به نام خدا

University of Tehran

Faculty of electrical and computer engineering

Digital Systems 1

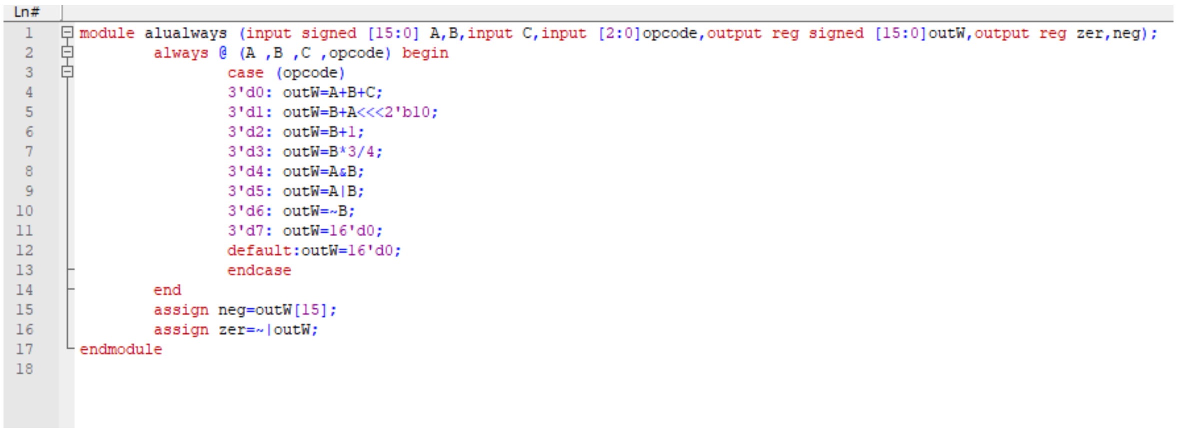
Computer Assignment 3

Amirhossein Yavarikhoo	نام و نام خانوادگی
810199514	شماره دانشجویی
1401/2/17	تاریخ ارسال گزارش

Q1

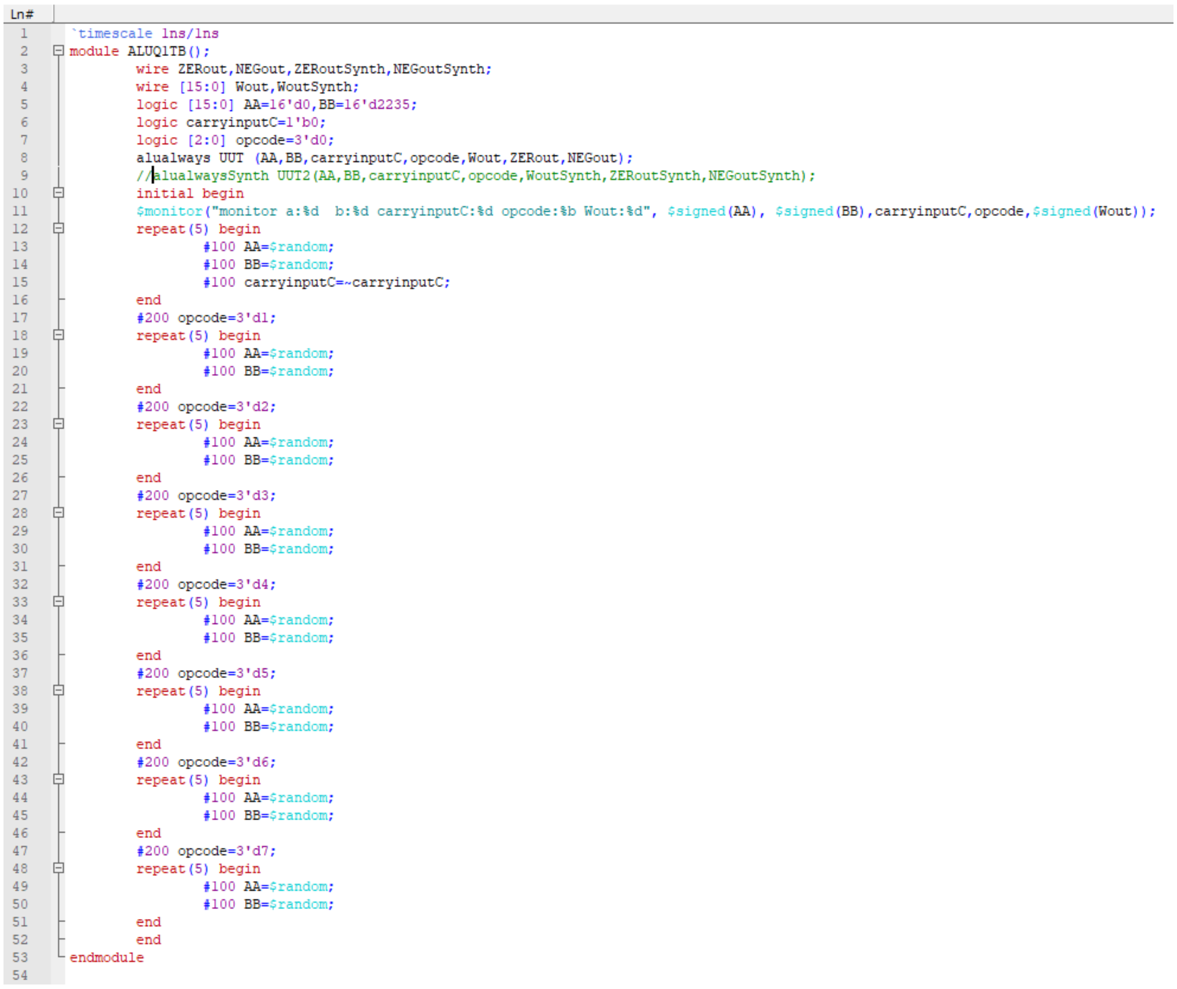
a)

The ALU with the operations given is shown below.



```
module alualways (input signed [15:0] A,B,input C,input [2:0]opcode,output reg signed [15:0]outW,output reg zer,neg);
        always @ (A ,B ,C ,opcode) begin
                case (opcode)
                3'd0: outW=A+B+C;
                3'd1: outW=B+A<<<2'b10;
                3'd2: outW=B+1;
                3'd3: outW=B*3/4;
                3'd4: outW=A&B;
                3'd5: outW=A|B;
                3'd6: outW=~B;
                3'd7: outW=16'd0;
                default:outW=16'd0;
                endcase
        end
        assign neg=outW[15];
        assign zer=~|outW;
endmodule
```

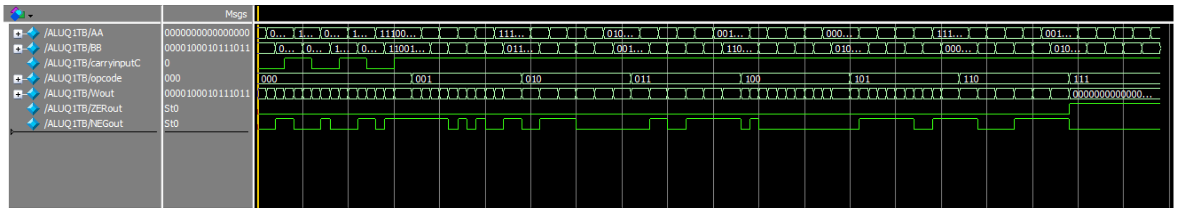
The picture below shows the testbench code.



```
`timescale 1ns/1ns
module ALUQ1TB();
        wire ZERout,NEGout,ZERoutSynth,NEGoutSynth;
        wire [15:0] Wout,WoutSynth;
        logic [15:0] AA=16'd0,BB=16'd2235;
        logic carryinputC=1'b0;
        logic [2:0] opcode=3'd0;
        alualways UUT (AA,BB,carryinputC,opcode,Wout,ZERout,NEGout);
        //alualwaysSynth UUT2(AA,BB,carryinputC,opcode,WoutSynth,ZERoutSynth,NEGoutSynth);
        initial begin
        $monitor("monitor a:%d  b:%d carryinputC:%d opcode:%b Wout:%d", $signed(AA), $signed(BB),carryinputC,opcode,$signed(Wout));
        repeat(5) begin
                #100 AA=$random;
                #100 BB=$random;
                #100 carryinputC=~carryinputC;
        end
        #200 opcode=3'd1;
        repeat(5) begin
                #100 AA=$random;
                #100 BB=$random;
        end
        #200 opcode=3'd2;
        repeat(5) begin
                #100 AA=$random;
                #100 BB=$random;
        end
        #200 opcode=3'd3;
        repeat(5) begin
                #100 AA=$random;
                #100 BB=$random;
        end
        #200 opcode=3'd4;
        repeat(5) begin
                #100 AA=$random;
                #100 BB=$random;
        end
        #200 opcode=3'd5;
        repeat(5) begin
                #100 AA=$random;
                #100 BB=$random;
        end
        #200 opcode=3'd6;
        repeat(5) begin
                #100 AA=$random;
                #100 BB=$random;
        end
        #200 opcode=3'd7;
        repeat(5) begin
                #100 AA=$random;
                #100 BB=$random;
        end
        end
endmodule
```

