ECE 585 Project 2 Spring 13 ver1
Simulation of CPU, Cache, Bus, and Memory Datapath

1. INTRODUCTION

Project 2 is due Thurs Apr 11 EOD. You may work as groups of up to two. Submit your report to the Project 2 folder in the Electronic Black Board for the course. NO OTHE FORM OF SUBMISSION OR LATE SUBMISSIONS WILL BE ACCEPTED.

In this project, you will simulate a CPU, cache, bus, and memory complex for a set of instructions with emphasis on the cache operation. The overall block diagram is shown in Figure 1.

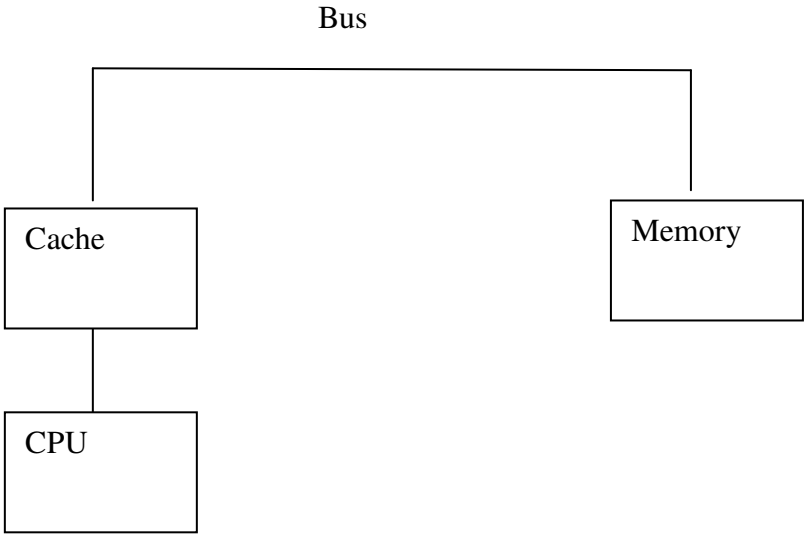

Figure 1: Overall Block Diagram

The goal of this project is to provide you with a more practical hands-on approach to computer architecture design problems. The processor complex you will be designing is a 32-bit version of the MIPS processor; however, the instruction set will be a small subset of the actual MIPS ISA. You should implement the end to end operation of the complex utilizing the VHDL hardware descriptive language. You may use any constructs within the VHDL language, however, the design must be of your own. Copying of any form from any other student or any internal or external sources is illegal and will not be accepted.

The processor supports the three instruction formats: R-format, I-format, and J-format as described in the text book and lectures. Table I Summarizes the core set of instructions for your ISA. The memory is assumed to be byte addressable and each word is 32 bits.

Table I: Core MIPS Instruction Set to be Designed (with example)

OpCode [31 : 26]	Function Field [5 : 0]	Instruction	Operation
100011	--	lw	lw $s1, 200($t3)
101011	--	sw	sw $s3, 100($t4)
000000	100000	add	add $s3, $t3, $t2
000100	--	beq	beq $s5, $t6, 400
		(Custom set)	

The total set you need to design is the core set as above + a custom set designated for you as follows.
Student ID ending in:

1. BNE, LUI
2. NOR, SLL
3. ADDI, LUI
4. BNE, LUI
5. NOR, LUI
6. ANDI, JR
7. BNE, LUI
8. NOR, LUI
9. ANDI, JR
0. ADDI, LUI

2. Implementation Details

2.1 CPU: You need to treat the CPU as a block diagram and show only the inputs, outputs and the changes in the Register file. Note that all the source values for the instructions are derived from the CPU registers and immediate value in the instruction itself. The results will also be stored in the register except for the store instruction. For both load and store instructions, ALU operation is needed for address calculation. You need not simulate the detailed internal operation of the CPU complex

2.2 Bus

Used only for transfer of words and blocks. The bus (between cache and memory) has the following specifications: Bandwidth of 32 words/cycle.

2.3 Cache/Bus/Memory Specifications

The focus of the report is on the cache operation.
The cache has the following specifications

1. Size 256 Byte I-cache, 128 Byte D-cache; block size of 8 words; word size of 4 Bytes
2. The cache access time is 1 cycle
3. Direct Memory access approach is used for cache block placement
4. The parameters for cache operation include IHc (Icache hit), DHc (Dcache hit) and dirty bit set flag for a block to be replaced (dbset)

The memory has the following specifications

1. Size 1,024 Bytes; Byte addressable

2. Memory port access time is 5 cycles/word for reads; 3 cycles/word for writes.
3. Additional memory read time: 3 cycles/word, write time 4 cycles/word

Additional common specifications

1. The instruction address is available in the Program Counter (PC); the accessed instruction is placed in the Instruction Register (IR); the data read (for loads) is loaded in the Memory Data Register (MDR) – all are 32 bit registers.

Additional student specific specifications

Last digit of student ID	Write Strategy	Write miss Strategy
0	Write Thru	Write Allocate
1	Write back	No-write allocate
2	Write Thru	No-write allocate
3	Write back	Write Allocate
4	Write Thru	Write Allocate
5	Write Thru	Write Allocate
6	Write back	No-write allocate
7	Write back	Write Allocate
8	Write back	No-write allocate
9	Write Thru	No-write allocate

General Guidelines

1. All parameters must be defined as variables (or data inputs) so that different parameters can be used for testing your code.
2. You should annotate your code with appropriate/sufficient comments so that the code is self explanatory.
3. You may use additional meaningful assumptions and state them clearly in your report.
4. Section 5 provides some useful hints for cache operation.

3. Test Program: Design a test program to verify the operation of your code. It needs to account for the following variables: PC address value; range of PC's for the instruction type (e.g. 0-500: ALU_BR type; 504-600: Loads; 604-700: stores); Icache hit flag (1 or 0); Daddr; Dcache hit flag (1 or 0); range of addresses (dbaddr1 to dbaddr2) for which the dirty bit is set. Your output should explicitly indicate which type of instruction is complete after completion of the operation.

4. Report

You are required to turn in a report that describes the design along with the VHDL code. The report should be typed, well written, and well organized. The suggested contents of the report are as follows:

- An overview of your design
- Appropriate sections to convey your report
- A discussion on how you tried to optimize your design
- A discussion on any improvements or additional features made to your design
- A discussion on what does not work correctly in your design
- An overview block diagram of your design.
- A sample simulation of your design that is annotated to show its correct operation.
- Copying of code will not be acceptable. Any code copied will automatically result in a 0 for your project and may be subject to additional disciplinary action.
- Start the project right away
- Good luck and have fun

5 Hint: Basic operations are summarized in the following. Please note that you need to modify it appropriately to account for placement/replacement, dirty bit status, write strategies, write miss strategies etc.