Computer Architecture Homework #4

Verilog Exercise Single-cycle MIPS Implementation

TA: Chi-Hung Weng (r08943010@ntu.edu.tw) contact me if you have problem with HW spec
Yan-Lun Wu (r08943016@ntu.edu.tw)

You can contact us if you have problem in Verilog.

Due: 2019/12/31 13:00 Tuesday (CEIBA, no late homework is allowed.)

1. Introduction

Microprocessor without Interlocked Pipeline Stages (MIPS) is a widely used instruction set architecture. We've already had comprehensive understandings of it in our Computer Architecture course. In this exercise, we're going to implement the single-cycle MIPS architecture using Verilog. This exercise will give you a hardware viewpoint to this architecture.

The MIPS architecture is shown in Fig. 1. In this exercise, the instruction memory and data memory are implemented in the testbench. Except the memories, you need to implement everything else by yourself.

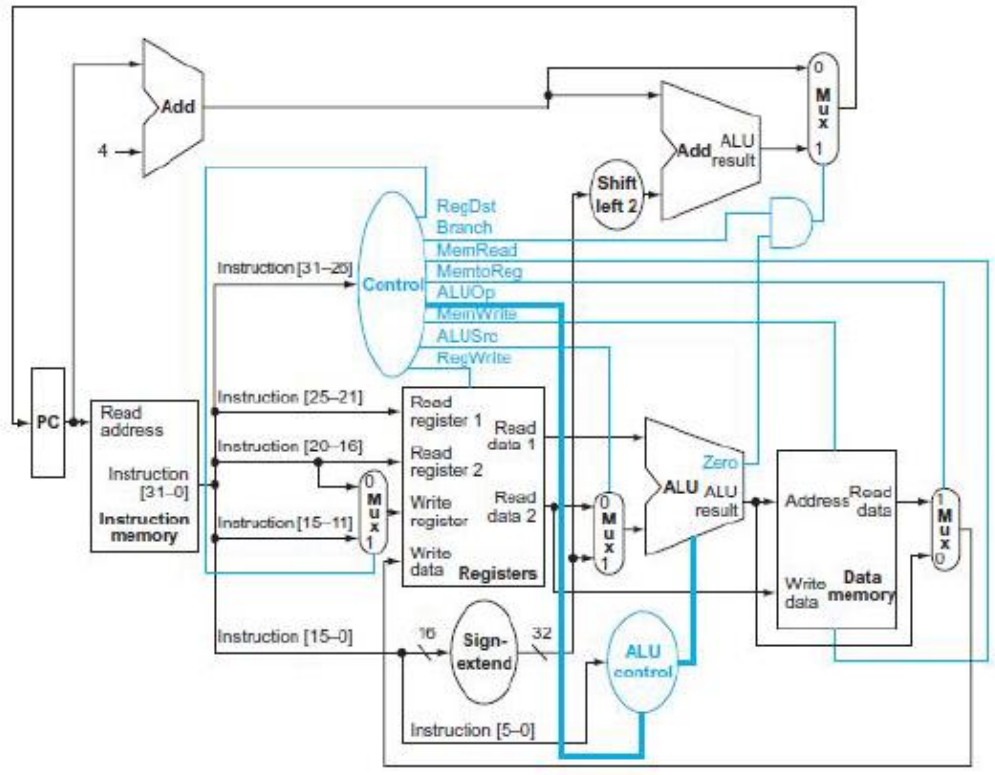


Figure1: Single-cycle MIPS architecture.

2. Specification

The input/output pins are defined in Table. 1. The required instructions you need to support in the baseline are listed in Table. 2. The complete machine codes for the required instructions are not listed in this document. Please refer to your textbook or MIPS Green Sheet for the full machine code, our testbench follows the standard machine code rules. (We also provide you MIPS Green Sheet with marker on the instructions that we asked you to do. The password is same as course slides)

Tabel1: I/O pins specification

Signal name	I / O	Bit width	Description
clk	I	1	Clock signal. Positive edge trigger.
rst_n	I	1	Active low **synchronous** reset signal.
IR_addr	O	32	Output address of the instruction memory.
IR	I	32	Instruction read from instruction memory
ReadDataMem	I	32	Signals from data memory.
CEN	O	1	Chip-enable; set it low to read/write data.
WEN	O	1	Write-enable; set it low to write data into memory.
A	O	7	Address for data memory.
Data2Mem	O	32	Registers output signal.
OEN	O	1	Read-enable; set it low to read data from memory. Note that OEN should be the inverse of WEN.