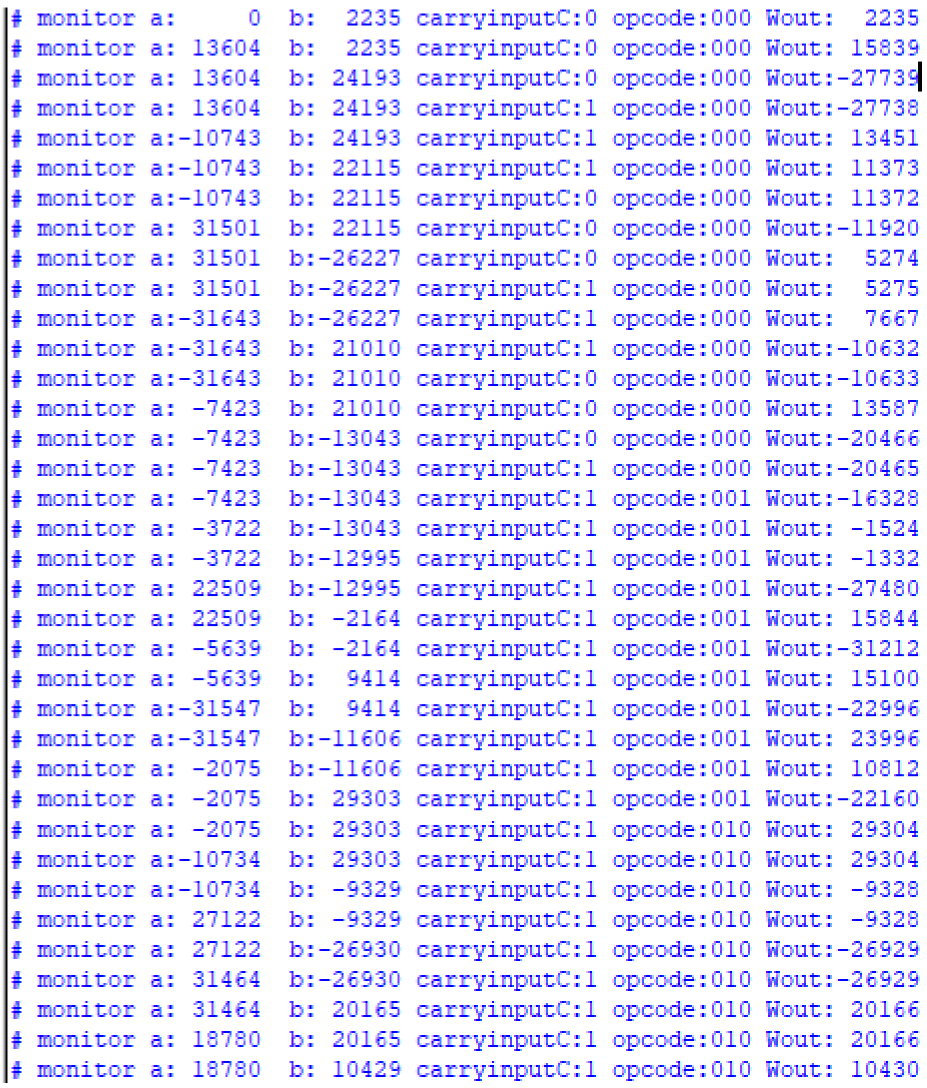
For each operation code we used a repeat statement that randomizes our inputs 5 times.

Test result is shown below:



To verify the module easier, we use monitor command to convert binary numbers to decimal.

Part of transcript of the simulation above is shown below:



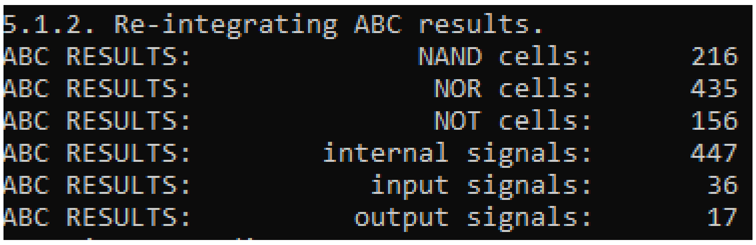
```
# monitor a:     0  b:  2235 carryinputC:0 opcode:000 Wout:  2235
# monitor a: 13604  b:  2235 carryinputC:0 opcode:000 Wout: 15839
# monitor a: 13604  b: 24193 carryinputC:0 opcode:000 Wout:-27739
# monitor a: 13604  b: 24193 carryinputC:1 opcode:000 Wout:-27738
# monitor a:-10743  b: 24193 carryinputC:1 opcode:000 Wout: 13451
# monitor a:-10743  b: 22115 carryinputC:1 opcode:000 Wout: 11373
# monitor a:-10743  b: 22115 carryinputC:0 opcode:000 Wout: 11372
# monitor a: 31501  b: 22115 carryinputC:0 opcode:000 Wout:-11920
# monitor a: 31501  b:-26227 carryinputC:0 opcode:000 Wout:  5274
# monitor a: 31501  b:-26227 carryinputC:1 opcode:000 Wout:  5275
# monitor a:-31643  b:-26227 carryinputC:1 opcode:000 Wout:  7667
# monitor a:-31643  b: 21010 carryinputC:1 opcode:000 Wout:-10632
# monitor a:-31643  b: 21010 carryinputC:0 opcode:000 Wout:-10633
# monitor a: -7423  b: 21010 carryinputC:0 opcode:000 Wout: 13587
# monitor a: -7423  b:-13043 carryinputC:0 opcode:000 Wout:-20466
# monitor a: -7423  b:-13043 carryinputC:1 opcode:000 Wout:-20465
# monitor a: -7423  b:-13043 carryinputC:1 opcode:001 Wout:-16328
# monitor a: -3722  b:-13043 carryinputC:1 opcode:001 Wout: -1524
# monitor a: -3722  b:-12995 carryinputC:1 opcode:001 Wout: -1332
# monitor a: 22509  b:-12995 carryinputC:1 opcode:001 Wout:-27480
# monitor a: 22509  b: -2164 carryinputC:1 opcode:001 Wout: 15844
# monitor a: -5639  b: -2164 carryinputC:1 opcode:001 Wout:-31212
# monitor a: -5639  b:  9414 carryinputC:1 opcode:001 Wout: 15100
# monitor a:-31547  b:  9414 carryinputC:1 opcode:001 Wout:-22996
# monitor a:-31547  b:-11606 carryinputC:1 opcode:001 Wout: 23996
# monitor a: -2075  b:-11606 carryinputC:1 opcode:001 Wout: 10812
# monitor a: -2075  b: 29303 carryinputC:1 opcode:001 Wout:-22160
# monitor a: -2075  b: 29303 carryinputC:1 opcode:010 Wout: 29304
# monitor a:-10734  b: 29303 carryinputC:1 opcode:010 Wout: 29304
# monitor a:-10734  b: -9329 carryinputC:1 opcode:010 Wout: -9328
# monitor a: 27122  b: -9329 carryinputC:1 opcode:010 Wout: -9328
# monitor a: 27122  b:-26930 carryinputC:1 opcode:010 Wout:-26929
# monitor a: 31464  b:-26930 carryinputC:1 opcode:010 Wout:-26929
# monitor a: 31464  b: 20165 carryinputC:1 opcode:010 Wout: 20166
# monitor a: 18780  b: 20165 carryinputC:1 opcode:010 Wout: 20166
# monitor a: 18780  b: 10429 carryinputC:1 opcode:010 Wout: 10430
```

Note that sometimes overflow happens and the reason some outputs aren’t as we expected is because of that.

We can easily verify that the intended commands work properly.

b)

First we synthesize the ALU from question 1 then we optimize it and run the synthesis again.

