverter
entity wagInvAll is
  Port ( x : in  STD_LOGIC_VECTOR(7 downto 0);
        y : out  STD_LOGIC_VECTOR(7 downto 0));
end wagInvAll;

architecture Behavioral of wagInvAll is

component wag1inv
  Port (a:in std_logic;
        z:out std_logic);
end component;

begin
  InvAll0 : wag1inv Port Map (a =>x(0) , z=>y(0));
  InvAll1 : wag1inv Port Map (a =>x(1) , z=>y(1));
  InvAll2 : wag1inv Port Map (a =>x(2) , z=>y(2));
  InvAll3 : wag1inv Port Map (a =>x(3) , z=>y(3));
  InvAll4 : wag1inv Port Map (a =>x(4) , z=>y(4));
  InvAll5 : wag1inv Port Map (a =>x(5) , z=>y(5));
  InvAll6 : wag1inv Port Map (a =>x(6) , z=>y(6));
  InvAll7 : wag1inv Port Map (a =>x(7) , z=>y(7));
end Behavioral;

-----------------------------------------------------------------------------------
-- Designer: Wilfred A. Glover-Akpey
-- Date: August 2008
-- Course: CSE600
-- Affiliation: The Pennsylvania State University
-- Description: 8-bit XOR
-----------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity wag8bitXor is
  Port ( x : in  STD_LOGIC_VECTOR(7 downto 0);
         y : in  STD_LOGIC_VECTOR(7 downto 0);
         z : out  STD_LOGIC_VECTOR(7 downto 0));
end wag8bitXor;
architecture Behavioral of wag8bitXor is

component wagxor2 is
  Port (a : in std_logic;
       b : in std_logic;
       z : out std_logic);
end component;

begin

  bitXor0 : wagxor2 Port Map (a => x(0), b => y(0), z => z(0));
  bitXor1 : wagxor2 Port Map (a => x(1), b => y(1), z => z(1));
  bitXor2 : wagxor2 Port Map (a => x(2), b => y(2), z => z(2));
  bitXor3 : wagxor2 Port Map (a => x(3), b => y(3), z => z(3));
  bitXor4 : wagxor2 Port Map (a => x(4), b => y(4), z => z(4));
  bitXor5 : wagxor2 Port Map (a => x(5), b => y(5), z => z(5));
  bitXor6 : wagxor2 Port Map (a => x(6), b => y(6), z => z(6));
  bitXor7 : wagxor2 Port Map (a => x(7), b => y(7), z => z(7));

end Behavioral;

--------------------------------------------------------------------------------

Designer: Wilfred A. Glover-Akpey
Date: August 2008
Course: CSE600
Affiliation: The Pennsylvania State University
Description: 8-input MUX
----------------------------------------------------------------------------------
process (a,b,c,d,e,f,g,h,opcode_2) is
begin
  if (opcode_2="00010000" or opcode_2 ="00011000") then -- add/sub
    pmuxout<=a after 100 ps;
  elsif opcode_2 ="00011010" then -- logical shift left shl
    pmuxout<=b after 100 ps;
  elsif opcode_2 ="00000110" then -- and
    pmuxout<=c after 100 ps;
  elsif opcode_2 ="00011100" then -- logical shift right shr
    pmuxout<=d after 100 ps;
  elsif opcode_2 ="00000100" then -- or
    pmuxout<=e after 100 ps;
  elsif opcode_2 ="00000111" then -- not
    pmuxout<=f after 100 ps;
  elsif opcode_2 ="00000010" then -- xor
    pmuxout<=g after 100 ps;
  else
    pmuxout<=h after 100 ps; -- do nothing;
  end if;
end process;
end Behavioral;

--Designer: Wilfred A. Glover-Akpey
--Date: August 2008
--Course: CSE600
--Affiliation: The Pennsylvania State University
--Description: 8-bit LSR

library IEEE;
use IEEE.numeric_std;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity wag8bitlsr is
  Port ( x : in STD_LOGIC_VECTOR(7 downto 0);
    Cin : in STD_LOGIC;
    Cout : out STD_LOGIC;
  );
architecture Behavioral of wag8bitlsr is

begin

process(Cin, x)
begin
if Cin = '1' then
    z(7) <= '1'; -- shift a one into LSB
    z(6 downto 0) <= x(7 downto 1);
    Cout <= x(0) after 100 ps;
else
    z(7) <= '0'; -- shift a zero into LSB
    z(6 downto 0) <= x(7 downto 1);
    Cout <= x(0) after 100 ps;
end if;
end process;

end Behavioral;

library IEEE;
use IEEE.numeric_std;
--use IEEE.numeric_unsigned.all;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

--Uncomment the following library declaration if instantiating
--any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity wag8bitlsr is
  Port ( x : in STD_LOGIC_VECTOR(7 downto 0);
         Cin : in STD_LOGIC;
         Cout : out STD_LOGIC;
         z : out STD_LOGIC_VECTOR(7 downto 0));
end wag8bitlsr;
architecture Behavioral of wag8bitAnd is

begin
    process(Cin, x)
    begin
        if Cin = '1' then
            z(0) <= '1'; -- shift a one into LSB
            z(7 downto 1) <= x(6 downto 0);
            Cout <= x(7);
        else
            z(0) <= '0'; -- shift a zero into LSB
            z(7 downto 1) <= x(6 downto 0);
            Cout <= x(7);
        end if;
    end process;
end Behavioral;

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity wag8bitAnd is
    Port ( x : in STD_LOGIC_VECTOR(7 downto 0);
            y : in STD_LOGIC_VECTOR(7 downto 0);
            z : out STD_LOGIC_VECTOR(7 downto 0));
end wag8bitAnd;

architecture Behavioral of wag8bitAnd is

component wag1and2 is
    Port ( a : in std_logic;
            b : in std_logic;
            out : out std_logic);
end component;

The Designer is
Wilfred A. Glover-Akpey

The Date is
August 2008

The Course is
CSE600

The Affiliation is
The Pennsylvania State University

The Description is
8-bit AND
b : in std_logic;
z : out std_logic);
end component;

begin

bitand0 : wag1and2 Port Map (a => x(0), b => y(0), z => z(0));
bitand1 : wag1and2 Port Map (a => x(1), b => y(1), z => z(1));
bitand2 : wag1and2 Port Map (a => x(2), b => y(2), z => z(2));
bitand3 : wag1and2 Port Map (a => x(3), b => y(3), z => z(3));
bitand4 : wag1and2 Port Map (a => x(4), b => y(4), z => z(4));
bitand5 : wag1and2 Port Map (a => x(5), b => y(5), z => z(5));
bitand6 : wag1and2 Port Map (a => x(6), b => y(6), z => z(6));
bitand7 : wag1and2 Port Map (a => x(7), b => y(7), z => z(7));
end Behavioral;

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

/* Uncomment the following library declaration if instantiating
any Xilinx primitives in this code. */
library UNISIM;
use UNISIM.VComponents.all;

entity wag8bitOr is
    Port ( x : in STD_LOGIC_VECTOR(7 downto 0);
           y : in STD_LOGIC_VECTOR(7 downto 0);
           z : out STD_LOGIC_VECTOR(7 downto 0));
end wag8bitOr;

architecture Behavioral of wag8bitOr is

component wag1or2 is
    Port ( a : in std_logic;
           b : in std_logic;
           z : out std_logic);
end component;

begin


end component;

begin

  bitor0 : wag1or2 Port Map (a => x(0), b => y(0), z => z(0));
  bitor1 : wag1or2 Port Map (a => x(1), b => y(1), z => z(1));
  bitor2 : wag1or2 Port Map (a => x(2), b => y(2), z => z(2));
  bitor3 : wag1or2 Port Map (a => x(3), b => y(3), z => z(3));
  bitor4 : wag1or2 Port Map (a => x(4), b => y(4), z => z(4));
  bitor5 : wag1or2 Port Map (a => x(5), b => y(5), z => z(5));
  bitor6 : wag1or2 Port Map (a => x(6), b => y(6), z => z(6));
  bitor7 : wag1or2 Port Map (a => x(7), b => y(7), z => z(7));

end Behavioral;

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity wag2fa is
  Port(
    a: in std_logic;
    b: in std_logic;
    ci: in std_logic;
    s: out std_logic;
    co: out std_logic);
end wag2fa;

architecture struc of wag2fa is

begin

  process(a, b, ci)
  begin
    s <= a xor b xor ci after 200 ps;
  end process;

end;
co<= (a and b) or (ci and b) or (a and ci) after 200 ps;
end process;
end struct;

-----------------------------------------------------------------------------------------
--Designer: Wilfred A. Glover-Akpey
--Date: August 2008
--Course: CSE471
--Affiliation: The Pennsylvania State University
--Description: Inverter, 1bit, behavioral VHDL

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity wag1inv is
  Port( a:in std_logic;
       z:out std_logic);
end wag1inv;

architecture Behavioral of wag1inv is
begin
  z <= (not a) after 101 ps;--
end Behavioral;

-----------------------------------------------------------------------------------------
--Designer: Wilfred A. Glover-Akpey
--Date: August 2008
--Course: CSE600
--Affiliation: The Pennsylvania State University
--Description: 2-Input OR gate

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity wag1or2 is
  Port( a : in std_logic;
        b : in std_logic;
        z : out std_logic)
end wag1or2;
entity wag1or2 is
  Port ( a : in std_logic;
        b : in std_logic;
        z : out std_logic);
end wag1or2;

architecture Behavioral of wag1or2 is
begin
  z <= (a or b) after 121 ps;
end Behavioral;

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity wag1and2 is
  Port ( a : in std_logic;
         b : in std_logic;
         z : out std_logic);
end wag1and2;

architecture Behavioral of wag1and2 is
begin
  z <= (a and b) after 141 ps;
end Behavioral;

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity wag1and2 is
  Port ( a : in std_logic;
         b : in std_logic;
         z : out std_logic);
end wag1and2;

architecture Behavioral of wag1and2 is
begin
  z <= (a and b) after 141 ps;
end Behavioral;

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity wagLookupTable is
  Port (CLK2 : in std_logic;
        IPCODE : in std_logic_vector (7 downto 0);
        MDW : out std_logic;
        DX0 : out std_logic;
        DX1 : out std_logic);
end wagLookupTable;

architecture Behavioral of wagLookupTable is

  SIGNAL dx1_sig : std_logic;
  SIGNAL dx0_sig : std_logic;
  SIGNAL mdw_sig : std_logic;

begin

  process (CLK2,IPCODE) is
  begin
    if (IPCODE="00100000") then dx1_sig <='1'; -- mv#
elsif (IPCODE="00000000" and CLK2='1') then dx1_sig <='0'; -- mv / nop
elsif (IPCODE="00000000" and CLK2='0') then dx1_sig <='0'; -- mv
elsif (IPCODE="01000000") then dx1_sig <='1'; -- in
elsif (IPCODE="00010000" and CLK2='1') then dx1_sig <='0'; -- add
elsif (IPCODE="00010000" and CLK2='0') then dx1_sig <='0'; -- add
elsif (IPCODE="00011000" and CLK2='0') then dx1_sig <='0'; -- sub
elsif (IPCODE="00111010" and CLK2='0') then dx1_sig <='0'; -- sub
elsif (IPCODE="00000110" and CLK2='0') then dx1_sig <='0'; -- AND
elsif (IPCODE="00000010" and CLK2='0') then dx1_sig <='0'; -- OR
elsif (IPCODE="00000100" and CLK2='0') then dx1_sig <='0'; -- OR
elsif (IPCODE="00000000" and CLK2='1') then dx1_sig <='0'; -- XOR
elsif (IPCODE="00000010" and CLK2='0') then dx1_sig <='0'; -- XOR
elsif (IPCODE="00000100" and CLK2='0') then dx1_sig <='0'; -- SHR
elsif (IPCODE="00001100" and CLK2='0') then dx1_sig <='0'; -- SHR
elsif (IPCODE="00111010" and CLK2='0') then dx1_sig <='0'; -- SHRC
elsif (IPCODE="00111100" and CLK2='0') then dx1_sig <='0'; -- SHRC
  end process;

end Behavioral;
elsif (IPCODE="00111010" and CLK2='1') then dx1_sig <='0'; -- SHL
elsif (IPCODE="00011010" and CLK2='0') then dx1_sig <='0'; -- SHL
elsif (IPCODE="00011000" and CLK2='1') then dx1_sig <='0'; -- SHLC
elsif (IPCODE="00011000" and CLK2='0') then dx1_sig <='0'; -- SHLC
elsif (IPCODE="00000111" and CLK2='1') then dx1_sig <='0'; -- NOT
elsif (IPCODE="00000111" and CLK2='0') then dx1_sig <='0'; -- NOT
elsif (IPCODE="10000000") then dx1_sig <='0'; -- bc
elsif (IPCODE="01001000") then dx1_sig <='0'; -- out
elsif (IPCODE="10001000") then dx1_sig <='0'; -- ba
else dx1_sig <='0';
end if;
end process;

process (IPCODE, CLK2) is
begin
if (IPCODE="00100000") then dx0_sig <='0'; -- mv#
elsif (IPCODE="01000000") then dx0_sig <='1'; -- in
elsif (IPCODE="00000000" and CLK2='1') then dx0_sig <='0'; -- mv / nop
elsif (IPCODE="00000000" and CLK2='0') then dx0_sig <='0'; -- mv
elsif (IPCODE="00010000" and CLK2='1') then dx0_sig <='1'; -- add
elsif (IPCODE="00010000" and CLK2='0') then dx0_sig <='1'; -- add
elsif (IPCODE="00011000" and CLK2='1') then dx0_sig <='1'; -- sub
elsif (IPCODE="00011000" and CLK2='0') then dx0_sig <='1'; -- sub
elsif (IPCODE="00000110" and CLK2='1') then dx0_sig <='1'; -- AND
elsif (IPCODE="00000110" and CLK2='0') then dx0_sig <='1'; -- AND
elsif (IPCODE="00000100" and CLK2='1') then dx0_sig <='1'; -- OR
elsif (IPCODE="00000100" and CLK2='0') then dx0_sig <='1'; -- OR
elsif (IPCODE="00000010" and CLK2='1') then dx0_sig <='1'; -- XOR
elsif (IPCODE="00000010" and CLK2='0') then dx0_sig <='1'; -- XOR
elsif (IPCODE="00011100" and CLK2='1') then dx0_sig <='1'; -- SHR
elsif (IPCODE="00011100" and CLK2='0') then dx0_sig <='1'; -- SHR
elsif (IPCODE="00111010" and CLK2='1') then dx0_sig <='1'; -- SHL
elsif (IPCODE="00111010" and CLK2='0') then dx0_sig <='1'; -- SHLC
elsif (IPCODE="00011100" and CLK2='0') then dx0_sig <='1'; -- SHRC
elsif (IPCODE="00011010" and CLK2='1') then dx0_sig <='1'; -- SHL
elsif (IPCODE="00011010" and CLK2='0') then dx0_sig <='1'; -- SHLC
elsif (IPCODE="00011000" and CLK2='1') then dx0_sig <='1'; -- SHLC
elsif (IPCODE="00011000" and CLK2='0') then dx0_sig <='1'; -- SHRC
elsif (IPCODE="00000111" and CLK2='1') then dx0_sig <='1'; -- NOT
elsif (IPCODE="00000111" and CLK2='0') then dx0_sig <='1'; -- NOT
elsif (IPCODE="10000000") then dx0_sig <='0'; -- bc
elsif (IPCODE="01001000") then dx0_sig <='0'; -- out
elsif (IPCODE="10001000") then dx0_sig <='0'; -- ba
else dx0_sig <='0';
end if;
end process;

process (IPCODE, CLK2) is
begin

if (IPCODE="00100000" and CLK2='0') then mdw_sig <='1'; -- mv#
elsif (IPCODE="00100000" and CLK2='1') then mdw_sig <='0'; -- mv#
elsif (IPCODE="01000000" and CLK2='0') then mdw_sig <='1'; -- in
elsif (IPCODE="10000000") then mdw_sig <='0'; -- bc
elsif (IPCODE="01001000") then mdw_sig <='0'; -- out
elsif (IPCODE="10001000") then mdw_sig <='0'; -- ba
elsif (IPCODE="00000000" or IPCODE="00010000" or IPCODE="00011000")
and (CLK2='0') then mdw_sig <='1'; --add or sub or mv / nop
elsif (IPCODE="00000110" or IPCODE="00000010" or IPCODE="00000110")
and (CLK2='0') then mdw_sig <='1'; -- AND or OR or XOR
elsif (IPCODE="00011100" or IPCODE="00111010") and (CLK2='0') then
mdw_sig <='1'; -- SHR or SHRC
elsif (IPCODE="00011010" or IPCODE="00011000") and (CLK2='0') then
mdw_sig <='1'; -- SHL or SHLC
elsif (IPCODE="00000111") and (CLK2='0') then mdw_sig <='1'; -- NOT
else mdw_sig <='0';
end if;
end process;

DX1 <= dx1_sig;
DX0 <= dx0_sig;
MDW <= mdw_sig;

end Behavioral;

--Designer: Wilfred A. Glover-Akpey
--Date: August 2008
--Course: CSE600
--Affiliation: The Pennsylvania State University
--Description: 4by16 Address Decoder

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

library UNISIM;
use UNISIM.VComponents.all;

entity wag4to16AddDec is
    Port ( a : in std_logic_vector(3 downto 0);
           t : out std_logic_vector(15 downto 0));
end wag4to16AddDec;

architecture Behavioral of wag4to16AddDec is

COMPONENT wag1and4
    Port ( a : in std_logic;
           b : in std_logic;
           c : in std_logic;
           d : in std_logic;
           z : out std_logic);
END COMPONENT;

