Computer Organization
Final Project

Simple CPU

Design Description (1/3)

- An arithmetic logic unit (ALU) is a digital electronic circuit that performs arithmetic and bitwise logical operations on integer binary numbers. In order to increase clock frequency and throughputs, pipeline technology can be used.
- In this project, a string of instructions will be given. Your job is to decode these instructions and execute. You should try to design a simple 16-bits CPU with 16 registers.

Design Description (2/3)

- The following is the basic required instruction set: (similar to MIPS)

Type	**Format (19 bits)**				
Logic	opcode (3 bits)	rs (4 bits)	rt (4 bits)	rd (4 bits)	func (4 bits)
Operation	opcode (3 bits)	rs (4 bits)	rt (4 bits)	rd (4 bits)	rl (4 bits)
Immediate	opcode (3 bits)	rs (4 bits)	rt (4 bits)	immediate (8 bits)	

Design Description (3/3)

- Register s(rs), Register t(rt), Register d(rd) and Register l(rl) represent the address of registers. Since the instruction takes 4 bits to store the address, it means we have 16 registers, from r0 to r15. Each register reserve 16 bits to store data, e.g. rs = 4’b0000 means one of operands is “r0”. rt = 4’b0101 means one of operands is “r5”, and so on. And the outputs are corresponding to the registers, ex: out = r0 when we check your answer.
- There will be a 8192*16 data memory in this project. You can access this memory according to the instruction.

Instruction

Function	Meaning	Instruction Binary Encode
AND	rd = rs & rt (bit wise)	000-ssss-tttt-dddd-0000
OR	rd = rs | rt (bit wise)	000-ssss-tttt-dddd-0001
XOR	rd = rs ^ rt (bit wise)	000-ssss-tttt-dddd-0010
ADD	rd = rs + rt	000-ssss-tttt-dddd-0011
SUB	rd = rs - rt	000-ssss-tttt-dddd-0100
MULT	{rd, rl} = rs * rt	001-ssss-tttt-dddd-llll
SQUARE	{rd, rl} = rs^2	010-ssss-tttt-dddd-llll
ADDI	rt = rs + immediate (sign)	011-ssss-tttt-iiii-iiii
SUBI	rt = rs - immediate (sign)	100-ssss-tttt-iiii-iiii
STORE	Memory[rs[12:0] + immediate (sign)] = rt	101-ssss-tttt-iiii-iiii
LOAD	rt = Memory[rs[12:0] + immediate (sign)]	110-ssss-tttt-iiii-iiii

Example

- instruction[18:0] = 19h’0_1230

  opcode = 0, rs = 1, rt = 2, rd = 3, func = 0 → r3 = r1 & r2

- instruction[18:0] = 19h’1_2345

  opcode = 1, rs = 2, rt = 3, rd = 4, rl = 5 → {r4, r5} = r2 * r3

- instruction[18:0] = 19h’3_45ff

  opcode = 3, rs = 4, rt = 5, immediate = (-1) → r5 = r4 + (-1)

- instruction[18:0] = 19h’5_6701

  opcode = 5, rs = 6, rt = 7, immediate = 1 → Memory[r6[12:0] + 1] = r7

- instruction[18:0] = 19h’6_78ef

  opcode = 6, rs = 7, rt = 8, immediate = 127 → r8 = Memory[r7[12:0] + 127]

Inputs

- All input signals will be **synchronized** at **negative edge** of the clock.

Input	Bit Width	Definition and Description
clk	1	Positive edge trigger clock.
rst_n	1	Asynchronous active-low reset.
in_valid	1	**instruction[18:0]** is valid when **in_valid** is high.
instruction	19	As above.
MEM_out	16	When **WEN** is high, you can load data **MEM_out[15:0]** from memory.