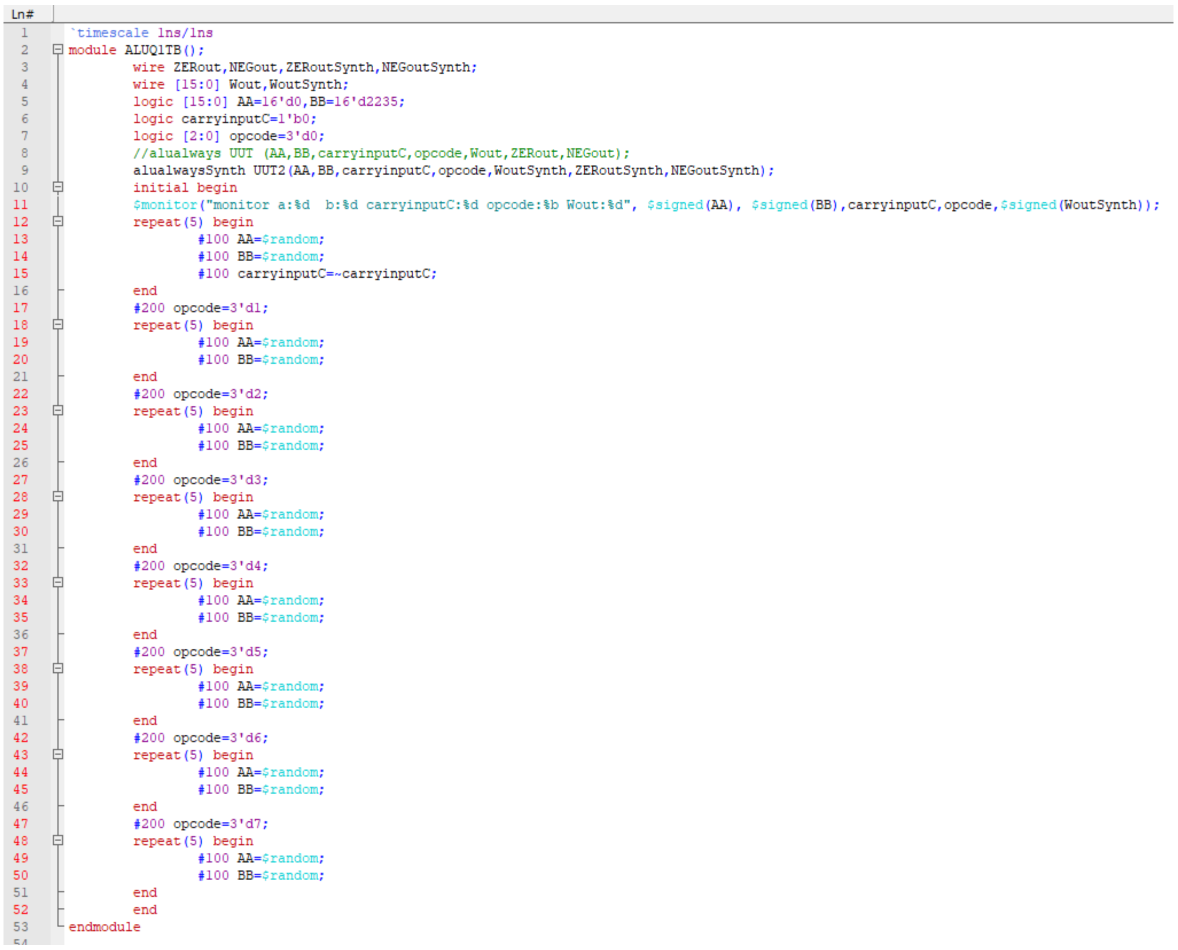


```
5.1.2. Re-integrating ABC results.
ABC RESULTS:             NAND cells:       216
ABC RESULTS:              NOR cells:       435
ABC RESULTS:              NOT cells:       156
ABC RESULTS:       internal signals:       447
ABC RESULTS:          input signals:        36
ABC RESULTS:         output signals:        17
```

C)

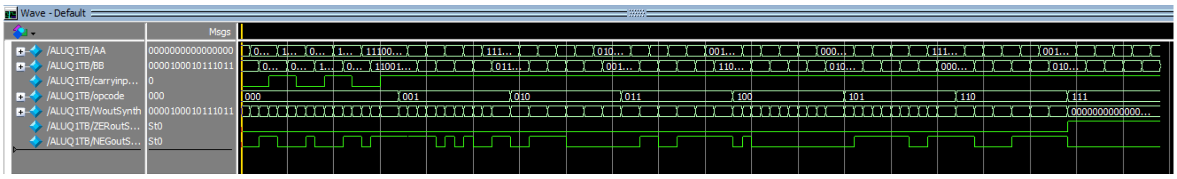
The code will be added in zip file.

Here is the testbench code for simulating the result.

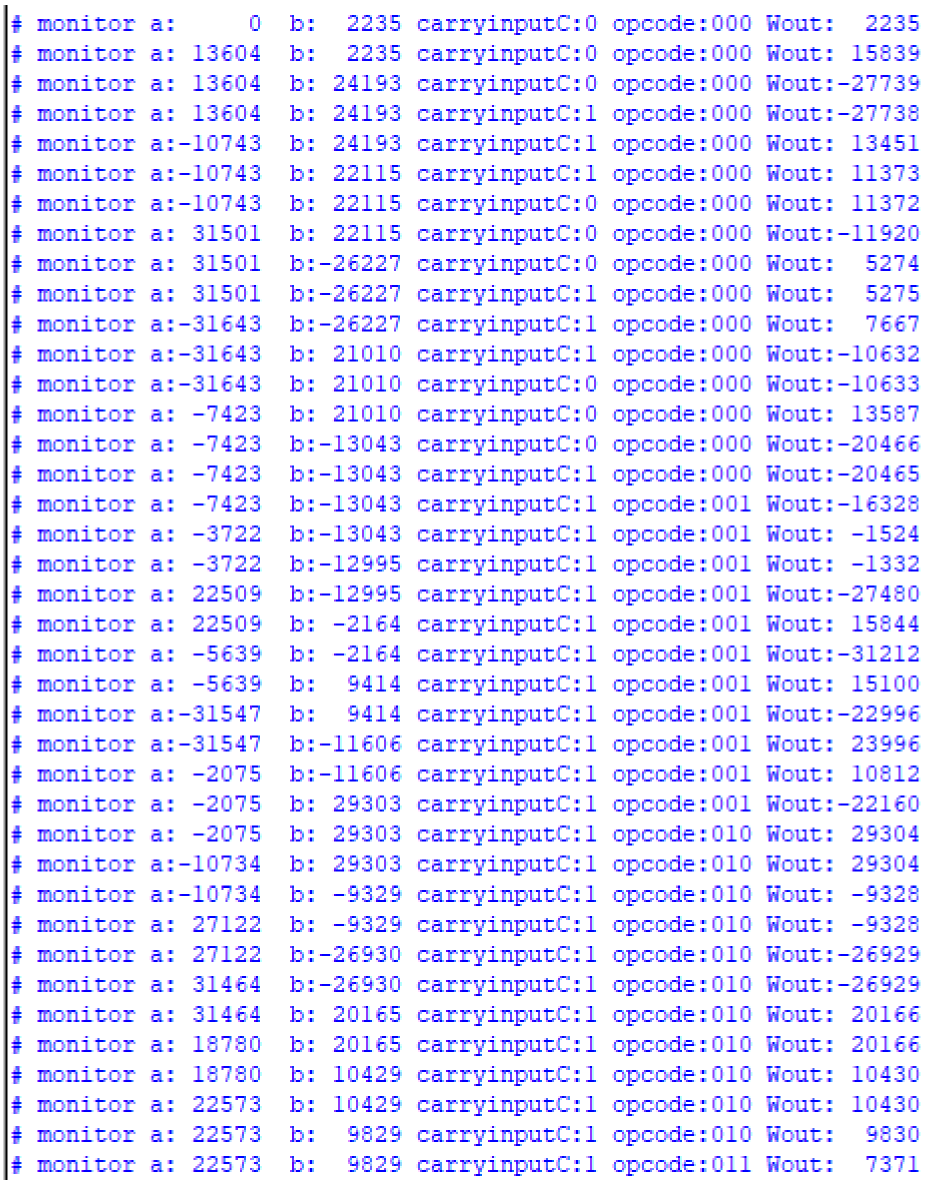


```
`timescale 1ns/1ns
module ALUQ1TB();
        wire ZERout,NEGout,ZERoutSynth,NEGoutSynth;
        wire [15:0] Wout,WoutSynth;
        logic [15:0] AA=16'd0,BB=16'd2235;
        logic carryinputC=1'b0;
        logic [2:0] opcode=3'd0;
        //alualways UUT (AA,BB,carryinputC,opcode,Wout,ZERout,NEGout);
        alualwaysSynth UUT2(AA,BB,carryinputC,opcode,WoutSynth,ZERoutSynth,NEGoutSynth);
        initial begin
        $monitor("monitor a:%d  b:%d carryinputC:%d opcode:%b Wout:%d", $signed(AA), $signed(BB),carryinputC,opcode,$signed(WoutSynth));
        repeat(5) begin
                #100 AA=$random;
                #100 BB=$random;
                #100 carryinputC=~carryinputC;
        end
        #200 opcode=3'd1;
        repeat(5) begin
                #100 AA=$random;
                #100 BB=$random;
        end
        #200 opcode=3'd2;
        repeat(5) begin
                #100 AA=$random;
                #100 BB=$random;
        end
        #200 opcode=3'd3;
        repeat(5) begin
                #100 AA=$random;
                #100 BB=$random;
        end
        #200 opcode=3'd4;
        repeat(5) begin
                #100 AA=$random;
                #100 BB=$random;
        end
        #200 opcode=3'd5;
        repeat(5) begin
                #100 AA=$random;
                #100 BB=$random;
        end
        #200 opcode=3'd6;
        repeat(5) begin
                #100 AA=$random;
                #100 BB=$random;
        end
        #200 opcode=3'd7;
        repeat(5) begin
                #100 AA=$random;
                #100 BB=$random;
        end
        end
endmodule
```

Waveform result is shown below:



```
/ALUQ1TB/AA          0000000000000000
/ALUQ1TB/BB          0000100010111011
/ALUQ1TB/carryinp... 0
/ALUQ1TB/opcode      000
/ALUQ1TB/WoutSynth   0000100010111011
/ALUQ1TB/ZERoutS...  St0
/ALUQ1TB/NEGoutS...  St0
```

