

Digital design Lab 5

- Piyush Arora (B20CS086)

1. ALU (32 bit)

Code -

```
`timescale 1ns / 1ps
module alu(a,b,op,result);

input [31:0] a,b;
input[2:0] op;
output reg [31:0] result;

always @(a,b,op,result)
begin
    case(op)
        0 : result = 0;
        1 : result = a+b;
        2 : result = a-b;
        3 : result = a<<1;
        4 : result = a>>1;
        5 : result = a&b;
        6 : result = a|b;
        7 : result = a^b;
    endcase
end

endmodule
```

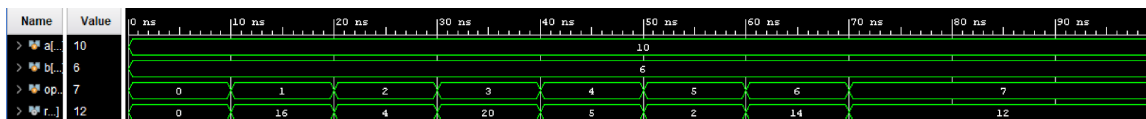
Test Bench -

```
timescale 1ns / 1ps
module testBench;
    reg [31:0] a,b;
    reg [2:0] op;
    wire [31:0] result;

    alu abc (a,b,op,result);

    initial
        begin
            a = 31'b00000000000000000000000000001010;
            b = 31'b00000000000000000000000000000110;
            op = 0;
            #10 op = 1;
            #10 op = 2;
            #10 op = 3;
            #10 op = 4;
            #10 op = 5;
            #10 op = 6;
            #10 op = 7;
        end
    initial #100 $finish;
endmodule
```

Waveform -



2. BCDAdder

Code-

```
timescale 1ns / 1ps

module bcdAdder(input [3:0]a,input [3:0]b,input carryInput,output reg [3:0]sum,output reg carry);

    reg [4:0] sumTemp;

    always @(a,b,carryInput)
    begin
        sumTemp = a+b+carryInput;
        if(sumTemp > 9)
            begin
                sumTemp = sumTemp+6;
                carry = 1;
                sum = sumTemp[3:0];
            end
        else
            begin
                carry = 0;
                sum = sumTemp[3:0];
            end
        end
    end

endmodule
```

Testbench-

```

`timescale 1ns / 1ps

module testBcdAdder;

    reg [3:0] a,b;
    reg carryInput;
    wire [3:0] sum;
    wire carry;
    bcdAdder bcd(a,b,carryInput,sum,carry);
    initial
        begin

            a = 4'b0000;  b = 4'b0000;  carryInput = 1'b0;

            #20 a = 4'b0011;  b = 4'b0010;
            #20 a = 4'b0100;  b = 4'b0011;
            #20 a = 4'b0110;  b = 4'b1000;
            #20 a = 4'b1011;  b = 4'b0010;
            #20 a = 4'b1000;  b = 4'b0101;
        end

    initial #100 $finish;

endmodule

```

Waveform-

Name	Value	0 ns	20 ns	40 ns	60 ns	80 ns
> a...	11	0	3	4	6	11
> b...	2	0	2	3	8	2
> c...	0					
> s...	3	0	5	7	4	3
> car	1					

3. BCDSubtractor

Code-

```

`timescale 1ns / 1ps

module bcdSubtractor(a,b,c,s);

input [3:0] a;
input [3:0] b;
output [3:0] c;
output s;
    wire [3:0] tenscomp, tsum, tsum2;
    wire ct;
    tensComp t(b, tenscomp);
    bcdAdder A(a, tenscomp, 1'b0, tsum, ct);
    assign s = !ct;
    tensComp t2(tsum, tsum2);
    assign c = s?tsum2:tsum;
endmodule

module tensComp(a,b);

input [0:3]a;
output [0:3]b;

assign b = 10 -a;

endmodule

module bcdAdder(input [3:0]a,input [3:0]b,input carryInput,output reg [3:0]sum,output reg carry);...e

```

Testbench-

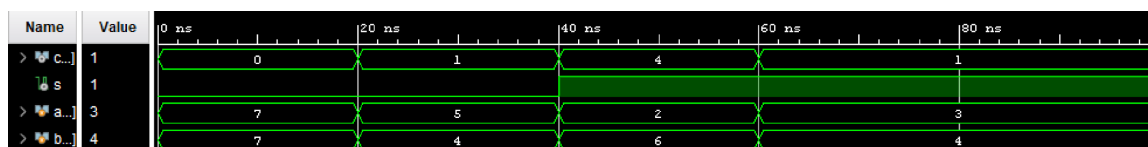
```

`timescale 1ns / 1ps
module test_bench;
wire [3:0] c;
wire s;
reg[3:0] a, b;

bcdSubtractor bl(a, b, c, s);
initial
    begin
        a = 7; b=7;
        #20 a = 5; b = 4;
        #20 a = 2; b = 6;
        #20 a = 3; b = 4;
    end
initial #100 $finish;
endmodule

```

Waveform-



4. Multiply by 5

Code-

```

`timescale 1ns / 1ps

module testBench;

reg [31:0] a;
wire [31:0] b;

multiplyByFive mbfive(a,b);

initial
begin
    a = 32'b000000000000000000000000000000;
    #10 a = 32'b000000000000000000000000000010000;
    #10 a = 32'b000000000000000000000000000001100;
    #10 a = 32'b000000000000000000000000000000010;
    #10 a = 32'b00000000000000000000000000000101000;
end
initial #50 $finish;

endmodule

