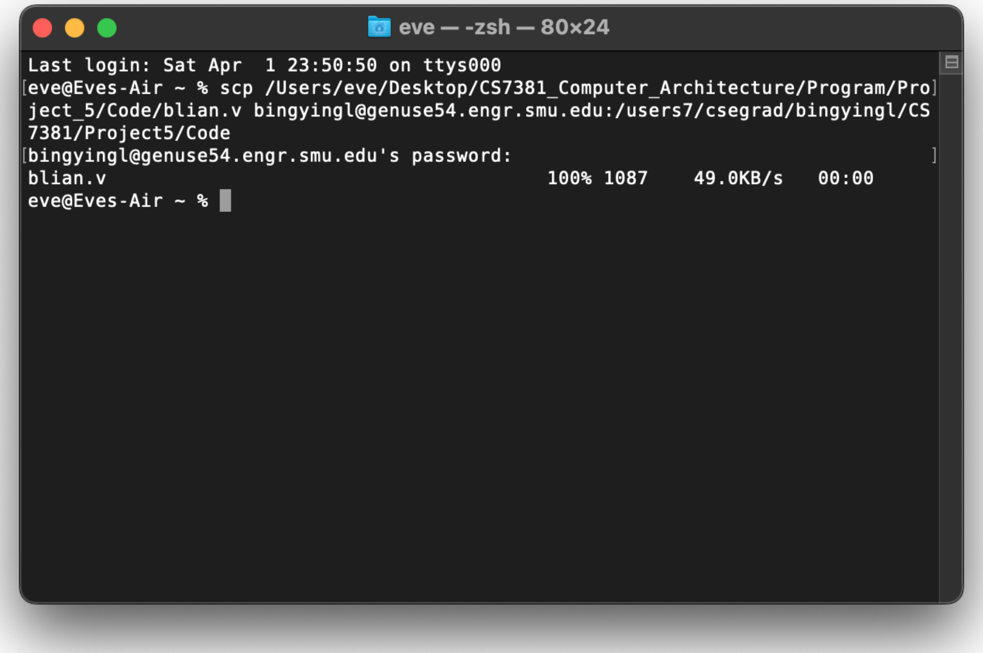
CS7381 Project 5: Verilog Code Development – MIPS ALU Design

Name: Bingying Liang
ID: 48999397
Distance

March 31 2023

In the previous project (Project 4), you were introduced to the Verilog code development using the Cadence Xcelium tool. Next, we will use Verilog to modify the basic MIPS ALU of Project 4 and simulate this modification. **NOTE: for additional Verilog resources, please see my web link**

1. Create a copy of the S23_MIPS_ALU_basic.v file from Project 4. Similar to the MARS assignment, name your file using the first initial of your first name and the first 4 letters of your last name. For example, my ALU file name would be **tmani.v**.



```
Last login: Sat Apr  1 23:50:50 on ttys000
[eve@Eves-Air ~ % scp /Users/eve/Desktop/CS7381_Computer_Architecture/Program/Pro]
ject_5/Code/blian.v bingyingl@genuse54.engr.smu.edu:/users7/csegrad/bingyingl/CS
7381/Project5/Code
[bingyingl@genuse54.engr.smu.edu's password:                                    ]
blian.v                                       100% 1087    49.0KB/s   00:00
eve@Eves-Air ~ % 
```

2. **Add** the following functions to the ALU:

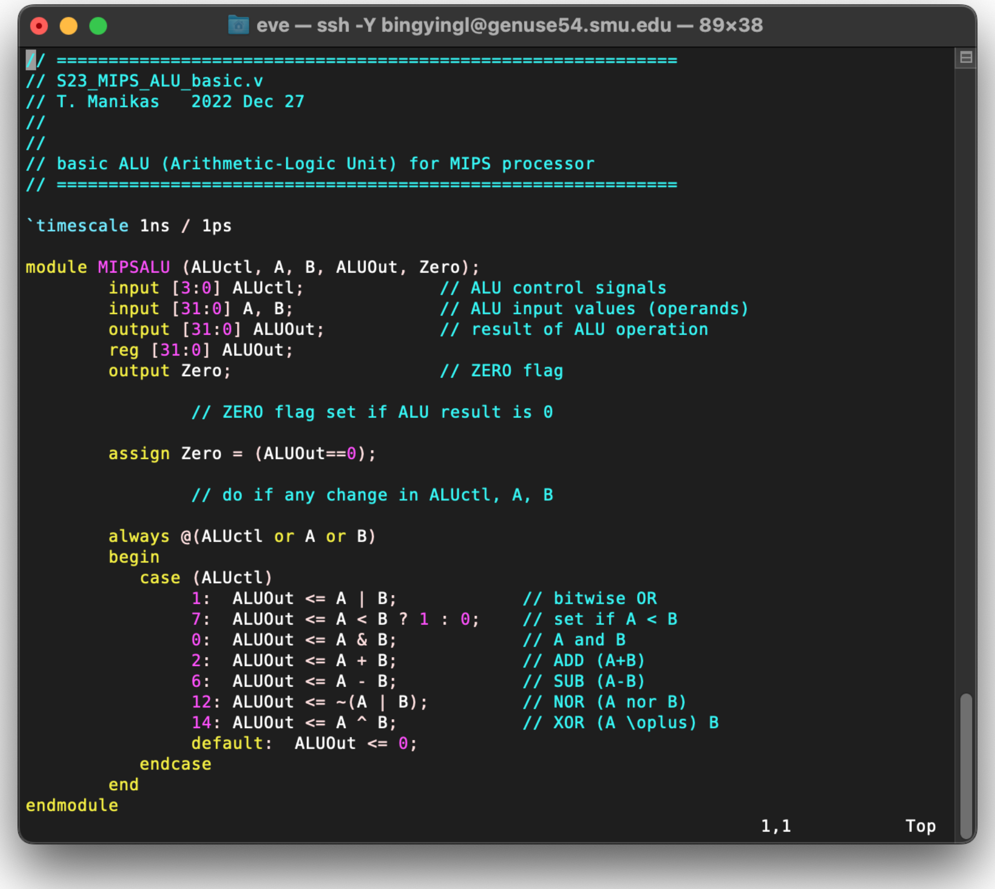
(a) Function code 0 to implement bitwise AND: $A \; and \; B$

(b) Function code 2 to implement ADD $(A + B)$

(c) Function code 6 to implement SUB $(A - B)$

(d) Function code 12 to implement bitwise NOR $(A \; nor \; B)$

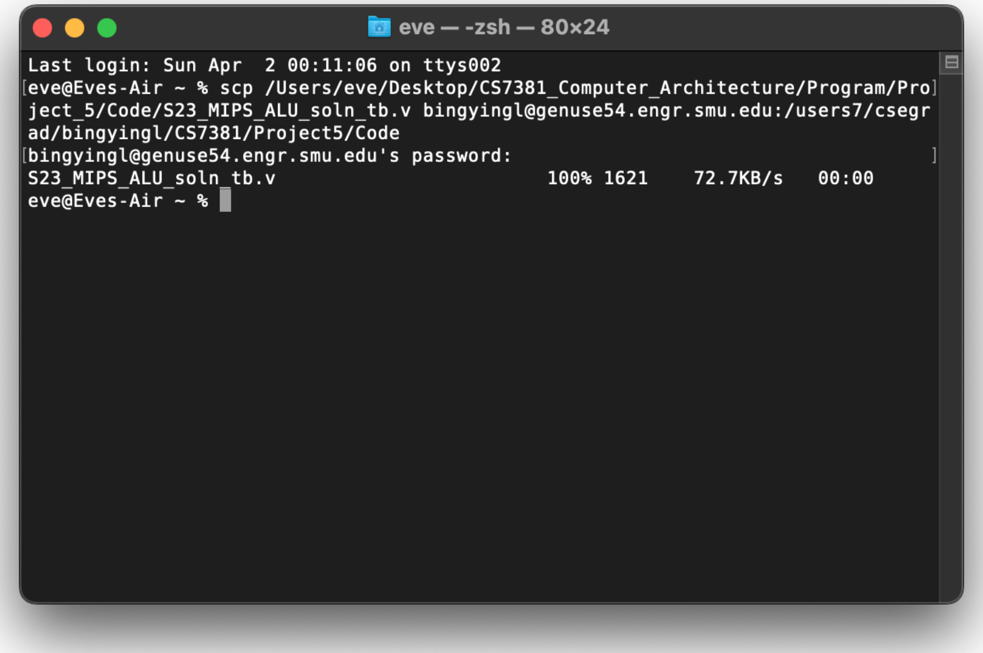
(e) Function code 14 to implement bitwise XOR $(A \oplus B)$



```
// ============================================================
// S23_MIPS_ALU_basic.v
// T. Manikas   2022 Dec 27
//
//
// basic ALU (Arithmetic-Logic Unit) for MIPS processor
// ============================================================

`timescale 1ns / 1ps

module MIPSALU (ALUctl, A, B, ALUOut, Zero);
        input [3:0] ALUctl;                     // ALU control signals
        input [31:0] A, B;                      // ALU input values (operands)
        output [31:0] ALUOut;                   // result of ALU operation
        reg [31:0] ALUOut;
        output Zero;                            // ZERO flag

                // ZERO flag set if ALU result is 0

        assign Zero = (ALUOut==0);

                // do if any change in ALUctl, A, B

        always @(ALUctl or A or B)
        begin
           case (ALUctl)
                1:  ALUOut <= A | B;                    // bitwise OR
                7:  ALUOut <= A < B ? 1 : 0;    // set if A < B
                0:  ALUOut <= A & B;                    // A and B
                2:  ALUOut <= A + B;                    // ADD (A+B)
                6:  ALUOut <= A - B;                    // SUB (A-B)
                12: ALUOut <= ~(A | B);                 // NOR (A nor B)
                14: ALUOut <= A ^ B;                    // XOR (A \oplus) B
                default:  ALUOut <= 0;
           endcase
        end
endmodule
                                                                       1,1           Top
```