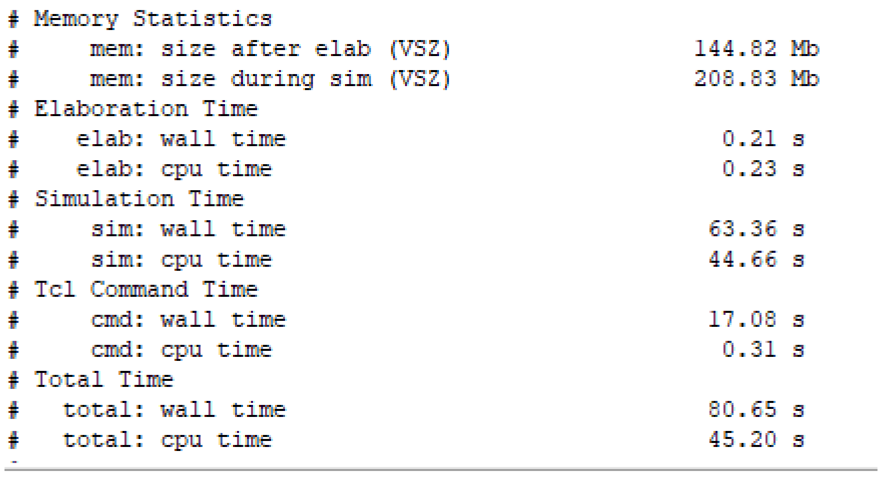


```
# monitor a:      0  b:   2235 carryinputC:0 opcode:000 Wout:  2235
# monitor a: 13604  b:   2235 carryinputC:0 opcode:000 Wout: 15839
# monitor a: 13604  b: 24193 carryinputC:0 opcode:000 Wout:-27739
# monitor a: 13604  b: 24193 carryinputC:1 opcode:000 Wout:-27738
# monitor a:-10743  b: 24193 carryinputC:1 opcode:000 Wout: 13451
# monitor a:-10743  b: 22115 carryinputC:1 opcode:000 Wout: 11373
# monitor a:-10743  b: 22115 carryinputC:0 opcode:000 Wout: 11372
# monitor a: 31501  b: 22115 carryinputC:0 opcode:000 Wout:-11920
# monitor a: 31501  b:-26227 carryinputC:0 opcode:000 Wout:  5274
# monitor a: 31501  b:-26227 carryinputC:1 opcode:000 Wout:  5275
# monitor a:-31643  b:-26227 carryinputC:1 opcode:000 Wout:  7667
# monitor a:-31643  b: 21010 carryinputC:1 opcode:000 Wout:-10632
# monitor a:-31643  b: 21010 carryinputC:0 opcode:000 Wout:-10633
# monitor a: -7423  b: 21010 carryinputC:0 opcode:000 Wout: 13587
# monitor a: -7423  b:-13043 carryinputC:0 opcode:000 Wout:-20466
# monitor a: -7423  b:-13043 carryinputC:1 opcode:000 Wout:-20465
# monitor a: -7423  b:-13043 carryinputC:1 opcode:001 Wout:-16328
# monitor a: -3722  b:-13043 carryinputC:1 opcode:001 Wout: -1524
# monitor a: -3722  b:-12995 carryinputC:1 opcode:001 Wout: -1332
# monitor a: 22509  b:-12995 carryinputC:1 opcode:001 Wout:-27480
# monitor a: 22509  b: -2164 carryinputC:1 opcode:001 Wout: 15844
# monitor a: -5639  b: -2164 carryinputC:1 opcode:001 Wout:-31212
# monitor a: -5639  b:   9414 carryinputC:1 opcode:001 Wout: 15100
# monitor a:-31547  b:   9414 carryinputC:1 opcode:001 Wout:-22996
# monitor a:-31547  b:-11606 carryinputC:1 opcode:001 Wout: 23996
# monitor a: -2075  b:-11606 carryinputC:1 opcode:001 Wout: 10812
# monitor a: -2075  b: 29303 carryinputC:1 opcode:001 Wout:-22160
# monitor a: -2075  b: 29303 carryinputC:1 opcode:010 Wout: 29304
# monitor a:-10734  b: 29303 carryinputC:1 opcode:010 Wout: 29304
# monitor a:-10734  b: -9329 carryinputC:1 opcode:010 Wout: -9328
# monitor a: 27122  b: -9329 carryinputC:1 opcode:010 Wout: -9328
# monitor a: 27122  b:-26930 carryinputC:1 opcode:010 Wout:-26929
# monitor a: 31464  b:-26930 carryinputC:1 opcode:010 Wout:-26929
# monitor a: 31464  b: 20165 carryinputC:1 opcode:010 Wout: 20166
# monitor a: 18780  b: 20165 carryinputC:1 opcode:010 Wout: 20166
# monitor a: 18780  b: 10429 carryinputC:1 opcode:010 Wout: 10430
# monitor a: 22573  b: 10429 carryinputC:1 opcode:010 Wout: 10430
# monitor a: 22573  b:   9829 carryinputC:1 opcode:010 Wout:  9830
# monitor a: 22573  b:   9829 carryinputC:1 opcode:011 Wout:  7371
```

The transcript can verify that system works as expected. Overflow problem still occurs.

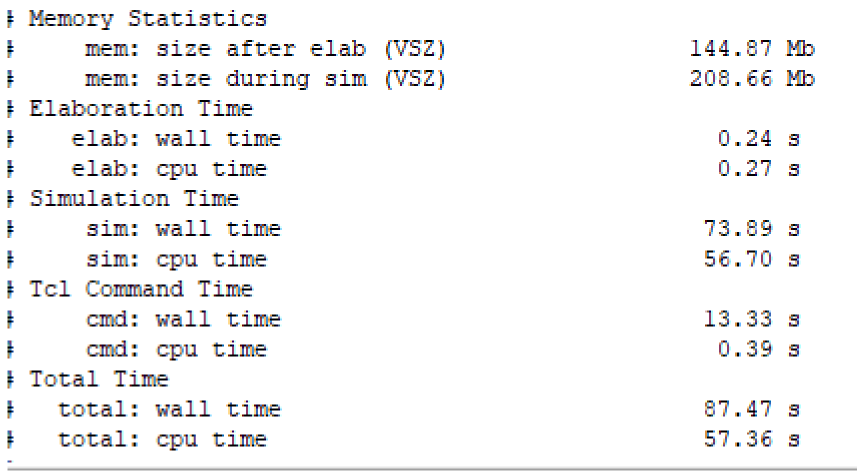
D) To compare simulation speed, we have to give inputs many more times than 5 so we replace each repeat with 50000.

This screenshot is for pre-synthesis ALU.



```
# Memory Statistics
#     mem: size after elab (VSZ)              144.82 Mb
#     mem: size during sim (VSZ)              208.83 Mb
# Elaboration Time
#    elab: wall time                            0.21 s
#    elab: cpu time                             0.23 s
# Simulation Time
#     sim: wall time                           63.36 s
#     sim: cpu time                            44.66 s
# Tcl Command Time
#     cmd: wall time                           17.08 s
#     cmd: cpu time                             0.31 s
# Total Time
#   total: wall time                           80.65 s
#   total: cpu time                            45.20 s
```

Screenshot below is for post synthesis simulation.

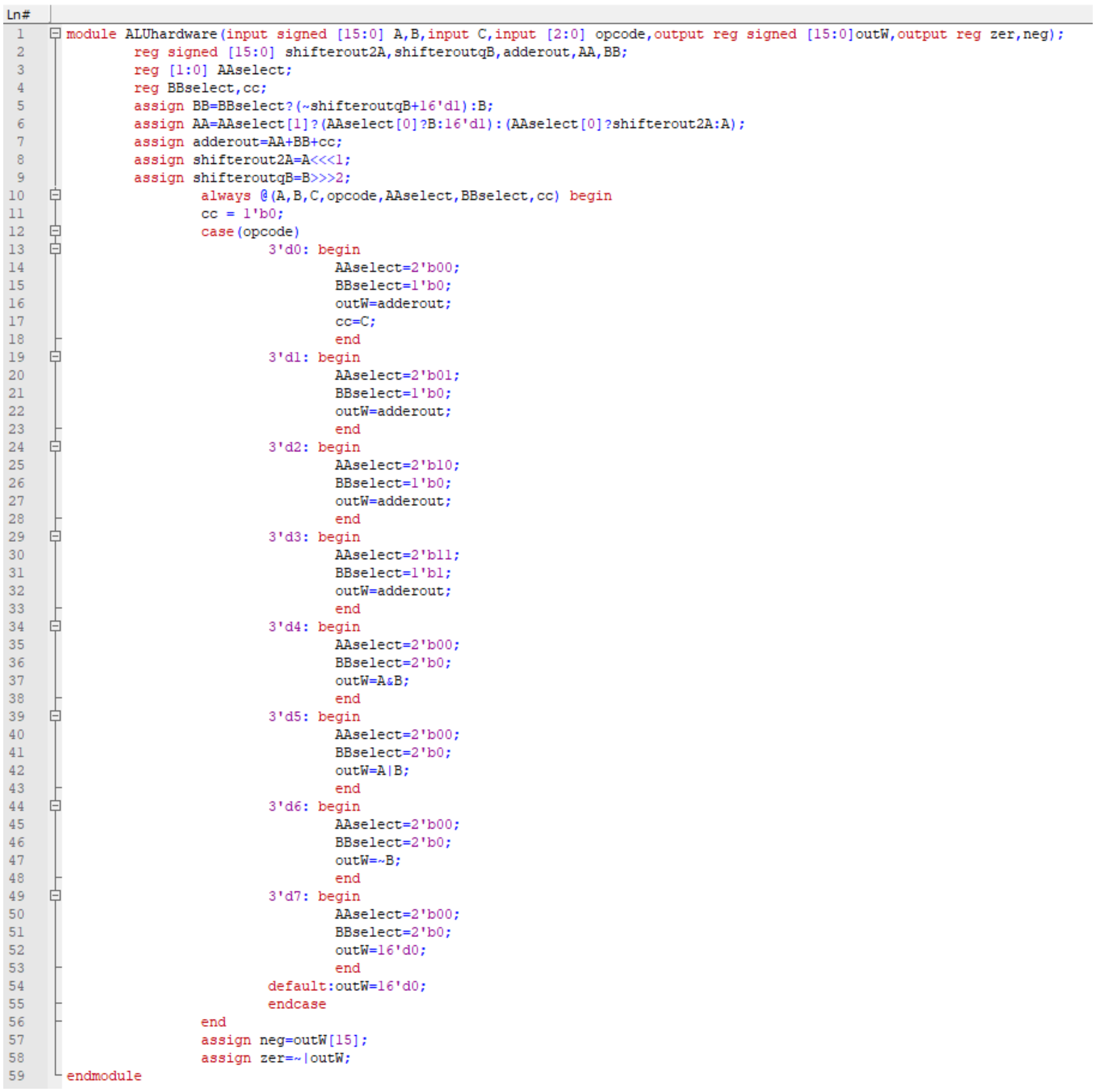


```
# Memory Statistics
#     mem: size after elab (VSZ)              144.87 Mb
#     mem: size during sim (VSZ)              208.66 Mb
# Elaboration Time
#    elab: wall time                            0.24 s
#    elab: cpu time                             0.27 s
# Simulation Time
#     sim: wall time                           73.89 s
#     sim: cpu time                            56.70 s
# Tcl Command Time
#     cmd: wall time                           13.33 s
#     cmd: cpu time                             0.39 s
# Total Time
#   total: wall time                           87.47 s
#   total: cpu time                            57.36 s
```