COMPONENT wag1inv
    Port(a:in std_logic;
         z:out std_logic);
END COMPONENT;
SIGNAL a_n : std_logic_vector(3 downto 0);

begin

inv3 : wag1inv PORT MAP(a=>a(3), z=>a_n(3));
inv2 : wag1inv PORT MAP(a=>a(2), z=>a_n(2));
inv1 : wag1inv PORT MAP(a=>a(1), z=>a_n(1));
inv0 : wag1inv PORT MAP(a=>a(0), z=>a_n(0));
p00 : wag1and4 PORT MAP (a=>a_n(3), b=>a_n(2), c=>a_n(1), d=>a_n(0), z=>t(0));
p01 : wag1and4 PORT MAP (a=>a_n(3), b=>a_n(2), c=>a_n(1), d=>a(0), z=>t(1));
p02 : wag1and4 PORT MAP (a=>a_n(3), b=>a_n(2), c=>a(1), d=>a_n(0), z=>t(2));
p03 : wag1and4 PORT MAP (a=>a_n(3), b=>a_n(2), c=>a(1), d=>a(0), z=>t(3));
p04 : wag1and4 PORT MAP (a=>a_n(3), b=>a(2), c=>a_n(1), d=>a_n(0), z=>t(4));
p05 : wag1and4 PORT MAP (a=>a_n(3), b=>a(2), c=>a(1), d=>a(0), z=>t(5));
p06 : wag1and4 PORT MAP (a=>a_n(3), b=>a(2), c=>a(1), d=>a(0), z=>t(6));
p07 : wag1and4 PORT MAP (a=>a_n(3), b=>a(2), c=>a(1), d=>a(0), z=>t(7));
p08 : wag1and4 PORT MAP (a=>a(3), b=>a_n(2), c=>a_n(1), d=>a_n(0), z=>t(8));
p09 : wag1and4 PORT MAP (a=>a(3), b=>a_n(2), c=>a_n(1), d=>a(0), z=>t(9));
p10 : wag1and4 PORT MAP (a=>a(3), b=>a_n(2), c=>a(1), d=>a_n(0), z=>t(10));
p11 : wag1and4 PORT MAP (a=>a(3), b=>a_n(2), c=>a(1), d=>a(0), z=>t(11));
p12 : wag1and4 PORT MAP (a=>a(3), b=>a(2), c=>a_n(1), d=>a_n(0), z=>t(12));
p13 : wag1and4 PORT MAP (a=>a(3), b=>a(2), c=>a_n(1), d=>a(0), z=>t(13));
p14 : wag1and4 PORT MAP (a=>a(3), b=>a(2), c=>a(1), d=>a_n(0), z=>t(14));
p15 : wag1and4 PORT MAP (a=>a(3), b=>a(2), c=>a(1), d=>a(0), z=>t(15));

end Behavioral;

------------------------------------------------------------------
--Designer:  Wilfred A. Glover-Akpey
--Date:     August 2008
--Course:   CSE600
--Affiliation:  The Pennsylvania State University
--Description:  16by8 Word RAM
------------------------------------------------------------------
we : in std_logic;
din : in STD_LOGIC_VECTOR(7 downto 0);
adr : in STD_LOGIC_VECTOR(3 downto 0);
ra : in STD_LOGIC_VECTOR(3 downto 0);
dpo : out STD_LOGIC_VECTOR(7 downto 0);
spo : out STD_LOGIC_VECTOR(7 downto 0));
end wagRam16X8D;

architecture Behavioral of wagRam16X8D is

component wagRam16X4D
  Port (clk1 : in std_logic;
        we : in std_logic;
        Din : in STD_LOGIC_VECTOR(3 downto 0);
        Adr : in STD_LOGIC_VECTOR(3 downto 0);
        RA : in STD_LOGIC_VECTOR(3 downto 0);
        DPO : out STD_LOGIC_VECTOR(3 downto 0);
        SPO : out STD_LOGIC_VECTOR(3 downto 0));
end component;

signal dpo_sig : std_logic_vector(7 downto 0);
signal spo_sig : std_logic_vector(7 downto 0);

begin

  Inst1_16X4D : wagRam16X4D Port map (  clk1=> clk2 ,
                                       we => we,
                                       Din => din(3 downto 0),
                                       Adr => adr,
                                       RA => ra,
                                       DPO => dpo_sig(3 downto 0),
                                       SPO => spo_sig(3 downto 0));

  Inst2_16X4D : wagRam16X4D Port map (  clk1=> clk2 ,
                                       we => we,
                                       Din => din(7 downto 4),
                                       Adr => adr,
                                       RA => ra,
                                       DPO => dpo_sig(7 downto 4),
                                       SPO => spo_sig(7 downto 4));

  dpo(3 downto 0)<=dpo_sig(3 downto 0);
  dpo(7 downto 4)<=dpo_sig(7 downto 4);

  spo(3 downto 0)<=spo_sig(3 downto 0);
  spo(7 downto 4)<=spo_sig(7 downto 4);

end Behavioral;
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

--- Uncomment the following library declaration if instantiating
--- any Xilinx primitives in this code.
library UNISIM;
--use UNISIM.VComponents.all;

entity wag16to1Mux is
  Port ( a : in  std_logic_vector(7 downto 0);
         b : in  std_logic_vector(7 downto 0);
         c : in  std_logic_vector(7 downto 0);
         d : in  std_logic_vector(7 downto 0);
         e : in  std_logic_vector(7 downto 0);
         f : in  std_logic_vector(7 downto 0);
         g : in  std_logic_vector(7 downto 0);
         h : in  std_logic_vector(7 downto 0);
         i : in  std_logic_vector(7 downto 0);
         j : in  std_logic_vector(7 downto 0);
         k : in  std_logic_vector(7 downto 0);
         l : in  std_logic_vector(7 downto 0);
         m : in  std_logic_vector(7 downto 0);
         n : in  std_logic_vector(7 downto 0);
         o : in  std_logic_vector(7 downto 0);
         p : in  std_logic_vector(7 downto 0);
         sel : in STD_LOGIC_VECTOR(3 downto 0);
         z   : out std_logic_vector(7 downto 0));
end wag16to1Mux;

architecture Behavioral of wag16to1Mux is
begin

  process (sel,a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p)
  begin
    case sel is
      when "0000" => z <= a;
      when "0001" => z <= b;
      when others => z <= "0000";
    end case;
  end process;

end Behavioral;
when "0010" => z <= c;
when "0011" => z <= d;
when "0100" => z <= e;
when "0101" => z <= f;
when "0110" => z <= g;
when "0111" => z <= h;
when "1000" => z <= i;
when "1001" => z <= j;
when "1010" => z <= k;
when "1011" => z <= l;
when "1100" => z <= m;
when "1101" => z <= n;
when "1110" => z <= o;
when "1111" => z <= p;
when others => z <= a;
end case;
end process;
end Behavioral;
begin

    z <= (a and b and c and d) after 181 ps;

end Behavioral;

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity wagRam16X4D is
    Port (clk1    : in std_logic;
        we      : in std_logic;
        Din     : in STD_LOGIC_VECTOR(3 downto 0);
        Adr     : in STD_LOGIC_VECTOR(3 downto 0);
        RA      : in STD_LOGIC_VECTOR(3 downto 0);
        DPO     : out STD_LOGIC_VECTOR(3 downto 0);
        SPO     : out STD_LOGIC_VECTOR(3 downto 0));
end wagRam16X4D;

architecture Behavioral of wagRam16X4D is

component wagRam16X1D is
    port ( A0 : in std_logic;
        A1 : in std_logic;
        A2 : in std_logic;
        A3 : in std_logic;
        clk : in std_logic;
        D_in : in std_logic;
        RA0 : in std_logic;
        RA1 : in std_logic;
        RA2 : in std_logic;
        RA3 : in std_logic;
        WE : in std_logic;
        DPO_out : out std_logic;
    )
end wagRam16X1D;
SPO_out : out std_logic);
end component;

begin

U_0 : wagRam16X1D
port map ( A0 => Adr(0),
           A1 => Adr(1),
           A2 => Adr(2),
           A3 => Adr(3),
           clk => clk1,
           D_in => Din(0),
           RA0 => RA(0),
           RA1 => RA(1),
           RA2 => RA(2),
           RA3 => RA(3),
           WE => we,
           DPO_out => DPO(0),
           SPO_out => SPO(0));

U_1 : wagRam16X1D
port map ( A0 => Adr(0),
           A1 => Adr(1),
           A2 => Adr(2),
           A3 => Adr(3),
           clk => clk1,
           D_in => Din(1),
           RA0 => RA(0),
           RA1 => RA(1),
           RA2 => RA(2),
           RA3 => RA(3),
           WE => we,
           DPO_out => DPO(1),
           SPO_out => SPO(1));

U_2 : wagRam16X1D
port map ( A0 => Adr(0),
           A1 => Adr(1),
           A2 => Adr(2),
           A3 => Adr(3),
           clk => clk1,
           D_in => Din(2),
           RA0 => RA(0),
           RA1 => RA(1),
           RA2 => RA(2),
           RA3 => RA(3),
           WE => we,
           DPO_out => DPO(2),
SPO_out => SPO(2));

U_3 : wagRam16X1D
port map (A0 => Adr(0),
          A1 => Adr(1),
          A2 => Adr(2),
          A3 => Adr(3),
          clk => clk1,
          D_in => Din(3),
          RA0 => RA(0),
          RA1 => RA(1),
          RA2 => RA(2),
          RA3 => RA(3),
          WE => we,
          DPO_out => DPO(3),
          SPO_out => SPO(3));

end Behavioral;

-------------------------------------------------------------------------------------
--Designer: Wilfred A. Glover-Akpey
--Date: August 2008
--Course: CSE600
--Affiliation: The Pennsylvania State University
--Description: 16by1D
-------------------------------------------------------------------------------------

library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;

entity wagRam16X1D is
  port ( A0 : in std_logic;
          A1 : in std_logic;
          A2 : in std_logic;
          A3 : in std_logic;
          clk : in std_logic;
          D_in : in std_logic;
          RA0 : in std_logic;
          RA1 : in std_logic;
          RA2 : in std_logic;
          RA3 : in std_logic;
          WE : in std_logic;
          DPO_out : out std_logic;
          SPO_out : out std_logic);
end wagRam16X1D;

architecture BEHAVIORAL of wagRam16X1D is

begin

RAM16X1D_inst : RAM16X1D
generic map (
    INIT => X"0000"
)
port map (
    DPO => DPO_out, -- Read-only 1-bit data output for DPRA
    SPO => SPO_out, -- R/W 1-bit data output for A0-A3
    A0 => A0, -- R/W address[0] input bit
    A1 => A1, -- R/W address[1] input bit
    A2 => A2, -- R/W address[2] input bit
    A3 => A3, -- R/W address[3] input bit
    D => D_in, -- Write 1-bit data input
    DPRA0 => RA0, -- Read-only address[0] input bit
    DPRA1 => RA1, -- Read-only address[1] input bit
    DPRA2 => RA2, -- Read-only address[2] input bit
    DPRA3 => RA3, -- Read-only address[3] input bit
    WCLK => clk, -- Write clock input
    WE => WE -- Write enable input
);

end BEHAVIORAL;

------------------------------------------------------------------------------
-- Designer: Wilfred A. Glover-Akpey
-- Date: August 2008
-- Course: CSE471
-- Affiliation: The Pennsylvania State University
-- Description: D-Flip-flop
------------------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;

entity wagfcd is
port( C, D, CLR : in std_logic;
    Q : out std_logic:=0');
end wagfcd;

architecture archi of wagfcd is

begin
    process (C, CLR)
    begin
        if (CLR = '1')then

end process (C, CLR);

end BEHAVIORAL;
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity wag8bitff is
  Port (D : in std_logic_vector(7 downto 0);
        W : in std_logic;
        Y : out std_logic_vector(7 downto 0));
end wag8bitff;

architecture Behavioral of wag8bitff is

begin

  process (W)
  begin
    if (rising_edge(W)) then
      Y <= D after 200 ps;
    end if;
  end process;

end Behavioral;

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity wag8bit2_1mux is
    Port ( I0 : in std_logic_vector(7 downto 0);
           I1 : in std_logic_vector(7 downto 0);
           R : in std_logic;
           Z : out std_logic_vector(7 downto 0));
end wag8bit2_1mux;

architecture Behavioral of wag8bit2_1mux is
begin
    Z <= I0 WHEN R = '1' ELSE I1;
end Behavioral;

--
-- Copyright (c) 1995-2007 Xilinx, Inc. All rights reserved.
--

library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;
entity tx is
  port ( txck16i : in    std_logic;
         txpdi   : in    std_logic_vector (7 downto 0);
         txwi    : in    std_logic;
         txfo    : out   std_logic;
         txfwro  : out   std_logic;
         txsdo   : out   std_logic);
end tx;

architecture BEHAVIORAL of tx is
  signal txsh : std_logic;
  signal XLXN_7 : std_logic;
  component txdata
    port ( txdi   : in    std_logic_vector (7 downto 0);
          txdw   : in    std_logic;
          txdl   : in    std_logic;
          txshck : in    std_logic;
          tdo    : out   std_logic);
  end component;

  component txcntc1a
    port ( tcki  : in    std_logic;
           txwi  : in    std_logic;
           shcko : out   std_logic;
           txlo  : out   std_logic;
           txf   : out   std_logic;
           txrwo : out   std_logic);
  end component;

  begin
    XLXI_51 : txdata
      port map (txdi(7 downto 0) => txpdi(7 downto 0),
                 txdl => XLXN_7,
                 txdw => txwi,
                 txshck => txsh,
                 tdo => txsdo);

    XLXI_53 : txcntc1a
      port map (tcki => txck16i,
                txwi => txwi,
                shcko => txsh,
                txf => txfo,
                txlo => XLXN_7,
                txrwo => txfwro);
  end BEHAVIORAL;
library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;

entity rx is
  port (rxck16i : in std_logic;
         rxsdi   : in std_logic;
         rxwi    : in std_logic;
         rxfo    : out std_logic;
         rxpdo   : out std_logic_vector (7 downto 0));
end rx;

architecture BEHAVIORAL of rx is
  signal XLXN_12 : std_logic;
  signal rxfo_DUMMY : std_logic;
component rxdata
  port ( sdi   : in std_logic;
         cki   : in std_logic;
         wri   : in std_logic;
         datao : out std_logic_vector (7 downto 0));
end component;

component rxcntcl1a
  port ( sdi  : in std_logic;
          ck21i : in std_logic;
          rdi  : in std_logic;
          ...
end component;
begin
rxfo <= rxfo_DUMMY;
XLXI_2 : rxdata
    port map (cki=>XLXN_12,
              sdi=>rxsdi,
             wri=>rxfo_DUMMY,
            datao(7 downto 0)=>rxpdo(7 downto 0));

XLXI_3 : rxcntc1a
    port map (ck21i=>rxck16i,
              rdi=>rxwi,
             sdi=>rxsdi,
            rxf=>rxfo_DUMMY,
              shcko=>XLXN_12);

end BEHAVIORAL;

-- Copyright (c) 1995-2007 Xilinx, Inc. All rights reserved.
--

-- Vendor: Xilinx
-- Version : 9.2.04i
-- Application : sch2vhdl
-- Filename : fdiv11v1.vhf
-- Timestamp : 07/04/2008 09:22:15
--
-- Command: C:\Xilinx\bin\nt\sch2vhdl.exe -intstyle ise -family spartan3 -flat -suppress -w
C:\Thesis\Sim2_new\fdiv11v1.sch fdiv11v1.vhf

library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;
entity FTRSE_MXILINX_fdiv11v1 is
  port ( C : in    std_logic;
         CE : in    std_logic;
         R  : in    std_logic;
         S  : in    std_logic;
         T  : in    std_logic;
         Q  : out   std_logic);
end FTRSE_MXILINX_fdiv11v1;

architecture BEHAVIORAL of FTRSE_MXILINX_fdiv11v1 is
  attribute BOX_TYPE   : string   is "BLACK_BOX";
  attribute INIT       : string   is "0";
  attribute RLOC       : string   is "X0Y0";
  signal CE_S    : std_logic;
  signal D_S     : std_logic;
  signal TQ      : std_logic;
  signal Q_DUMMY : std_logic;
  component XOR2
    port ( I0 : in    std_logic;
           I1 : in    std_logic;
           O  : out   std_logic);
  end component;
  attribute BOX_TYPE of XOR2 : component is "BLACK_BOX";

  component FDRE
    -- synopsys translate_off
    generic( INIT : bit := '0');
    -- synopsys translate_on
    port ( C : in    std_logic;
           CE : in    std_logic;
           D  : in    std_logic;
           R  : in    std_logic;
           Q  : out   std_logic);
  end component;
  attribute INIT of FDRE : component is "0";
  attribute BOX_TYPE of FDRE : component is "BLACK_BOX";

  component OR2
    port ( I0 : in    std_logic;
           I1 : in    std_logic;
           O  : out   std_logic);
  end component;
  attribute BOX_TYPE of OR2 : component is "BLACK_BOX";

  attribute RLOC of I_36_35 : label is "X0Y0";
begin
  Q <= Q_DUMMY;
  I_36_32 : XOR2
library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;

entity CB4RE_MXILINX_fdiv11v1 is
  port ( C   : in    std_logic;
         CE  : in    std_logic;
         R   : in    std_logic;
         CEO : out   std_logic;
         Q0  : out   std_logic;
         Q1  : out   std_logic;
         Q2  : out   std_logic;
         Q3  : out   std_logic;
         TC  : out   std_logic);
end CB4RE_MXILINX_fdiv11v1;

architecture BEHAVIORAL of CB4RE_MXILINX_fdiv11v1 is
attribute HU_SET     : string := "BEHAVIORAL";
attribute BOX_TYPE   : string := "behavioural";
signal T2       : std_logic;
signal T3       : std_logic;
signal XLXN_1   : std_logic;
begin
end BEHAVIORAL;
signal XLXN_2 : std_logic;
signal Q0_DUMMY : std_logic;
signal Q1_DUMMY : std_logic;
signal Q2_DUMMY : std_logic;
signal Q3_DUMMY : std_logic;
signal TC_DUMMY : std_logic;
component FTRSE_MXILINX_fdiv11v1
  port ( C : in  std_logic;
         CE : in  std_logic;
         R : in  std_logic;
         S : in  std_logic;
         T : in  std_logic;
         Q : out std_logic);
end component;

component AND4
  port ( I0 : in  std_logic;
         I1 : in  std_logic;
         I2 : in  std_logic;
         I3 : in  std_logic;
         O  : out std_logic);
end component;
attribute BOX_TYPE of AND4 : component is "BLACK_BOX";

component AND3
  port ( I0 : in  std_logic;
         I1 : in  std_logic;
         I2 : in  std_logic;
         O  : out std_logic);
end component;
attribute BOX_TYPE of AND3 : component is "BLACK_BOX";

component AND2
  port ( I0 : in  std_logic;
         I1 : in  std_logic;
         O  : out std_logic);
end component;
attribute BOX_TYPE of AND2 : component is "BLACK_BOX";

component VCC
  port ( P : out std_logic);
end component;
attribute BOX_TYPE of VCC : component is "BLACK_BOX";

component GND
  port ( G : out std_logic);
end component;
attribute BOX_TYPE of GND : component is "BLACK_BOX";
attribute HU_SET of I_Q0 : label is "I_Q0_0";
attribute HU_SET of I_Q1 : label is "I_Q1_1";
attribute HU_SET of I_Q2 : label is "I_Q2_2";
attribute HU_SET of I_Q3 : label is "I_Q3_3";

begin
    Q0 <= Q0_DUMMY;
    Q1 <= Q1_DUMMY;
    Q2 <= Q2_DUMMY;
    Q3 <= Q3_DUMMY;
    TC <= TC_DUMMY;

