System Synthesis and Modeling (CSCE 3953)

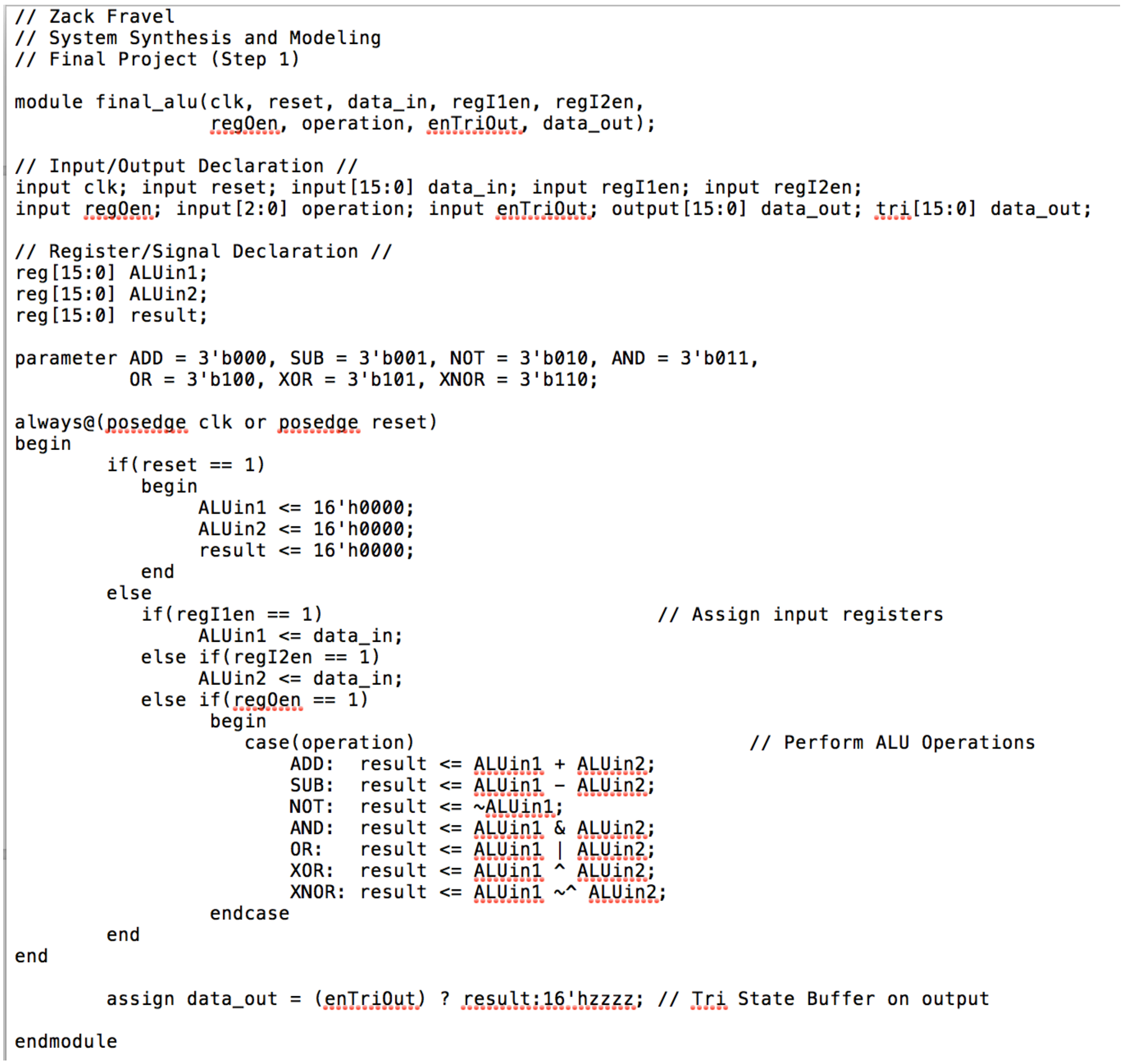
Final Project (Step 1)