As we can see, even in small simulations like 1000 post-synthesis simulation takes almost twice the time as the pre-synthesis simulation. The reason is clear. Because in post synthesis we have multiple gates and the simulation is gate level.

Q2

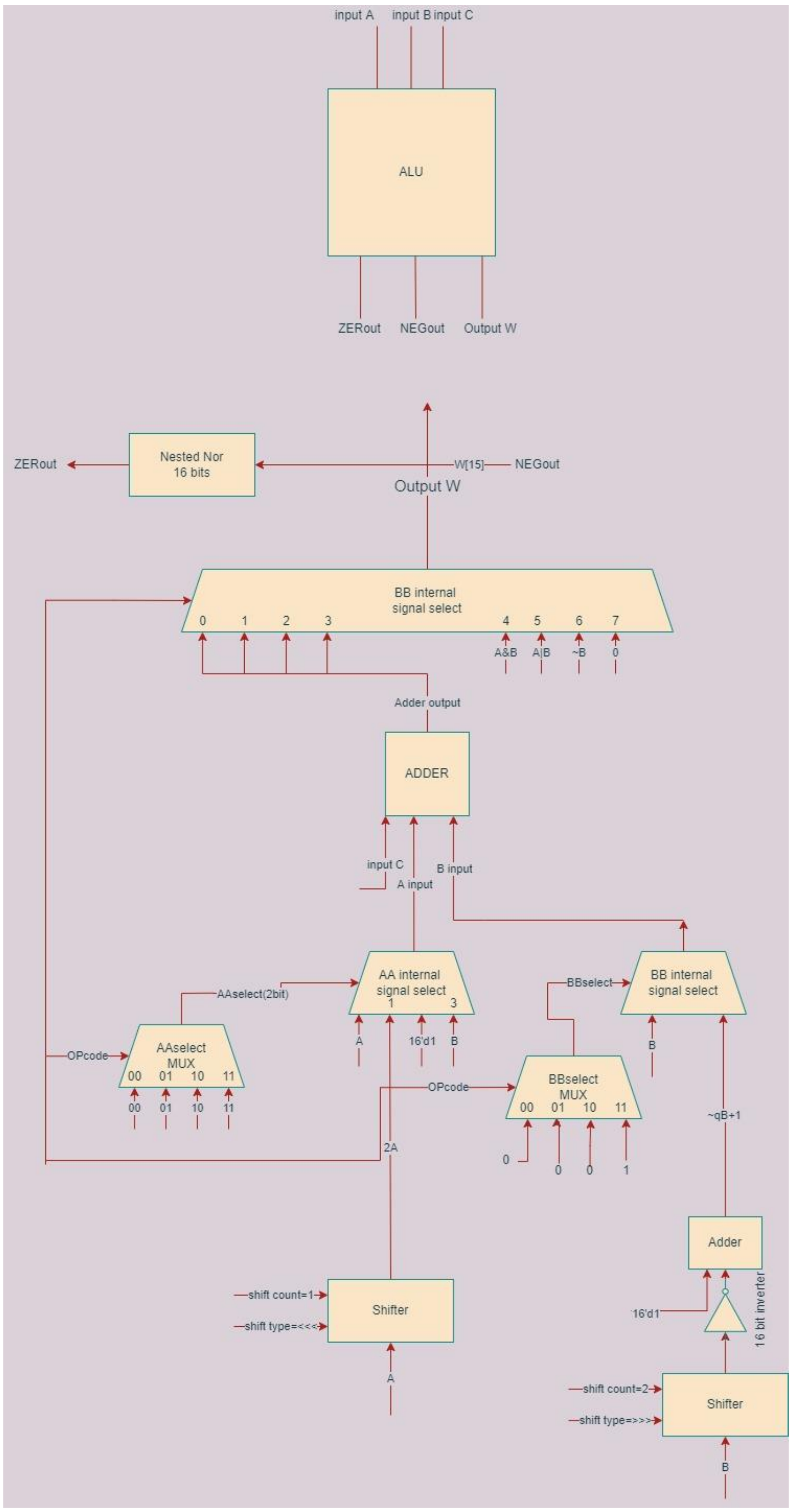
A) For an optimized design, 2 shifters, 1 adder/subtractor,2 multiplexers and nested AND,OR and NOT gates are used.



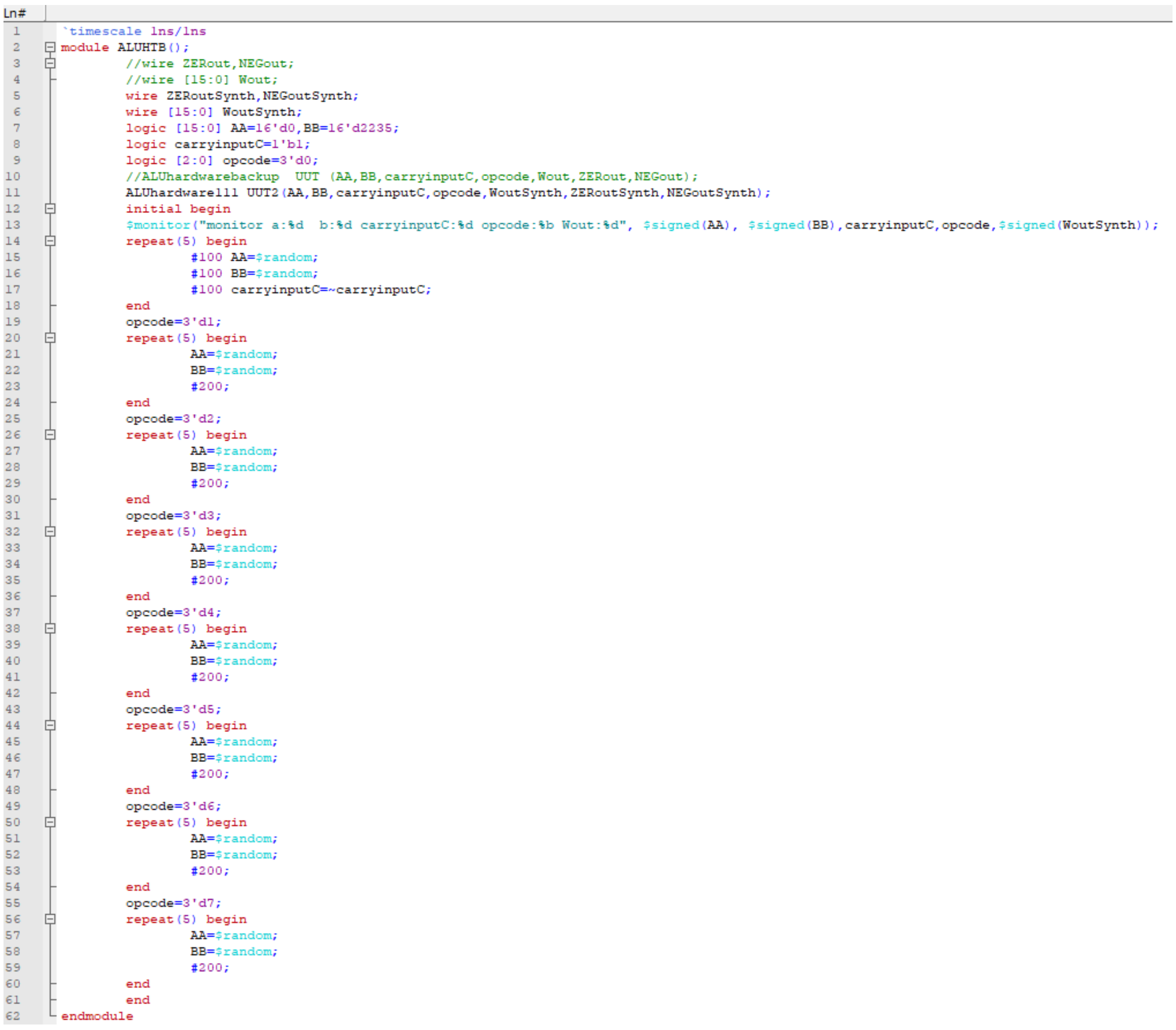
```verilog
module ALUhardware(input signed [15:0] A,B,input C,input [2:0] opcode,output reg signed [15:0]outW,output reg zer,neg);
        reg signed [15:0] shifterout2A,shifteroutqB,adderout,AA,BB;
        reg [1:0] AAselect;
        reg BBselect,cc;
        assign BB=BBselect?(~shifteroutqB+16'd1):B;
        assign AA=AAselect[1]?(AAselect[0]?B:16'd1):(AAselect[0]?shifterout2A:A);
        assign adderout=AA+BB+cc;
        assign shifterout2A=A<<<1;
        assign shifteroutqB=B>>>2;
                always @(A,B,C,opcode,AAselect,BBselect,cc) begin
                cc = 1'b0;
                case(opcode)
                        3'd0: begin
                                AAselect=2'b00;
                                BBselect=1'b0;
                                outW=adderout;
                                cc=C;
                                end
                        3'd1: begin
                                AAselect=2'b01;
                                BBselect=1'b0;
                                outW=adderout;
                                end
                        3'd2: begin
                                AAselect=2'b10;
                                BBselect=1'b0;
                                outW=adderout;
                                end
                        3'd3: begin
                                AAselect=2'b11;
                                BBselect=1'b1;
                                outW=adderout;
                                end
                        3'd4: begin
                                AAselect=2'b00;
                                BBselect=2'b0;
                                outW=A&B;
                                end
                        3'd5: begin
                                AAselect=2'b00;
                                BBselect=2'b0;
                                outW=A|B;
                                end
                        3'd6: begin
                                AAselect=2'b00;
                                BBselect=2'b0;
                                outW=~B;
                                end
                        3'd7: begin
                                AAselect=2'b00;
                                BBselect=2'b0;
                                outW=16'd0;
                                end
                        default:outW=16'd0;
                        endcase
                end
                assign neg=outW[15];
                assign zer=~|outW;
endmodule
```