I_Q0 : FTRSE_MXILINX_fdiv11v1
    port map (C=>C,
             CE=>CE,
             R=>R,
             S=>XLXN_2,
             T=>XLXN_1,
             Q=>Q0_DUMMY);

I_Q1 : FTRSE_MXILINX_fdiv11v1
    port map (C=>C,
             CE=>CE,
             R=>R,
             S=>XLXN_2,
             T=>Q0_DUMMY,
             Q=>Q1_DUMMY);

I_Q2 : FTRSE_MXILINX_fdiv11v1
    port map (C=>C,
             CE=>CE,
             R=>R,
             S=>XLXN_2,
             T=>T2,
             Q=>Q2_DUMMY);

I_Q3 : FTRSE_MXILINX_fdiv11v1
    port map (C=>C,
             CE=>CE,
             R=>R,
             S=>XLXN_2,
             T=>T3,
             Q=>Q3_DUMMY);

I_36_31 : AND4
    port map (I0=>Q3_DUMMY,
              I1=>Q2_DUMMY,
              I2=>Q1_DUMMY,
              I3=>Q0_DUMMY,
              O=>TC_DUMMY);
I_36_32 : AND3
    port map (I0=>Q2_DUMMY,
              I1=>Q1_DUMMY,
              I2=>Q0_DUMMY,
              O=>T3);

I_36_33 : AND2
    port map (I0=>Q1_DUMMY,
              I1=>Q0_DUMMY,
              O=>T2);

I_36_58 : VCC
    port map (P=>XLXN_1);

I_36_64 : GND
    port map (G=>XLXN_2);

I_36_69 : AND2
    port map (I0=>CE,
              I1=>TC_DUMMY,
              O=>CEO);

dend BEHAVIORAL;

library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;

entity M2_1_MUXILINX_fdiv11v1 is
    port ( D0 : in    std_logic;
            D1 : in    std_logic;
            S0 : in    std_logic;
            O  : out   std_logic);
end M2_1_MUXILINX_fdiv11v1;

architecture BEHAVIORAL of M2_1_MUXILINX_fdiv11v1 is
    attribute BOX_TYPE : string := "BLACK_BOX";
    signal M0 : std_logic;
    signal M1 : std_logic;
    component AND2B1
        port ( I0 : in    std_logic;
                I1 : in    std_logic;
                O : out   std_logic);
    end component;
    begin
        end architecture;
component OR2
  port ( I0 : in    std_logic;
        I1 : in    std_logic;
        O  : out   std_logic);
end component;
attribute BOX_TYPE of OR2 : component is "BLACK_BOX";

component AND2
  port ( I0 : in    std_logic;
        I1 : in    std_logic;
        O  : out   std_logic);
end component;
attribute BOX_TYPE of AND2 : component is "BLACK_BOX";

begin
  I_36_7 : AND2B1
    port map (I0=>S0,
              I1=>D0,
              O=>M0);

  I_36_8 : OR2
    port map (I0=>M1,
              I1=>M0,
              O=>O);

  I_36_9 : AND2
    port map (I0=>D1,
              I1=>S0,
              O=>M1);

end BEHAVIORAL;

library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;

entity FTCE_MXILINX_fdiv11v1 is
  port ( C   : in    std_logic;
         CE  : in    std_logic;
         CLR : in    std_logic;
         T   : in    std_logic;
         Q   : out   std_logic);
end FTCE_MXILINX_fdiv11v1;
architecture BEHAVIORAL of FTCE_MXILINX_fdiv11v1 is
attribute BOX_TYPE : string ;
attribute INIT : string ;
attribute RLOC : string ;
signal TQ : std_logic;
signal Q_DUMMY : std_logic;
component XOR2
  port ( I0 : in    std_logic;
     I1 : in    std_logic;
      O : out   std_logic);
end component;
attribute BOX_TYPE of XOR2 : component is "BLACK_BOX";
component FDCE
  -- synopsys translate_off
  generic( INIT : bit := '0');
  -- synopsys translate_on
  port ( C : in    std_logic;
    CE : in    std_logic;
    CLR : in    std_logic;
      D : in    std_logic;
       Q : out   std_logic);
end component;
attribute INIT of FDCE : component is "0";
attribute BOX_TYPE of FDCE : component is "BLACK_BOX";
attribute RLOC of I_36_35 : label is "X0Y0";
begin
  Q <= Q_DUMMY;
I_36_32 : XOR2
  port map (I0=>T,
    I1=>Q_DUMMY,
      O=>TQ);
I_36_35 : FDCE
  port map (C=>C,
    CE=>CE,
    CLR=>CLR,
    D=>TQ,
      Q=>Q_DUMMY);
end BEHAVIORAL;

library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;

entity fdiv11v1 is
port ( cki   : in    std_logic;
       dvcko : out   std_logic);
end fdiv11v1;

architecture BEHAVIORAL of fdiv11v1 is

attribute BOX_TYPE   : string ;
attribute HU_SET     : string ;
attribute INIT       : string ;
signal cti         : std_logic_vector (3 downto 0);
signal dti4        : std_logic_vector (3 downto 0);
signal dti5        : std_logic_vector (3 downto 0);
signal ms6         : std_logic;
signal mt04o4      : std_logic;
signal mt04o5      : std_logic;
signal res6        : std_logic;
signal XLXN_2      : std_logic;
signal XLXN_3      : std_logic;
signal XLXN_4      : std_logic;
signal dvcko_DUMMY : std_logic;

component VCC
port ( P : out   std_logic);
end component;

attribute BOX_TYPE of VCC : component is "BLACK_BOX";

component FTCE_MXILINX_fdiv11v1
port ( C   : in    std_logic;
       CE  : in    std_logic;
       CLR : in    std_logic;
       T   : in    std_logic;
       Q   : out   std_logic);
end component;

component GND
port ( G : out   std_logic);
end component;

attribute BOX_TYPE of GND : component is "BLACK_BOX";

component match04
port ( ctin   : in    std_logic_vector (3 downto 0);
       dtin   : in    std_logic_vector (3 downto 0);
       matcho : out   std_logic);
end component;

component M2_1_MXILINX_fdiv11v1
port ( D0 : in    std_logic;
       D1 : in    std_logic;
       ...
S0 : in std_logic;
O  : out std_logic);
end component;

component CB4RE_MXILINX_fdiv11v1
port ( C   : in    std_logic;
      CE : in    std_logic;
      R   : in    std_logic;
      CEO : out   std_logic;
      Q0  : out   std_logic;
      Q1  : out   std_logic;
      Q2  : out   std_logic;
      Q3  : out   std_logic;
      TC  : out   std_logic);
end component;

component FD
  -- synopsys translate_off
  generic( INIT : bit := '0');
  -- synopsys translate_on
port ( C : in    std_logic;
      D : in    std_logic;
      Q : out std_logic);
end component;
attribute INIT of FD : component is "0";
attribute BOX_TYPE of FD : component is "BLACK_BOX";

attribute HU_SET of XLXI_4 : label is "XLXI_4_4";
attribute HU_SET of XLXI_12 : label is "XLXI_12_5";
attribute HU_SET of XLXI_13 : label is "XLXI_13_6";
begins
dvcko <= dvcko_DUMMY;
XLXI_2 : VCC
  port map (P=>XLXN_3);

XLXI_4 : FTCE_MXILINX_fdiv11v1
  port map (C=>cki,
              CE=>res6,
              CLR=>XLXN_4,
              T=>XLXN_2,
              Q=>dvcko_DUMMY);

XLXI_5 : VCC
  port map (P=>XLXN_2);

XLXI_6 : GND
  port map (G=>XLXN_4);

XLXI_7 : match04
port map (ctin(3 downto 0)=>cti(3 downto 0),
    dtin(3 downto 0)=>dti4(3 downto 0),
    matcho=>mt04o4);

XLXI_8 : match04
port map (ctin(3 downto 0)=>cti(3 downto 0),
    dtin(3 downto 0)=>dti5(3 downto 0),
    matcho=>mt04o5);

XLXI_12 : M2_1_MXILINX_fdiv11v1
port map (D0=>mt04o4,
    D1=>mt04o5,
    S0=>ms6,
    O=>res6);

XLXI_13 : CB4RE_MXILINX_fdiv11v1
port map (C=>cki,
    CE=>XLXN_3,
    R=>res6,
    CEO=>open,
    Q0=>cti(0),
    Q1=>cti(1),
    Q2=>cti(2),
    Q3=>cti(3),
    TC=>open);

XLXI_24 : VCC
port map (P=>dti4(2));

XLXI_26 : GND
port map (G=>dti4(3));

XLXI_27 : GND
port map (G=>dti4(1));

XLXI_28 : GND
port map (G=>dti4(0));

XLXI_29 : GND
port map (G=>dti5(1));

XLXI_30 : GND
port map (G=>dti5(3));

XLXI_31 : VCC
port map (P=>dti5(2));

XLXI_32 : VCC
port map (P=>dti5(0));
XLXI_92 : FD
    port map (C=>cki,
            D=>dvcko_DUMMY,
            Q=>ms6);
end BEHAVIORAL;

-- Copyright (c) 1995-2007 Xilinx, Inc. All rights reserved.
--------------------------------------------------------------------------------
-- ____  ____   ____   ____   ____   ____   ____   ____   ____   ____   ____   ____
-- /   / /   /   / /   /   / /   /   / /   /   / /   /   / /   /   / /   /   / /    
-- /___/   /___/   /___/   /___/   /___/   /___/   /___/   /___/   /___/   /___/   
-- Vendor: Xilinx
-- Version : 9.2.04i
-- Application : sch2vhdl
-- Filename : fdiv1250v1.vhf
-- Timestamp : 07/04/2008 09:21:57
-- Design Name: fdiv1250v1
-- Device: spartan3
-- Purpose:  
--   This vhdl netlist is translated from an ECS schematic. It can be  
--   synthesis and simulated, but it should not be modified.  
--
library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;

entity fdiv1250v1 is
    port ( ckin : in std_logic;
           dvcko : out std_logic);
end fdiv1250v1;

architecture BEHAVIORAL of fdiv1250v1 is
    attribute BOX_TYPE : string := "";
    signal ckdiv : std_logic_vector (15 downto 0);
    signal gggnd : std_logic;
    signal vvvcc : std_logic;
    component ckuart
        port ( dti : in std_logic_vector (15 downto 0);
component BUF
  port ( I : in std_logic;
        O : out std_logic);
end component;
attribute BOX_TYPE of BUF : component is "BLACK_BOX";

component GND
  port ( G : out std_logic);
end component;
attribute BOX_TYPE of GND : component is "BLACK_BOX";

component VCC
  port ( P : out std_logic);
end component;
attribute BOX_TYPE of VCC : component is "BLACK_BOX";

begin
  XLXI_1 : ckuart
  port map (cki=>ckin,
            dti(15 downto 0)=>ckdiv(15 downto 0),
            dvcko=>dvcko);

  XLXI_5 : BUF
  port map (I=>gggnd,
            O=>ckdiv(8));

  XLXI_6 : BUF
  port map (I=>gggnd,
            O=>ckdiv(9));

  XLXI_7 : BUF
  port map (I=>vvvcc,
            O=>ckdiv(10));

  XLXI_8 : BUF
  port map (I=>gggnd,
            O=>ckdiv(11));

  XLXI_9 : BUF
  port map (I=>gggnd,
            O=>ckdiv(12));

  XLXI_10 : BUF
  port map (I=>gggnd,
            O=>ckdiv(13));
XLXI_11 : BUF
port map (I=>gggnd,
   O=>ckdiv(14));

XLXI_12 : BUF
port map (I=>gggnd,
   O=>ckdiv(15));

XLXI_13 : BUF
port map (I=>vvvcc,
   O=>ckdiv(7));

XLXI_14 : BUF
port map (I=>vvvcc,
   O=>ckdiv(6));

XLXI_15 : BUF
port map (I=>vvvcc,
   O=>ckdiv(5));

XLXI_16 : BUF
port map (I=>gggnd,
   O=>ckdiv(4));

XLXI_17 : BUF
port map (I=>gggnd,
   O=>ckdiv(3));

XLXI_18 : BUF
port map (I=>gggnd,
   O=>ckdiv(2));

XLXI_19 : BUF
port map (I=>gggnd,
   O=>ckdiv(1));

XLXI_20 : BUF
port map (I=>vvvcc,
   O=>ckdiv(0));

XLXI_21 : GND
port map (G=>gggnd);

XLXI_22 : VCC
port map (P=>vvvcc);

end BEHAVIORAL;
library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;

entity match04 is
   port ( ctin   : in    std_logic_vector (3 downto 0);
          dtin   : in    std_logic_vector (3 downto 0);
          matcho : out   std_logic);
end match04;

architecture BEHAVIORAL of match04 is
   attribute BOX_TYPE   : string ;
   signal XLXN_64 : std_logic;
   signal XLXN_65 : std_logic;
   signal XLXN_66 : std_logic;
   signal XLXN_67 : std_logic;
   component XNOR2
      port ( I0 : in    std_logic;
             I1 : in    std_logic;
             O  : out   std_logic);
   end component;
   attribute BOX_TYPE of XNOR2 : component is "BLACK_BOX";
   component AND4

begin

XLXI_1 : XNOR2
port map (I0=>ctin(1),
I1=>dtin(1),
O=>XLXN_64);

XLXI_2 : XNOR2
port map (I0=>ctin(2),
I1=>dtin(2),
O=>XLXN_65);

XLXI_3 : XNOR2
port map (I0=>ctin(3),
I1=>dtin(3),
O=>XLXN_66);

XLXI_16 : XNOR2
port map (I0=>ctin(0),
I1=>dtin(0),
O=>XLXN_67);

XLXI_22 : AND4
port map (I0=>XLXN_66,
I1=>XLXN_65,
I2=>XLXN_64,
I3=>XLXN_67,
O=>matcho);

end BEHAVIORAL;

-- Copyright (c) 1995-2007 Xilinx, Inc. All rights reserved.