Zack Fravel

10/28/16

zpfravel@uark.edu

Code



```verilog
// Zack Fravel
// System Synthesis and Modeling
// Final Project (Step 1)

module final_alu(clk, reset, data_in, regI1en, regI2en,
                 regOen, operation, enTriOut, data_out);

// Input/Output Declaration //
input clk; input reset; input[15:0] data_in; input regI1en; input regI2en;
input regOen; input[2:0] operation; input enTriOut; output[15:0] data_out; tri[15:0] data_out;

// Register/Signal Declaration //
reg[15:0] ALUin1;
reg[15:0] ALUin2;
reg[15:0] result;

parameter ADD = 3'b000, SUB = 3'b001, NOT = 3'b010, AND = 3'b011,
          OR = 3'b100, XOR = 3'b101, XNOR = 3'b110;

always@(posedge clk or posedge reset)
begin
        if(reset == 1)
           begin
                ALUin1 <= 16'h0000;
                ALUin2 <= 16'h0000;
                result <= 16'h0000;
           end
        else
           if(regI1en == 1)                                     // Assign input registers
                ALUin1 <= data_in;
           else if(regI2en == 1)
                ALUin2 <= data_in;
           else if(regOen == 1)
                 begin
                    case(operation)                                     // Perform ALU Operations
                        ADD:  result <= ALUin1 + ALUin2;
                        SUB:  result <= ALUin1 - ALUin2;
                        NOT:  result <= ~ALUin1;
                        AND:  result <= ALUin1 & ALUin2;
                        OR:   result <= ALUin1 | ALUin2;
                        XOR:  result <= ALUin1 ^ ALUin2;
                        XNOR: result <= ALUin1 ~^ ALUin2;
                 endcase
        end
end

        assign data_out = (enTriOut) ? result:16'hzzzz; // Tri State Buffer on output

endmodule
```

Testbench

```
// Zack Fravel
// System Synthesis and Modeling
// Final Project (Step 1)

`timescale 1ns/1ps
module final_alu_tb;

// Inputs
        reg CLK; reg RESET; reg[15:0] BUS_in;
        reg R1EN; reg R2EN; reg ROEN;
        reg[2:0] OPERATION; reg ENTRIOUT;
// Outputs
        wire BUS_out;
// Declare ALU
        final_alu test (CLK, RESET, BUS_in, R1EN, R2EN, ROEN, OPERATION, ENTRIOUT);

// Testbench

        always #10 CLK = ~CLK;          // 20 ns clock cycle (50 MHz)

        initial
           begin
                CLK = 0;
                RESET = 0;
                BUS_in = 16'h0000;
                R1EN = 0; R2EN = 0; ROEN = 0;
                OPERATION = 3'b000; ENTRIOUT = 0; // Do nothing
             #40;
                BUS_in = 16'h0001;
                R1EN = 1; R2EN = 0; ROEN = 0;
                OPERATION = 3'b000; ENTRIOUT = 0; // Read 1 into reg 1
             #20;
                BUS_in = 16'h0001;
                R1EN = 0; R2EN = 1; ROEN = 0;
                OPERATION = 3'b000; ENTRIOUT = 0; // Read 1 into reg 2
             #20;
                BUS_in = 16'h1111;
                R1EN = 0; R2EN = 0; ROEN = 1;
                OPERATION = 3'b000; ENTRIOUT = 0; // Add reg 1 and reg 2
             #20;
                BUS_in = 16'h1001;
                R1EN = 0; R2EN = 0; ROEN = 0;     // also shows that changing data_in doesn't change anything at this stage
                OPERATION = 3'b000; ENTRIOUT = 1; // Output Result
             #20;
                BUS_in = 16'h1011;
                R1EN = 1; R2EN = 0; ROEN = 0;
                OPERATION = 3'b000; ENTRIOUT = 0; // read into reg 1
             #20;
                BUS_in = 16'h1101;
                R1EN = 0; R2EN = 1; ROEN = 0;
                OPERATION = 3'b001; ENTRIOUT = 0; // read into reg 2
             #20;
                BUS_in = 16'h1101;
                R1EN = 0; R2EN = 0; ROEN = 1;
                OPERATION = 3'b001; ENTRIOUT = 0; // SUB
             #20;
                BUS_in = 16'h0000;
                R1EN = 0; R2EN = 0; ROEN = 0;
                OPERATION = 3'b001; ENTRIOUT = 1; // Output
             #20;
                OPERATION = 3'b010; ENTRIOUT = 0; ROEN = 1; // NOT
             #20;
                ENTRIOUT = 1; ROEN = 0;                  // Output
             #20;
                OPERATION = 3'b011; ENTRIOUT = 0; ROEN = 1; // AND
             #20;
                ENTRIOUT = 1; ROEN = 0;                  // Output
             #20;
                OPERATION = 3'b100; ENTRIOUT = 0; ROEN = 1; // OR
             #20;
                ENTRIOUT = 1; ROEN = 0;                  // Output
             #20;
                OPERATION = 3'b101; ENTRIOUT = 0; ROEN = 1; // XOR
             #20;
                ENTRIOUT = 1; ROEN = 0;                  // Output
             #20;
                OPERATION = 3'b110; ENTRIOUT = 0; ROEN = 1; // XNOR
             #20;
                ENTRIOUT = 1; ROEN = 0;                  // Output
             #20;
                ENTRIOUT = 0; RESET = 1;                 // Reset
             #20;
                RESET = 0;
           end

endmodule
```