We have to change input for internal RTL components multiple times. So we create an internal signal for each input. Depending on the command and control signals sent from always command(in this design always command is equivalent to a multiplexer), internal signals are changed. To implement ¾ B, B is shifted 2 bits to the right (equivalent of ¼) and is subtracted from B.

Block diagram is shown below:



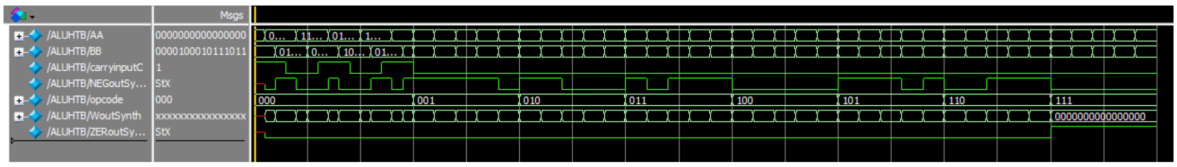
Code for the testbench is shown below:



```
`timescale 1ns/1ns
module ALUHTB();
        //wire ZERout,NEGout;
        //wire [15:0] Wout;
        wire ZERoutSynth,NEGoutSynth;
        wire [15:0] WoutSynth;
        logic [15:0] AA=16'd0,BB=16'd2235;
        logic carryinputC=1'b1;
        logic [2:0] opcode=3'd0;
        //ALUhardwarebackup  UUT (AA,BB,carryinputC,opcode,Wout,ZERout,NEGout);
        ALUhardware111 UUT2(AA,BB,carryinputC,opcode,WoutSynth,ZERoutSynth,NEGoutSynth);
        initial begin
        $monitor("monitor a:%d  b:%d carryinputC:%d opcode:%b Wout:%d", $signed(AA), $signed(BB),carryinputC,opcode,$signed(WoutSynth));
        repeat(5) begin
                #100 AA=$random;
                #100 BB=$random;
                #100 carryinputC=~carryinputC;
        end
        opcode=3'd1;
        repeat(5) begin
                AA=$random;
                BB=$random;
                #200;
        end
        opcode=3'd2;
        repeat(5) begin
                AA=$random;
                BB=$random;
                #200;
        end
        opcode=3'd3;
        repeat(5) begin
                AA=$random;
                BB=$random;
                #200;
        end
        opcode=3'd4;
        repeat(5) begin
                AA=$random;
                BB=$random;
                #200;
        end
        opcode=3'd5;
        repeat(5) begin
                AA=$random;
                BB=$random;
                #200;
        end
        opcode=3'd6;
        repeat(5) begin
                AA=$random;
                BB=$random;
                #200;
        end
        opcode=3'd7;
        repeat(5) begin
                AA=$random;
                BB=$random;
                #200;
        end
        end
endmodule
```

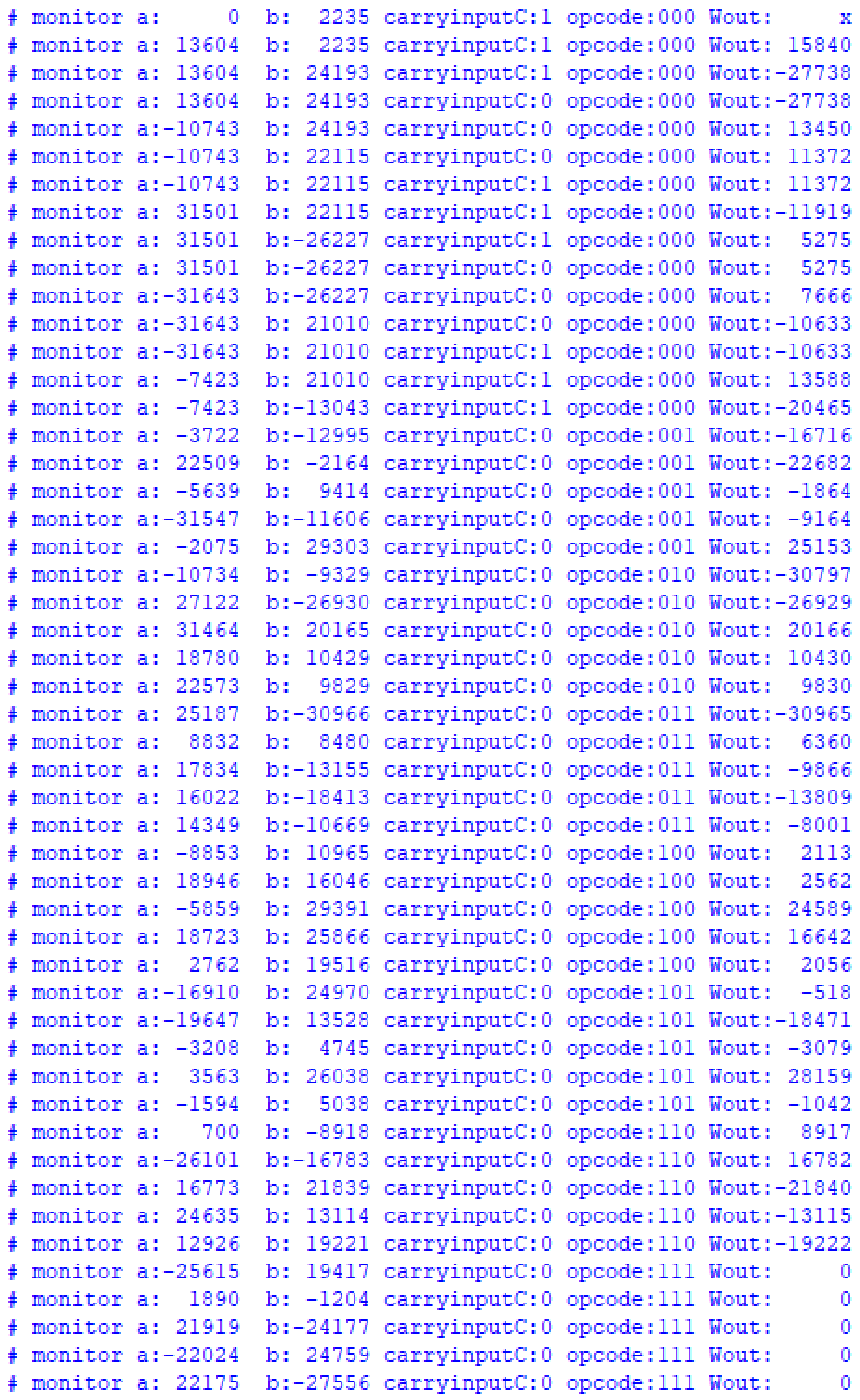
Same as part 1,monitor is used to understand and verify the design easier.

Testbench waveform result is shown below:



The part in the end of ZERout seems to change to 0 sometimes but it’s a display bug from modelsim and by using the cursor we can confirm that.

