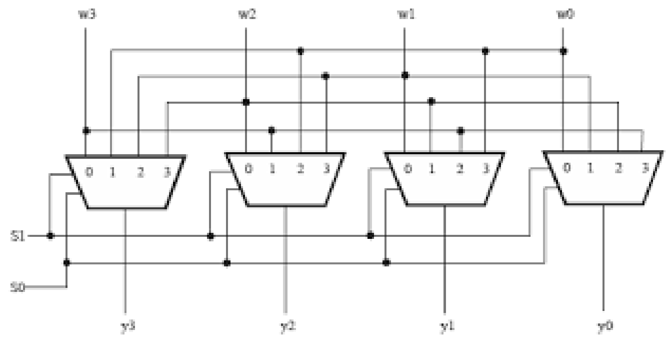
LAB REPORT-B20EE016

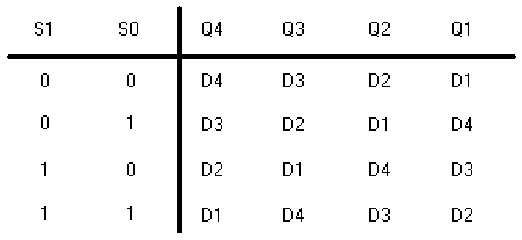
Digital Design - EEL2020

Dhruv

B20EE016

Barrel Shifter



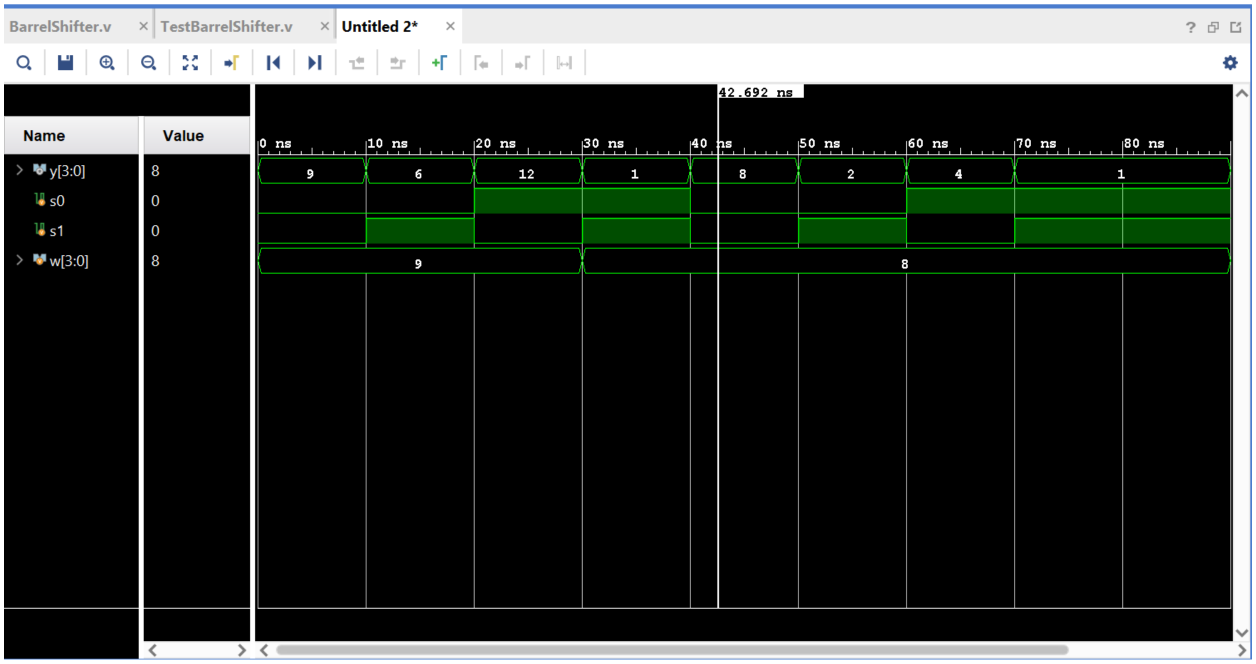


S1	S0	Q4	Q3	Q2	Q1
0	0	D4	D3	D2	D1
0	1	D3	D2	D1	D4
1	0	D2	D1	D4	D3
1	1	D1	D4	D3	D2

Code

```
`timescale 1ns / 1ps

module Mux(
input a,b,c,d,
input s0, s1,
output out);
assign out = s1 ? (s0 ? d : c) : (s0 ? b : a);
endmodule



module BarrelShifter(
input [3:0] w,

input s0,s1,
output [3:0] y
    );

Mux M1(w[3], w[0], w[1], w[2], s0, s1, y[3]);
Mux M2(w[2], w[3], w[0], w[1], s0, s1, y[2]);
Mux M3(w[1], w[2], w[3], w[0], s0, s1, y[1]);
Mux M4(w[0], w[1], w[2], w[3], s0,s1, y[0]);
endmodule
```