Report

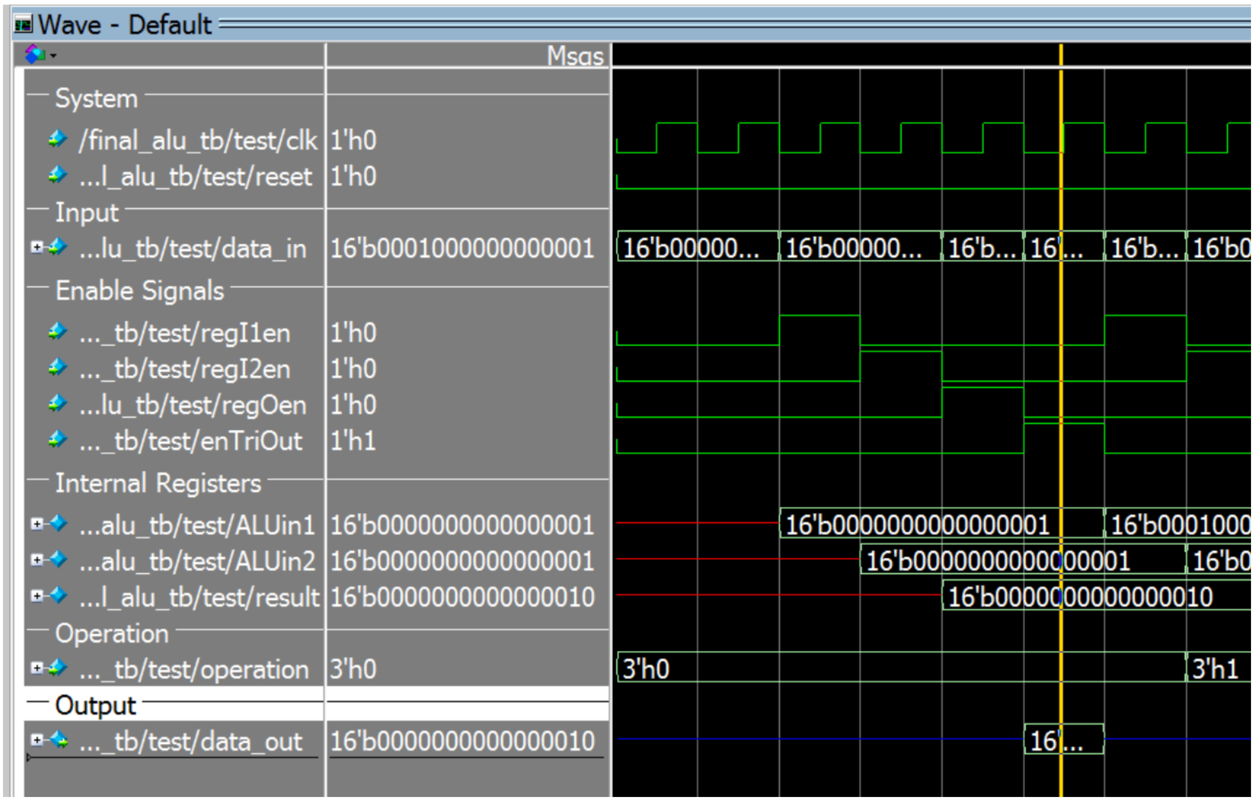
I designed my ALU with 8 inputs and 1 output. The ALU takes in the first operand from the bus, then takes in the second operand from the same bus input, and then calculates the specified operation given by the instruction/testbench and outputs the result to a tri-state buffer. The inputs are as follows: clock, reset, bus_in[15:0], in1En, in2En, outEn, operation[3:0], enTriOut. The only output is bus_out[15:0].

My code is condensed into one always block that is sensitive to the positive edge of the clock and reset. Within this process, I check if reset = 1 and set all three internal registers to 16h'0000. On the else condition, I have another 3 if-else-if statements that check for each of the three internal registers' enable signals. Whenever the output register is enabled, the module checks which operation to output using a case statement on the operation signal. Following the case statement the final line of code of the ALU assigns the output, which is a tri-state buffer, to 16'hzzzz or "disconnected" at all times except for when the tri-enable signal is active the buffer outputs the contents of the "result" register.

My testbench is designed to show all 7 possible operations. The testbench follows a standard input pattern, as well as shows the ALU handles improper inputs. Whenever I want to perform an operation, I simulate bringing data in for one clock cycle, bring in the other piece of data for another, operate on the third, and set the output on the fourth clock cycle. I change the bus_in a few times during an operation and the ALU correctly ignores the input. Attached to the end are screenshots of the simulation waveform, as well as a screenshot to show the design synthesized correctly and works with gate delay.