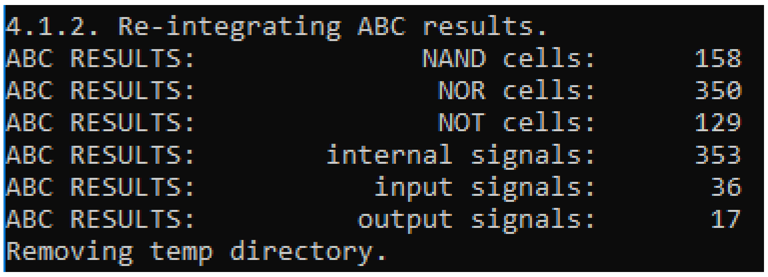
Part of the transcript result is shown below:



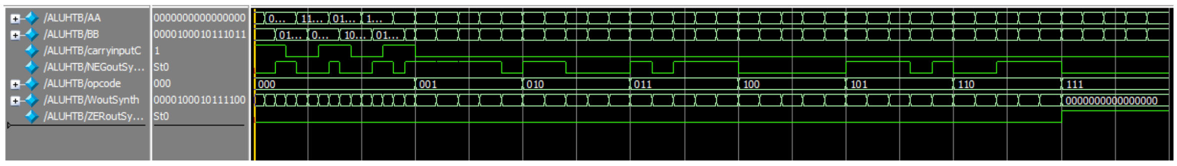
```
# monitor a:     0  b:  2235 carryinputC:1 opcode:000 Wout:     x
# monitor a: 13604  b:  2235 carryinputC:1 opcode:000 Wout: 15840
# monitor a: 13604  b: 24193 carryinputC:1 opcode:000 Wout:-27738
# monitor a: 13604  b: 24193 carryinputC:0 opcode:000 Wout:-27738
# monitor a:-10743  b: 24193 carryinputC:0 opcode:000 Wout: 13450
# monitor a:-10743  b: 22115 carryinputC:0 opcode:000 Wout: 11372
# monitor a:-10743  b: 22115 carryinputC:1 opcode:000 Wout: 11372
# monitor a: 31501  b: 22115 carryinputC:1 opcode:000 Wout:-11919
# monitor a: 31501  b:-26227 carryinputC:1 opcode:000 Wout:  5275
# monitor a: 31501  b:-26227 carryinputC:0 opcode:000 Wout:  5275
# monitor a:-31643  b:-26227 carryinputC:0 opcode:000 Wout:  7666
# monitor a:-31643  b: 21010 carryinputC:0 opcode:000 Wout:-10633
# monitor a:-31643  b: 21010 carryinputC:1 opcode:000 Wout:-10633
# monitor a: -7423  b: 21010 carryinputC:1 opcode:000 Wout: 13588
# monitor a: -7423  b:-13043 carryinputC:1 opcode:000 Wout:-20465
# monitor a: -3722  b:-12995 carryinputC:0 opcode:001 Wout:-16716
# monitor a: 22509  b: -2164 carryinputC:0 opcode:001 Wout:-22682
# monitor a: -5639  b:  9414 carryinputC:0 opcode:001 Wout: -1864
# monitor a:-31547  b:-11606 carryinputC:0 opcode:001 Wout: -9164
# monitor a: -2075  b: 29303 carryinputC:0 opcode:001 Wout: 25153
# monitor a:-10734  b: -9329 carryinputC:0 opcode:010 Wout:-30797
# monitor a: 27122  b:-26930 carryinputC:0 opcode:010 Wout:-26929
# monitor a: 31464  b: 20165 carryinputC:0 opcode:010 Wout: 20166
# monitor a: 18780  b: 10429 carryinputC:0 opcode:010 Wout: 10430
# monitor a: 22573  b:  9829 carryinputC:0 opcode:010 Wout:  9830
# monitor a: 25187  b:-30966 carryinputC:0 opcode:011 Wout:-30965
# monitor a:  8832  b:  8480 carryinputC:0 opcode:011 Wout:  6360
# monitor a: 17834  b:-13155 carryinputC:0 opcode:011 Wout: -9866
# monitor a: 16022  b:-18413 carryinputC:0 opcode:011 Wout:-13809
# monitor a: 14349  b:-10669 carryinputC:0 opcode:011 Wout: -8001
# monitor a: -8853  b: 10965 carryinputC:0 opcode:100 Wout:  2113
# monitor a: 18946  b: 16046 carryinputC:0 opcode:100 Wout:  2562
# monitor a: -5859  b: 29391 carryinputC:0 opcode:100 Wout: 24589
# monitor a: 18723  b: 25866 carryinputC:0 opcode:100 Wout: 16642
# monitor a:  2762  b: 19516 carryinputC:0 opcode:100 Wout:  2056
# monitor a:-16910  b: 24970 carryinputC:0 opcode:101 Wout:  -518
# monitor a:-19647  b: 13528 carryinputC:0 opcode:101 Wout:-18471
# monitor a: -3208  b:  4745 carryinputC:0 opcode:101 Wout: -3079
# monitor a:  3563  b: 26038 carryinputC:0 opcode:101 Wout: 28159
# monitor a: -1594  b:  5038 carryinputC:0 opcode:101 Wout: -1042
# monitor a:   700  b: -8918 carryinputC:0 opcode:110 Wout:  8917
# monitor a:-26101  b:-16783 carryinputC:0 opcode:110 Wout: 16782
# monitor a: 16773  b: 21839 carryinputC:0 opcode:110 Wout:-21840
# monitor a: 24635  b: 13114 carryinputC:0 opcode:110 Wout:-13115
# monitor a: 12926  b: 19221 carryinputC:0 opcode:110 Wout:-19222
# monitor a:-25615  b: 19417 carryinputC:0 opcode:111 Wout:     0
# monitor a:  1890  b: -1204 carryinputC:0 opcode:111 Wout:     0
# monitor a: 21919  b:-24177 carryinputC:0 opcode:111 Wout:     0
# monitor a:-22024  b: 24759 carryinputC:0 opcode:111 Wout:     0
# monitor a: 22175  b:-27556 carryinputC:0 opcode:111 Wout:     0
```

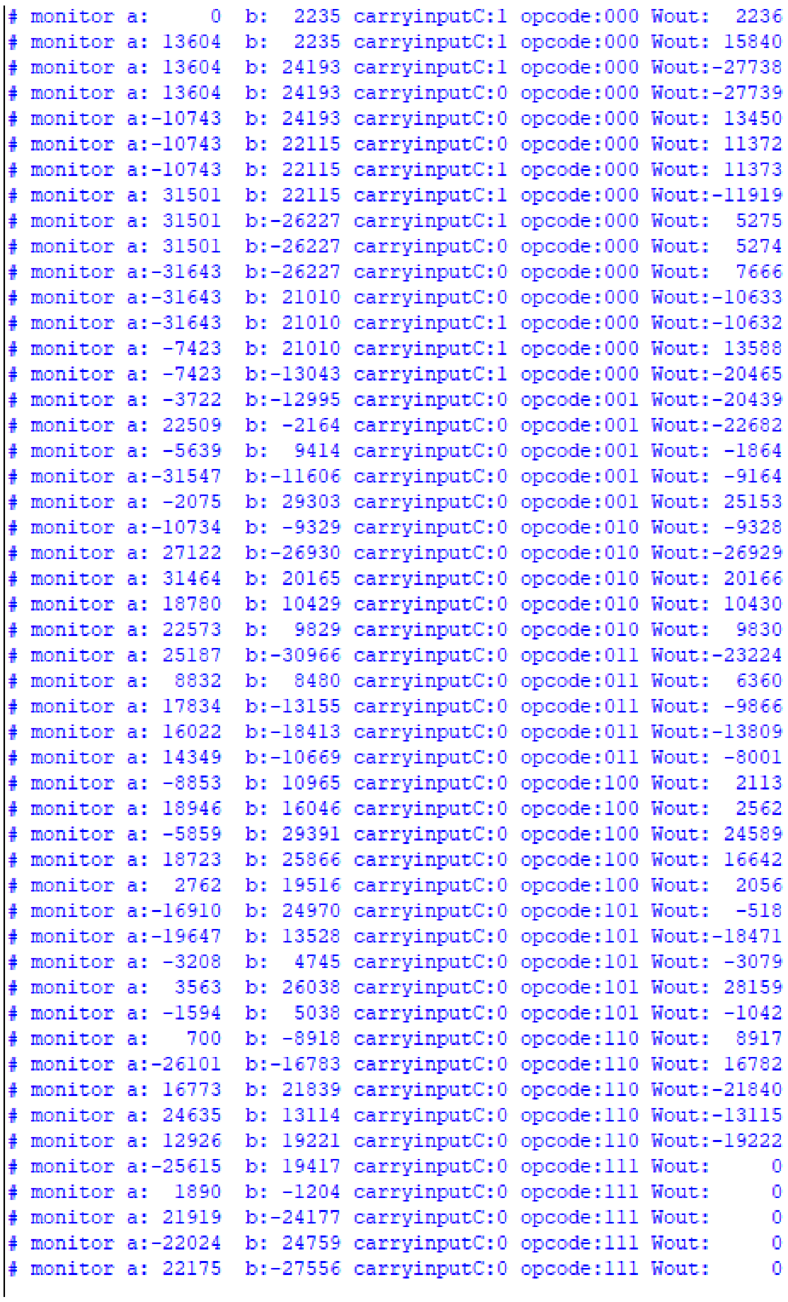
It can be confirmed that the ALU works correctly.