-- / \  _____
-- / \ /  
--/___\ / Vendor: Xilinx
\ \ V  Version : 9.2.04i
\  \ Application : sch2vhdl
\  /  Filename : ckuart.vhf
\/__\  \  Timestamp : 07/04/2008 09:22:11
library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;

entity FTCE_MXILINX_ckuart is
  port ( C : in    std_logic;
         CE : in    std_logic;
         CLR : in    std_logic;
         T : in    std_logic;
         Q : out   std_logic);
end FTCE_MXILINX_ckuart;

architecture BEHAVIORAL of FTCE_MXILINX_ckuart is
  attribute BOX_TYPE   : string ;
  attribute INIT       : string ;
  attribute RLOC       : string ;
  signal TQ      : std_logic;
  signal Q_DUMMY : std_logic;
  component XOR2
    port ( I0 : in    std_logic;
           I1 : in    std_logic;
           O  : out   std_logic);
  end component;
  attribute BOX_TYPE of XOR2 : component is "BLACK_BOX";
  component FDCE
    -- synopsys translate_off
    generic(INIT : bit := '0');
    -- synopsys translate_on
    port ( C : in    std_logic;
           CE : in    std_logic;
           CLR : in    std_logic;
           D  : in    std_logic;
           Q : out   std_logic);
  end component;
attribute INIT of FDCE : component is "0";
attribute BOX_TYPE of FDCE : component is "BLACK_BOX";

attribute RLOC of I_36_35 : label is "X0Y0";
begin
  Q <= Q_DUMMY;
  I_36_32 : XOR2
    port map (I0=>T,
              I1=>Q_DUMMY,
              O=>TQ);

  I_36_35 : FDCE
    port map (C=>C,
              CE=>CE,
              CLR=>CLR,
              D=>TQ,
              Q=>Q_DUMMY);
end BEHAVIORAL;

library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;

entity FTRSE_MXILINX_ckuart is
  port ( C  : in    std_logic;
         CE : in    std_logic;
         R  : in    std_logic;
         S  : in    std_logic;
         T  : in    std_logic;
         Q  : out   std_logic);
end FTRSE_MXILINX_ckuart;

architecture BEHAVIORAL of FTRSE_MXILINX_ckuart is
attribute BOX_TYPE : string :
attribute INIT : string :
attribute RLOC : string :
signal CE_S : std_logic;
signal D_S : std_logic;
signal TQ : std_logic;
signal Q_DUMMY : std_logic;
component XOR2
  port ( I0 : in    std_logic;
         I1 : in    std_logic;
         O : out   std_logic);
end component;
attribute BOX_TYPE of XOR2 : component is "BLACK_BOX";

component FDRE
  -- synopsys translate_off
generic( INIT : bit := '0');
  -- synopsys translate_on
  port ( C : in    std_logic;
        CE : in    std_logic;
        D  : in    std_logic;
        R  : in    std_logic;
        Q  : out   std_logic);
end component;
attribute INIT of FDRE : component is "0";
attribute BOX_TYPE of FDRE : component is "BLACK_BOX";

component OR2
  port ( I0 : in    std_logic;
        I1 : in    std_logic;
        O  : out   std_logic);
end component;
attribute BOX_TYPE of OR2 : component is "BLACK_BOX";

attribute RLOC of I_36_35 : label is "X0Y0";
beg
begin
  Q <= Q_DUMMY;
  I_36_32 : XOR2
  port map (I0=>T,
            I1=>Q_DUMMY,
            O=>TQ);

  I_36_35 : FDRE
  port map (C=>C,
            CE=>CE_S,
            D=>D_S,
            R=>R,
            Q=>Q_DUMMY);

  I_36_73 : OR2
  port map (I0=>S,
            I1=>TQ,
            O=>D_S);

  I_36_77 : OR2
  port map (I0=>CE,
            I1=>S,
            O=>CE_S);
end BEHAVIORAL;
library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;

entity CB16RE_MXILINX_ckuart is
port ( C   : in    std_logic;
      CE  : in    std_logic;
      R   : in    std_logic;
      CEO : out   std_logic;
      Q   : out   std_logic_vector (15 downto 0);
      TC  : out   std_logic);
end CB16RE_MXILINX_ckuart;

architecture BEHAVIORAL of CB16RE_MXILINX_ckuart is
attribute HU_SET     : string ;
attribute BOX_TYPE  : string ;
signal T2       : std_logic;
signal T3       : std_logic;
signal T4       : std_logic;
signal T5       : std_logic;
signal T6       : std_logic;
signal T7       : std_logic;
signal T8       : std_logic;
signal T9       : std_logic;
signal T10      : std_logic;
signal T11      : std_logic;
signal T12      : std_logic;
signal T13      : std_logic;
signal T14      : std_logic;
signal T15      : std_logic;
signal XLXN_1   : std_logic;
signal XLXN_2   : std_logic;
signal XLXN_3   : std_logic;
signal Q_DUMMY  : std_logic_vector (15 downto 0);
signal TC_DUMMY : std_logic;
component FTRSE_MXILINX_ckuart
port ( C  : in    std_logic;
      CE : in    std_logic;
      R  : in    std_logic;
      S  : in    std_logic;
      T  : in    std_logic;
      Q  : out   std_logic);
end component;
component GND
  port ( G : out std_logic);
end component;
attribute BOX_TYPE of GND : component is "BLACK_BOX";

component AND5
  port ( I0 : in std_logic;
         I1 : in std_logic;
         I2 : in std_logic;
         I3 : in std_logic;
         I4 : in std_logic;
         O  : out std_logic);
end component;
attribute BOX_TYPE of AND5 : component is "BLACK_BOX";

component AND2
  port ( I0 : in std_logic;
         I1 : in std_logic;
         O  : out std_logic);
end component;
attribute BOX_TYPE of AND2 : component is "BLACK_BOX";

component AND3
  port ( I0 : in std_logic;
         I1 : in std_logic;
         I2 : in std_logic;
         O  : out std_logic);
end component;
attribute BOX_TYPE of AND3 : component is "BLACK_BOX";

component AND4
  port ( I0 : in std_logic;
         I1 : in std_logic;
         I2 : in std_logic;
         I3 : in std_logic;
         O  : out std_logic);
end component;
attribute BOX_TYPE of AND4 : component is "BLACK_BOX";

component VCC
  port ( P : out std_logic);
end component;
attribute BOX_TYPE of VCC : component is "BLACK_BOX";

attribute HU_SET of I_Q0 : label is "I_Q0_15";
attribute HU_SET of I_Q1 : label is "I_Q1_14";
attribute HU_SET of I_Q2 : label is "I_Q2_13";
attribute HU_SET of I_Q3 : label is "I_Q3_12";
attribute HU_SET of I_Q4 : label is "I_Q4_11";
attribute HU_SET of I_Q5 : label is "I_Q5_10";
attribute HU_SET of I_Q6 : label is "I_Q6_9";
attribute HU_SET of I_Q7 : label is "I_Q7_8";
attribute HU_SET of I_Q8 : label is "I_Q8_0";
attribute HU_SET of I_Q9 : label is "I_Q9_1";
attribute HU_SET of I_Q10 : label is "I_Q10_2";
attribute HU_SET of I_Q11 : label is "I_Q11_3";
attribute HU_SET of I_Q12 : label is "I_Q12_4";
attribute HU_SET of I_Q13 : label is "I_Q13_5";
attribute HU_SET of I_Q14 : label is "I_Q14_6";
attribute HU_SET of I_Q15 : label is "I_Q15_7";
begin
Q(15 downto 0) <= Q_DUMMY(15 downto 0);
TC <= TC_DUMMY;
I_Q0 : FTRSE_MXILINX_ckuart
  port map (C=>C,
  CE=>CE,
  R=>R,
  S=>XLXN_2,
  T=>XLXN_1,
  Q=>Q_DUMMY(0));

I_Q1 : FTRSE_MXILINX_ckuart
  port map (C=>C,
  CE=>CE,
  R=>R,
  S=>XLXN_2,
  T=>Q_DUMMY(0),
  Q=>Q_DUMMY(1));

I_Q2 : FTRSE_MXILINX_ckuart
  port map (C=>C,
  CE=>CE,
  R=>R,
  S=>XLXN_2,
  T=>T2,
  Q=>Q_DUMMY(2));

I_Q3 : FTRSE_MXILINX_ckuart
  port map (C=>C,
  CE=>CE,
  R=>R,
  S=>XLXN_2,
  T=>T3,
  Q=>Q_DUMMY(3));

I_Q4 : FTRSE_MXILINX_ckuart
  port map (C=>C,
  CE=>CE,
R=>R,
S=>XLXN_2,
T=>T4,
Q=>Q_DUMMY(4));

I_Q5 : FTRSE_MXILINX_ckuart
port map (C=>C,
    CE=>CE,
    R=>R,
    S=>XLXN_2,
    T=>T5,
    Q=>Q_DUMMY(5));

I_Q6 : FTRSE_MXILINX_ckuart
port map (C=>C,
    CE=>CE,
    R=>R,
    S=>XLXN_2,
    T=>T6,
    Q=>Q_DUMMY(6));

I_Q7 : FTRSE_MXILINX_ckuart
port map (C=>C,
    CE=>CE,
    R=>R,
    S=>XLXN_2,
    T=>T7,
    Q=>Q_DUMMY(7));

I_Q8 : FTRSE_MXILINX_ckuart
port map (C=>C,
    CE=>CE,
    R=>R,
    S=>XLXN_3,
    T=>T8,
    Q=>Q_DUMMY(8));

I_Q9 : FTRSE_MXILINX_ckuart
port map (C=>C,
    CE=>CE,
    R=>R,
    S=>XLXN_3,
    T=>T9,
    Q=>Q_DUMMY(9));

I_Q10 : FTRSE_MXILINX_ckuart
port map (C=>C,
    CE=>CE,
    R=>R,
S=>XLXN_3, 
T=>T10, 
Q=>Q_DUMMY(10));

I_Q11 : FTRSE_MXILINX_ckuart 
port map (C=>C, 
CE=>CE, 
R=>R, 
S=>XLXN_3, 
T=>T11, 
Q=>Q_DUMMY(11));

I_Q12 : FTRSE_MXILINX_ckuart 
port map (C=>C, 
CE=>CE, 
R=>R, 
S=>XLXN_3, 
T=>T12, 
Q=>Q_DUMMY(12));

I_Q13 : FTRSE_MXILINX_ckuart 
port map (C=>C, 
CE=>CE, 
R=>R, 
S=>XLXN_3, 
T=>T13, 
Q=>Q_DUMMY(13));

I_Q14 : FTRSE_MXILINX_ckuart 
port map (C=>C, 
CE=>CE, 
R=>R, 
S=>XLXN_3, 
T=>T14, 
Q=>Q_DUMMY(14));

I_Q15 : FTRSE_MXILINX_ckuart 
port map (C=>C, 
CE=>CE, 
R=>R, 
S=>XLXN_3, 
T=>T15, 
Q=>Q_DUMMY(15));

I_36_1 : GND 
port map (G=>XLXN_3);

I_36_2 : GND 
port map (G=>XLXN_2);
I_36_20 : AND5
  port map (I0=>Q_DUMMY(11),
           I1=>Q_DUMMY(10),
           I2=>Q_DUMMY(9),
           I3=>Q_DUMMY(8),
           I4=>T8,
           O=>T12);

I_36_21 : AND2
  port map (I0=>Q_DUMMY(8),
           I1=>T8,
           O=>T9);

I_36_22 : AND3
  port map (I0=>Q_DUMMY(9),
           I1=>Q_DUMMY(8),
           I2=>T8,
           O=>T10);

I_36_23 : AND4
  port map (I0=>Q_DUMMY(10),
           I1=>Q_DUMMY(9),
           I2=>Q_DUMMY(8),
           I3=>T8,
           O=>T11);

I_36_24 : AND4
  port map (I0=>Q_DUMMY(14),
           I1=>Q_DUMMY(13),
           I2=>Q_DUMMY(12),
           I3=>T12,
           O=>T15);

I_36_25 : AND3
  port map (I0=>Q_DUMMY(13),
           I1=>Q_DUMMY(12),
           I2=>T12,
           O=>T14);

I_36_26 : AND2
  port map (I0=>Q_DUMMY(12),
           I1=>T12,
           O=>T13);

I_36_27 : AND5
  port map (I0=>Q_DUMMY(15),
           I1=>Q_DUMMY(14),
           I2=>Q_DUMMY(13),
I3=>Q_DUMMY(12),
I4=>T12,
O=>TC_DUMMY);

I_36_28 : AND4
port map (I0=>Q_DUMMY(6),
I1=>Q_DUMMY(5),
I2=>Q_DUMMY(4),
I3=>T4,
O=>T7);

I_36_29 : AND3
port map (I0=>Q_DUMMY(5),
I1=>Q_DUMMY(4),
I2=>T4,
O=>T6);

I_36_30 : AND2
port map (I0=>Q_DUMMY(4),
I1=>T4,
O=>T5);

I_36_31 : AND5
port map (I0=>Q_DUMMY(7),
I1=>Q_DUMMY(6),
I2=>Q_DUMMY(5),
I3=>Q_DUMMY(4),
I4=>T4,
O=>T8);

I_36_32 : AND4
port map (I0=>Q_DUMMY(3),
I1=>Q_DUMMY(2),
I2=>Q_DUMMY(1),
I3=>Q_DUMMY(0),
O=>T4);

I_36_33 : VCC
port map (P=>XLXN_1);

I_36_34 : AND2
port map (I0=>Q_DUMMY(1),
I1=>Q_DUMMY(0),
O=>T2);

I_36_35 : AND3
port map (I0=>Q_DUMMY(2),
I1=>Q_DUMMY(1),
I2=>Q_DUMMY(0),
library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;

component CB16RE_MXILINX_ckuart
port ( C : in    std_logic;
       CE : in    std_logic;
       R : in    std_logic;
       CEO : out  std_logic;
       Q : out  std_logic_vector (15 downto 0);
       TC : out  std_logic);
end component;

component FTCE_MXILINX_ckuart
port ( C : in    std_logic;
       CE : in    std_logic;
       CLR : in    std_logic;
       T : in    std_logic;
       Q : out  std_logic);
end component;

component GND
end component;

port (G : out std_logic);
end component;
attribute BOX_TYPE of GND : component is "BLACK_BOX";

component VCC
port (P : out std_logic);
end component;
attribute BOX_TYPE of VCC : component is "BLACK_BOX";

component match16
port (dtin : in std_logic_vector (15 downto 0);
     ctin : in std_logic_vector (15 downto 0);
     matcho : out std_logic);
end component;

attribute HU_SET of XLXI_1 : label is "XLXI_1_16";
attribute HU_SET of XLXI_2 : label is "XLXI_2_17";
begin
XLXI_1 : CB16RE_MXILINX_ckuart
port map (C=>cki,
         CE=>XLXN_3,
         R=>XLXN_6,
         CEO=>open,
         Q(15 downto 0)=>XLXN_7(15 downto 0),
         TC=>open);

XLXI_2 : FTCE_MXILINX_ckuart
port map (C=>cki,
         CE=>XLXN_6,
         CLR=>XLXN_4,
         T=>XLXN_2,
         Q=>dvcko);

XLXI_3 : GND
port map (G=>XLXN_4);

XLXI_4 : VCC
port map (P=>XLXN_3);

XLXI_5 : VCC
port map (P=>XLXN_2);

XLXI_6 : match16
port map (ctin(15 downto 0)=>XLXN_7(15 downto 0),
         dtin(15 downto 0)=>dti(15 downto 0),
         matcho=>XLXN_6);
end BEHAVIORAL;
library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;

entity AND16_MXILINX_match16 is
  port ( I0 : in std_logic;
         I1 : in std_logic;
         I2 : in std_logic;
         I3 : in std_logic;
         I4 : in std_logic;
         I5 : in std_logic;
         I6 : in std_logic;
         I7 : in std_logic;
         I8 : in std_logic;
         I9 : in std_logic;
         I10 : in std_logic;
         I11 : in std_logic;
         I12 : in std_logic;
         I13 : in std_logic;
         I14 : in std_logic;
         I15 : in std_logic;
         O : out std_logic);
end AND16_MXILINX_match16;
architecture BEHAVIORAL of AND16_MXILINX_match16 is
attribute BOX_TYPE : string;
attribute RLOC : string;
signal CIN : std_logic;
signal C0 : std_logic;
signal C1 : std_logic;
signal C2 : std_logic;
signal S0 : std_logic;
signal S1 : std_logic;
signal S2 : std_logic;
signal S3 : std_logic;
signal XLXN_46 : std_logic;
component MUXCY_L
  port ( CI : in std_logic;
        DI : in std_logic;
        S : in std_logic;
        LO : out std_logic);
end component;
attribute BOX_TYPE of MUXCY_L : component is "BLACK_BOX";
component FMAP
  port ( I1 : in std_logic;
        I2 : in std_logic;
        I3 : in std_logic;
        I4 : in std_logic;
        O : in std_logic);
end component;
attribute BOX_TYPE of FMAP : component is "BLACK_BOX";
component VCC
  port ( P : out std_logic);
end component;
attribute BOX_TYPE of VCC : component is "BLACK_BOX";
component GND
  port ( G : out std_logic);
end component;
attribute BOX_TYPE of GND : component is "BLACK_BOX";
component AND4
  port ( I0 : in std_logic;
        I1 : in std_logic;
        I2 : in std_logic;
        I3 : in std_logic;
        O : out std_logic);
end component;
attribute BOX_TYPE of AND4 : component is "BLACK_BOX";
component MUXCY
port (CI : in std_logic;
    DI : in std_logic;
    S : in std_logic;
    O : out std_logic);
end component;
attribute BOX_TYPE of MUXCY : component is "BLACK_BOX";

attribute RLOC of I_36_2 : label is "X0Y0";
attribute RLOC of I_36_29 : label is "X0Y0";
attribute RLOC of I_36_129 : label is "X0Y0";
attribute RLOC of I_36_138 : label is "X0Y0";
attribute RLOC of I_36_142 : label is "X0Y1";
attribute RLOC of I_36_147 : label is "X0Y1";
attribute RLOC of I_36_165 : label is "X0Y1";
attribute RLOC of I_36_170 : label is "X0Y1";
begi
I_36_2 : MUXCY_L
    port map (CI=>CIN,
            DI=>XLXN_46,
            S=>S0,
            LO=>C0);

I_36_29 : FMAP
    port map (I1=>I0,
              I2=>I1,
              I3=>I2,
              I4=>I3,
              O=>S0);

I_36_107 : VCC
    port map (P=>CIN);

I_36_109 : GND
    port map (G=>XLXN_46);

I_36_110 : AND4
    port map (I0=>I0,
             I1=>I1,
             I2=>I2,
             I3=>I3,
             O=>S0);

I_36_127 : AND4
    port map (I0=>I4,
             I1=>I5,
             I2=>I6,
             I3=>I7,
             O=>S1);
I_36_129 : MUXCY_L
  port map (CI=>C0,
           DI=>XLXN_46,
           S=>S1,
           LO=>C1);

I_36_138 : FMAP
  port map (I1=>I4,
           I2=>I5,
           I3=>I6,
           I4=>I7,
           O=>S1);

I_36_142 : FMAP
  port map (I1=>I8,
           I2=>I9,
           I3=>I10,
           I4=>I11,
           O=>S2);

I_36_147 : MUXCY_L
  port map (CI=>C1,
           DI=>XLXN_46,
           S=>S2,
           LO=>C2);

I_36_151 : AND4
  port map (I0=>I8,
           I1=>I9,
           I2=>I10,
           I3=>I11,
           O=>S2);