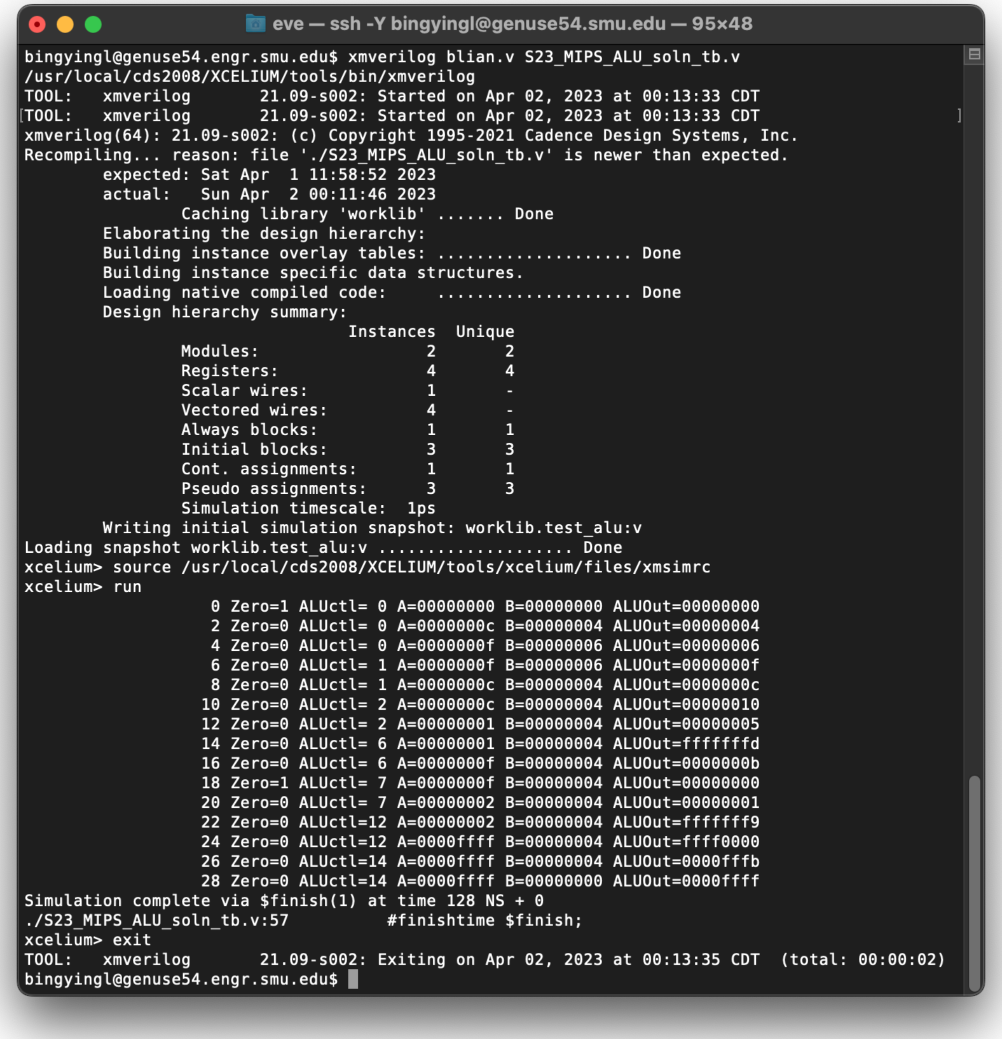
3. Please download the following Verilog file:

(a) S23_MIPS_ALU_soln_tb.v - the testbench for testing your updated ALU



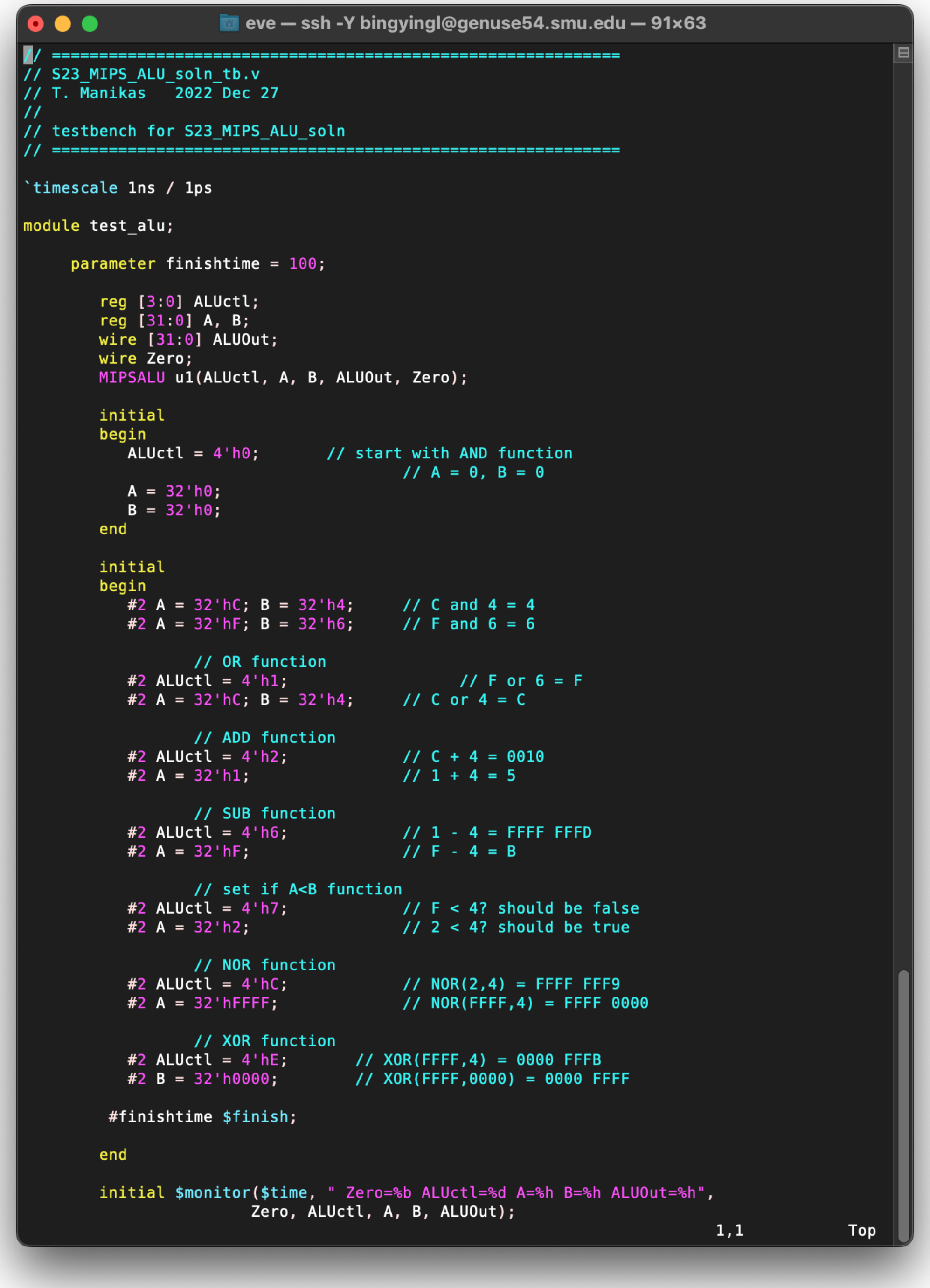
```
Last login: Sun Apr  2 00:11:06 on ttys002
[eve@Eves-Air ~ % scp /Users/eve/Desktop/CS7381_Computer_Architecture/Program/Pro]
ject_5/Code/S23_MIPS_ALU_soln_tb.v bingyingl@genuse54.engr.smu.edu:/users7/csegr
ad/bingyingl/CS7381/Project5/Code
[bingyingl@genuse54.engr.smu.edu's password:                                    ]
S23_MIPS_ALU_soln_tb.v                        100% 1621    72.7KB/s   00:00
eve@Eves-Air ~ % 
```