B,C)The result of yosis synthesis is shown below:



```
4.1.2. Re-integrating ABC results.
ABC RESULTS:               NAND cells:      158
ABC RESULTS:                NOR cells:      350
ABC RESULTS:                NOT cells:      129
ABC RESULTS:         internal signals:      353
ABC RESULTS:            input signals:       36
ABC RESULTS:           output signals:       17
Removing temp directory.
```



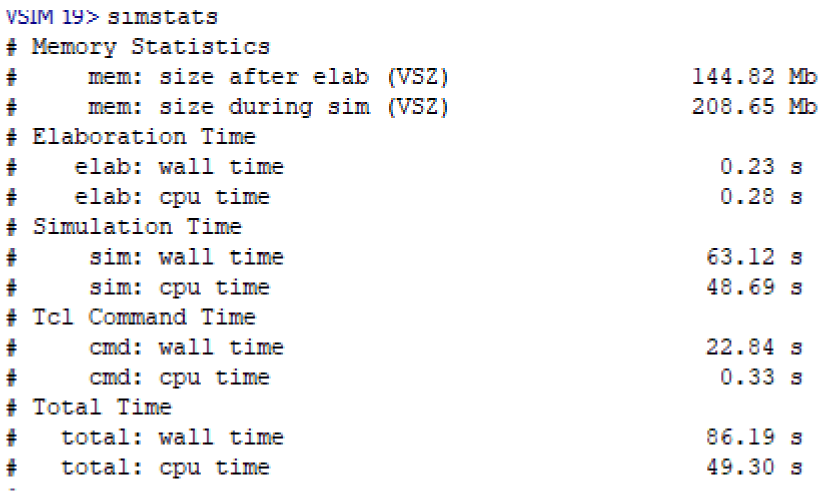


```
# monitor a:      0  b:  2235 carryinputC:1 opcode:000 Wout:  2236
# monitor a: 13604  b:  2235 carryinputC:1 opcode:000 Wout: 15840
# monitor a: 13604  b: 24193 carryinputC:1 opcode:000 Wout:-27738
# monitor a: 13604  b: 24193 carryinputC:0 opcode:000 Wout:-27739
# monitor a:-10743  b: 24193 carryinputC:0 opcode:000 Wout: 13450
# monitor a:-10743  b: 22115 carryinputC:0 opcode:000 Wout: 11372
# monitor a:-10743  b: 22115 carryinputC:1 opcode:000 Wout: 11373
# monitor a: 31501  b: 22115 carryinputC:1 opcode:000 Wout:-11919
# monitor a: 31501  b:-26227 carryinputC:1 opcode:000 Wout:  5275
# monitor a: 31501  b:-26227 carryinputC:0 opcode:000 Wout:  5274
# monitor a:-31643  b:-26227 carryinputC:0 opcode:000 Wout:  7666
# monitor a:-31643  b: 21010 carryinputC:0 opcode:000 Wout:-10633
# monitor a:-31643  b: 21010 carryinputC:1 opcode:000 Wout:-10632
# monitor a: -7423  b: 21010 carryinputC:1 opcode:000 Wout: 13588
# monitor a: -7423  b:-13043 carryinputC:1 opcode:000 Wout:-20465
# monitor a: -3722  b:-12995 carryinputC:0 opcode:001 Wout:-20439
# monitor a: 22509  b: -2164 carryinputC:0 opcode:001 Wout:-22682
# monitor a: -5639  b:  9414 carryinputC:0 opcode:001 Wout: -1864
# monitor a:-31547  b:-11606 carryinputC:0 opcode:001 Wout: -9164
# monitor a: -2075  b: 29303 carryinputC:0 opcode:001 Wout: 25153
# monitor a:-10734  b: -9329 carryinputC:0 opcode:010 Wout: -9328
# monitor a: 27122  b:-26930 carryinputC:0 opcode:010 Wout:-26929
# monitor a: 31464  b: 20165 carryinputC:0 opcode:010 Wout: 20166
# monitor a: 18780  b: 10429 carryinputC:0 opcode:010 Wout: 10430
# monitor a: 22573  b:  9829 carryinputC:0 opcode:010 Wout:  9830
# monitor a: 25187  b:-30966 carryinputC:0 opcode:011 Wout:-23224
# monitor a:  8832  b:  8480 carryinputC:0 opcode:011 Wout:  6360
# monitor a: 17834  b:-13155 carryinputC:0 opcode:011 Wout: -9866
# monitor a: 16022  b:-18413 carryinputC:0 opcode:011 Wout:-13809
# monitor a: 14349  b:-10669 carryinputC:0 opcode:011 Wout: -8001
# monitor a: -8853  b: 10965 carryinputC:0 opcode:100 Wout:  2113
# monitor a: 18946  b: 16046 carryinputC:0 opcode:100 Wout:  2562
# monitor a: -5859  b: 29391 carryinputC:0 opcode:100 Wout: 24589
# monitor a: 18723  b: 25866 carryinputC:0 opcode:100 Wout: 16642
# monitor a:  2762  b: 19516 carryinputC:0 opcode:100 Wout:  2056
# monitor a:-16910  b: 24970 carryinputC:0 opcode:101 Wout:  -518
# monitor a:-19647  b: 13528 carryinputC:0 opcode:101 Wout:-18471
# monitor a: -3208  b:  4745 carryinputC:0 opcode:101 Wout: -3079
# monitor a:  3563  b: 26038 carryinputC:0 opcode:101 Wout: 28159
# monitor a: -1594  b:  5038 carryinputC:0 opcode:101 Wout: -1042
# monitor a:   700  b: -8918 carryinputC:0 opcode:110 Wout:  8917
# monitor a:-26101  b:-16783 carryinputC:0 opcode:110 Wout: 16782
# monitor a: 16773  b: 21839 carryinputC:0 opcode:110 Wout:-21840
# monitor a: 24635  b: 13114 carryinputC:0 opcode:110 Wout:-13115
# monitor a: 12926  b: 19221 carryinputC:0 opcode:110 Wout:-19222
# monitor a:-25615  b: 19417 carryinputC:0 opcode:111 Wout:     0
# monitor a:  1890  b: -1204 carryinputC:0 opcode:111 Wout:     0
# monitor a: 21919  b:-24177 carryinputC:0 opcode:111 Wout:     0
# monitor a:-22024  b: 24759 carryinputC:0 opcode:111 Wout:     0
# monitor a: 22175  b:-27556 carryinputC:0 opcode:111 Wout:     0
```

It can be seen that the synthesized version works as intended.

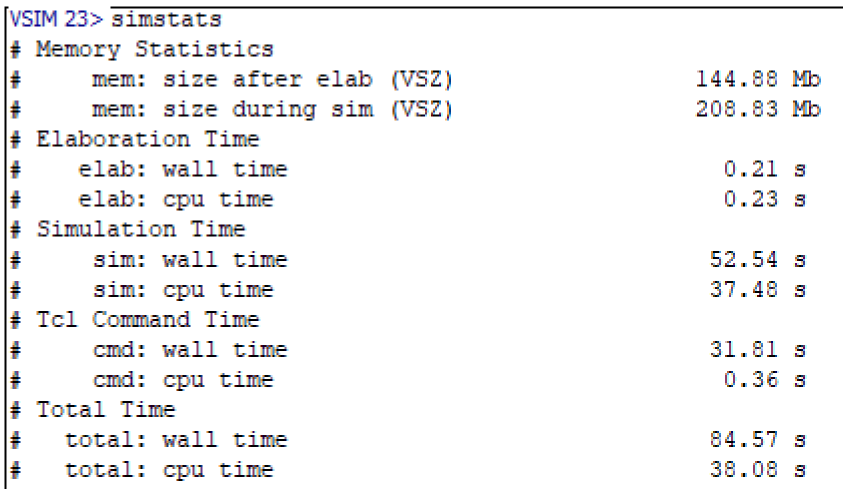
D)

Simstats post synthesize:



```
VSIM 19> simstats
# Memory Statistics
#     mem: size after elab (VSZ)                 144.82 Mb
#     mem: size during sim (VSZ)                 208.65 Mb
# Elaboration Time
#    elab: wall time                               0.23 s
#    elab: cpu time                                0.28 s
# Simulation Time
#     sim: wall time                              63.12 s
#     sim: cpu time                               48.69 s
# Tcl Command Time
#     cmd: wall time                              22.84 s
#     cmd: cpu time                                0.33 s
# Total Time
#   total: wall time                              86.19 s
#   total: cpu time                               49.30 s
```

Simstats pre synthesize:



```
VSIM 23> simstats
# Memory Statistics
#     mem: size after elab (VSZ)                 144.88 Mb
#     mem: size during sim (VSZ)                 208.83 Mb
# Elaboration Time
#    elab: wall time                               0.21 s
#    elab: cpu time                                0.23 s
# Simulation Time
#     sim: wall time                              52.54 s
#     sim: cpu time                               37.48 s
# Tcl Command Time
#     cmd: wall time                              31.81 s
#     cmd: cpu time                                0.36 s
# Total Time
#   total: wall time                              84.57 s
#   total: cpu time                               38.08 s
```

Simulation time in pre synthesize is lower by 10 seconds which is good because the chip calculates the output directly from algorithm while post synthesized version has to propagate the values by driving all of the gates.

Q3

To sum up all the effort in this assignment, we can safely say that yosis is not very intelligent and smart in synthesizing. As an electronic engineer we should design the hardware the most optimized way so synthesis would also be optimized. This is concluded from comparison of Q1 and Q2 synthesis which can be observed that in question 1 more gates are used. Difference is about 40 percent which is a lot for a hardware this small. In industry level designs, this can be very upsetting.