Table 2: Your MIPS needs to support the instructions in this table to pass baseline.

Instrucion	Type	Op code (Hex)	Func code (Hex)
sll	R	0	00
srl	R	0	02
add	R	0	20
sub	R	0	22
and	R	0	24
or	R	0	25
slt	R	0	2A
addi	I	8	N/A
lw	I	23	N/A
sw	I	2B	N/A
beq	I	4	N/A
bne	I	5	N/A

j	J	2	N/A
jal	J	3	N/A
jr	R	0	8

3. Timing Diagram for Memory

In the MIPS architecture shown in Fig. 1, there are two memories needed. Here we use a ROM for the instruction memory. As soon as you give it an address, the instruction memory sends the instruction immediately.

As for data memory, we use a simulated SRAM for it in this exercise. Fig. 2 shows the timing diagram for reading data from a memory. Fig. 3 shows the timing diagram for writing data into a memory.

Figure2: Read data from a memory.

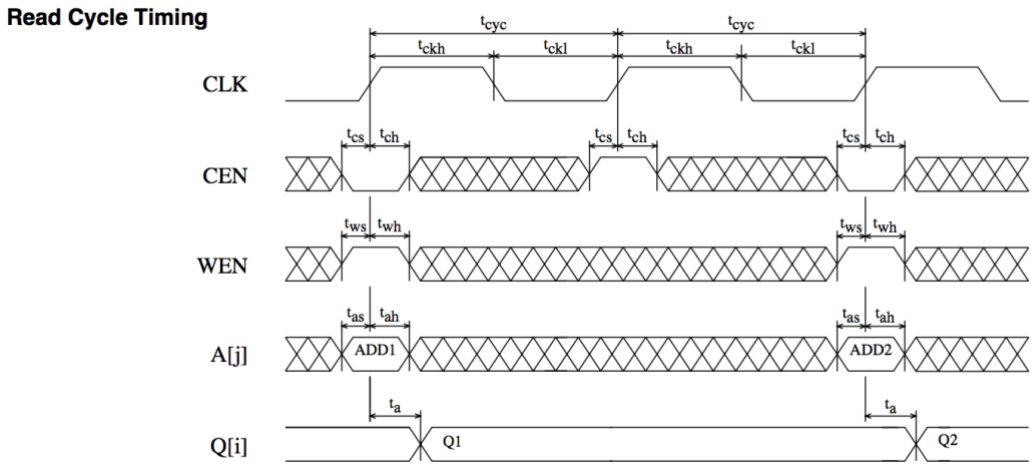
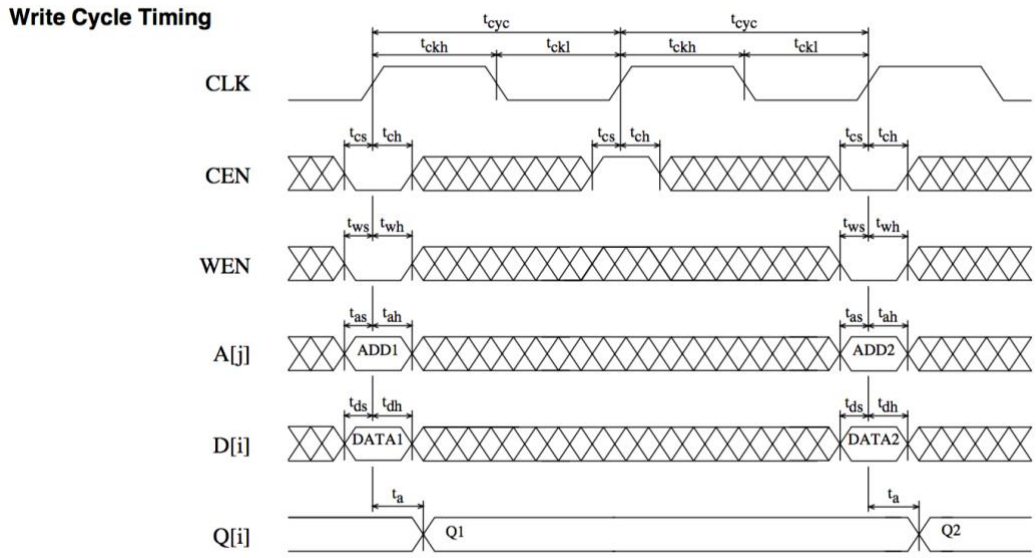


Figure3: Write data from a memory.