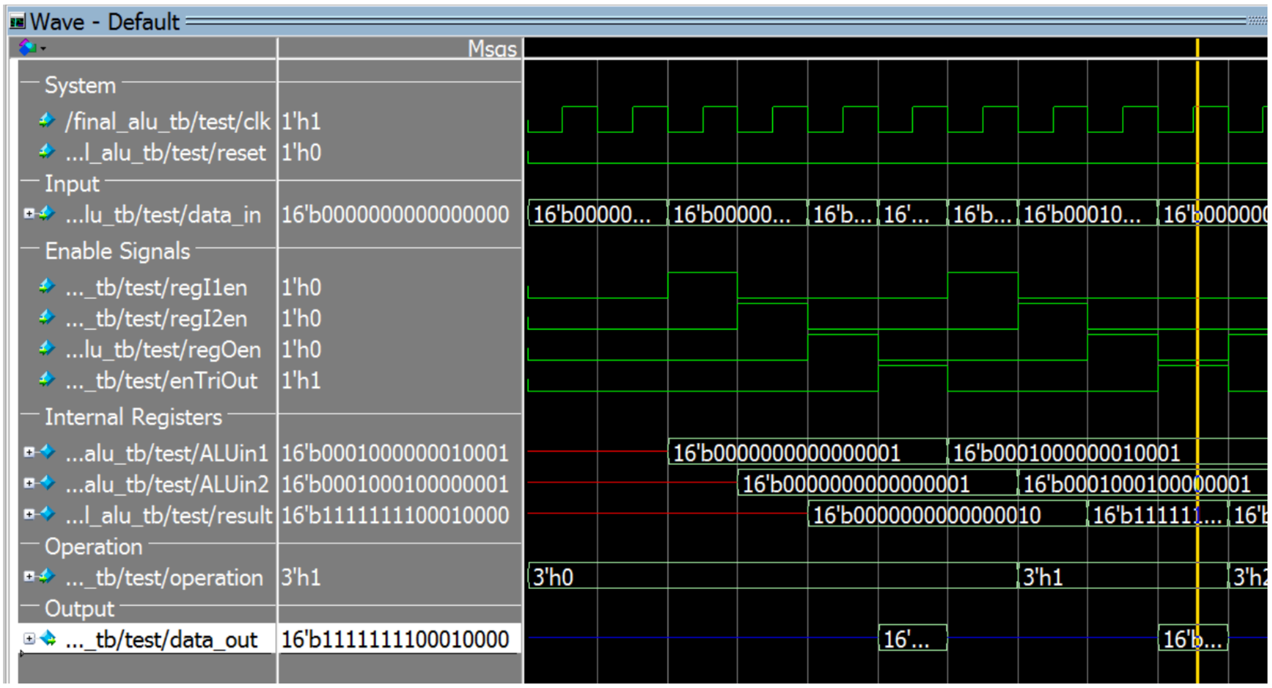
Waveforms

Add



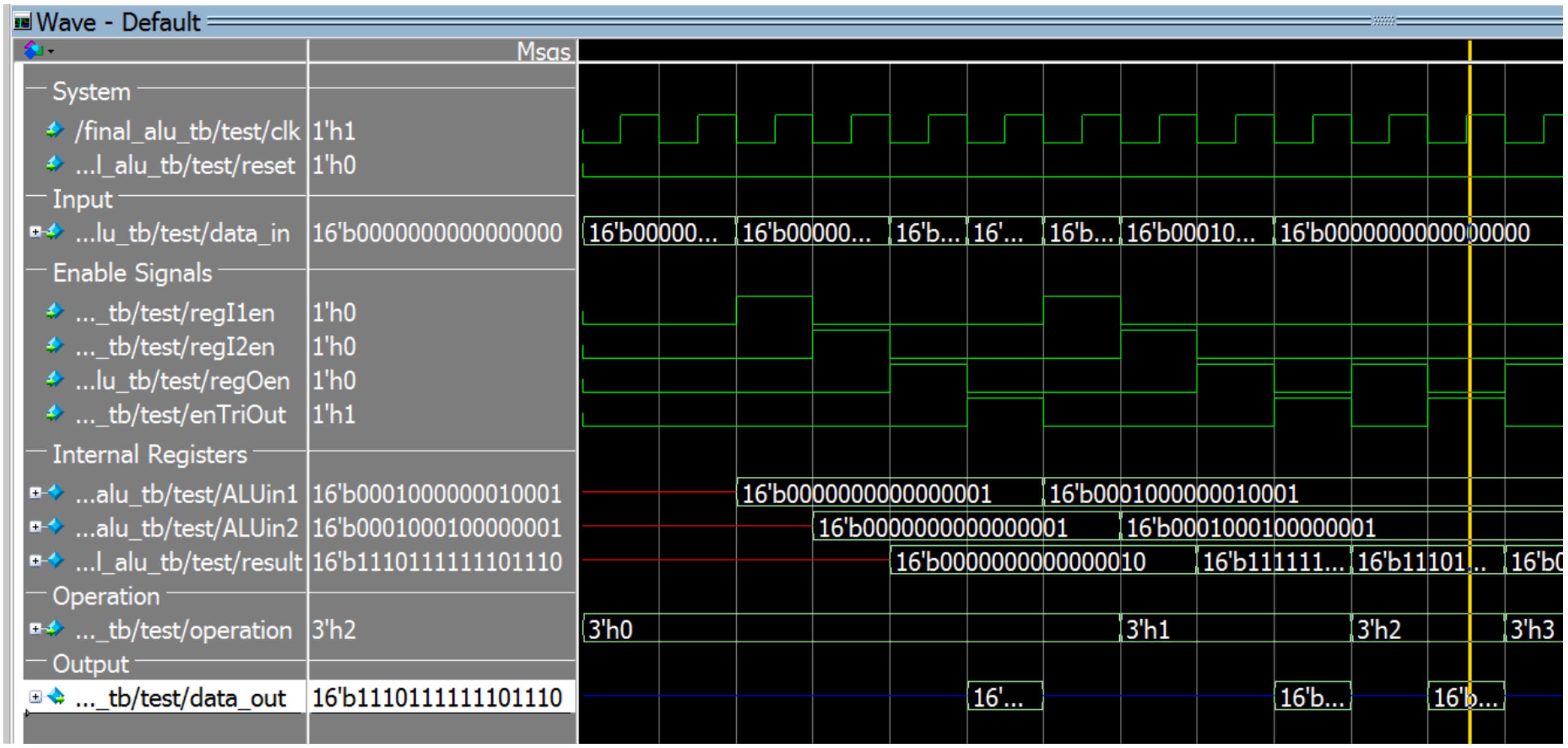
```
Wave - Default
                               Msgs
System
  /final_alu_tb/test/clk       1'h0
  ...l_alu_tb/test/reset       1'h0
Input
  ...lu_tb/test/data_in        16'b0001000000000001
Enable Signals
  ..._tb/test/regI1en          1'h0
  ..._tb/test/regI2en          1'h0
  ...lu_tb/test/regOen         1'h0
  ..._tb/test/enTriOut         1'h1
Internal Registers
  ...alu_tb/test/ALUin1        16'b0000000000000001
  ...alu_tb/test/ALUin2        16'b0000000000000001
  ...l_alu_tb/test/result      16'b0000000000000010
Operation
  ..._tb/test/operation        3'h0
Output
  ..._tb/test/data_out         16'b0000000000000010
```

