CS 223 Digital Design
Section 01
Lab 05

NAME: Ferhat
SURNAME: Korkmaz
ID: 21901940
DATE: 15.04.2021

a-)
```
module fulladder(input logic a, b, cin, output logic cout, s );
    assign s = a ^ b ^ cin;
    assign cout = (a & b) | (a & cin) | (b & cin);
endmodule

module ALU(input logic [3:0] a, b, input logic [1:0] sel, output logic [3:0] rel);
    logic [3:0] n0, n1, n2, n3;;
    logic [3:0] cout0, cout1, cout2;
    fulladder aPlusOne0(a[0], 1, 0, cout0[0], n0[0]);
    fulladder aPlusOne1(a[1], 0, cout0[0], cout0[1], n0[1]);
    fulladder aPlusOne2(a[2], 0, cout0[1], cout0[2], n0[2]);
    fulladder aPlusOne3(a[3], 0, cout0[2], cout0[3], n0[3]);
    fulladder aPlusB0(a[0], b[0], 0, cout1[0], n1[0]);
    fulladder aPlusB1(a[1], b[1], cout1[0], cout1[1], n1[1]);
    fulladder aPlusB2(a[2], b[2], cout1[1], cout1[2], n1[2]);
    fulladder aPlusB3(a[3], b[3], cout1[2], cout1[3], n1[3]);
    fulladder aMinusB0(a[0], ~b[0], 1, cout2[0], n2[0]);
    fulladder aMinusB1(a[1], ~b[1], cout2[0], cout2[1], n2[1]);
    fulladder aMinusB2(a[2], ~b[2], cout2[1], cout2[2], n2[2]);
    fulladder aMinusB3(a[3], ~b[3], cout2[2], cout2[3], n2[3]);
    assign n3[0] = a[0] | b[0];
    assign n3[1] = a[1] | b[1];
    assign n3[2] = a[2] | b[2];
    assign n3[3] = a[3] | b[3];
    fourToOneMux finalize(n0,n1,n2,n3,sel,rel);
endmodule
```

```
module twoToOneMux( input logic select, input logic [3:0] input1, input0, output logic [3:0] out );
    assign out = select ? input1 : input0;
endmodule

module fourToOneMux(input logic[3:0] in0, in1, in2, in3, input logic[1:0] select, output logic[3:0] out );
      assign out = select[1] ? ( select[0] ? in3 : in2 ) : ( select[0] ? in1: in0);
endmodule


module eightToOneMux(input logic [3:0] d0, d1, d2, d3, d4, d5, d6, d7, [2:0] s, output logic [3:0] y);
    assign y = s[2] ? ( s[1] ? ( s[0] ? d7 : d6 ) : ( s[0] ? d5 : d4 ) ) : ( s[1] ? ( s[0] ? d3 : d2 ) : ( s[0] ? d1 : d0 ) );
endmodule



module threeToEightDecoder( input logic a, b, c, enable, output logic [7:0] y );


    assign y[0] = enable & (~a & ~b & ~c);
    assign y[1] = enable & (~a & ~b & c);
    assign y[2] = enable & (~a & b & ~c);
    assign y[3] = enable & (~a & b & c);
    assign y[4] = enable & (a & ~b & ~c);
    assign y[5] = enable & (a & ~b & c);
    assign y[6] = enable & (a & b & ~c);
    assign y[7] = enable & (a & b & c);


endmodule



module RegisterFile( input logic clock, wr_en, reset, [2:0] rda1, rda2, wra, [3:0] wrd, output logic [3:0] rdd1 ,rdd2);
```

```
    logic [7:0] addr;
    logic [3:0] regDat1, regDat2, regDat3, regDat4, regDat5, regDat6, regDat7, regDat8;


module Register(input logic clock, load, reset, input logic[3:0] d, output logic [3:0] q);
    always@(posedge clock or posedge reset)
      if (reset)
        begin
        q[0] <=  0;
        q[1] <=  0;
        q[2] <=  0;
        q[3] <=  0;
        end
      else
         begin
         if (load)
         begin
         q[0] <=  d[0];
         q[1] <=  d[1];
         q[2] <=  d[2];
         q[3] <=  d[3];
         end
      end
endmodule
```

```
    threeToEightDecoder addresWrite( wra[2], wra[1], wra[0], wr_en, addr);
    Register reg1( clock, addr[0], reset, wrd, regDat1 );
    Register reg2( clock, addr[1], reset, wrd, regDat2 );
    Register reg3( clock, addr[2], reset, wrd, regDat3 );
    Register reg4( clock, addr[3], reset, wrd, regDat4 );
    Register reg5( clock, addr[4], reset, wrd, regDat5 );
    Register reg6( clock, addr[5], reset, wrd, regDat6 );
    Register reg7( clock, addr[6], reset, wrd, regDat7 );
    Register reg8( clock, addr[7], reset, wrd, regDat8 );
    eightToOneMux muxData0( regDat1, regDat2, regDat3, regDat4, regDat5, regDat6, regDat7, regDat8, rda1, rdd1 );
    eightToOneMux muxData1( regDat1, regDat2, regDat3, regDat4, regDat5, regDat6, regDat7, regDat8, rda2, rdd2 );

endmodule
```