I_36_161 : AND4
  port map (I0=>I12,
           I1=>I13,
           I2=>I14,
           I3=>I15,
           O=>S3);

I_36_165 : MUXCY
  port map (CI=>C2,
           DI=>XLXN_46,
           S=>S3,
           O=>O);

I_36_170 : FMAP
  port map (I1=>I12,
           I2=>I13,
I3 => I14,
I4 => I15,
O => S3);
end BEHAVIORAL;

library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;

entity match16 is
  port ( ctin : in    std_logic_vector (15 downto 0);
         dtin : in    std_logic_vector (15 downto 0);
         matcho : out  std_logic);
end match16;

architecture BEHAVIORAL of match16 is
  attribute BOX_TYPE : string ;
  attribute HU_SET : string ;
  signal XLXN_36 : std_logic;
  signal XLXN_37 : std_logic;
  signal XLXN_38 : std_logic;
  signal XLXN_39 : std_logic;
  signal XLXN_40 : std_logic;
  signal XLXN_41 : std_logic;
  signal XLXN_42 : std_logic;
  signal XLXN_43 : std_logic;
  signal XLXN_44 : std_logic;
  signal XLXN_45 : std_logic;
  signal XLXN_46 : std_logic;
  signal XLXN_47 : std_logic;
  signal XLXN_48 : std_logic;
  signal XLXN_49 : std_logic;
  signal XLXN_50 : std_logic;
  signal XLXN_51 : std_logic;
  component XNOR2
    port ( I0 : in    std_logic;
           I1 : in    std_logic;
           O : out  std_logic);
  end component;
  attribute BOX_TYPE of XNOR2 : component is "BLACK_BOX";
  component AND16_MXILINX_match16
    port ( I0 : in    std_logic;
           I1 : in    std_logic;
           I2 : in    std_logic;
           I3 : in    std_logic;
           I4 : in    std_logic;
           I5 : in    std_logic;
           I6 : in    std_logic;
           I7 : in    std_logic;
           I8 : in    std_logic;
           I9 : in    std_logic;
           I10 : in    std_logic;
           I11 : in    std_logic;
           I12 : in    std_logic;
           I13 : in    std_logic;
           I14 : in    std_logic;
           I15 : in    std_logic;
           I16 : in    std_logic;
           dtin : in    std_logic_vector (15 downto 0);
           matcho : out  std_logic);
  end component;
end;

I10 : in    std_logic;
I11 : in    std_logic;
I12 : in    std_logic;
I13 : in    std_logic;
I14 : in    std_logic;
I15 : in    std_logic;
I2  : in    std_logic;
I3  : in    std_logic;
I4  : in    std_logic;
I5  : in    std_logic;
I6  : in    std_logic;
I7  : in    std_logic;
I8  : in    std_logic;
I9  : in    std_logic;
O   : out   std_logic);
end component;

attribute HU_SET of XLXI_17 : label is "XLXI_17_0";
begin
XLXI_1 : XNOR2
port map (I0=>ctin(1),
I1=>dtin(1),
O=>XLXN_37);

XLXI_2 : XNOR2
port map (I0=>ctin(2),
I1=>dtin(2),
O=>XLXN_38);

XLXI_3 : XNOR2
port map (I0=>ctin(3),
I1=>dtin(3),
O=>XLXN_39);

XLXI_4 : XNOR2
port map (I0=>ctin(4),
I1=>dtin(4),
O=>XLXN_40);

XLXI_5 : XNOR2
port map (I0=>ctin(5),
I1=>dtin(5),
O=>XLXN_41);

XLXI_6 : XNOR2
port map (I0=>ctin(6),
I1=>dtin(6),
O=>XLXN_42);
XLXI_7 : XNOR2
  port map (I0=>ctin(7),
            I1=>dtin(7),
            O=>XLXN_43);

XLXI_8 : XNOR2
  port map (I0=>ctin(8),
            I1=>dtin(8),
            O=>XLXN_44);

XLXI_9 : XNOR2
  port map (I0=>ctin(9),
            I1=>dtin(9),
            O=>XLXN_45);

XLXI_10 : XNOR2
  port map (I0=>ctin(10),
            I1=>dtin(10),
            O=>XLXN_46);

XLXI_11 : XNOR2
  port map (I0=>ctin(11),
            I1=>dtin(11),
            O=>XLXN_47);

XLXI_12 : XNOR2
  port map (I0=>ctin(12),
            I1=>dtin(12),
            O=>XLXN_48);

XLXI_13 : XNOR2
  port map (I0=>ctin(13),
            I1=>dtin(13),
            O=>XLXN_49);

XLXI_14 : XNOR2
  port map (I0=>ctin(14),
            I1=>dtin(14),
            O=>XLXN_50);

XLXI_15 : XNOR2
  port map (I0=>ctin(15),
            I1=>dtin(15),
            O=>XLXN_51);

XLXI_16 : XNOR2
  port map (I0=>ctin(0),
            I1=>dtin(0),
            O=>XLXN_36);
XLXI_17 : AND16_MXILINX_match16
port map (I0=>XLXN_51,
    I1=>XLXN_50,
    I2=>XLXN_49,
    I3=>XLXN_48,
    I4=>XLXN_47,
    I5=>XLXN_46,
    I6=>XLXN_45,
    I7=>XLXN_44,
    I8=>XLXN_43,
    I9=>XLXN_42,
    I10=>XLXN_41,
    I11=>XLXN_40,
    I12=>XLXN_39,
    I13=>XLXN_38,
    I14=>XLXN_37,
    I15=>XLXN_36,
    O=>matcho);
end BEHAVIORAL;

-- Copyright (c) 1995-2007 Xilinx, Inc. All rights reserved.
---------------------------------------------------------------------------
-- ______   ______
-- / \ / \ Vendor: Xilinx
-- \ \ \ \ Version : 9.2.04i
-- \ \ \ \ Application : sch2vhdl
-- / / Filename : txcntc1a.vhf
-- \ \ / \ \ Timestamp : 07/04/2008 09:22:04
-- \ \ / \ \ -- \__\__\__\__\__
-- Commander: C:\Xilinx92i\bin\nt\sch2vhdl.exe -intstyle ise -family spartan3 -flat -suppress -w
C:\Thesis\Sim2_new/txcntc1a.sch txcntc1a.vhf
--Design Name: txcntc1a
--Device: spartan3
--Purpose:
-- This vhdl netlist is translated from an ECS schematic. It can be
-- synthesized and simulated, but it should not be modified.
--
library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;

entity FTRSE_MXILINX_txcntc1a is
  port ( C : in    std_logic;
         CE : in    std_logic;
         R  : in    std_logic;
         S  : in    std_logic;
         T  : in    std_logic;
         Q  : out   std_logic);
end FTRSE_MXILINX_txcntc1a;

architecture BEHAVIORAL of FTRSE_MXILINX_txcntc1a is
  attribute BOX_TYPE   : string ;
  attribute INIT       : string ;
  attribute RLOC       : string ;
  signal CE_S    : std_logic;
  signal D_S     : std_logic;
  signal TQ      : std_logic;
  signal Q_DUMMY : std_logic;
  component XOR2
      port ( I0 : in    std_logic;
             I1 : in    std_logic;
             O  : out   std_logic);
  end component;
  attribute BOX_TYPE of XOR2 : component is "BLACK_BOX";

  component FDRE
      -- synopsys translate_off
      generic( INIT : bit :=  '0');
      -- synopsys translate_on
      port ( C : in    std_logic;
             CE : in    std_logic;
             D  : in    std_logic;
             R  : in    std_logic;
             Q  : out   std_logic);
  end component;
  attribute INIT of FDRE : component is "0";
  attribute BOX_TYPE of FDRE : component is "BLACK_BOX";

  component OR2
      port ( I0 : in    std_logic;
             I1 : in    std_logic;
             O  : out   std_logic);
  end component;
  attribute BOX_TYPE of OR2 : component is "BLACK_BOX";

  attribute RLOC of I_36_35 : label is "X0Y0";
begin
  Q <= Q_DUMMY;
end;
I_36_32 : XOR2
port map (I0=>T,
I1=>Q_DUMMY,
O=>TQ);

I_36_35 : FDRE
port map (C=>C,
CE=>CE_S,
D=>D_S,
R=>R,
Q=>Q_DUMMY);

I_36_73 : OR2
port map (I0=>S,
I1=>TQ,
O=>D_S);

I_36_77 : OR2
port map (I0=>CE,
I1=>S,
O=>CE_S);
end BEHAVIORAL;

library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;

entity CB2RE_MXILINX_txcntc1a is
port ( C : in   std_logic;
   CE : in   std_logic;
    R : in   std_logic;
   CEO : out  std_logic;
   Q0 : out  std_logic;
   Q1 : out  std_logic;
    TC : out  std_logic);
end CB2RE_MXILINX_txcntc1a;

architecture BEHAVIORAL of CB2RE_MXILINX_txcntc1a is
attribute HU_SET     : string :=
attribute BOX_TYPE   : string :=
signal XLXN_1   : std_logic;
signal XLXN_2   : std_logic;
signal Q0_DUMMY : std_logic;
signal Q1_DUMMY : std_logic;
signal TC_DUMMY : std_logic;
component FTRSE_MXILINX_txcntc1a
  port ( C : in std_logic;
  CE : in std_logic;
  R : in std_logic;
  S : in std_logic;
  T : in std_logic;
  Q : out std_logic);
end component;

component AND2
  port ( I0 : in std_logic;
  I1 : in std_logic;
  O : out std_logic);
end component;
attribute BOX_TYPE of AND2 : component is "BLACK_BOX";

component VCC
  port ( P : out std_logic);
end component;
attribute BOX_TYPE of VCC : component is "BLACK_BOX";

component GND
  port ( G : out std_logic);
end component;
attribute BOX_TYPE of GND : component is "BLACK_BOX";

attribute HU_SET of I_Q0 : label is "I_Q0_0";
attribute HU_SET of I_Q1 : label is "I_Q1_1";
begin
  Q0 <= Q0_DUMMY;
  Q1 <= Q1_DUMMY;
  TC <= TC_DUMMY;
  I_Q0 : FTRSE_MXILINX_txcntc1a
    port map (C=>C,
               CE=>CE,
               R=>R,
               S=>XLXN_2,
               T=>XLXN_1,
               Q=>Q0_DUMMY);
  I_Q1 : FTRSE_MXILINX_txcntc1a
    port map (C=>C,
               CE=>CE,
               R=>R,
               S=>XLXN_2,
               T=>Q0_DUMMY,
               Q=>Q1_DUMMY);
I_36_37 : AND2
  port map (I0=>Q1_DUMMY,
           I1=>Q0_DUMMY,
           O=>TC_DUMMY);

I_36_47 : VCC
  port map (P=>XLXN_1);

I_36_54 : GND
  port map (G=>XLXN_2);

I_36_55 : AND2
  port map (I0=>CE,
           I1=>TC_DUMMY,
           O=>CEO);

end BEHAVIORAL;

library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;

entity FTCE_MXILINX_txcntc1a is
  port ( C   : in    std_logic;
         CE  : in    std_logic;
         CLR : in    std_logic;
         T   : in    std_logic;
         Q   : out   std_logic);
end FTCE_MXILINX_txcntc1a;

architecture BEHAVIORAL of FTCE_MXILINX_txcntc1a is
  attribute BOX_TYPE   : string := "BLACK_BOX";
  attribute INIT       : string :
  attribute RLOC       : string :
  signal TQ      : std_logic;
  signal Q_DUMMY : std_logic;
  component XOR2
    port ( I0 : in    std_logic;
           I1 : in    std_logic;
           O  : out   std_logic);
  end component;
  component FDCE
    -- synopsys translate_off
generic( INIT : bit := '0');
-- synopsys translate on
port ( C : in    std_logic;
    CE : in    std_logic;
    CLR : in    std_logic;
    D : in    std_logic;
    Q : out   std_logic);
end component;
attribute INIT of FDCE : component is "0";
attribute BOX_TYPE of FDCE : component is "BLACK_BOX";

attribute RLOC of I_36_35 : label is "X0Y0";
begin
  Q <= Q_DUMMY;
I_36_32 : XOR2
    port map (I0=>T,
              I1=>Q_DUMMY,
              O=>TQ);

I_36_35 : FDCE
    port map (C=>C,
              CE=>CE,
              CLR=>CLR,
              D=>TQ,
              Q=>Q_DUMMY);
end BEHAVIORAL;

library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;

entity CB8CE_MXILINX_txcntc1a is
  port ( C   : in    std_logic;
         CE  : in    std_logic;
         CLR : in    std_logic;
         CEO : out   std_logic;
         Q   : out   std_logic_vector (7 downto 0);
         TC  : out   std_logic);
end CB8CE_MXILINX_txcntc1a;

architecture BEHAVIORAL of CB8CE_MXILINX_txcntc1a is
  attribute HU_SET     : string ;
  attribute BOX_TYPE   : string ;
signal T2 : std_logic;
signal T3       : std_logic;
signal T4       : std_logic;
signal T5       : std_logic;
signal T6       : std_logic;
signal T7       : std_logic;
signal XLXN_1   : std_logic;
signal Q_DUMMY  : std_logic_vector (7 downto 0);
signal TC_DUMMY : std_logic;
component FTCE_MXILINX_t
  port ( C   : in    std_logic;
         CE  : in    std_logic;
         CLR : in    std_logic;
         T   : in    std_logic;
         Q   : out   std_logic);
end component;
component AND5
  port ( I0 : in    std_logic;
         I1 : in    std_logic;
         I2 : in    std_logic;
         I3 : in    std_logic;
         I4 : in    std_logic;
         O  : out   std_logic);
end component;
attribute BOX_TYPE of AND5 : component is "BLACK_BOX";
component AND2
  port ( I0 : in    std_logic;
         I1 : in    std_logic;
         O  : out   std_logic);
end component;
attribute BOX_TYPE of AND2 : component is "BLACK_BOX";
component AND3
  port ( I0 : in    std_logic;
         I1 : in    std_logic;
         I2 : in    std_logic;
         O  : out   std_logic);
end component;
attribute BOX_TYPE of AND3 : component is "BLACK_BOX";
component AND4
  port ( I0 : in    std_logic;
         I1 : in    std_logic;
         I2 : in    std_logic;
         I3 : in    std_logic;
         O  : out   std_logic);
end component;
attribute BOX_TYPE of AND4 : component is "BLACK_BOX";
component VCC
    port ( P : out std_logic);
end component;
attribute BOX_TYPE of VCC : component is "BLACK_BOX";

attribute HU_SET of I_Q0 : label is "I_Q0_8";
attribute HU_SET of I_Q1 : label is "I_Q1_9";
attribute HU_SET of I_Q2 : label is "I_Q2_5";
attribute HU_SET of I_Q3 : label is "I_Q3_6";
attribute HU_SET of I_Q4 : label is "I_Q4_7";
attribute HU_SET of I_Q5 : label is "I_Q5_4";
attribute HU_SET of I_Q6 : label is "I_Q6_3";
attribute HU_SET of I_Q7 : label is "I_Q7_2";
begin
Q(7 downto 0) <= Q_DUMMY(7 downto 0);
TC <= TC_DUMMY;
I_Q0 : FTCE_MXILINX_txcntc1a
    port map (C=>C,
              CE=>CE,
              CLR=>CLR,
              T=>XLXN_1,
              Q=>Q_DUMMY(0));
I_Q1 : FTCE_MXILINX_txcntc1a
    port map (C=>C,
              CE=>CE,
              CLR=>CLR,
              T=>Q_DUMMY(0),
              Q=>Q_DUMMY(1));
I_Q2 : FTCE_MXILINX_txcntc1a
    port map (C=>C,
              CE=>CE,
              CLR=>CLR,
              T=>T2,
              Q=>Q_DUMMY(2));
I_Q3 : FTCE_MXILINX_txcntc1a
    port map (C=>C,
              CE=>CE,
              CLR=>CLR,
              T=>T3,
              Q=>Q_DUMMY(3));
I_Q4 : FTCE_MXILINX_txcntc1a
    port map (C=>C,
              CE=>CE,
              CLR=>CLR,
T=>T4,
Q=>Q_DUMMY(4));

I_Q5 : FTCE_MXILINX_txcntc1a
port map (C=>C,
    CE=>CE,
    CLR=>CLR,
    T=>T5,
    Q=>Q_DUMMY(5));

I_Q6 : FTCE_MXILINX_txcntc1a
port map (C=>C,
    CE=>CE,
    CLR=>CLR,
    T=>T6,
    Q=>Q_DUMMY(6));

I_Q7 : FTCE_MXILINX_txcntc1a
port map (C=>C,
    CE=>CE,
    CLR=>CLR,
    T=>T7,
    Q=>Q_DUMMY(7));

I_36_1 : AND5
port map (I0=>Q_DUMMY(7),
    I1=>Q_DUMMY(6),
    I2=>Q_DUMMY(5),
    I3=>Q_DUMMY(4),
    I4=>T4,
    O=>TC_DUMMY);

I_36_2 : AND2
port map (I0=>Q_DUMMY(4),
    I1=>T4,
    O=>T5);

I_36_11 : AND3
port map (I0=>Q_DUMMY(5),
    I1=>Q_DUMMY(4),
    I2=>T4,
    O=>T6);

I_36_15 : AND4
port map (I0=>Q_DUMMY(3),
    I1=>Q_DUMMY(2),
    I2=>Q_DUMMY(1),
    I3=>Q_DUMMY(0),
    O=>T4);
library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;

entity txcntc1a is
  port ( tcki : in std_logic;
         txwi : in std_logic;
         shcko : out std_logic;
         txf : out std_logic;
         txlo : out std_logic;
         txrwo : out std_logic);
end txcntc1a;

architecture BEHAVIORAL of txcntc1a is
  attribute BOX_TYPE : string ;
  attribute HU_SET : string ;