Sub



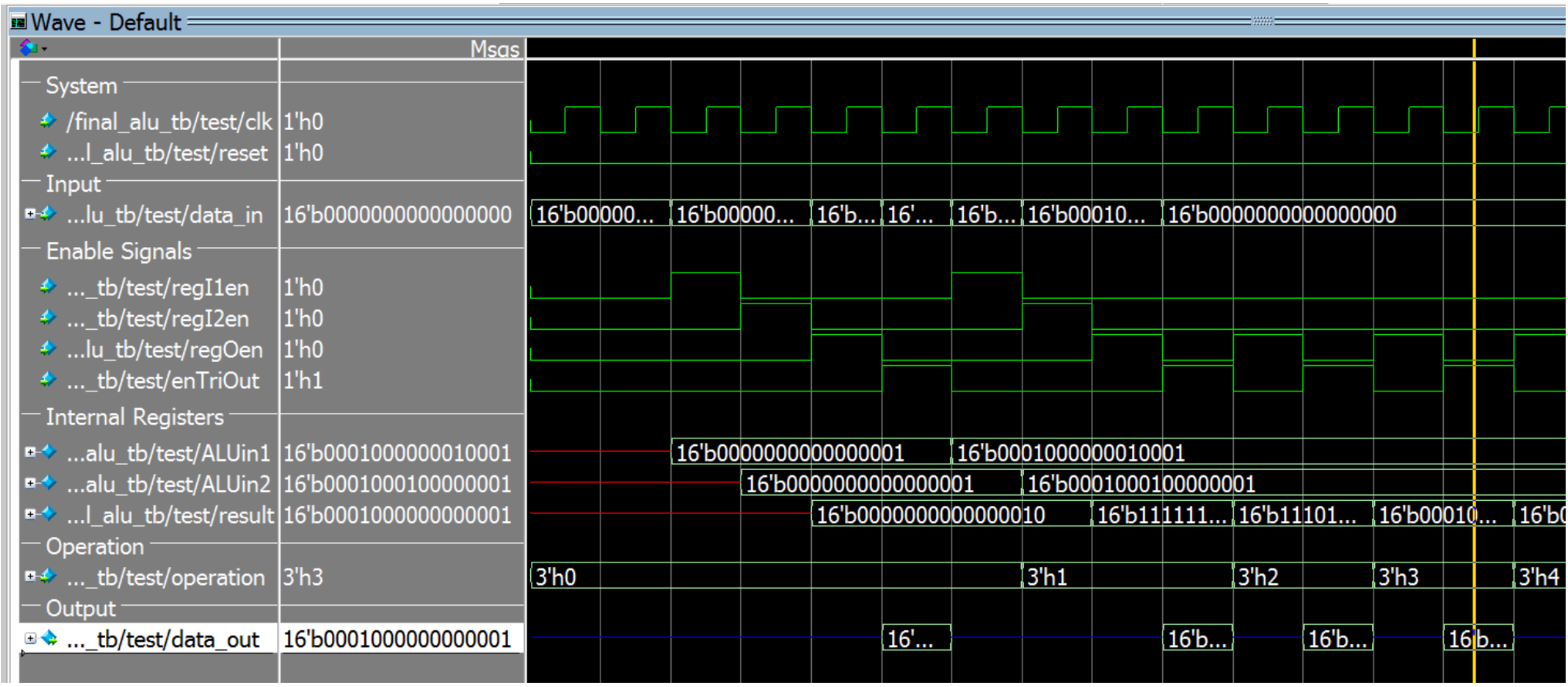
```
Wave - Default
                               Msgs
System
  /final_alu_tb/test/clk       1'h1
  ...l_alu_tb/test/reset       1'h0
Input
  ...lu_tb/test/data_in        16'b0000000000000000
Enable Signals
  ..._tb/test/regI1en          1'h0
  ..._tb/test/regI2en          1'h0
  ...lu_tb/test/regOen         1'h0
  ..._tb/test/enTriOut         1'h1
Internal Registers
  ...alu_tb/test/ALUin1        16'b0001000000010001
  ...alu_tb/test/ALUin2        16'b0001000100000001
  ...l_alu_tb/test/result      16'b1111111100010000
Operation
  ..._tb/test/operation        3'h1
Output
  ..._tb/test/data_out         16'b1111111100010000
```