Simulation

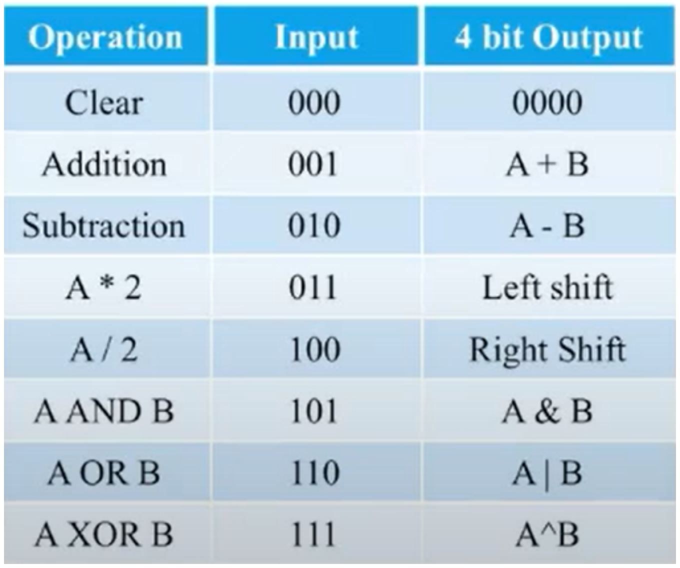


Test Bench

```
`timescale 1ns / 1ps
module TestBarrelShifter;

wire [3:0]y;
reg s0,s1;
reg [3:0]w;

BarrelShifter B(w,s0,s1,y);
initial
begin
assign w = 4'b1001;
s0 = 1'b0; s1 = 1'b0;
#10 s0 = 1'b0; s1 = 1'b1;
#10 s0 = 1'b1; s1 = 1'b0;
#10 s0 = 1'b1; s1 = 1'b1;
assign w = 4'b1000;
```

```
#10 s0 = 1'b0; s1 = 1'b0;
#10 s0 = 1'b0; s1 = 1'b1;
#10 s0 = 1'b1; s1 = 1'b0;
#10 s0 = 1'b1; s1 = 1'b1;
end
initial #90 $finish;
endmodule
```

ALU



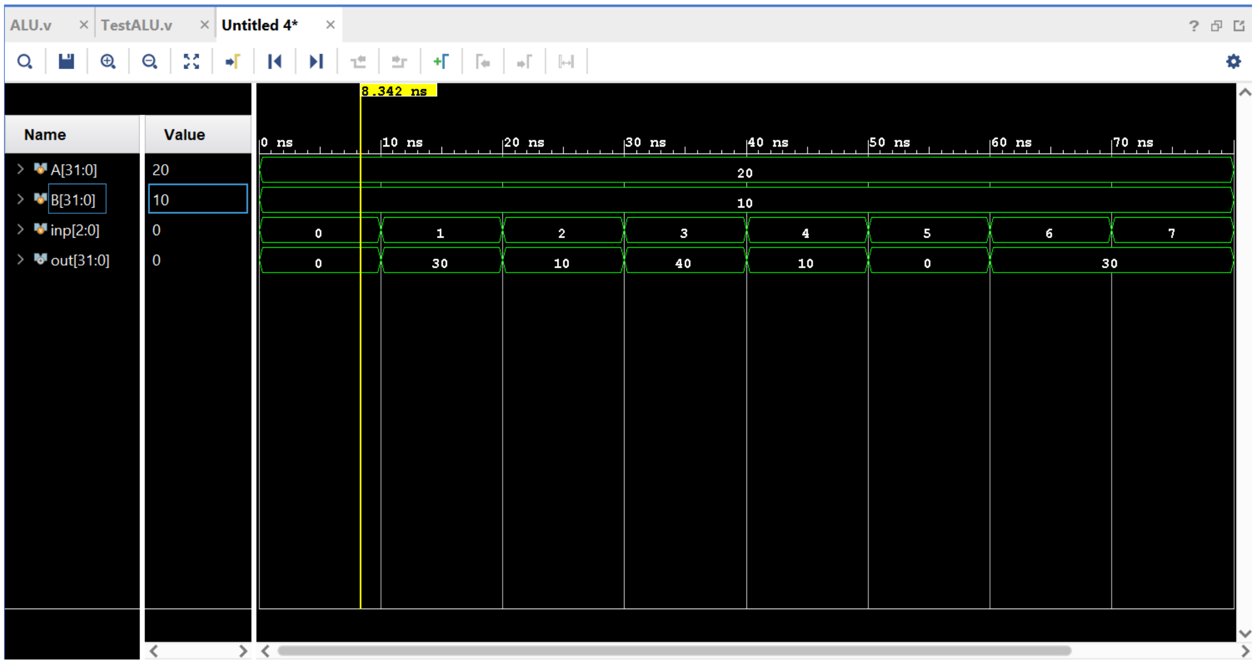
Operation	Input	4 bit Output
Clear	000	0000
Addition	001	A + B
Subtraction	010	A - B
A * 2	011	Left shift
A / 2	100	Right Shift
A AND B	101	A & B
A OR B	110	A | B
A XOR B	111	A^B

