SimpleMIPS: A Prototype of 5-stages Pipelined MIPS Microprocessor

Peiyao Sheng
MS108 : Computer System-I
Shanghai Jiao Tong University

June 25, 2017

Contents

1 Introduction

This is the final coursework of course MS108, Computer System-I. In this task, I implement a simple prototype of 32-bits, Harvard structural and 5-stages pipelined MIPS microprocessor in Verilog HDL, which has following features:

- Pipeline includes such five stages:Instruction Fetch,Instruction Decode & Data Preparation,Execution,Memory Access and Write Back.
- It implements Harvard architecture with separated storage for instructions and data.
- It has 32 registers, 32-bits width of data and address bus
- It make use of bypassing to deal with data hazard

2 Simulation

SimpleMIPS has following modules:

2.1 SOPC module

This module contains three main parts:

- SimpleMIPS : top module of CPU
- rom : instruction storage
- ram : data storage

2.2 SimpleMIPS

SimpleMIPS is the Top-level module, it connects all stages of pipeline. Besides, it contains three special registers, pc, hi, lo and a control module to issue stall request.

Table 1: Ports Description of module SimpleMIPS

	Port name	Description
input	rst	reset signal
	clk	clock signal
	rom_data_i	instruction read from ROM
	ram_data_i	data read from RAM
output	rom_addr_o	address sent to ROM
	rom_ce_o	chip enable signal for ROM

2.3 InsFetch

This module simulates the stage instructions fetch. The main function of this module is to modify the program counter and communicate with ROM to fetch instruction.

Table 2: Ports Description of module InsFetch

	Port name	Description
input	rst	reset signal
	clk	clock signal
	stall	stall signal
	branch_flag	branch flag
	branch_target	branch target address
output	pc_data	data of pc register

2.4 IF_ID module

It is the latch module between InsFetch and InsDecode.

Table 3: Ports Description of module IF_ID

	Port name	Description
input	rst	reset signal
	clk	clock signal
	stall	stall signal
	if_pc_data	data of pc register in IF
	if_instruction	instruction in IF
output	id_pc_data	data of pc register in ID
	id_instruction	instruction in ID

2.5 InsDecode module

This module decode the instruction and fetch source operands from registers according to decoding results. To deal with data hazards, some ports are designed to fetch data directly from stage EX and MEM. To deal with control hazard and the special type of data hazard, a stall port is set to receive stall request.

2.6 ID_EX module

It is the latch module between InsDecode and Execute.

2.7 Execute module

This module execute arithmetic operations based on the operands passed in the decoding phase.

2.8 EX_MEM module

It is the latch module between Execute and Memory Access.

Table 4: Ports Description of module InsDecode

	Port name	Description
input	rst	reset signal
	instruction_i	input instruction
	reg1_read_data	first input from Regfile
	reg2_read_data	second input from Regfile
	pc_data_i	input data of pc register
	ex_op	for data dependence
	ex_reg_we	
	ex_reg_write_address	
	ex_reg_write_data	
	mem_reg_we	
	mem_reg_write_address	
	mem_reg_write_data	
output	instruction_o	output instruction
	pc_data_o	output data of pc register
	branch_flag	branch flag
	op	type of operator
	inskind	type of instruction
	reg1	first source register
	reg2	second source register
	reg1_re	first source register read enable
	reg2_re	second source register read enable
	reg1_read_address	first source register read address
	reg2_read_address	second source register read address
	reg_we	target register write enable
	reg_write_address	target register write address
	reg_write_data	target register write data
	stall_request	stall request

2.9 Memory Access

This module calculate memory address and send it to RAM to execute the load or store instructions.

2.10 MEM_WB

latch module of Memory and WriteBack

2.11 Write Back

This module write results of execution back to register files.

2.12 CTRL module

This module receive the stall request and send the stall signal to latches.

Table 5: Ports Description of module ID_EX

	Port name	Description
input	rst	reset signal
	clk	clock signal
	stall	stall signal
	id_instruction	instruction in ID
	id_op	operator in ID
	id_ins_kind	instruction type in ID
	id_reg1	first source register in ID
	id_reg2	second source register in ID
	id_reg_we	register write enable in ID
	id_reg_write_address	register write address in ID
	id_reg_write_data	register write data in ID
output	ex_instruction	instruction in EX
	ex_op	operator in EX
	ex_ins_kind	instruction type in EX
	ex_reg1	first source register in EX
	ex_reg2	second source register in EX
	ex_reg_we	register write enable in EX
	ex_reg_write_address	register write address in XE
	ex_reg_write_data	register write data in EX

2.13 Regfile module

This module handle the instructions of requiring data and request to write registers.

2.14 ROM and RAM module

ROM module receive the address from InsFetch module and return the instruction in memory. RAM module receive two types of address signals : write and read. And handle the data memory.

3 Implementation on FPGA

3.1 Instructions Fetch

As first step to program on FPGA, I implement a simple InsFetch module on FPGA. It receive an original program point value from PC, then increment and return to PC receiver.