Outputs

Output	Bit Width	Definition and Description
busy	1	1. **busy** should be low if you want to get the next **instruction[18:0]**. 2. If output port **busy** is low, the next **instruction[18:0]** and **in_valid** will be given immediately and stay one cycle. 3. As long as **busy** is low, **instruction[18:0]** will be sent as input every cycle.
out_valid	1	**out_valid** is high when you want to output your answer. We will check your answer when your **out_valid** is high.
out0 ~ out15	16	**out0[15:0]** ~ **out15[15:0]** are synchronous to the positive edge of **clk**, and TA's pattern will sample your result at the following negative edge of **clk**.
WEN	1	Write Enable Negative is an input signal of memory. When **WEN** is high, you can load data from memory. When **WEN** is low, you can write data into memory.
ADDR	13	Memory address.
MEM_in	16	When **WEN** is low, **MEM_in[15:0]** will be written into memory.

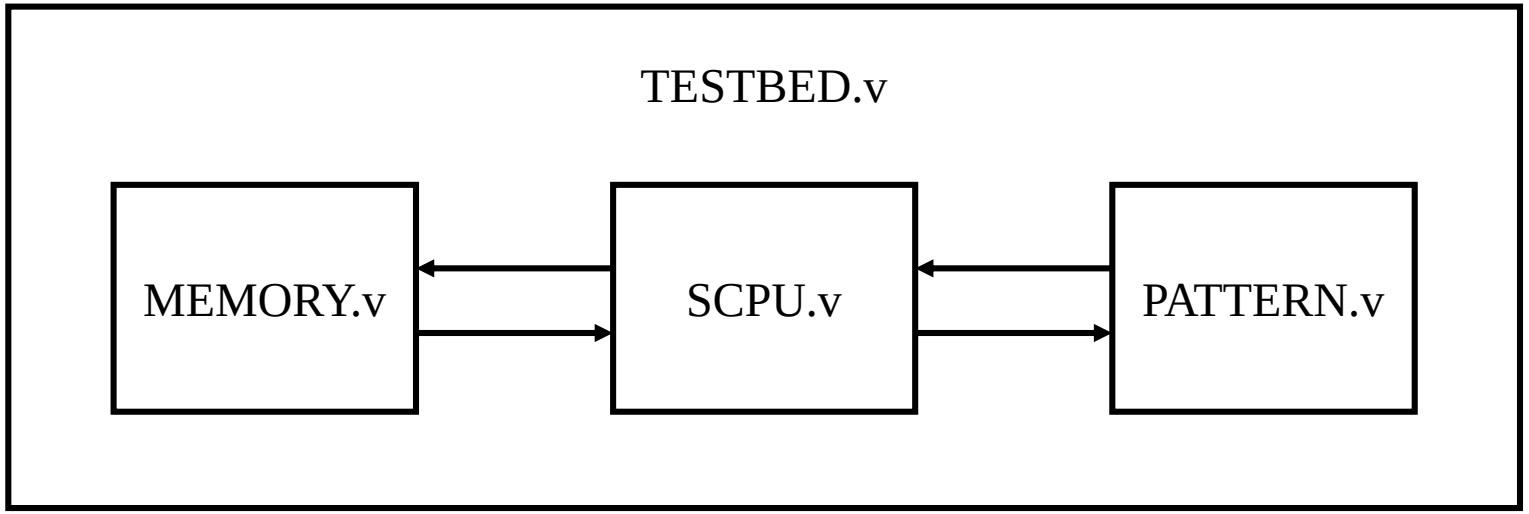
Specifications (1/2)

- Top module name : SCPU (File name : SCPU.v)
- Input pins: clk, rst_n, in_valid, instruction[18:0], MEM_out[15:0]
- Output pins: busy, out_valid, out0[15:0] ~ out15[15:0], WEN, ADDR[12:0], MEM_in[15:0].
- It is an asynchronous active-low reset and positive edge clk architecture.
- The reset signal would be given only once at the beginning of the simulation.

Specifications (2/2)

- All output signals should be reset after the reset signal is asserted.
- All of data in registers are signed number, including your output.
- The busy should be only kept high within 10 cycles.
- The rd won’t be equal to rl when doing MULT and SQUARE.
- The first sixteen instructions are LOAD for out0 ~ out15.
- Overflow is allowed in your design.