b-)

```
module testbenchALU();
logic [1:0] select;
logic [3:0] a, b, rel;

ALU testRun(a, b, select, rel);
    initial begin
        select = 0; a = 3; b = 1;
        #100;
        select = 0; a = 10; b = 1;
        #100;
        select = 1; a = 3; b = 1;
        #100;
        select = 1; a = 3; b = 5;
        #100;
```

```
        select = 2; a = 3; b = 1;
        #100;
        select = 2; a = 3; b = 5;
        #100;
        select = 3; a = 3; b = 5;
        #100;
        select = 3; a = 3; b = 1;
        #100;
    end
endmodule

module testbenchRegisterFile();
     logic clk, wr_en, rst;
    logic [2:0] rda1, rda2, wra;
    logic [3:0] wrd, rdd1, rdd2;

    RegisterFile regfile(clk, wr_en, rst, rda1, rda2, wra, wrd, rdd1, rdd2);

    always #10 clk = ~clk;

     initial
        begin
        clk <= 0;
        wr_en <= 1;
        rst <= 1;
        rst <= 0;
        #20;
        wra <= 0;
        wrd <= 0;
        rda1 <= 0;
```

```
        #20
        wra <= 1;
        wrd <= 1;
        rda2 <= 1;
        #20
        wr_en <= 0;
        wra <= 1;
        wrd <= 2;
        rda1 <= 1;
        rda2 <= 0;
        #20
        wr_en <= 1;
        wra <= 3;
        wrd <= 10;
        rda1 <= 3;
        rda2 <= 1;
        end
endmodule
```

c-)

```
module topModule( input logic [3:0] EXTDATA, input logic [2:0] AddrSrc1, AddrSrc2,
AddrDest, input logic isExternal, clock, pushButn, rst, input logic [1:0] ALUSel, output logic
[3:0] Res );
    logic [3:0] data, rdd1, rdd2;
    logic pulse;
    twoToOneMux writeDataSelect( isExternal, EXTDATA, Res, data );
    debouncer dbcr( clock, pushButn, pulse );
    RegisterFile regfile( clock, pulse, rst, AddrSrc1, AddrSrc2, AddrDest, data, rdd1, rdd2 );
    ALU alu( rdd1, rdd2, ALUSel, Res );
endmodule
```

```
module testbenchTopModule();
    logic[3:0] extData;
    logic isExternal;
    logic[2:0] AddrSrc1;
    logic[2:0] AddrSrc2;
    logic[2:0] AddrDest;
    logic pushButn;
    logic clk;
    logic[1:0] ALUSel;
    logic reset;
    logic[3:0] ALUOutput;


    topModule pathmodule(extData, AddrSrc1, AddrSrc2, AddrDest, isExternal, clk, pushButn, reset, ALUSel, ALUOutput);
    initial
        clk = 1;


    always
        begin
        #1;


        clk = ~clk;


        end


    initial begin
    reset = 1; #10;
    reset = 0; #10;
    extData = 1; isExternal = 1; AddrSrc1 = 1; AddrSrc2 = 1; AddrDest = 0;  pushButn = 0; ALUSel = 2'b00; #10;
    extData = 3; isExternal = 1; AddrSrc1 = 1; AddrSrc2 = 1; AddrDest = 1;  pushButn = 1; ALUSel = 2'b01; #10;
```

```
    extData = 2; isExternal = 1; AddrSrc1 = 1; AddrSrc2 = 1; AddrDest = 2;  pushButn = 0; ALUSel = 2'b10; #10;

    extData = 0; isExternal = 1; AddrSrc1 = 1; AddrSrc2 = 1; AddrDest = 3;  pushButn = 1; ALUSel = 2'b11; #10;


    end

endmodule
```