4. **Test your modified ALU using this testbench**



```
bingyingl@genuse54.engr.smu.edu$ xmverilog blian.v S23_MIPS_ALU_soln_tb.v
/usr/local/cds2008/XCELIUM/tools/bin/xmverilog
TOOL:   xmverilog       21.09-s002: Started on Apr 02, 2023 at 00:13:33 CDT
TOOL:   xmverilog       21.09-s002: Started on Apr 02, 2023 at 00:13:33 CDT
xmverilog(64): 21.09-s002: (c) Copyright 1995-2021 Cadence Design Systems, Inc.
Recompiling... reason: file './S23_MIPS_ALU_soln_tb.v' is newer than expected.
        expected: Sat Apr  1 11:58:52 2023
        actual:   Sun Apr  2 00:11:46 2023
                Caching library 'worklib' ....... Done
        Elaborating the design hierarchy:
        Building instance overlay tables: .................... Done
        Building instance specific data structures.
        Loading native compiled code:     .................... Done
        Design hierarchy summary:
                                 Instances  Unique
                Modules:                 2       2
                Registers:               4       4
                Scalar wires:            1       -
                Vectored wires:          4       -
                Always blocks:           1       1
                Initial blocks:          3       3
                Cont. assignments:       1       1
                Pseudo assignments:      3       3
                Simulation timescale:  1ps
        Writing initial simulation snapshot: worklib.test_alu:v
Loading snapshot worklib.test_alu:v .................... Done
xcelium> source /usr/local/cds2008/XCELIUM/tools/xcelium/files/xmsimrc
xcelium> run
                   0 Zero=1 ALUctl= 0 A=00000000 B=00000000 ALUOut=00000000
                   2 Zero=0 ALUctl= 0 A=0000000c B=00000004 ALUOut=00000004
                   4 Zero=0 ALUctl= 0 A=0000000f B=00000006 ALUOut=00000006
                   6 Zero=0 ALUctl= 1 A=0000000f B=00000006 ALUOut=0000000f
                   8 Zero=0 ALUctl= 1 A=0000000c B=00000004 ALUOut=0000000c
                  10 Zero=0 ALUctl= 2 A=0000000c B=00000004 ALUOut=00000010
                  12 Zero=0 ALUctl= 2 A=00000001 B=00000004 ALUOut=00000005
                  14 Zero=0 ALUctl= 6 A=00000001 B=00000004 ALUOut=fffffffd
                  16 Zero=0 ALUctl= 6 A=0000000f B=00000004 ALUOut=0000000b
                  18 Zero=1 ALUctl= 7 A=0000000f B=00000004 ALUOut=00000000
                  20 Zero=0 ALUctl= 7 A=00000002 B=00000004 ALUOut=00000001
                  22 Zero=0 ALUctl=12 A=00000002 B=00000004 ALUOut=fffffff9
                  24 Zero=0 ALUctl=12 A=0000ffff B=00000004 ALUOut=ffff0000
                  26 Zero=0 ALUctl=14 A=0000ffff B=00000004 ALUOut=0000fffb
                  28 Zero=0 ALUctl=14 A=0000ffff B=00000000 ALUOut=0000ffff
Simulation complete via $finish(1) at time 128 NS + 0
./S23_MIPS_ALU_soln_tb.v:57          #finishtime $finish;
xcelium> exit
TOOL:   xmverilog       21.09-s002: Exiting on Apr 02, 2023 at 00:13:35 CDT  (total: 00:00:02)
bingyingl@genuse54.engr.smu.edu$
```

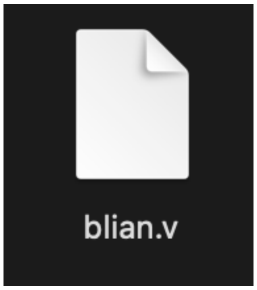
Compare the the result with testbench annotation, they are the same.



