A Single Cycle CPU by Verilog

B01902023 章莉

Environment

Although I've tried on ModelSim at last but I mainly implement the whole thing in MAC OSX using terminal with iverilog compilation and gtkwave for debugging. The two are applications for UNIX-like environment.

After installing MacPort, iverilog and gtkwave using sudo. Then you can easily type in make all on the terminal to compile the file and get the corresponding executable file. After compiling, simply type in "./testbench" without quote to run the simulation and thereafter an output.txt will be created. If you want to take a look at the wave, remember to install gtkwave and then type in "gtkwave TestBench.vcd" without quote.
(http://kreegerresearch.com/wiki/doku.php?id=hdl:verilog:icarusverilogsetupmac) FYI
ps. You'll have to install MacPort first. (macports.org)

Implementation

CPU.v

This is the body of the whole project and is the first module that i implemented. It connects all the others modules and combine them together. Inside this code, there's no implementation of modules, the only thing I've done in this file is connecting each wire between different modules.

Control.v

This module reads in a 5 bit input from the instruction and decided as R-type if the value is 6'b000000 and I type if the value is 6'b001000. The signals are then decided due to the type.

	RegDst	**ALUOp**	**ALUSrc**	**RegWrite**
R-type	1'b1	1'b1	1'b0	1'b1
I-type	1'b0	1'b0	1'b1	1'b1

Adder.v

This module simply eats two 32 bit input and add them together and assign the value to the output.

PC.v

This module is controlled by clock ,rst_i, and start_i signals. And output the pc address every time when positive edge of clk_i or negative edge of rst_i occur.

Instruction_Memory.v

This module reads in the pc_o from the module PC and find out the corresponding instructions in the memory and output it to the instruction wire in the module CPU.

Registers.v

This module is the register file in the CPU module, it allocates the 32 registers and is the main control unit of the registers.

MUX5.v & MUX32.v

This module output input2 if the select value is 1 and input1 if the select value is 0 then the input1 is outputted.

Signed_Extend.v

This module reads in a 16 bits and assigned it to the output of a 32 bits value.

ALU.v

This module reads in the 32 bit inputs as operands and a 3 bit control input and do the arithmetic operations due to the 3 bit input.

	AND	OR	ADD II ADDI	SUB	MUL
Control bits	0	1	2	3	4

ALU_Control.v

This module reads in the value from instruction bits and ALUOp and then output the corresponding ALU control bits. If the ALUOp is set to 0 then it will output the control bits as add.

	AND	OR	ADD II ADDI	ADDI	SUB	MUL
Control bits	0	1	2	2	3	4
func code	6’b100100	6’b100101	6’b100000	x	6’b100010	6’b011000
ALUOP	1	1	1	0	1	1

Problems Encountered & Solutions

The first problem is that because my computer is of Mac OSX environment and there is certainly no mac version of modelSim to install and that I don’t want to install Windows on my desktop. And in many aspects that I don’t like modelSim. I’ve been searching for alternatives. And at last I found that by using iverilog + gtkwave can make be get to something near modelSim.

And the second one is that at first the output value is a bit strange and so that I checked the output wave form and found out that I’ve made a mistake in Singed_Extend module that I shift the value left 16 times to make it to be 32 bits which is apparently wrong.

What I’ve learned

I’ve gotten a clearer view of how a CPU works and how to compile Verilog under LINUX environment.