I_36_16 : VCC
  port map (P=>XLXN_1);

I_36_24 : AND2
  port map (I0=>Q_DUMMY(1),
            I1=>Q_DUMMY(0),
            O=>T2);

I_36_26 : AND3
  port map (I0=>Q_DUMMY(2),
            I1=>Q_DUMMY(1),
            I2=>Q_DUMMY(0),
            O=>T3);

I_36_28 : AND4
  port map (I0=>Q_DUMMY(6),
            I1=>Q_DUMMY(5),
            I2=>Q_DUMMY(4),
            I3=>T4,
            O=>T7);

I_36_31 : AND2
  port map (I0=>CE,
            I1=>TC_DUMMY,
            O=>CEO);

end BEHAVIORAL;
attribute INIT : string;
signal cto : std_logic_vector (7 downto 0);
signal XLXN_80 : std_logic;
signal XLXN_115 : std_logic;
signal XLXN_116 : std_logic;
signal XLXN_127 : std_logic;
signal txrwo_DUMMY : std_logic;
signal txf_DUMMY : std_logic;

component VCC
  port ( P : out std_logic);
end component;
attribute BOX_TYPE of VCC : component is "BLACK_BOX";

component BUF
  port ( I : in std_logic;
        O : out std_logic);
end component;
attribute BOX_TYPE of BUF : component is "BLACK_BOX";

component CB8CE_MXILINX_txcntc1a
  port ( C : in std_logic;
        CE : in std_logic;
        CLR : in std_logic;
        CEO : out std_logic;
        Q : out std_logic_vector (7 downto 0);
        TC : out std_logic);
end component;

component FDCE
  -- synopsys translate_off
  generic(INIT : bit := '0');
  -- synopsys translate_on
  port ( C : in std_logic;
        CE : in std_logic;
        CLR : in std_logic;
        D : in std_logic;
        Q : out std_logic);
end component;
attribute INIT of FDCE : component is "0";
attribute BOX_TYPE of FDCE : component is "BLACK_BOX";

component CB2RE_MXILINX_txcntc1a
  port ( C : in std_logic;
        CE : in std_logic;
        R : in std_logic;
        CEO : out std_logic;
        Q0 : out std_logic;
        Q1 : out std_logic;
        TC : out std_logic);
end component;

component txdt009
    port ( dti : in    std_logic_vector (7 downto 0);
          dtcki : in    std_logic;
          dto   : out   std_logic);
end component;

component txdt169
    port ( dti : in    std_logic_vector (7 downto 0);
          dtcki : in    std_logic;
          dto   : out   std_logic);
end component;

attribute HU_SET of XLXI_52 : label is "XLXI_52_10";
attribute HU_SET of XLXI_65 : label is "XLXI_65_11";
begin
    txf <= txf_DUMMY;
    txrwo <= txrwo_DUMMY;
    XLXI_32 : VCC
        port map (P=>XLXN_80);
    XLXI_51 : BUF
        port map (I=>cto(3),
                  O=>shcko);
    XLXI_52 : CB8CE_MXILINX_txcntc1a
        port map (C=>tcki,
                  CE=>txf_DUMMY,
                  CLR=>XLXN_116,
                  CEO=>open,
                  Q(7 downto 0)=>cto(7 downto 0),
                  TC=>open);
    XLXI_53 : FDCE
        port map (C=>txwi,
                  CE=>XLXN_80,
                  CLR=>XLXN_115,
                  D=>XLXN_80,
                  Q=>txlo);
    XLXI_54 : FDCE
        port map (C=>txwi,
                  CE=>XLXN_80,
                  CLR=>XLXN_116,
                  D=>XLXN_80,
                  Q=>txf_DUMMY);
    XLXI_65 : CB2RE_MXILINX_txcntc1a

port map (C=>tcki,
    CE=>XLXN_127,
    R=>txrwo_DUMMY,
    CEO=>open,
    Q0=>open,
    Q1=>txrwo_DUMMY,
    TC=>open);

XLXI_66 : FDCE
    port map (C=>XLXN_116,
        CE=>XLXN_80,
        CLR=>txrwo_DUMMY,
        D=>XLXN_80,
        Q=>XLXN_127);

XLXI_67 : txdt009
    port map (dtcki=>tcki,
        dti(7 downto 0)=>cto(7 downto 0),
        dto=>XLXN_115);

XLXI_68 : txdt169
    port map (dtcki=>tcki,
        dti(7 downto 0)=>cto(7 downto 0),
        dto=>XLXN_116);

end BEHAVIORAL;

-- Copyright (c) 1995-2007 Xilinx, Inc. All rights reserved.
library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;

entity txdt009 is
  port ( dtcki : in    std_logic;
         dti   : in    std_logic_vector (7 downto 0);
         dto   : out   std_logic);
end txdt009;

architecture BEHAVIORAL of txdt009 is
  attribute BOX_TYPE   : string := "BLACK_BOX";
  attribute INIT       : string := "0";
  signal XLXN_110  : std_logic;
  signal XLXN_111  : std_logic;
  signal XLXN_124  : std_logic;
  signal XLXN_125  : std_logic;
  signal dto_DUMMY : std_logic;
  component AND2
    port ( I0 : in    std_logic;
           I1 : in    std_logic;
           O  : out   std_logic);
  end component;
  signal XLXN_124  : std_logic;
  signal XLXN_125  : std_logic;
  signal dto_DUMMY : std_logic;
  component AND4B4
    port ( I0 : in    std_logic;
           I1 : in    std_logic;
           I2 : in    std_logic;
           I3 : in    std_logic;
           O  : out   std_logic);
  end component;
  signal XLXN_124  : std_logic;
  signal XLXN_125  : std_logic;
  signal dto_DUMMY : std_logic;
  component AND4B2
    port ( I0 : in    std_logic;
           I1 : in    std_logic;
           I2 : in    std_logic;
           I3 : in    std_logic;
           O  : out   std_logic);
  end component;
  signal XLXN_124  : std_logic;
  signal XLXN_125  : std_logic;
  signal dto_DUMMY : std_logic;
  component FDRE
    -- synopsys translate_off
    generic( INIT : bit := '0');
-- synopsys translate_on
port ( C : in    std_logic;
      CE : in    std_logic;
      D : in    std_logic;
      R : in    std_logic;
      Q : out   std_logic);
end component;
attribute INIT of FDRE : component is "0";
attribute BOX_TYPE of FDRE : component is "BLACK_BOX";

component VCC
  port ( P : out std_logic);
end component;
attribute BOX_TYPE of VCC : component is "BLACK_BOX";

begin
  dto <= dto_DUMMY;
  XLXI_51 : AND2
    port map (I0=>XLXN_110,
              I1=>XLXN_111,
              O=>XLXN_124);

  XLXI_52 : AND4B4
    port map (I0=>dti(4),
              I1=>dti(5),
              I2=>dti(6),
              I3=>dti(7),
              O=>XLXN_111);

  XLXI_53 : AND4B2
    port map (I0=>dti(1),
              I1=>dti(2),
              I2=>dti(0),
              I3=>dti(3),
              O=>XLXN_110);

  XLXI_54 : FDRE
    port map (C=>dtcki,
              CE=>XLXN_124,
              D=>XLXN_125,
              R=>dto_DUMMY,
              Q=>dto_DUMMY);

  XLXI_58 : VCC
    port map (P=>XLXN_125);

end BEHAVIORAL;
-- Copyright (c) 1995-2007 Xilinx, Inc. All rights reserved.

-- Command: C:\Xilinx92i\bin\nt\sch2vhdl.exe -intstyle ise -family spartan3 -flat -suppress -w C:/Thesis/Sim2_new/txdt169.sch txdt169.vhf
-- Design Name: txdt169
-- Device: spartan3
-- Purpose:
-- This vhdl netlist is translated from an ECS schematic. It can be
-- synthesis and simulated, but it should not be modified.
--

library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;

entity txdt169 is
  port ( dtcki : in    std_logic;
         dti   : in    std_logic_vector (7 downto 0);
         dto   : out   std_logic);
end txdt169;

architecture BEHAVIORAL of txdt169 is
  attribute BOX_TYPE   : string ;
  attribute INIT       : string ;
  signal XLXN_110  : std_logic;
  signal XLXN_111  : std_logic;
  signal XLXN_112  : std_logic;
  signal XLXN_113  : std_logic;
  signal dto_DUMMY : std_logic;
  component AND4
    port ( I0 : in    std_logic;
          I1 : in    std_logic;
          I2 : in    std_logic;
          I3 : in    std_logic;
          O  : out   std_logic);
  end component;
end architecture;
attribute BOX_TYPE of AND4 : component is "BLACK_BOX";

component AND4B4
    port ( I0 : in    std_logic;
          I1 : in    std_logic;
          I2 : in    std_logic;
          I3 : in    std_logic;
          O  : out   std_logic);
end component;
attribute BOX_TYPE of AND4B4 : component is "BLACK_BOX";

component AND2
    port ( I0 : in    std_logic;
          I1 : in    std_logic;
          O  : out   std_logic);
end component;
attribute BOX_TYPE of AND2 : component is "BLACK_BOX";

component FDRE
    -- synopsys translate_off
    generic( INIT : bit := '0');
    -- synopsys translate_on
    port ( C  : in    std_logic;
           CE : in    std_logic;
           D  : in    std_logic;
           R  : in    std_logic;
           Q  : out   std_logic);
end component;
attribute INIT of FDRE : component is "0";
attribute BOX_TYPE of FDRE : component is "BLACK_BOX";

component VCC
    port ( P : out   std_logic);
end component;
attribute BOX_TYPE of VCC : component is "BLACK_BOX";

begin
    dto <= dto_DUMMY;
    XLXI_49 : AND4
        port map (I0=>dti(0),
                  I1=>dti(3),
                  I2=>dti(5),
                  I3=>dti(7),
                  O=>XLXN_111);

    XLXI_50 : AND4B4
        port map (I0=>dti(1),
                  I1=>dti(2),
                  I2=>dti(4),
library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;

I3=>dti(6),
O=>XLXN_110);

XLXI_51 : AND2
  port map (I0=>XLXN_110,
            I1=>XLXN_111,
            O=>XLXN_112);

XLXI_52 : FDRE
  port map (C=>dtcki,
            CE=>XLXN_112,
            D=>XLXN_113,
            R=>dto_DUMMY,
            Q=>dto_DUMMY);

XLXI_53 : VCC
  port map (P=>XLXN_113);

end BEHAVIORAL;

-- Copyright (c) 1995-2007 Xilinx, Inc. All rights reserved.

-- Command: C:\Xilinx92i\bin\nt\sch2vhdl.exe -intstyle ise -family spartan3 -flat -suppress -w
C:\Thesis/Sim2_new/txdata.sch txdata.vhf
--Design Name: txdata
--Device: spartan3
--Purpose:
-- This vhdl netlist is translated from an ECS schematic. It can be
-- synthesis and simulated, but it should not be modified.
--
entity FD8CE_MXILINX_txdata is
  port ( C : in    std_logic;
         CE : in    std_logic;
         CLR : in    std_logic;
         D : in    std_logic_vector (7 downto 0);
         Q : out   std_logic_vector (7 downto 0));
end FD8CE_MXILINX_txdata;

architecture BEHAVIORAL of FD8CE_MXILINX_txdata is
  attribute INIT       : string ;
  attribute BOX_TYPE   : string ;
  component FDCE
    -- synopsys translate_off
    generic( INIT : bit := '0');
    -- synopsys translate_on
    port ( C : in    std_logic;
           CE : in    std_logic;
           CLR : in    std_logic;
           D : in    std_logic;
           Q : out   std_logic);
  end component;
  attribute INIT of FDCE : component is "0";
  attribute BOX_TYPE of FDCE : component is "BLACK_BOX";
begin
  I_Q0 : FDCE
    port map (C=>C,
              CE=>CE,
              CLR=>CLR,
              D=>D(0),
              Q=>Q(0));

  I_Q1 : FDCE
    port map (C=>C,
              CE=>CE,
              CLR=>CLR,
              D=>D(1),
              Q=>Q(1));

  I_Q2 : FDCE
    port map (C=>C,
              CE=>CE,
              CLR=>CLR,
              D=>D(2),
              Q=>Q(2));

  I_Q3 : FDCE
    port map (C=>C,
CE=>CE,
CLR=>CLR,
D=>D(3),
Q=>Q(3));

I_Q4 : FDCE
port map (C=>C,
CE=>CE,
CLR=>CLR,
D=>D(4),
Q=>Q(4));

I_Q5 : FDCE
port map (C=>C,
CE=>CE,
CLR=>CLR,
D=>D(5),
Q=>Q(5));

I_Q6 : FDCE
port map (C=>C,
CE=>CE,
CLR=>CLR,
D=>D(6),
Q=>Q(6));

I_Q7 : FDCE
port map (C=>C,
CE=>CE,
CLR=>CLR,
D=>D(7),
Q=>Q(7));

drop BEHAVIORAL;

library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;

entity M2_1_MXILINX_txdata is
port ( D0 : in std_logic;
       D1 : in std_logic;
       S0 : in std_logic;
       Q : out std_logic);
end M2_1_MXILINX_txdata;
architecture BEHAVIORAL of M2_1_MXILINX_txdata is
  attribute BOX_TYPE : string :
  signal M0 : std_logic;
  signal M1 : std_logic;
  component AND2B1
    port ( I0 : in std_logic;
           I1 : in std_logic;
           O  : out std_logic);
  end component;
  attribute BOX_TYPE of AND2B1 : component is "BLACK_BOX";

  component OR2
    port ( I0 : in std_logic;
           I1 : in std_logic;
           O  : out std_logic);
  end component;
  attribute BOX_TYPE of OR2 : component is "BLACK_BOX";

  component AND2
    port ( I0 : in std_logic;
           I1 : in std_logic;
           O  : out std_logic);
  end component;
  attribute BOX_TYPE of AND2 : component is "BLACK_BOX";

begin
  I_36_7 : AND2B1
    port map (I0=>S0,
              I1=>D0,
              O=>M0);

  I_36_8 : OR2
    port map (I0=>M1,
              I1=>M0,
              O=>O);

  I_36_9 : AND2
    port map (I0=>D1,
              I1=>S0,
              O=>M1);
end BEHAVIORAL;

library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;

entity SR16CLE_MXILINX_txdata is
  port ( C   : in    std_logic;
         CE  : in    std_logic;
         CLR : in    std_logic;
         D   : in    std_logic_vector (15 downto 0);
         L   : in    std_logic;
         SLI : in    std_logic;
         Q   : out   std_logic_vector (15 downto 0));
end SR16CLE_MXILINX_txdata;

architecture BEHAVIORAL of SR16CLE_MXILINX_txdata is
  attribute HU_SET     : string     :
  attribute INIT       : string     :
  attribute BOX_TYPE   : string     :
  signal L_OR_CE : std_logic;
  signal MD0     : std_logic;
  signal MD1     : std_logic;
  signal MD2     : std_logic;
  signal MD3     : std_logic;
  signal MD4     : std_logic;
  signal MD5     : std_logic;
  signal MD6     : std_logic;
  signal MD7     : std_logic;
  signal MD8     : std_logic;
  signal MD9     : std_logic;
  signal MD10    : std_logic;
  signal MD11    : std_logic;
  signal MD12    : std_logic;
  signal MD13    : std_logic;
  signal MD14    : std_logic;
  signal MD15    : std_logic;
  signal Q_DUMMY : std_logic_vector (15 downto 0);
component M2_1_MXILINX_txdata
  port ( D0 : in    std_logic;
         D1 : in    std_logic;
         S0 : in    std_logic;
         O  : out   std_logic);
end component;

component FDCE
  -- synopsys translate_off
  generic( INIT : bit := '0');
  -- synopsys translate_on
  port ( C   : in    std_logic;
         CE  : in    std_logic;
         CLR : in    std_logic;
D : in std_logic;
Q : out std_logic);
end component;
attribute INIT of FDCE : component is "0";
attribute BOX_TYPE of FDCE : component is "BLACK_BOX";

component OR2
port ( I0 : in std_logic;
I1 : in std_logic;
O : out std_logic);
end component;
attribute BOX_TYPE of OR2 : component is "BLACK_BOX";

begin
Q(15 downto 0) <= Q_DUMMY(15 downto 0);
I_MQ0 : M2_1_MXILINX_txdata
port map (D0=>SLI,
          D1=>D(0),
          S0=>L,
          O=>MD0);

I_MQ1 : M2_1_MXILINX_txdata
port map (D0=>Q_DUMMY(0),
          D1=>D(1),
          S0=>L,
          O=>MD1);

I_MQ2 : M2_1_MXILINX_txdata
port map (D0=>Q_DUMMY(1),
          D1=>D(2),
          S0=>L,
          O=>MD2);
I_MQ3 : M2_1_MXILINX_txdata
    port map (D0=>Q_DUMMY(2),
             D1=>D(3),
             S0=>L,
             O=>MD3);

I_MQ4 : M2_1_MXILINX_txdata
    port map (D0=>Q_DUMMY(3),
             D1=>D(4),
             S0=>L,
             O=>MD4);

I_MQ5 : M2_1_MXILINX_txdata
    port map (D0=>Q_DUMMY(4),
             D1=>D(5),
             S0=>L,
             O=>MD5);

I_MQ6 : M2_1_MXILINX_txdata
    port map (D0=>Q_DUMMY(5),
             D1=>D(6),
             S0=>L,
             O=>MD6);

I_MQ7 : M2_1_MXILINX_txdata
    port map (D0=>Q_DUMMY(6),
             D1=>D(7),
             S0=>L,
             O=>MD7);