Not



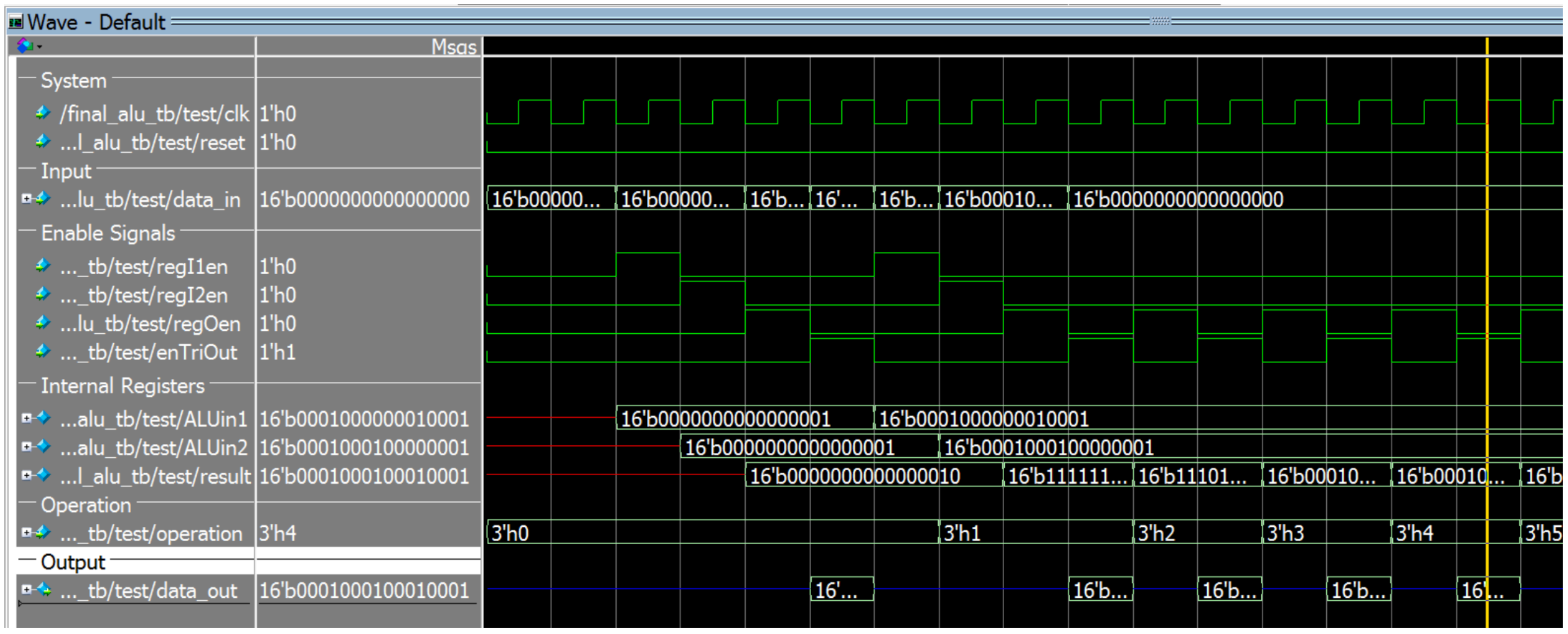
```
Wave - Default
                              Msgs
System
  /final_alu_tb/test/clk      1'h1
  ...l_alu_tb/test/reset      1'h0
Input
  ...lu_tb/test/data_in       16'b0000000000000000
Enable Signals
  ..._tb/test/regI1en         1'h0
  ..._tb/test/regI2en         1'h0
  ...lu_tb/test/regOen        1'h0
  ..._tb/test/enTriOut        1'h1
Internal Registers
  ...alu_tb/test/ALUin1       16'b0001000000010001
  ...alu_tb/test/ALUin2       16'b0001000100000001
  ...l_alu_tb/test/result     16'b1110111111101110
Operation
  ..._tb/test/operation       3'h2
Output
  ..._tb/test/data_out        16'b1110111111101110
```