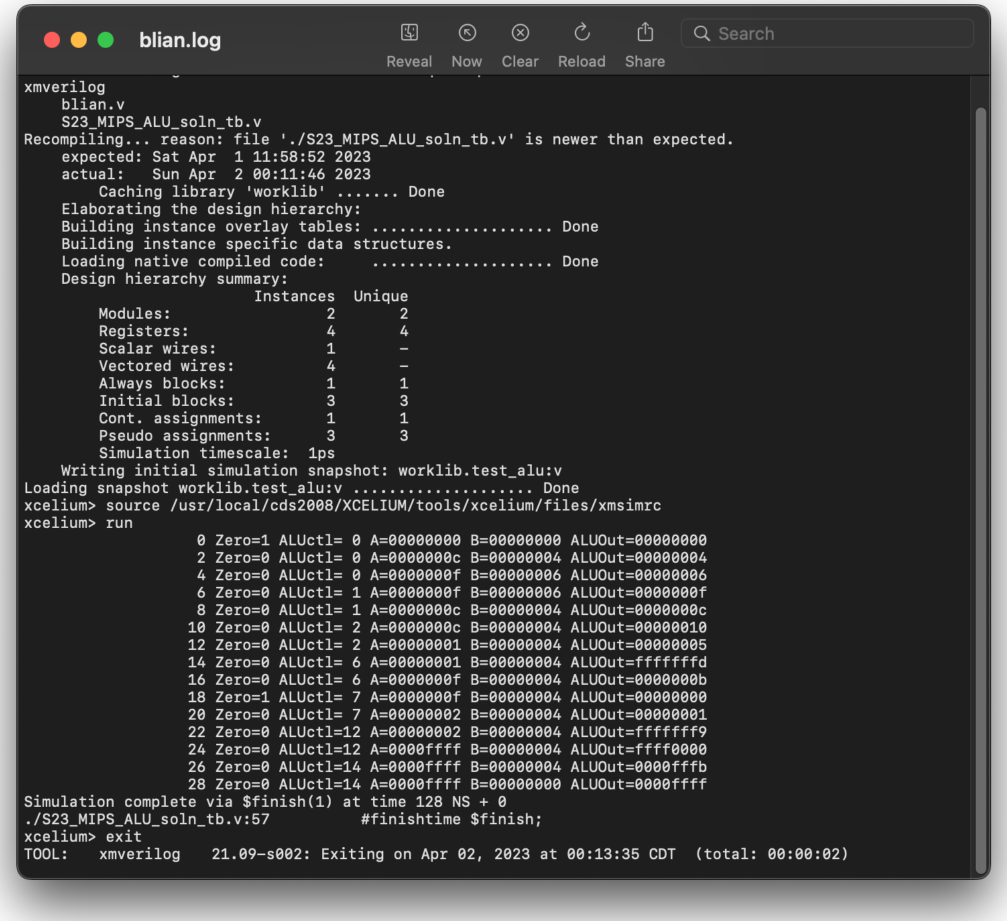
```
// ============================================================
// S23_MIPS_ALU_soln_tb.v
// T. Manikas   2022 Dec 27
//
// testbench for S23_MIPS_ALU_soln
// ============================================================

`timescale 1ns / 1ps

module test_alu;

     parameter finishtime = 100;

        reg [3:0] ALUctl;
        reg [31:0] A, B;
        wire [31:0] ALUOut;
        wire Zero;
        MIPSALU u1(ALUctl, A, B, ALUOut, Zero);

        initial
        begin
           ALUctl = 4'h0;       // start with AND function
                                        // A = 0, B = 0
           A = 32'h0;
           B = 32'h0;
        end

        initial
        begin
           #2 A = 32'hC; B = 32'h4;     // C and 4 = 4
           #2 A = 32'hF; B = 32'h6;     // F and 6 = 6

                  // OR function
           #2 ALUctl = 4'h1;                  // F or 6 = F
           #2 A = 32'hC; B = 32'h4;     // C or 4 = C

                  // ADD function
           #2 ALUctl = 4'h2;            // C + 4 = 0010
           #2 A = 32'h1;                // 1 + 4 = 5

                  // SUB function
           #2 ALUctl = 4'h6;            // 1 - 4 = FFFF FFFD
           #2 A = 32'hF;                // F - 4 = B

                  // set if A<B function
           #2 ALUctl = 4'h7;            // F < 4? should be false
           #2 A = 32'h2;                // 2 < 4? should be true

                  // NOR function
           #2 ALUctl = 4'hC;            // NOR(2,4) = FFFF FFF9
           #2 A = 32'hFFFF;             // NOR(FFFF,4) = FFFF 0000

                  // XOR function
           #2 ALUctl = 4'hE;       // XOR(FFFF,4) = 0000 FFFB
           #2 B = 32'h0000;        // XOR(FFFF,0000) = 0000 FFFF

         #finishtime $finish;

        end

        initial $monitor($time, " Zero=%b ALUctl=%d A=%h B=%h ALUOut=%h",
                        Zero, ALUctl, A, B, ALUOut);
                                                                         1,1           Top
```

