

Problem Set3

Names: Klunpaitoon, Benjamin - 100% contribution
Le Alex - 100% contribution
Liang Su - 100% contribution

Student Team Name: Le & Liang

1. Describe the schematic design of the processor that connects all the necessary components, data paths, and control signals.

The processor has a register, ALU, and a control unit. It also accesses memory for information. The processor has flags that can trigger the control unit and register to do things and also have the ALU process something. The control unit has `rf_read`, `rf_write`, `rf_address`, `rf_data`, `opcode`, `op1`, `op2`, and `result` as flags. `Rf_read` and `write` are for reading from and writing to registers and memory. `Rf_address` is for an address of some data, `rf_data`. The `opcode` is for what operation the ALU would do, `op1` and `op2` being the two operands used in that operation. The `result` is the output of the ALU. The processor starts the control unit, which creates a program counter. The program counter runs its first instruction which sets `rf_read` to 1 and `rf_write` to 0. This means that the program wants to read something and not write any information yet. The control unit does all the fetching and decoding based on that information and saves all the variables needed for the operations. The flag in the control unit, `rf_read`, is linked to another flag with the same name in the processor and another in the register. When one updates, all of them update as well. After a cycle passes and everything is read, the processor now has an updated `rf_data` flag. The register gets a read signal, loads `rf_data`, and then the data is sent to the processor and the control unit. This is due to `rf_data` being a flag that exists in the control unit, processor, and register. Now that we've retrieved the information, `rf_read` and `rf_write` are set to 0, and we start to process information using the ALU. Based on the data we have, the `opcode` is set for an operation and `op1` and `op2` are loaded with the correct data. The ALU doesn't use any of the flags like the other components, instead the ALU uses a clock. For our clock, it only runs on a positive edge. The ALU only processes its information when the correct data is loaded in. When it does produce a result, it passes it into the processor via a wire. The wire also exists in the register and control unit and simultaneously updated, and returned since it was the result.

2. Describe the main updates of Verilog implementation

For our Verilog information, we implemented a control unit and a 32bit register. We put in additional parameters for the control unit that allows reading, writing, and other such functions. The 32 bit register is a 32x32 array of virtual memory that we've allocated. More specifically, for the ALU we added registers for use and also added a ZERO output functionality that returns if the output is 0 or not. For the register, we created the 32x32 storage, as well as initializing it to 0. We reset the block on a negative edge of a RST signal. We return the content from `ADDR_R1` and `ADDR_R2` for a read and modify `ADDR_W's`

content to DATA_W on a write request. The control unit implements a state machine which determines what state the machine is and what it should be doing. We implemented the ability to generate control signals such as fetching and decoding. We also created access to the memory and created a program counter to keep track of where we are. Finally, the processor is there to give an instruction to the control unit rather than making the control unit fetch the instruction itself. For each of these files, we created test-benches to make sure they worked.

3. Capture the screen shots of running the simulation and results (screen output, waveforms)

The screenshot displays the ModelSim PE Student Edition 10.4a interface during a simulation. The main window shows the Design unit hierarchy on the left, the Objects window in the center, and the Wave window on the right. The Transcript window at the bottom shows the simulation results.

Design unit hierarchy:

Instance	Design unit	Design unit type	Top Category	Visibility	Total coverage
alu_tester	alu_tester	Module	DU Instance	+acc...	
test_and_coun...	alu_tester	Task	-	+acc...	
test_golden	alu_tester	Function	-	+acc...	
alu_inst_01	ALU	Module	DU Instance	+acc...	
#INITIAL#19	alu_tester	Process	-	+acc...	
#vsm_capacity#	Capacity	Statistics	+acc...		

Objects window:

Name	Value	Kind	Mode	Now
total_test	32'h0...	Inte...	Internal	
pass_test	32'h0...	Inte...	Internal	
oprn_reg	6'h09	Pack...	Internal	
op1_reg	32'h0...	Pack...	Internal	
op2_reg	32'h0...	Pack...	Internal	
r_net	32'h0...	Net	Internal	
z_net	32'h0...	Net	Internal	

Processes (Active) window:

Name	Type (filtered)	State	Order	Parent P
#INITIAL#19	Initial	Active	1	/alu_tes

Wave window:

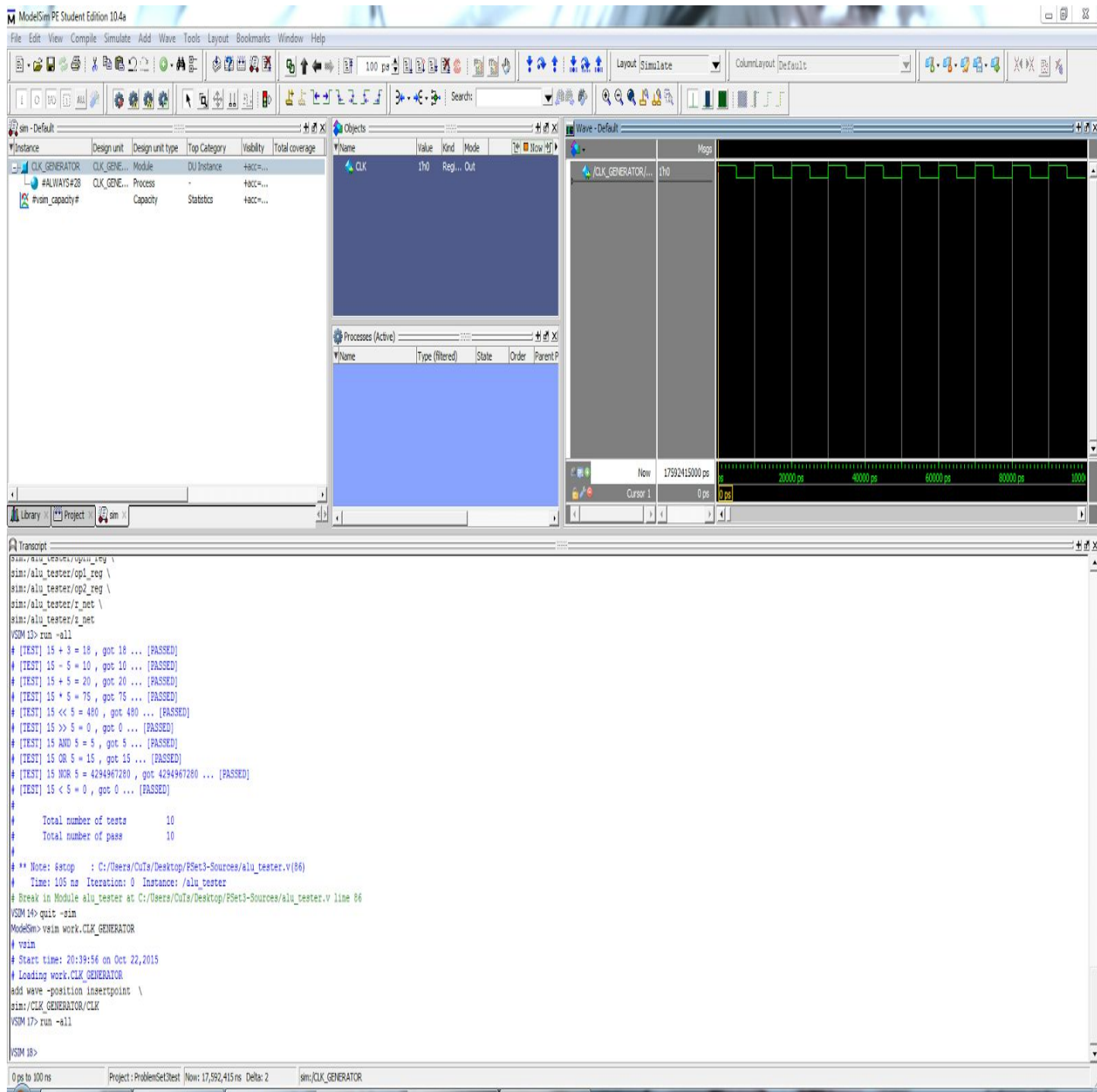
Signal	Value	Time
/alu_tester/total_test	32'h00000000	Now
/alu_tester/pass_test	32'h00000000	Now
/alu_tester/oprn_reg	6'h00	Now
/alu_tester/op1_reg	32'h00000000	Now
/alu_tester/op2_reg	32'h00000000	Now
/alu_tester/r_net	32'h00000000	Now
/alu_tester/z_net	32'h00000000	Now

Transcript window:

```

ModelSim> vsim work.alu_tester
# vsim
# Start time: 21:14:46 on Oct 22, 2015
# Loading work.alu_tester
# Loading work.ALU
add wave -position insertpoint \
sim:/alu_tester/total_test \
sim:/alu_tester/pass_test \
sim:/alu_tester/oprn_reg \
sim:/alu_tester/op1_reg \
sim:/alu_tester/op2_reg \
sim:/alu_tester/r_net \
sim:/alu_tester/z_net
VSM 40> run -all
# [TEST] 15 + 3 = 18 , got 18 ... [PASSED][ZERO FLAG SET]
# [TEST] 15 - 5 = 10 , got 10 ... [PASSED][ZERO FLAG SET]
# [TEST] 15 + 5 = 20 , got 20 ... [PASSED][ZERO FLAG SET]
# [TEST] 15 * 5 = 75 , got 75 ... [PASSED][ZERO FLAG SET]
# [TEST] 15 < 5 = 480 , got 480 ... [PASSED][ZERO FLAG SET]
# [TEST] 15 > 5 = 0 , got 0 ... [PASSED]
# [TEST] 15 AND 5 = 5 , got 5 ... [PASSED][ZERO FLAG SET]
# [TEST] 15 OR 5 = 15 , got 15 ... [PASSED][ZERO FLAG SET]
# [TEST] 15 NOR 5 = 4294967280 , got 4294967280 ... [PASSED][ZERO FLAG SET]
# [TEST] 15 < 5 = 0 , got 0 ... [PASSED]
#
# Total number of tests      10
# Total number of pass      10
#
# ** Note: $stop      : C:/Users/CuTs/Desktop/PSet3-Sources/alu_tester.v(88)
# Time: 105 ns Iteration: 0 Instance: /alu_tester
# Break in Module alu_tester at C:/Users/CuTs/Desktop/PSet3-Sources/alu_tester.v line 88
VSM 41>
  
```

Status bar: 104050 ps to 105050 ps | Project: ProblemSet3test | Now: 105 ns Delta: 0 | sim:/alu_tester/#INITIAL#19



4. Highlight key experiences encountered while implementing the processor.

We encountered many issues during the implementation of the processor. We had no idea how Verilog worked, so we had to figure out the syntax and such of that language. Secondly, we were not well versed in computer circuits and design, so we had to haphazardly try out many different things. While we were implementing the processor, we had to learn how data traveled, what signals needed to be sent, and how information was

sent. We had to look up many of these things. The control unit was mind boggling and we had a lot of difficulty with it. When we finally implemented the test bench for the ALU, it was difficult even though we had a basis from problem set 2.