4. Floating-Point Unit

For more advanced exercise, you can add some function to your MIPS to support some floating-point instructions. You need to extend from your baseline code, adding new registers $FR0 - $FR31 to store floating-point numbers ($FR0 is not always 0 !).

Since some floating-point operations are too complex to implement, you have to use DesignWare modules to accomplish this part. DesignWare modules are stored in the following directory.

```
/usr/cad/synopsys/synthesis/cur/dw/sim_ver/
```

During RTL Simulation, include these modules by adding the following in command line :

```
-y /usr/cad/synopsys/synthesis/cur/dw/sim_ver/    \
+libext+.v +incdir+/usr/cad/synopsys/synthesis/cur/dw/sim_ver/
```

Table 3 are the instructions that will be tested in our testbench, where double precision instructions are marked with darker background. You can also just do single precision part to get partial points.

Table 3: floating-point instructions that your MIPS have to support

Instrucion	Type	Op code (Hex)	FMT(Hex)	Func code (Hex)
add.s	FR	11	10	00
sub.s	FR	11	10	01
mul.s	FR	11	10	02
div.s	FR	11	10	03
lwcl	I	31	N/A	N/A
swcl	I	39	N/A	N/A
c.eq.s	FR	11	10	32
bclt	FI	11	8	N/A
add.d	FR	11	11	00
sub.d	FR	11	11	01
ldcl	I	35	N/A	N/A
sdcl	I	3D	N/A	N/A

Grading:

1. (8%) RTL Simulation pass Single precision testbench
2. (8%) RTL Simulation also pass Double precision testbench
3. (4%) Gate-Level Simulation pass testbench that you pass in RTL Simulation

5. Simulation Scripts

I. Environmental Settings

Source the ./cshrc file provided in HW3 for the environmental setting

source ./cshrc

II. RTL Simulation

ncverilog tb.v SingleCycleMIPS.v +define+Baseline +access+r

III. Synthesis

dc_shell -f run.tcl

Please check if there's any error message (you can ignore most warning messages). If any, read the error messages and try to resolve them; if not, enter exit to leave Design Compiler.

IV. Gate-level Simulation

ncverilog tb.v SingleCycleMIPS_syn.v tsmc13.v +define+Baseline+SDF

+access+r

V. RTL Simulation for FPU

ncverilog tb.v SingleCycleMIPS_FPU.v +define+Baseline +access+r

ncverilog tb.v SingleCycleMIPS_FPU.v +define+FPU+Single +access+r

ncverilog tb.v SingleCycleMIPS_FPU.v +define+FPU+Double +access+r

Your **SingleCycleMIPS_FPU.v** should also pass the baseline testbench, which means you have to extend from your baseline Verilog code instead of writing a new one.

VI. Gate-level Simulation for FPU

ncverilog tb.v SingleCycleMIPS_FPU_syn.v tsmc13.v +define+FPU+Single

+access+r

ncverilog tb.v SingleCycleMIPS_FPU_syn.v tsmc13.v

+define+FPU+Double +access+r

VII. Debug

nWave &

Use program nWave to view the simulated signals. This will be very helpful for this exercise.

<u>TA will use these commands to test your files.</u>

6. Bonus (A*T value)

When designing a digital system, there are some metrics to verify how well it performs, the common metrics are timing, area and power. We will use timing and area in this exercise.

All groups in class will compete their A*T value(cost) of baseline design.

- Top 50% with the smallest cost get 5 points bonus.
- Top 25% with the smallest cost get additional 5 points bonus.(Total 10)

You have to calculate your A*T value and put it in your report to enter the competition. Please refer to Appendix I to see how to calculate A*T in this exercise.

7. Report

Your report should include the following parts:

A. Snapshot
   1. Readme.txt content (see sec. 8)
   2. All RTL Simulation you pass (see Appendix II for example)
   3. All Gate-level Simulation you pass
   4. Timing report
   5. Area report
B. Your baseline A*T value calculated from numbers in the snapshot
C. Please describe how you design this circuit and what difficulties you encountered when working on this exercise.
D. Work distribution
E. (optional) how you improve your A*T value.

8. Files

The deadline for this exercise is 13:00, Dec. 31th. Please pack your files in a folder named CA_hw4_yourgroupindex_member1ID_member2_ID, compress it into a HW4.zip file and then submit to CEIBA. There's a 5% penalty for incorrect upload format. No late submission is accepted.