5. **Please include the following for your homework submission:**

   (a) Your modified MIPS ALU Verilog file – **submit the actual *.v file so that the grader can run it.**



(b) Your testbench results – this can be a copy of the results in a Word document.



```
xmverilog
    blian.v
    S23_MIPS_ALU_soln_tb.v
Recompiling... reason: file './S23_MIPS_ALU_soln_tb.v' is newer than expected.
    expected: Sat Apr  1 11:58:52 2023
    actual:   Sun Apr  2 00:11:46 2023
        Caching library 'worklib' ....... Done
    Elaborating the design hierarchy:
    Building instance overlay tables: .................... Done
    Building instance specific data structures.
    Loading native compiled code:     .................... Done
    Design hierarchy summary:
                         Instances  Unique
        Modules:                 2       2
        Registers:               4       4
        Scalar wires:            1       -
        Vectored wires:          4       -
        Always blocks:           1       1
        Initial blocks:          3       3
        Cont. assignments:       1       1
        Pseudo assignments:      3       3
        Simulation timescale:  1ps
    Writing initial simulation snapshot: worklib.test_alu:v
Loading snapshot worklib.test_alu:v .................... Done
xcelium> source /usr/local/cds2008/XCELIUM/tools/xcelium/files/xmsimrc
xcelium> run
                   0 Zero=1 ALUctl= 0 A=00000000 B=00000000 ALUOut=00000000
                   2 Zero=0 ALUctl= 0 A=0000000c B=00000004 ALUOut=00000004
                   4 Zero=0 ALUctl= 0 A=0000000f B=00000006 ALUOut=00000006
                   6 Zero=0 ALUctl= 1 A=0000000f B=00000006 ALUOut=0000000f
                   8 Zero=0 ALUctl= 1 A=0000000c B=00000004 ALUOut=0000000c
                  10 Zero=0 ALUctl= 2 A=0000000c B=00000004 ALUOut=00000010
                  12 Zero=0 ALUctl= 2 A=00000001 B=00000004 ALUOut=00000005
                  14 Zero=0 ALUctl= 6 A=00000001 B=00000004 ALUOut=fffffffd
                  16 Zero=0 ALUctl= 6 A=0000000f B=00000004 ALUOut=0000000b
                  18 Zero=1 ALUctl= 7 A=0000000f B=00000004 ALUOut=00000000
                  20 Zero=0 ALUctl= 7 A=00000002 B=00000004 ALUOut=00000001
                  22 Zero=0 ALUctl=12 A=00000002 B=00000004 ALUOut=fffffff9
                  24 Zero=0 ALUctl=12 A=0000ffff B=00000004 ALUOut=ffff0000
                  26 Zero=0 ALUctl=14 A=0000ffff B=00000004 ALUOut=0000fffb
                  28 Zero=0 ALUctl=14 A=0000ffff B=00000000 ALUOut=0000ffff
Simulation complete via $finish(1) at time 128 NS + 0
./S23_MIPS_ALU_soln_tb.v:57          #finishtime $finish;
xcelium> exit
TOOL:   xmverilog   21.09-s002: Exiting on Apr 02, 2023 at 00:13:35 CDT  (total: 00:00:02)
```

I save the result in the document as blian.log

(c) **PLEASE MAKE SURE THAT YOUR NAME APPEARS ON ALL SUBMITTED ITEMS FOR PROPER CREDIT**