Code

```
`timescale 1ns / 1ps

module ALU(
input [31:0] A,B,
input [2:0] inp,
output reg [31:0] out

    );


always @(A,B,inp)
    begin
        case(inp)

            3'b000: out= 0;
```

```
              3'b001: out = A+B;

              3'b010: out= A - B;

              3'b011: out = A << 1;

              3'b100: out = A>> 1;

              3'b101: out = A & B;

              3'b110: out = A | B;

              3'b111: out = A^B;
          endcase
      end
  endmodule
```

Simulation



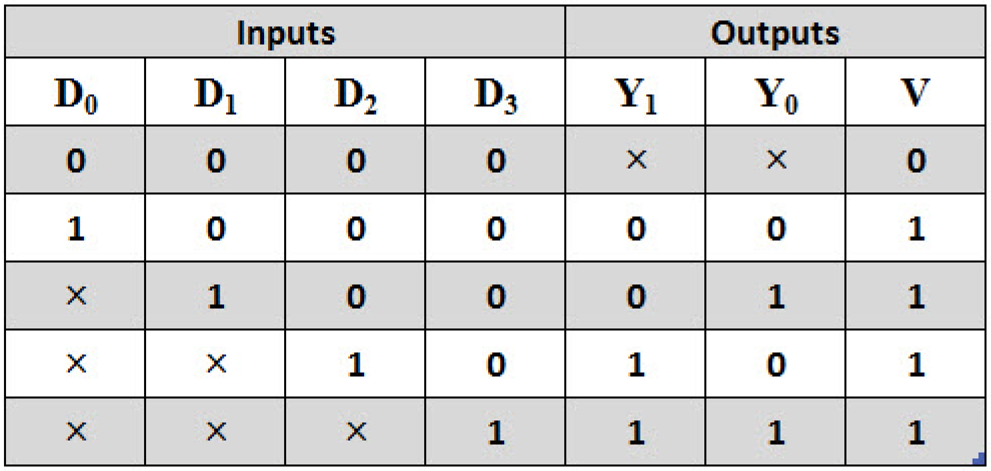
Name	Value
A[31:0]	20
B[31:0]	10
inp[2:0]	0
out[31:0]	0

Test Bench

```
`timescale 1ns / 1ps

module TestALU();
```

```
reg [31:0] A,B;
reg  [2:0] inp;
wire [31:0] out;

ALU A1(A,B,inp,out);
initial
    begin
        assign A =20;
        assign B = 10;
        inp = 0; #10;
        inp = 1; #10;
        inp = 2; #10;
        inp = 3; #10;
        inp = 4; #10;
        inp = 5; #10;
        inp= 6; #10;
        inp = 7; #10;
    end

initial #80 $finish;
endmodule
```

Priority Encoder (Case Statement)