```

Testbench-

```

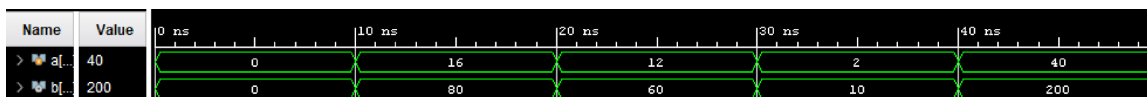
`timescale 1ns / 1ps
module multiplyByFive(a, b);
input [31:0] a;
output [31:0] b;

assign b = (a<<2) + a;

endmodule

```

Waveform-



5. Priority Encoder using casex

Code-

```

`timescale 1ns / 1ps
module fourLine2(d,x,v);
input [3:0] d;
output reg [1:0] x;
output reg v;

always@(d)
begin
assign v = 1 ;
casex(d)
4'b1xxx : x = 2'b11;
4'b01xx : x = 2'b10;
4'b001x : x = 2'b01;
4'b0001 : x = 2'b00;
default :
begin
x = 2'bxx;
assign v = 0;
end
endcase
end
endmodule

```

TestBench-

```

`timescale 1ns / 1ps

module testBench;

reg [3:0]d;
wire [1:0]x;
wire v;

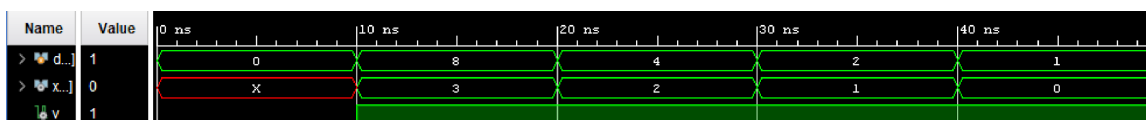
fourLine2 abc(d,x,v);

initial
begin
d = 4'b0000;
#10 d = 4'b1000;
#10 d = 4'b0100;
#10 d = 4'b0010;
#10 d = 4'b0001;
end
initial #50 $finish;

endmodule

```

Waveform-



6. Priority Encoder using for loop

Code-

```
`timescale 1ns / 1ps

module priorityEncoder(d, x, v);

input [3:0] d;
output reg [1:0] x;
output reg v;
integer k = 0;
always@(d)
begin
    x = 2'bxx;
    v = 0;
    for(k=0;k<4;k=k+1)
        if (d[k])
            begin
                x = k;
                v = 1;
            end
        end
    end

endmodule
```

TestBench-

```
`timescale 1ns / 1ps
| module testBench;

    reg [3:0] d;

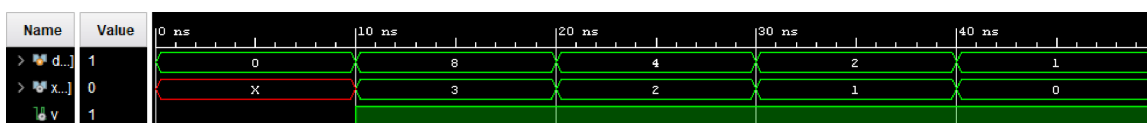
    wire [1:0] x;
    wire v;

    priorityEncoder pe(d,x,v);

| initial
| begin
    d = 4'b0000;
    #10 d = 4'b1000;
    #10 d = 4'b0100;
    #10 d = 4'b0010;
    #10 d = 4'b0001;
| end
    initial #50 $finish;

| endmodule
```

Waveform-



7. Barrel Shifter

Code-

```
`timescale 1ns / 1ps

module barrel_shifter(w,s,y);
    input [3:0] w;
    input [1:0] s;
    output [3:0] y;

    mux m1(w[3],w[0],w[1],w[2],y[3],s);
    mux m2(w[2],w[3],w[0],w[1],y[2],s);
    mux m3(w[1],w[2],w[3],w[0],y[1],s);
    mux m4(w[0],w[1],w[2],w[3],y[0],s);

endmodule
```

code mux-


```

module mux(i1,i2,i3,i4,op,control);
input i1,i2,i3,i4;
input [1:0] control;
output reg op;

always @*
begin
    case(control)
        2'b00: op = i1;
        2'b01: op = i2;
        2'b10: op = i3;
        2'b11: op = i4;
        default : op=1'bz;
    endcase
end

endmodule

```

testbench-

```




`timescale 1ns / 1ps
module test_bench;
wire [3:0] y;
reg [3:0] w;
reg [1:0] s;

barrel_shifter bs(w,s,y);

initial
begin
    w=5;s=0;
    #50 w=4; s = 1;
    #50 w=5; s = 2;
    #50 w=5; s = 3;
    #50 w=6; s = 0;
    #50 w=7; s = 1;
    #50 w=4; s = 2;
    #50 w=7; s = 3;
end
endmodule

```

waveform-

Name	Value	0 ns	200 ns
>  y[3:0]	1110	0101 0010 0101 1010	0110 1011 0001
>  w[3:0]	0111	0101 0100 0101	0110 0111 0100
>  s[1:0]	11	00 01 10 11	00 01 10