I_MQ8 : M2_1_MXILINX_txdata
    port map (D0=>Q_DUMMY(7),
             D1=>D(8),
             S0=>L,
             O=>MD8);

I_MQ9 : M2_1_MXILINX_txdata
    port map (D0=>Q_DUMMY(8),
             D1=>D(9),
             S0=>L,
             O=>MD9);

I_MQ10 : M2_1_MXILINX_txdata
    port map (D0=>Q_DUMMY(9),
              D1=>D(10),
              S0=>L,
              O=>MD10);

I_MQ11 : M2_1_MXILINX_txdata
port map (D0=>Q_DUMMY(10),
        D1=>D(11),
        S0=>L,
        O=>MD11);

I_MQ12 : M2_1_MXILINX_txdata
    port map (D0=>Q_DUMMY(11),
              D1=>D(12),
              S0=>L,
              O=>MD12);

I_MQ13 : M2_1_MXILINX_txdata
    port map (D0=>Q_DUMMY(12),
              D1=>D(13),
              S0=>L,
              O=>MD13);

I_MQ14 : M2_1_MXILINX_txdata
    port map (D0=>Q_DUMMY(13),
              D1=>D(14),
              S0=>L,
              O=>MD14);

I_MQ15 : M2_1_MXILINX_txdata
    port map (D0=>Q_DUMMY(14),
              D1=>D(15),
              S0=>L,
              O=>MD15);

I_Q0 : FDCE
    port map (C=>C,
              CE=>L_OR_CE,
              CLR=>CLR,
              D=>MD0,
              Q=>Q_DUMMY(0));

I_Q1 : FDCE
    port map (C=>C,
              CE=>L_OR_CE,
              CLR=>CLR,
              D=>MD1,
              Q=>Q_DUMMY(1));

I_Q2 : FDCE
    port map (C=>C,
              CE=>L_OR_CE,
              CLR=>CLR,
              D=>MD2,
              Q=>Q_DUMMY(2));
I_Q3 : FDCE
    port map (C=>C,
              CE=>L_OR_CE,
              CLR=>CLR,
              D=>MD3,
              Q=>Q_DUMMY(3));

I_Q4 : FDCE
    port map (C=>C,
              CE=>L_OR_CE,
              CLR=>CLR,
              D=>MD4,
              Q=>Q_DUMMY(4));

I_Q5 : FDCE
    port map (C=>C,
              CE=>L_OR_CE,
              CLR=>CLR,
              D=>MD5,
              Q=>Q_DUMMY(5));

I_Q6 : FDCE
    port map (C=>C,
              CE=>L_OR_CE,
              CLR=>CLR,
              D=>MD6,
              Q=>Q_DUMMY(6));

I_Q7 : FDCE
    port map (C=>C,
              CE=>L_OR_CE,
              CLR=>CLR,
              D=>MD7,
              Q=>Q_DUMMY(7));

I_Q8 : FDCE
    port map (C=>C,
              CE=>L_OR_CE,
              CLR=>CLR,
              D=>MD8,
              Q=>Q_DUMMY(8));

I_Q9 : FDCE
    port map (C=>C,
              CE=>L_OR_CE,
              CLR=>CLR,
              D=>MD9,
              Q=>Q_DUMMY(9));
I_Q10 : FDCE
port map (C=>C,
    CE=>L_OR_CE,
    CLR=>CLR,
    D=>MD10,
    Q=>Q_DUMMY(10));

I_Q11 : FDCE
port map (C=>C,
    CE=>L_OR_CE,
    CLR=>CLR,
    D=>MD11,
    Q=>Q_DUMMY(11));

I_Q12 : FDCE
port map (C=>C,
    CE=>L_OR_CE,
    CLR=>CLR,
    D=>MD12,
    Q=>Q_DUMMY(12));

I_Q13 : FDCE
port map (C=>C,
    CE=>L_OR_CE,
    CLR=>CLR,
    D=>MD13,
    Q=>Q_DUMMY(13));

I_Q14 : FDCE
port map (C=>C,
    CE=>L_OR_CE,
    CLR=>CLR,
    D=>MD14,
    Q=>Q_DUMMY(14));

I_Q15 : FDCE
port map (C=>C,
    CE=>L_OR_CE,
    CLR=>CLR,
    D=>MD15,
    Q=>Q_DUMMY(15));

I_36_67 : OR2
port map (I0=>L,
    I1=>CE,
    O=>L_OR_CE);

end BEHAVIORAL;
library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;

entity txdata is
  port ( txdi   : in    std_logic_vector (7 downto 0);
        txdl   : in    std_logic;
        txdw   : in    std_logic;
        txshck : in    std_logic;
        tdo    : out   std_logic);
end txdata;

architecture BEHAVIORAL of txdata is
  attribute HU_SET     : string :
  attribute BOX_TYPE   :
  signal do      : std_logic_vector (15 downto 0);
  signal pdi     : std_logic_vector (15 downto 0);
  signal tdi     : std_logic_vector (7 downto 0);
  signal XLXN_1  : std_logic;
  signal XLXN_4  : std_logic;
  signal XLXN_33 : std_logic;
  signal XLXN_80 : std_logic;
  signal XLXN_81 : std_logic;
  component SR16CLE_MXILINX_txdata
    port ( C   : in    std_logic;
           CE  : in    std_logic;
           CLR : in    std_logic;
           D   : in    std_logic_vector (15 downto 0);
           L   : in    std_logic;
           SLI : in    std_logic;
           Q   : out   std_logic_vector (15 downto 0));
  end component;

  component GND
    port ( G : out   std_logic);
  end component;
  attribute BOX_TYPE of GND : component is "BLACK_BOX";

  component VCC
    port ( P : out   std_logic);
  end component;
  attribute BOX_TYPE of VCC : component is "BLACK_BOX";

  component BUF
port (I : in std_logic;
    O : out std_logic);
end component;
attribute BOX_TYPE of BUF : component is "BLACK_BOX";

component FD8CE_MXILINX_txdata
    port (C : in std_logic;
            CE : in std_logic;
            CLR : in std_logic;
            D : in std_logic_vector (7 downto 0);
            Q : out std_logic_vector (7 downto 0));
end component;

attribute HU_SET of XLXI_3 : label is "XLXI_3_16";
attribute HU_SET of XLXI_28 : label is "XLXI_28_17";
begin
    XLXI_3 : SR16CLE_MXILINX_txdata
        port map (C=>txshck,
                  CE=>XLXN_33,
                  CLR=>XLXN_1,
                  D(15 downto 0)=>pdi(15 downto 0),
                  L=>txdl,
                  SLI=>XLXN_4,
                  Q(15 downto 0)=>do(15 downto 0));
    XLXI_4 : GND
        port map (G=>XLXN_1);
    XLXI_5 : VCC
        port map (P=>XLXN_4);
    XLXI_6 : GND
        port map (G=>pdi(15));
    XLXI_7 : BUF
        port map (I=>tdi(0),
                  O=>pdi(14));
    XLXI_8 : BUF
        port map (I=>tdi(1),
                  O=>pdi(13));
    XLXI_9 : BUF
        port map (I=>tdi(2),
                  O=>pdi(12));
    XLXI_10 : BUF
        port map (I=>tdi(3),
                   O=>pdi(11));
XLXI_11 : BUF
  port map (I=>tdi(4),
           O=>pdi(10));

XLXI_12 : BUF
  port map (I=>tdi(5),
           O=>pdi(9));

XLXI_13 : BUF
  port map (I=>tdi(6),
           O=>pdi(8));

XLXI_14 : BUF
  port map (I=>tdi(7),
           O=>pdi(7));

XLXI_15 : VCC
  port map (P=>pdi(6));

XLXI_16 : VCC
  port map (P=>pdi(5));

XLXI_18 : VCC
  port map (P=>pdi(4));

XLXI_19 : VCC
  port map (P=>pdi(3));

XLXI_20 : VCC
  port map (P=>pdi(2));

XLXI_21 : VCC
  port map (P=>pdi(1));

XLXI_22 : VCC
  port map (P=>pdi(0));

XLXI_23 : BUF
  port map (I=>do(15),
           O=>tdo);

XLXI_24 : VCC
  port map (P=>XLXN_33);

XLXI_28 : FD8CE_MXILINX_txdata
  port map (C=>txdw,
            CE=>XLXN_80,
            CLR=>XLXN_81,
D(7 downto 0)=>txdi(7 downto 0),
Q(7 downto 0)=>tdi(7 downto 0));

XLXI_29 : GND
port map (G=>XLXN_81);

XLXI_30 : VCC
port map (P=>XLXN_80);

end BEHAVIORAL;

library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;

entity FD8CE_MXILINX_rxdata is
    port ( C   : in    std_logic;
            CE : in    std_logic;
            CLR : in    std_logic;
            D   : in    std_logic_vector (7 downto 0);
            Q   : out   std_logic_vector (7 downto 0));
end FD8CE_MXILINX_rxdata;

architecture BEHAVIORAL of FD8CE_MXILINX_rxdata is
attribute INIT : string;
attribute BOX_TYPE : string;
component FDCE
-- synopsys translate_off
generic ( INIT : bit := '0');
-- synopsys translate_on
port ( C : in    std_logic;
    CE : in    std_logic;
    CLR : in    std_logic;
    D  : in    std_logic;
    Q  : out   std_logic);
end component;
attribute INIT of FDCE : component is "0";
attribute BOX_TYPE of FDCE : component is "BLACK_BOX";
begin
  I_Q0 : FDCE
  port map (C=>C,
    CE=>CE,
    CLR=>CLR,
    D=>D(0),
    Q=>Q(0));

  I_Q1 : FDCE
  port map (C=>C,
    CE=>CE,
    CLR=>CLR,
    D=>D(1),
    Q=>Q(1));

  I_Q2 : FDCE
  port map (C=>C,
    CE=>CE,
    CLR=>CLR,
    D=>D(2),
    Q=>Q(2));

  I_Q3 : FDCE
  port map (C=>C,
    CE=>CE,
    CLR=>CLR,
    D=>D(3),
    Q=>Q(3));

  I_Q4 : FDCE
  port map (C=>C,
    CE=>CE,
    CLR=>CLR,
    D=>D(4),
Q=>Q(4));

I_Q5 : FDCE
  port map (C=>C,
           CE=>CE,
           CLR=>CLR,
           D=>D(5),
           Q=>Q(5));

I_Q6 : FDCE
  port map (C=>C,
           CE=>CE,
           CLR=>CLR,
           D=>D(6),
           Q=>Q(6));

I_Q7 : FDCE
  port map (C=>C,
           CE=>CE,
           CLR=>CLR,
           D=>D(7),
           Q=>Q(7));

end BEHAVIORAL;

library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;

entity SR8CE_MXILINX_rxdata is
  port ( C   : in    std_logic;
         CE  : in    std_logic;
         CLR : in    std_logic;
         SLI : in    std_logic;
         Q   : out   std_logic_vector (7 downto 0));
end SR8CE_MXILINX_rxdata;

architecture BEHAVIORAL of SR8CE_MXILINX_rxdata is
  attribute INIT       : string ;
  attribute BOX_TYPE   : string ;
  signal Q_DUMMY : std_logic_vector (7 downto 0);
  component FDCE
    -- synopsys translate_off
    generic( INIT : bit := '0');
    -- synopsys translate_on

port (C : in std_logic;
    CE : in std_logic;
    CLR : in std_logic;
    D : in std_logic;
    Q : out std_logic);
end component;
attribute INIT of FDCE : component is "0";
attribute BOX_TYPE of FDCE : component is "BLACK_BOX";

begin
  Q(7 downto 0) <= Q_DUMMY(7 downto 0);
  I_Q0 : FDCE
    port map (C=>C,
              CE=>CE,
              CLR=>CLR,
              D=>SLI,
              Q=>Q_DUMMY(0));

  I_Q1 : FDCE
    port map (C=>C,
              CE=>CE,
              CLR=>CLR,
              D=>Q_DUMMY(0),
              Q=>Q_DUMMY(1));

  I_Q2 : FDCE
    port map (C=>C,
              CE=>CE,
              CLR=>CLR,
              D=>Q_DUMMY(1),
              Q=>Q_DUMMY(2));

  I_Q3 : FDCE
    port map (C=>C,
              CE=>CE,
              CLR=>CLR,
              D=>Q_DUMMY(2),
              Q=>Q_DUMMY(3));

  I_Q4 : FDCE
    port map (C=>C,
              CE=>CE,
              CLR=>CLR,
              D=>Q_DUMMY(3),
              Q=>Q_DUMMY(4));

  I_Q5 : FDCE
    port map (C=>C,
              CE=>CE,
CLR=>CLR,
D=>Q_DUMMY(4),
Q=>Q_DUMMY(5));

I_Q6 : FDCE
port map (C=>C,
CE=>CE,
CLR=>CLR,
D=>Q_DUMMY(5),
Q=>Q_DUMMY(6));

I_Q7 : FDCE
port map (C=>C,
CE=>CE,
CLR=>CLR,
D=>Q_DUMMY(6),
Q=>Q_DUMMY(7));
end BEHAVIORAL;

library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;

entity SR4CE_MXILINX_rxdata is
port ( C   : in    std_logic;
      CE  : in std_logic;
      CLR : in std_logic;
      SLI : in std_logic;
      Q0  : out std_logic;
      Q1  : out std_logic;
      Q2  : out std_logic;
      Q3  : out std_logic);
end SR4CE_MXILINX_rxdata;

architecture BEHAVIORAL of SR4CE_MXILINX_rxdata is
attribute INIT : string ;
attribute BOX_TYPE : string ;
signal Q0_DUMMY : std_logic;
signal Q1_DUMMY : std_logic;
signal Q2_DUMMY : std_logic;
component FDCE
  -- synopsys translate_off
  generic( INIT : bit := '0');
  -- synopsys translate_on
port ( C : in    std_logic;
    CE : in    std_logic;
    CLR : in    std_logic;
    D : in    std_logic;
    Q : out   std_logic);
end component;
attribute INIT of FDCE : component is "0";
attribute BOX_TYPE of FDCE : component is "BLACK_BOX";

begin
Q0 <= Q0_DUMMY;
Q1 <= Q1_DUMMY;
Q2 <= Q2_DUMMY;
I_Q0 : FDCE
    port map (C=>C,
               CE=>CE,
               CLR=>CLR,
               D=>Sli,
               Q=>Q0_DUMMY);

I_Q1 : FDCE
    port map (C=>C,
               CE=>CE,
               CLR=>CLR,
               D=>Q0_DUMMY,
               Q=>Q1_DUMMY);

I_Q2 : FDCE
    port map (C=>C,
               CE=>CE,
               CLR=>CLR,
               D=>Q1_DUMMY,
               Q=>Q2_DUMMY);

I_Q3 : FDCE
    port map (C=>C,
               CE=>CE,
               CLR=>CLR,
               D=>Q2_DUMMY,
               Q=>Q3);
end BEHAVIORAL;

library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;

entity rxdata is
  port ( cki : in    std_logic;
          sdi : in    std_logic;
          wri : in    std_logic;
          datao : out  std_logic_vector (7 downto 0));
end rxdata;

architecture BEHAVIORAL of rxdata is
  attribute HU_SET     : string ;
  attribute BOX_TYPE   : string ;
  signal cei     : std_logic;
  signal srd     : std_logic_vector (7 downto 0);
  signal sro     : std_logic_vector (7 downto 0);
  signal XLXN_8  : std_logic;
  signal XLXN_47 : std_logic;
  signal XLXN_51 : std_logic;
  signal XLXN_52 : std_logic;
  component SR4CE_MXILINX_rxdata
    port ( C   : in    std_logic;
           CE  : in    std_logic;
           CLR : in    std_logic;
           SLI : in    std_logic;
           Q0  : out   std_logic;
           Q1  : out   std_logic;
           Q2  : out   std_logic;
           Q3  : out   std_logic);
  end component;

  component SR8CE_MXILINX_rxdata
    port ( C   : in    std_logic;
           CE  : in    std_logic;
           CLR : in    std_logic;
           SLI : in    std_logic;
           Q   : out   std_logic_vector (7 downto 0));
  end component;

  component FD8CE_MXILINX_rxdata
    port ( C   : in    std_logic;
           CE  : in    std_logic;
           CLR : in    std_logic;
           D   : in    std_logic_vector (7 downto 0);
           Q   : out   std_logic_vector (7 downto 0));
  end component;

  component GND
    port ( G : out  std_logic);
  end component;
attribute BOX_TYPE of GND : component is "BLACK_BOX";

component BUF
  port ( I : in  std_logic;
         O : out std_logic);
end component;
attribute BOX_TYPE of BUF : component is "BLACK_BOX";

component VCC
  port ( P : out std_logic);
end component;
attribute BOX_TYPE of VCC : component is "BLACK_BOX";

attribute HU_SET of XLXI_1 : label is "XLXI_1_0";
attribute HU_SET of XLXI_2 : label is "XLXI_2_1";
attribute HU_SET of XLXI_3 : label is "XLXI_3_2";
begin
XLXI_1 : SR4CE_MXILINX_rxdata
  port map (C=>cki, 
            CE=>cei, 
            CLR=>XLXN_8, 
            SLI=>sro(7), 
            Q0=>XLXN_47, 
            Q1=>open, 
            Q2=>open, 
            Q3=>open);

XLXI_2 : SR8CE_MXILINX_rxdata
  port map (C=>cki, 
            CE=>cei, 
            CLR=>XLXN_8, 
            SLI=>sdi, 
            Q(7 downto 0)=>sro(7 downto 0));

XLXI_3 : FD8CE_MXILINX_rxdata
  port map (C=>wri, 
            CE=>XLXN_52, 
            CLR=>XLXN_51, 
            D(7 downto 0)=>srd(7 downto 0), 
            Q(7 downto 0)=>datao(7 downto 0));