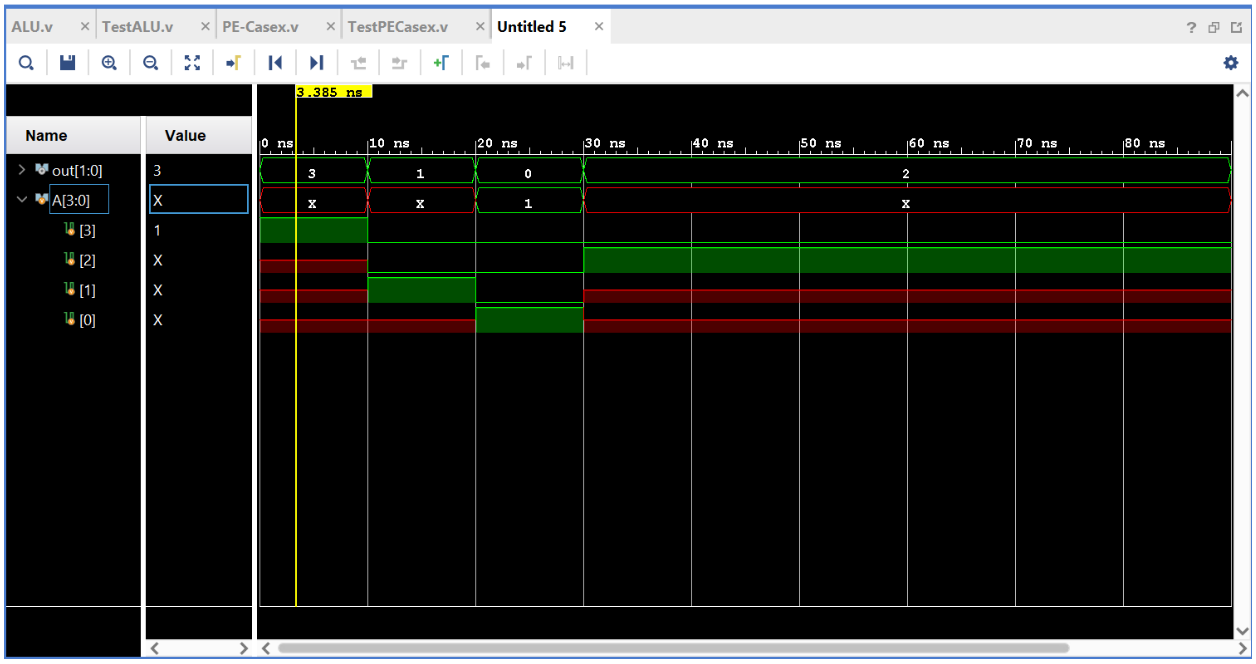


Inputs				Outputs		
D_0	D_1	D_2	D_3	Y_1	Y_0	V
0	0	0	0	×	×	0
1	0	0	0	0	0	1
×	1	0	0	0	1	1
×	×	1	0	1	0	1
×	×	×	1	1	1	1

Code

```
`timescale 1ns / 1ps

module PECasex(
input [3:0] A,
output reg [1:0] out
);


always@(A)
    begin
    case(A)
        4'b1XXX:out=2'b11;
        4'b01XX:out=2'b10;
        4'b001X:out=2'b01;
        4'b0001:out=2'b00;
        default:out=2'bX;

        endcase
    end
endmodule
```