And



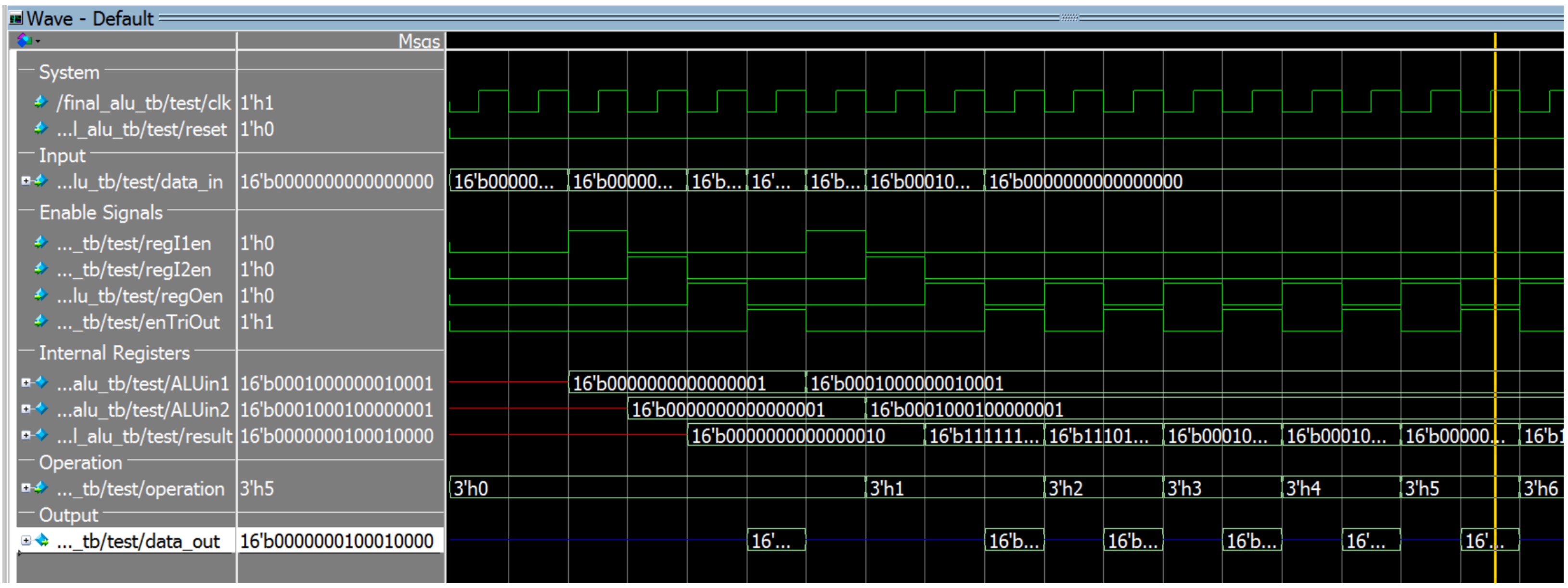
```
Wave - Default
                              Msgs
System
  /final_alu_tb/test/clk      1'h0
  ...l_alu_tb/test/reset      1'h0
Input
  ...lu_tb/test/data_in       16'b0000000000000000
Enable Signals
  ..._tb/test/regI1en         1'h0
  ..._tb/test/regI2en         1'h0
  ...lu_tb/test/regOen        1'h0
  ..._tb/test/enTriOut        1'h1
Internal Registers
  ...alu_tb/test/ALUin1       16'b0001000000010001
  ...alu_tb/test/ALUin2       16'b0001000100000001
  ...l_alu_tb/test/result     16'b0001000000000001
Operation
  ..._tb/test/operation       3'h3
Output
  ..._tb/test/data_out        16'b0001000000000001
```

