2010 R&E: Computer System Education & Research

Project

Design the pipelined MIPS CPU in Verilog, based on the single-cycle MIPS processor. You have an option to design the pipelined MIPS CPU from scratch if you want. The CPU design should satisfy the following requirements

- Implement the MIPS instructions in page 2
- Implement a 5-stage pipeline (IF, ID, EX, MA, WB)
- Implement hazard detection and forwarding logic
- Implement the delayed branch and jump
- You don't have to implement the exception detection and handling logic
- All the writes occur at the rising-edge of the clock
    a. Writes in register file
    b. Writes in data memory
    c. Program counter updates
- Branch outcome (taken or not-taken) and destination are calculated in the ID (decoding) stage of the pipeline

You should write many many testvectors with assembly code to validate your CPU design. Prof. Suh is also going to write his own assembly code and run it on your CPU design. Your CPU design should be able to run Prof. Suh's assembly codes correctly at the expected clock cycle. It will take the major part of the grading!

< Opcodes of MIPS Instructions>

Opcode	Name	Description	Implemented in Single-Cycle MIPS	Term Project	
000000 (0)	R-type	All R-type instructions		√	
000010 (2)	`j`	jump	√	√	
000011 (3)	`jal`	jump and link		√	
000100 (4)	`beq`	branch if equal	√	√	
000101 (5)	`bne`	branch if not equal		√	
001000 (8)	`addi`	add immediate	√	√	
001001 (9)	`addiu`	add immediate unsigned		√	
001010 (10)	`slti`	set less than immediate			
001011 (11)	`sltiu`	set less than immediate unsigned			
001100 (12)	`andi`	and immediate		√	
001101 (13)	`ori`	or immediate		√	
001110 (14)	`xori`	xor immediate		√	
001111 (15)	`lui`	load upper immediate		√	
100000 (32)	`lb`	load byte			
100001 (33)	`lh`	load halfword			
100011 (35)	`lw`	load word	√	√	
100100 (36)	`lbu`	load byte unsigned			
100101 (37)	`lhu`	load halfword unsigned			
101000 (40)	`sb`	store byte			
101001 (41)	`sh`	store halfword			
101011 (43)	`sw`	store word	√	√	

< Funct Field of MIPS Instructions>

Funct	Name	Description	Implemented in Single-Cycle MIPS	Term Project	
000000 (0)	`sll`	shift left logical		√	
000010 (2)	`srl`	shift right logical		√	
000011 (3)	`sra`	shift right arithmetic		√	
000100 (4)	`sllv`	shift left logical variable			
000110 (6)	`srlv`	shift right logical variable			
000111 (7)	`srav`	shift right arithmetic variable			
001000 (8)	`jr`	jump register		√	
001001 (9)	`jalr`	jump and link register			
001100 (12)	`syscall`	system call			
011000 (24)	`mult`	multiply			
011001 (25)	`multu`	multiply unsigned			
011010 (26)	`div`	divide			
011011 (27)	`divu`	divide unsigned			
100000 (32)	`add`	add	√	√	
100001 (33)	`addu`	add unsigned			
100010 (34)	`sub`	subtract	√	√	
100011 (35)	`subu`	subtract unsigned			
100100 (36)	`and`	and	√	√	
100101 (37)	`or`	or	√	√	
100110 (38)	`xor`	xor		√	
100111 (39)	`nor`	nor		√	
101010 (42)	`slt`	set less than	√	√	
101011 (43)	`sltu`	set less than unsigned		√	