4 Test Bench

According to the textbook, I set 10 test to test bypassing, logic, shift, move, arithmetic, jump, branch, memory access, stall and combination of all operations.

Table 6: Ports Description of module Execute

	Port name	Description
input	rst	reset signal
	instruction_i	input instruction
	op_i	operator in ID
	ins_kind	instruction type
	reg1_i	first source register
	reg2_i	second source register
	reg_we_i	register write enable
	reg_write_address_i	register write address
	reg_write_data_i	register write data
	mem_hi_we	for data dependence
	mem_hi_write_data	
	mem_lo_we	
	mem_lo_write_data	
	wb_hi_read_data	
	wb_hi_we	
	wb_hi_write_data	
	wb_lo_read_data	
	wb_lo_we	
	wb_lo_write_data	
output	instruction_i	input instruction
	op_o	operator in ID
	reg1_o	first source register
	reg2_o	second source register
	reg_we_o	register write enable
	reg_write_address_o	register write address
	reg_write_data_o	register write data
	hi_we	hi register write enable
	hi_write_data	hi register write data
	lo_we	lo register write enable
	lo_write_data	lo register write data
	stall_request	stall request

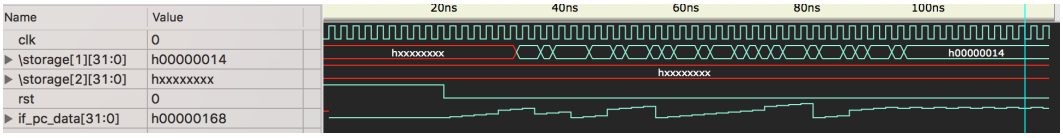

Figure 1: PC trace of Branch test

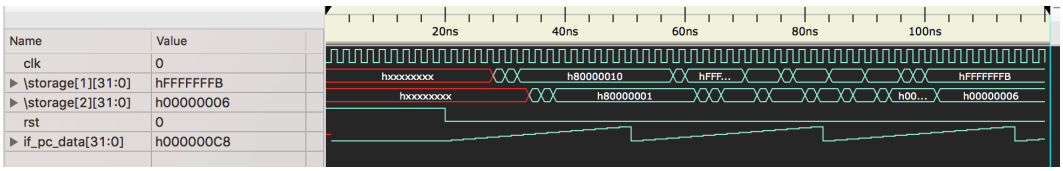

Figure 2: Registers trace of arithmetic test

Table 7: Ports Description of module EX_MEM

	Port name	Description
input	rst	reset signal
	clk	clock signal
	stall	stall signal
	ex_instruction	instruction in EX
	ex_op	operator in EX
	ex_reg1	first source register in EX
	ex_reg2	second source register in EX
	ex_reg_we	register write enable in EX
	ex_reg_write_address	register write address in EX
	ex_reg_write_data	register write data in EX
	ex_hi_we	hi register write enable in EX
	ex_hi_write_data	hi register write data in EX
	ex_lo_we	lo register write enable in EX
	ex_lo_write_data	lo register write data in EX
output	mem_instruction	instruction in MEM
	mem_op	operator in MEM
	mem_ins_kind	instruction type in MEM
	mem_reg1	first source register in MEM
	mem_reg2	second source register in MEM
	mem_reg_we	register write enable in MEM
	mem_reg_write_address	register write address in MEM
	mem_reg_write_data	register write data in MEM
	mem_hi_we	hi register write enable in MEM
	mem_hi_write_data	hi register write data in MEM
	mem_lo_we	lo register write enable in MEM
	mem_lo_write_data	lo register write data in MEM

Table 8: Ports Description of module Memory Access

	Port name	Description
input	rst	reset signal
	instruction	instruction
	op	type of operator
	reg1	first source register
	reg2	second source register
	memory_read_data	data read from memory
	reg_we_i	input register write enable signal
	reg_write_address_i	input register write address
	reg_write_data_i	input register write data
	hi_we_i	input hi register write enable
	hi_write_data_i	input hi register write data
	lo_we_i	input lo register write enable
	lo_write_data_i	input lo register write data
output	mem_instruction	instruction in MEM
	mem_op	operator in MEM
	mem_ins_kind	instruction type in MEM
	mem_reg1	first source register in MEM
	mem_reg2	second source register in MEM
	mem_reg_we	register write enable in MEM
	mem_reg_write_address	register write address in MEM
	mem_reg_write_data	register write data in MEM
	mem_hi_we	hi register write enable in MEM
	mem_hi_write_data	hi register write data in MEM
	mem_lo_we	lo register write enable in MEM
	mem_lo_write_data	lo register write data in MEM

Table 9: Ports Description of module ctrl

	Port name	Description
input	rst	reset signal
	id_stallreq	stall request from InsDecode module
	ex_stallreq	stall request from Execute module
output	stall	stall signal