Or



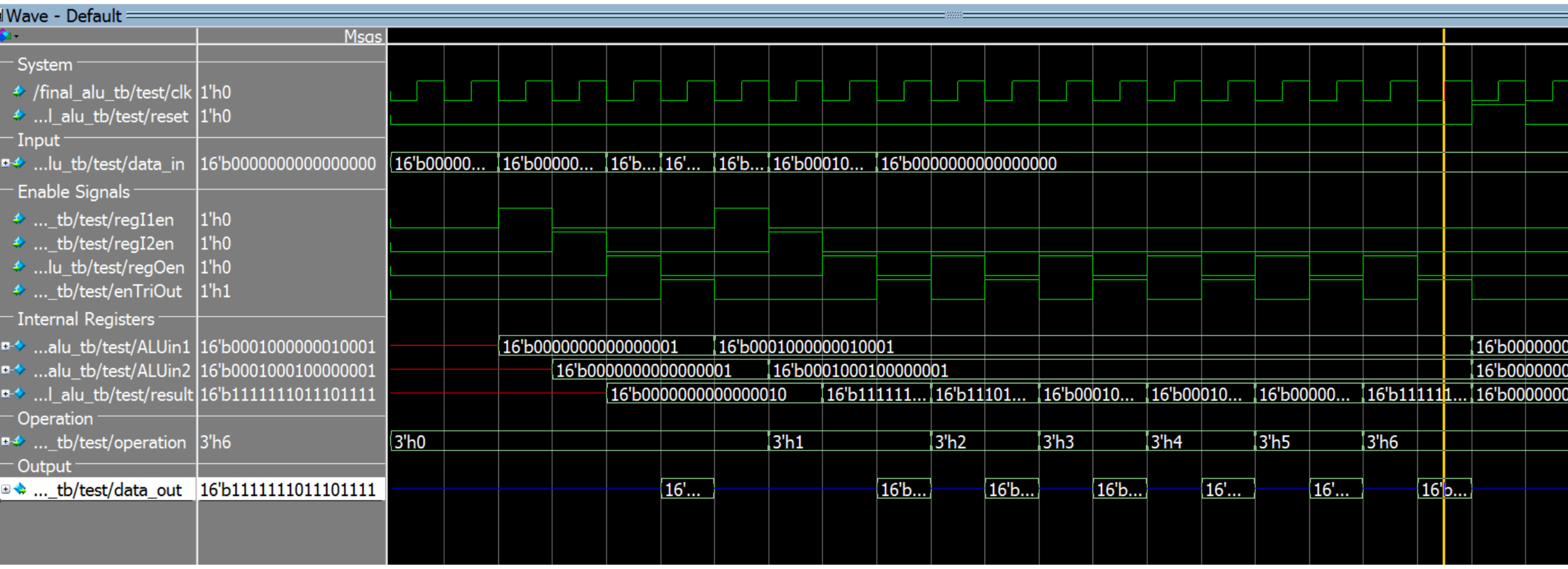
```
Wave - Default
                                  Msgs
System
/final_alu_tb/test/clk            1'h0
...l_alu_tb/test/reset            1'h0
Input
...lu_tb/test/data_in             16'b0000000000000000
Enable Signals
..._tb/test/regI1en               1'h0
..._tb/test/regI2en               1'h0
...lu_tb/test/regOen              1'h0
..._tb/test/enTriOut              1'h1
Internal Registers
...alu_tb/test/ALUin1             16'b0001000000010001
...alu_tb/test/ALUin2             16'b0001000100000001
...l_alu_tb/test/result           16'b0001000100010001
Operation
..._tb/test/operation             3'h4
Output
..._tb/test/data_out              16'b0001000100010001
```

Xor



```
Wave - Default
                                  Msgs
System
/final_alu_tb/test/clk            1'h1
...l_alu_tb/test/reset            1'h0
Input
...lu_tb/test/data_in             16'b0000000000000000
Enable Signals
..._tb/test/regI1en               1'h0
..._tb/test/regI2en               1'h0
...lu_tb/test/regOen              1'h0
..._tb/test/enTriOut              1'h1
Internal Registers
...alu_tb/test/ALUin1             16'b0001000000010001
...alu_tb/test/ALUin2             16'b0001000100000001
...l_alu_tb/test/result           16'b0000000100010000
Operation
..._tb/test/operation             3'h5
Output
..._tb/test/data_out              16'b0000000100010000
```

Xnor/Reset



```
Wave - Default
                          Msgs
System
/final_alu_tb/test/clk    1'h0
...l_alu_tb/test/reset    1'h0
Input
...lu_tb/test/data_in     16'b0000000000000000
Enable Signals
..._tb/test/regI1en       1'h0
..._tb/test/regI2en       1'h0
...lu_tb/test/regOen      1'h0
..._tb/test/enTriOut      1'h1
Internal Registers
...alu_tb/test/ALUin1     16'b0001000000010001
...alu_tb/test/ALUin2     16'b0001000100000001
...l_alu_tb/test/result   16'b1111111011101111
Operation
..._tb/test/operation     3'h6
Output
..._tb/test/data_out      16'b1111111011101111
```

Gate Delay (only noticeable when tri-state buffer outputs because the rest are inputs)