XLXI_6 : GND
  port map (G=>XLXN_8);

XLXI_7 : BUF
  port map (I=>XLXN_47, 
            O=>srd(0));

XLXI_9 : BUF
port map (I=>sro(7),
    O=>srd(1));

XLXI_10 : BUF
    port map (I=>sro(6),
               O=>srd(2));

XLXI_11 : BUF
    port map (I=>sro(5),
               O=>srd(3));

XLXI_12 : BUF
    port map (I=>sro(4),
               O=>srd(4));

XLXI_13 : BUF
    port map (I=>sro(3),
               O=>srd(5));

XLXI_14 : BUF
    port map (I=>sro(2),
               O=>srd(6));

XLXI_15 : BUF
    port map (I=>sro(1),
               O=>srd(7));

XLXI_38 : GND
    port map (G=>XLXN_51);

XLXI_39 : VCC
    port map (P=>XLXN_52);

XLXI_40 : VCC
    port map (P=>cei);
library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;

entity FTRSE_MXILINX_rxcntc1a is
  port ( C  : in    std_logic;
         CE : in    std_logic;
         R  : in    std_logic;
         S  : in    std_logic;
         T  : in    std_logic;
         Q  : out   std_logic);
end FTRSE_MXILINX_rxcntc1a;

architecture BEHAVIORAL of FTRSE_MXILINX_rxcntc1a is
  attribute BOX_TYPE   : string;
  attribute INIT       : string;
  attribute RLOC       : string;
  signal CE_S    : std_logic;
  signal D_S     : std_logic;
  signal TQ      : std_logic;
  signal Q_DUMMY : std_logic;
  component XOR2
    port ( I0 : in    std_logic;
           I1 : in    std_logic;
           O  : out   std_logic);
  end component;
  attribute BOX_TYPE of XOR2 : component is "BLACK_BOX";

  component FDRE
  -- synopsys translate_off
  generic( INIT : bit := '0');
  -- synopsys translate_on
    port ( C  : in    std_logic;
           CE : in    std_logic;
           D : in    std_logic;
           C1 : in    std_logic;
           R  : in    std_logic;
           S  : in    std_logic;
           T  : in    std_logic;
           Q  : out   std_logic);
  end component;
R : in std_logic;
Q : out std_logic);
end component;
attribute INIT of FDRE : component is "0";
attribute BOX_TYPE of FDRE : component is "BLACK_BOX";

component OR2
port ( I0 : in std_logic;
I1 : in std_logic;
O : out std_logic);
end component;
attribute BOX_TYPE of OR2 : component is "BLACK_BOX";

attribute RLOC of I_36_35 : label is "X0Y0";
begn
Q <= Q_DUMMY;
I_36_32 : XOR2
port map (I0=>T,
I1=>Q_DUMMY,
O=>TQ);
I_36_35 : FDRE
port map (C=>C,
CE=>CE_S,
D=>D_S,
R=>R,
Q=>Q_DUMMY);
I_36_73 : OR2
port map (I0=>S,
I1=>TQ,
O=>D_S);
I_36_77 : OR2
port map (I0=>CE,
I1=>S,
O=>CE_S);
end BEHAVIORAL;

library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;

entity CB8RE_MXILINX_rxcntc1a is
port ( C : in std_logic;
    CE : in std_logic;
    R : in std_logic;
    CEO : out std_logic;
    Q : out std_logic_vector (7 downto 0);
    TC : out std_logic);
end CB8RE_MXILINX_rxcntc1a;

architecture BEHAVIORAL of CB8RE_MXILINX_rxcntc1a is
attribute HU_SET : string;
attribute BOX_TYPE : string;
signal T2 : std_logic;
signal T3 : std_logic;
signal T4 : std_logic;
signal T5 : std_logic;
signal T6 : std_logic;
signal T7 : std_logic;
signal XLXN_1 : std_logic;
signal XLXN_2 : std_logic;
signal Q_DUMMY : std_logic_vector (7 downto 0);
signal TC_DUMMY : std_logic;
component FTRSE_MXILINX_rxcntc1a
port ( C : in std_logic;
    CE : in std_logic;
    R : in std_logic;
    S : in std_logic;
    T : in std_logic;
    Q : out std_logic);
end component;

component GND
    port ( G : out std_logic);
end component;
attribute BOX_TYPE of GND : component is "BLACK_BOX";

component VCC
    port ( P : out std_logic);
end component;
attribute BOX_TYPE of VCC : component is "BLACK_BOX";

component AND2
    port ( I0 : in std_logic;
        I1 : in std_logic;
        O : out std_logic);
end component;
attribute BOX_TYPE of AND2 : component is "BLACK_BOX";

component AND3
    port ( I0 : in std_logic;
I1 : in std_logic;
I2 : in std_logic;
  O : out std_logic);
end component;
attribute BOX_TYPE of AND3 : component is "BLACK_BOX";

component AND4
port ( I0 : in std_logic;
  I1 : in std_logic;
  I2 : in std_logic;
  I3 : in std_logic;
  O : out std_logic);
end component;
attribute BOX_TYPE of AND4 : component is "BLACK_BOX";

component AND5
port ( I0 : in std_logic;
  I1 : in std_logic;
  I2 : in std_logic;
  I3 : in std_logic;
  I4 : in std_logic;
  O : out std_logic);
end component;
attribute BOX_TYPE of AND5 : component is "BLACK_BOX";

attribute HU_SET of I_Q0 : label is "I_Q0_7";
attribute HU_SET of I_Q1 : label is "I_Q1_6";
attribute HU_SET of I_Q2 : label is "I_Q2_5";
attribute HU_SET of I_Q3 : label is "I_Q3_4";
attribute HU_SET of I_Q4 : label is "I_Q4_3";
attribute HU_SET of I_Q5 : label is "I_Q5_2";
attribute HU_SET of I_Q6 : label is "I_Q6_1";
attribute HU_SET of I_Q7 : label is "I_Q7_0";

begin
Q(7 downto 0) <= Q_DUMMY(7 downto 0);
TC <= TC_DUMMY;
I_Q0 : FTRSE_MXILINX_rxcntc1a
port map (C=>C,
    CE=>CE,
    R=>R,
    S=>XLXN_2,
    T=>XLXN_1,
    Q=>Q_DUMMY(0));

I_Q1 : FTRSE_MXILINX_rxcntc1a
port map (C=>C,
    CE=>CE,
    R=>R,
    S=>XLXN_2,
T => Q_DUMMY(0),
Q => Q_DUMMY(1));

I_Q2 : FTRSE_MXILINX_rxcntc1a
port map (C => C,
CE => CE,
R => R,
S => XLXN_2,
T => T2,
Q => Q_DUMMY(2));

I_Q3 : FTRSE_MXILINX_rxcntc1a
port map (C => C,
CE => CE,
R => R,
S => XLXN_2,
T => T3,
Q => Q_DUMMY(3));

I_Q4 : FTRSE_MXILINX_rxcntc1a
port map (C => C,
CE => CE,
R => R,
S => XLXN_2,
T => T4,
Q => Q_DUMMY(4));

I_Q5 : FTRSE_MXILINX_rxcntc1a
port map (C => C,
CE => CE,
R => R,
S => XLXN_2,
T => T5,
Q => Q_DUMMY(5));

I_Q6 : FTRSE_MXILINX_rxcntc1a
port map (C => C,
CE => CE,
R => R,
S => XLXN_2,
T => T6,
Q => Q_DUMMY(6));

I_Q7 : FTRSE_MXILINX_rxcntc1a
port map (C => C,
CE => CE,
R => R,
S => XLXN_2,
T => T7,
Q=>Q_DUMMY(7));

I_36_7 : GND
port map (G=>XLXN_2);

I_36_13 : VCC
port map (P=>XLXN_1);

I_36_21 : AND2
port map (I0=>Q_DUMMY(1),
I1=>Q_DUMMY(0),
O=>T2);

I_36_22 : AND3
port map (I0=>Q_DUMMY(2),
I1=>Q_DUMMY(1),
I2=>Q_DUMMY(0),
O=>T3);

I_36_23 : AND4
port map (I0=>Q_DUMMY(3),
I1=>Q_DUMMY(2),
I2=>Q_DUMMY(1),
I3=>Q_DUMMY(0),
O=>T4);

I_36_25 : AND2
port map (I0=>Q_DUMMY(4),
I1=>T4,
O=>T5);

I_36_26 : AND3
port map (I0=>Q_DUMMY(5),
I1=>Q_DUMMY(4),
I2=>T4,
O=>T6);

I_36_28 : AND4
port map (I0=>Q_DUMMY(6),
I1=>Q_DUMMY(5),
I2=>Q_DUMMY(4),
I3=>T4,
O=>T7);

I_36_29 : AND5
port map (I0=>Q_DUMMY(7),
I1=>Q_DUMMY(6),
I2=>Q_DUMMY(5),
I3=>Q_DUMMY(4),
library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;

entity M2_1_MXILINX_rxcntc1a is
    port ( D0 : in    std_logic;
             D1 : in    std_logic;
             S0 : in    std_logic;
             O  : out   std_logic);
end M2_1_MXILINX_rxcntc1a;

architecture BEHAVIORAL of M2_1_MXILINX_rxcntc1a is
    attribute BOX_TYPE   : string;
    signal M0 : std_logic;
    signal M1 : std_logic;

    component AND2B1
        port ( I0 : in    std_logic;
               I1 : in    std_logic;
               O  : out   std_logic);
    end component;
    attribute BOX_TYPE of AND2B1 : component is "BLACK_BOX";

    component OR2
        port ( I0 : in    std_logic;
               I1 : in    std_logic;
               O  : out   std_logic);
    end component;
    attribute BOX_TYPE of OR2 : component is "BLACK_BOX";

    component AND2
        port ( I0 : in    std_logic;
               I1 : in    std_logic;
               O  : out   std_logic);
    end component;
    attribute BOX_TYPE of AND2 : component is "BLACK_BOX";

begin

I4=>T4,
O=>TC_DUMMY);

I_36_32 : AND2
    port map (I0=>CE,
              I1=>TC_DUMMY,
              O=>CEO);

end BEHAVIORAL;
begin
  I_36_7 : AND2B1
  port map (I0=>S0,
           I1=>D0,
           O=>M0);

  I_36_8 : OR2
  port map (I0=>M1,
           I1=>M0,
           O=>O);

  I_36_9 : AND2
  port map (I0=>D1,
           I1=>S0,
           O=>M1);
end BEHAVIORAL;

library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;

entity rxcntc1a is
  port ( ck21i : in    std_logic;
         rdi   : in    std_logic;
         sdi   : in    std_logic;
         rxf   : out   std_logic;
         shcko : out   std_logic);
end rxcntc1a;

architecture BEHAVIORAL of rxcntc1a is
  attribute BOX_TYPE   : string   :
  attribute HU_SET     : string   :
  attribute INIT       : string   :
  signal cto      : std_logic_vector (7 downto 0);
  signal done     : std_logic;
  signal shckop   : std_logic;
  signal start    : std_logic;
  signal XLXN_8   : std_logic;
  signal XLXN_11  : std_logic;
  signal XLXN_12  : std_logic;
  signal XLXN_13  : std_logic;
  signal XLXN_46  : std_logic;
  signal XLXN_61  : std_logic;
signal XLXN_114 : std_logic;
signal XLXN_115 : std_logic;
component INV
  port ( I : in    std_logic;
        O : out   std_logic);
end component;
attribute BOX_TYPE of INV : component is "BLACK_BOX";

component M2_1_MXILINX_rxcntc1a
  port ( D0 : in    std_logic;
         D1 : in    std_logic;
         S0 : in    std_logic;
         O  : out   std_logic);
end component;

component FDRE
  -- synopsys translate_off
  generic( INIT : bit := '0');
  -- synopsys translate_on
  port ( C  : in    std_logic;
         CE : in    std_logic;
         D  : in    std_logic;
         R  : in    std_logic;
         Q  : out   std_logic);
end component;
attribute INIT of FDRE : component is "0";
attribute BOX_TYPE of FDRE : component is "BLACK_BOX";

component VCC
  port ( P : out   std_logic);
end component;
attribute BOX_TYPE of VCC : component is "BLACK_BOX";

component CB8RE_MXILINX_rxcntc1a
  port ( C   : in    std_logic;
         CE  : in    std_logic;
         R   : in    std_logic;
         CEO : out   std_logic;
         Q   : out   std_logic_vector (7 downto 0);
         TC  : out   std_logic);
end component;

component GND
  port ( G : out   std_logic);
end component;
attribute BOX_TYPE of GND : component is "BLACK_BOX";

component FDCE
  -- synopsys translate_off
generic( INIT : bit := '0');
-- synopsys translate_on
port ( C : in    std_logic;
     CE : in    std_logic;
     CLR : in    std_logic;
     D : in    std_logic;
     Q : out   std_logic);
end component;
attribute INIT of FDCE : component is "0";
attribute BOX_TYPE of FDCE : component is "BLACK_BOX";

component det160
port ( ia0 : in    std_logic;
      ia1 : in    std_logic;
      ia2 : in    std_logic;
      ia3 : in    std_logic;
      ia5 : in    std_logic;
      ia7 : in    std_logic;
      ia4 : in    std_logic;
      ia6 : in    std_logic;
      od160 : out   std_logic);
end component;

component FTRSE_MXILINX_rxcntc1a
port ( C : in    std_logic;
      CE : in    std_logic;
      R : in    std_logic;
      S : in    std_logic;
      T : in    std_logic;
      Q : out   std_logic);
end component;

attribute HU_SET of XLXI_15 : label is "XLXI_15_8";
attribute HU_SET of XLXI_32 : label is "XLXI_32_9";
attribute HU_SET of XLXI_61 : label is "XLXI_61_10";
begin
XLXI_9 : INV
  port map (I=>sdi,
            O=>XLXN_8);

XLXI_10 : INV
  port map (I=>start,
            O=>XLXN_11);

XLXI_15 : M2_1_MXILINX_rxcntc1a
  port map (D0=>start,
            D1=>XLXN_61,
            S0=>done,
            O=>XLXN_13);
XLXI_18 : FDRE
port map (C=>ck21i,
CE=>done,
D=>XLXN_46,
R=>rdi,
Q=>rfi);

XLXI_20 : VCC
port map (P=>XLXN_12);

XLXI_30 : VCC
port map (P=>XLXN_46);

XLXI_32 : CB8RE_MXILNX_rxcntc1a
port map (C=>ck21i,
CE=>XLXN_13,
R=>rdi,
CEO=>open,
Q(7 downto 0)=>cto(7 downto 0),
TC=>open);

XLXI_35 : INV
port map (I=>cto(2),
O=>shckop);

XLXI_36 : GND
port map (G=>XLXN_61);

XLXI_37 : FDCE
port map (C=>XLXN_8,
CE=>XLXN_11,
CLR=>rdi,
D=>XLXN_12,
Q=>start);

XLXI_53 : det160
port map (ia0=>cto(0),
ia1=>cto(1),
ia2=>cto(2),
ia3=>cto(3),
ia4=>cto(4),
ia5=>cto(5),
ia6=>cto(6),
ia7=>cto(7),
od160=>done);

XLXI_61 : FTRSE_MXILNX_rxcntc1a
port map (C=>shckop,
library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;

entity det160 is
port ( ia0 : in std_logic;
ia1 : in std_logic;
ia2 : in std_logic;
ia3 : in std_logic;
ia4 : in std_logic;
CE=>XLXN_13,
R=>rdi,
S=>XLXN_114,
T=>XLXN_115,
Q=>shcko);

XLXI_64 : GND
port map (G=>XLXN_114);

XLXI_65 : VCC
port map (P=>XLXN_115);
end BEHAVIORAL;

-- Copyright (c) 1995-2007 Xilinx, Inc. All rights reserved.
--
-- Command: C:\Xilinx92i\bin\nt\sch2vhdl.exe -intstyle ise -family spartan3 -flat -suppress -w C:/Thesis/Sim2_new/det160.sch det160.vhf
-- Design Name: det160
-- Device: spartan3
-- Purpose:
--  This vhdl netlist is translated from an ECS schematic. It can be
--  synthesis and simulated, but it should not be modified.
--
library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;
architecture BEHAVIORAL of det160 is
attribute BOX_TYPE : string :
signal XLXN_1 : std_logic;
signal XLXN_2 : std_logic;
component AND4B2
    port ( I0 : in    std_logic;
          I1 : in    std_logic;
          I2 : in    std_logic;
          I3 : in    std_logic;
          O  : out   std_logic);
end component;
attribute BOX_TYPE of AND4B2 : component is "BLACK_BOX";
component AND2
    port ( I0 : in    std_logic;
          I1 : in    std_logic;
          O  : out   std_logic);
end component;
attribute BOX_TYPE of AND2 : component is "BLACK_BOX";
component AND4B4
    port ( I0 : in    std_logic;
          I1 : in    std_logic;
          I2 : in    std_logic;
          I3 : in    std_logic;
          O  : out   std_logic);
end component;
attribute BOX_TYPE of AND4B4 : component is "BLACK_BOX";
begin
    XLXI_2 : AND4B2
    port map (I0=>ia6,
            I1=>ia4,
            I2=>ia7,
            I3=>ia5,
            O=>XLXN_2);

    XLXI_3 : AND2
    port map (I0=>XLXN_2,
              I1=>XLXN_1,
              O=>od160);

    XLXI_4 : AND4B4
port map (I0=>ia3,
     I1=>ia2,
     I2=>ia1,
     I3=>ia0,
     O=>XLXN_1);

end BEHAVIORAL;
Appendix B

Simulation Results

Functional Simulation of sample program 1

Post-Route Simulation of Sample program 1

Behavioral Simulation of Sample Program 2
Post-Route Simulation of Sample Program 2
Appendix C

Schematic for our implementation board.