Block Diagram

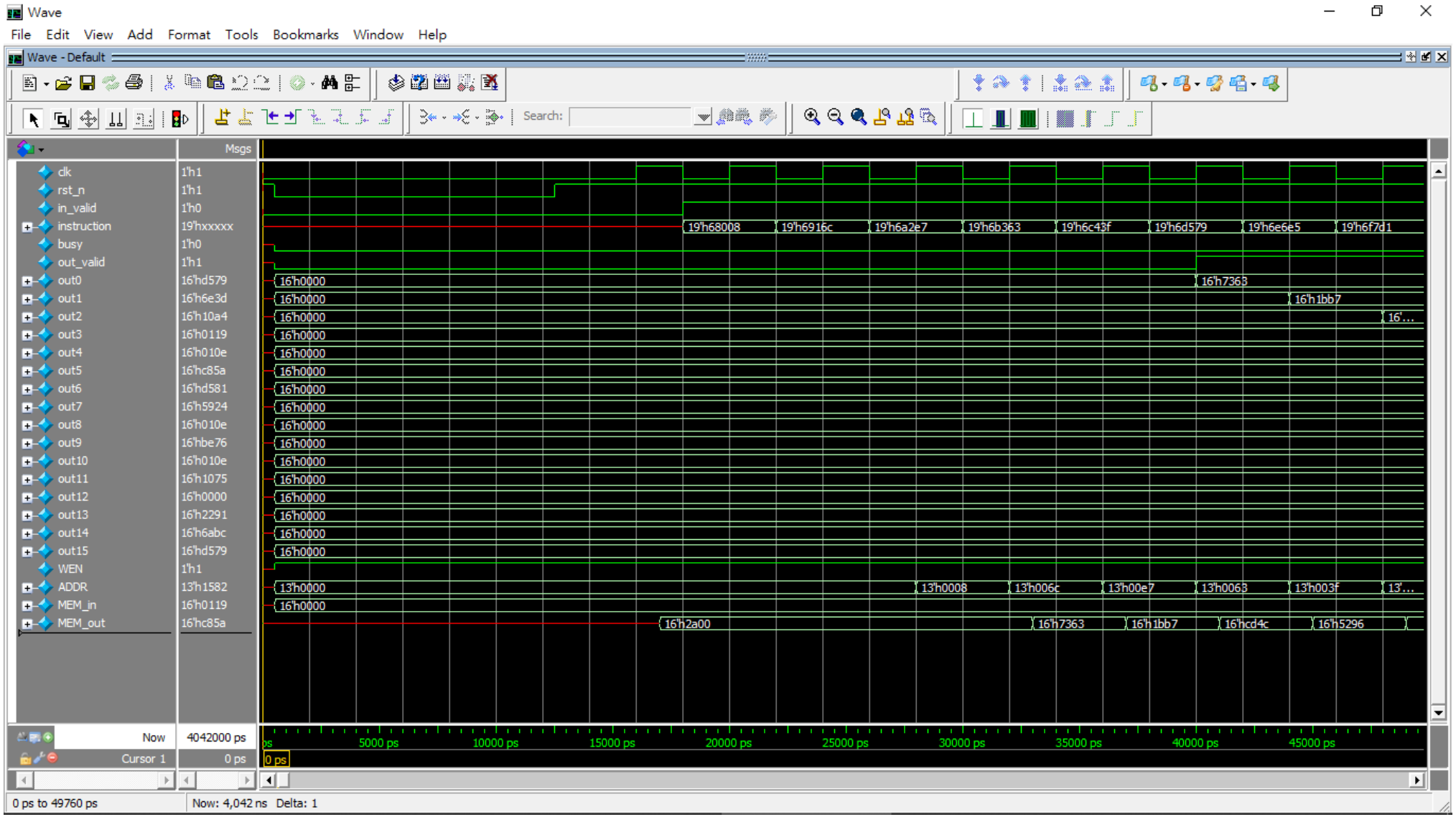


Grading Policy

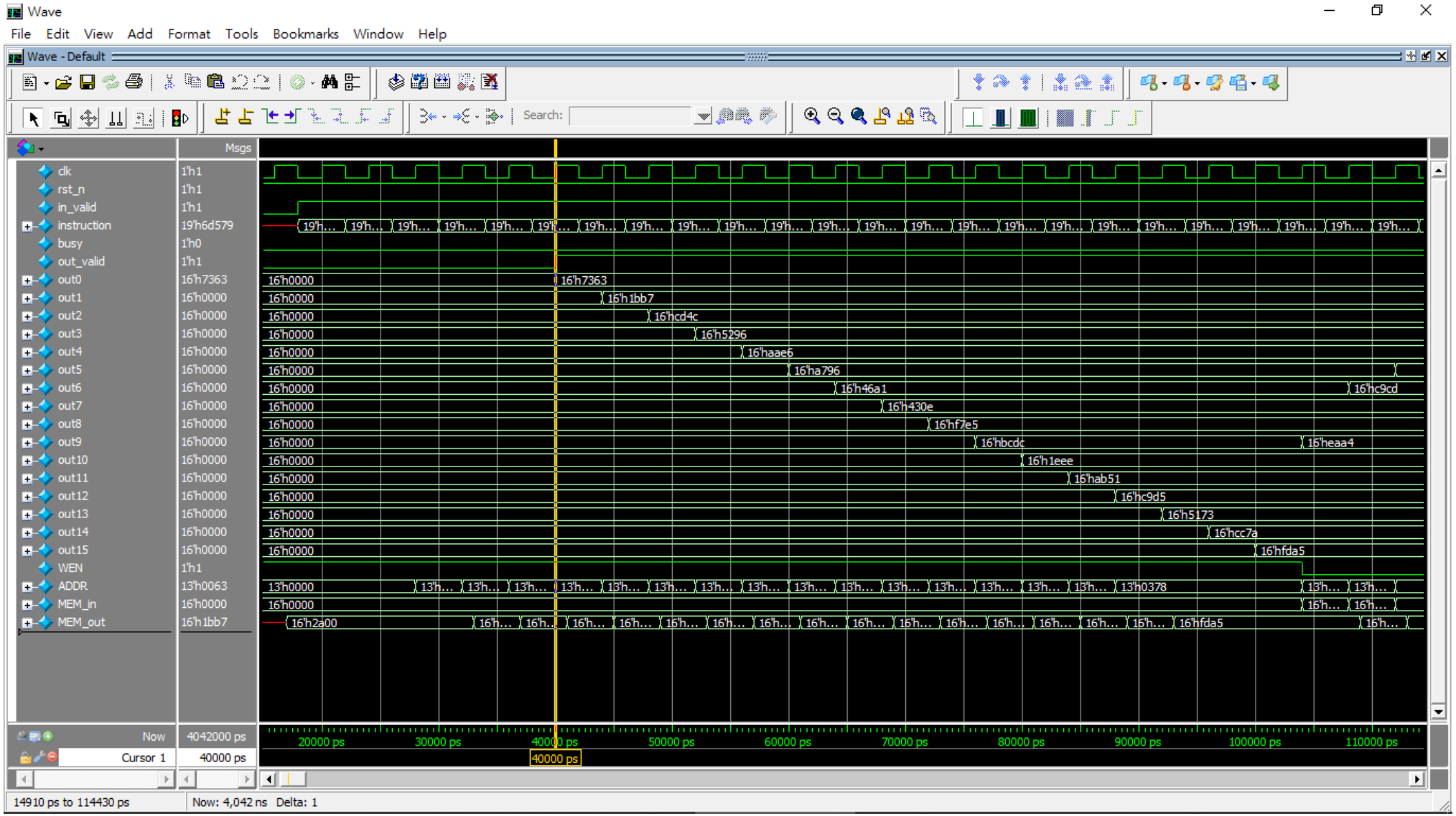
- Function Validity: 75%
- Pipeline: 10% (Without using **busy**)
- Execution Cycles: 15% (You will get full score in this part if you use pipeline architecture)

- Please upload “SCPU_studentID.v” on New e3 platform
- Due Date: 23:59, January 12, Sunday, 2020
- If you have a late submission, you can only get 80% of your original score.

Example Waveform (Pipeline)



Example Waveform (Pipeline)



Example Waveform (With busy)