Example:

./CA_hw4_G0_r08943010_r08943016/

*.v	(RTL files)
*_syn.v	(synthesized gate-level netlist)
*_syn.ddc	(Design database generated by Synopsys Design Compiler)
*_syn.sdf	(Pre-layout gate-level sdf)
G0_report.pdf	(Replace with your group index)
Readme.txt	(show cycle time and describe what you have done)

Readme format:

```
cycle time: 10
FPU Single: Y/N
FPU Double: Y/N
```

cycle time: refer to Appendix I

"Y" if you did the part, otherwise "N".

9. Grading Criteria

Item	Description
Baseline correctness(70%)	1. Pass baseline RTL simulation (50%) 2. Pass baseline Gate-level Simulation (20%)
FPU(20%)	Describe in detail in sec 4
Report (10%)	Describe in detail in sec 7
Bonus(10%)	Top 25% groups for smallest A*T will get 10. Top 50% groups for smallest A*T will get 5.

10. Appendix

I. Calculate A*T

A: from area report (Total cell area)

(this example is from other circuit)

```
Number of ports:                           30
Number of nets:                          1902
Number of cells:                         1769
Number of combinational cells:           1571
Number of sequential cells:               197
Number of macros/black boxes:               1
Number of buf/inv:                        215
Number of references:                     130

Combinational area:              13698.017932
Buf/Inv area:                     1082.941187
Noncombinational area:            6933.878880
Macro/Black Box area:            69557.296875
Net Interconnect area:          220547.789185

Total cell area:                 90189.193686
Total area:                     310736.982871
```

T: clock cycle you use to synthesis and pass gate-level simulation

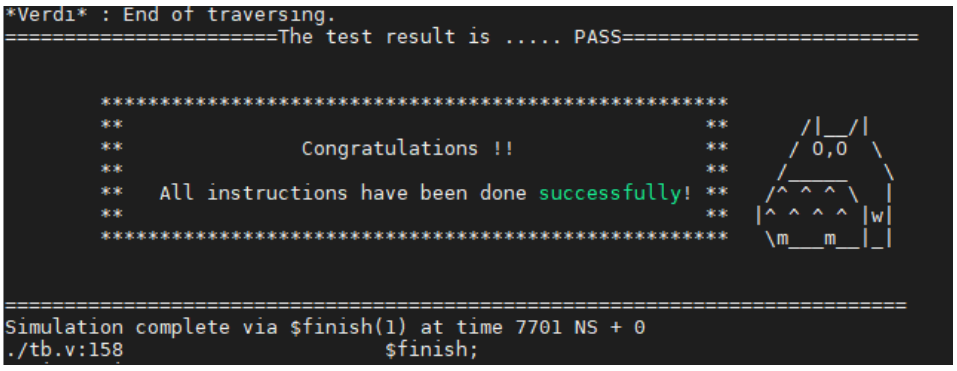
If you use 10 to synthesis but pass simulation at 12, please use 12 to calculate and also write 12 in Readme.txt . We will run your gate-level simulation on the cycle time you provide.

```
1  cycle time: 10
2  FPU Single: Y/N
3  FPU Double: Y/N
```

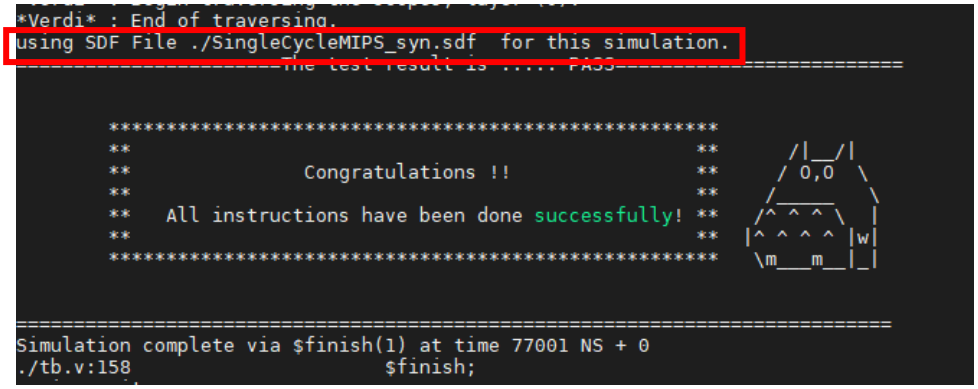
Based on the A, T got above, multiply them to get the result.

II. Snapshot examples

1. Baseline RTL



2. Gate-level



3. FPU