Simulation



Test-Bench

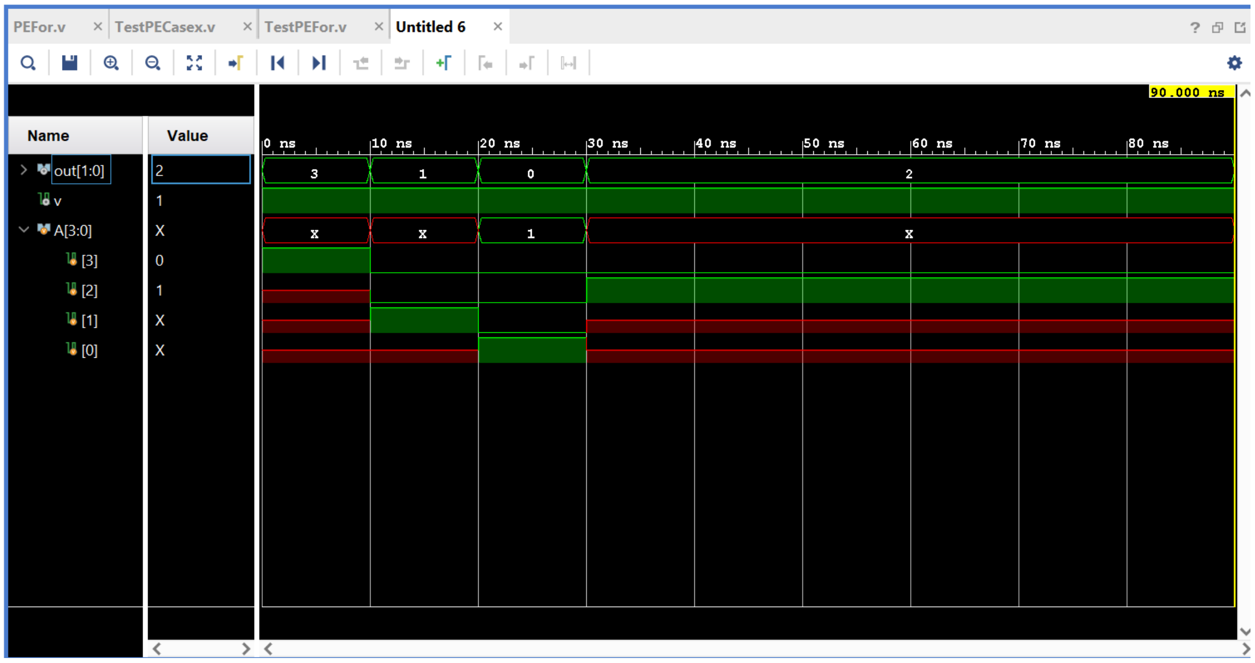
```
`timescale 1ns / 1ps
module TestPECasex();
    wire[1:0]out;
    reg[3:0]A;
    PECasex PE1(A,out);
    initial
    begin
        A=4'b0000;
        A = 4'b1XXX; #10;
        A = 4'b001X; #10;
        A = 4'b0001; #10;

        A = 4'b01XX; #10;

    end
 initial #90 $finish;
endmodule
```

Priority Encoder (For loop)

Code

```
`timescale 1ns / 1ps
module PEFor(
    input [4:0]A,
    output reg v,
    output reg [1:0]out
    );

    integer i;
    always @(A)
    begin
        out = 2'bxx;
        v = 0;
        for(i = 0;i<4; i=i+1)
        if(A[i])
        begin
        out = i;
        v = 1;
        end
    end
endmodule
```

Simulation

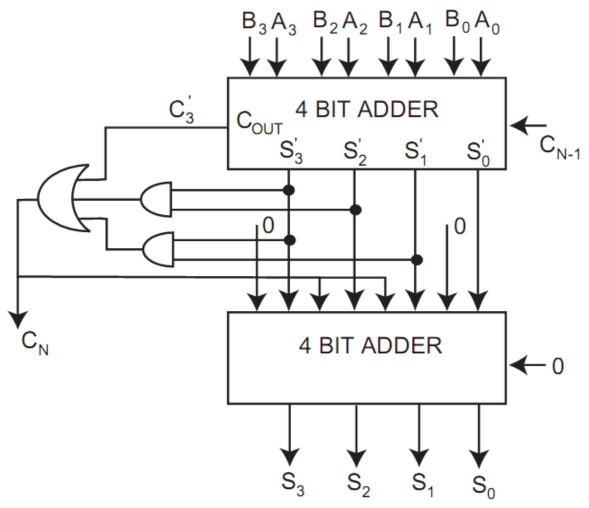


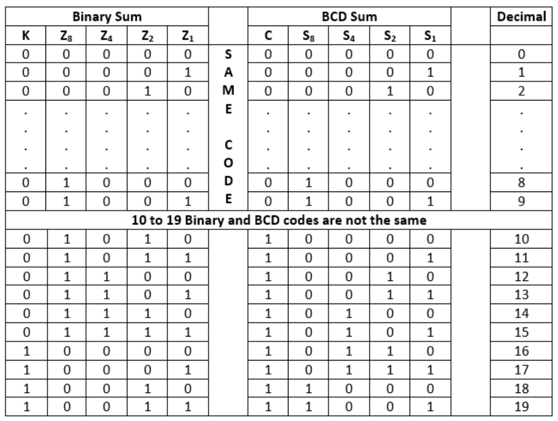
Name	Value
out[1:0]	2
v	1
A[3:0]	X
[3]	0
[2]	1
[1]	X
[0]	X

Test-Bench

```
`timescale 1ns / 1ps
module TestPEFor();
     wire[1:0]out;
     wire v;
     reg[3:0]A;
    PEFor PE1(A,v,out);
    initial
    begin
        A=4'b0000;
        A = 4'b1XXX; #10;
        A = 4'b001X; #10;
        A = 4'b0001; #10;

        A = 4'b01XX; #10;

    end
 initial #90 $finish;
endmodule
```

BCD Adder



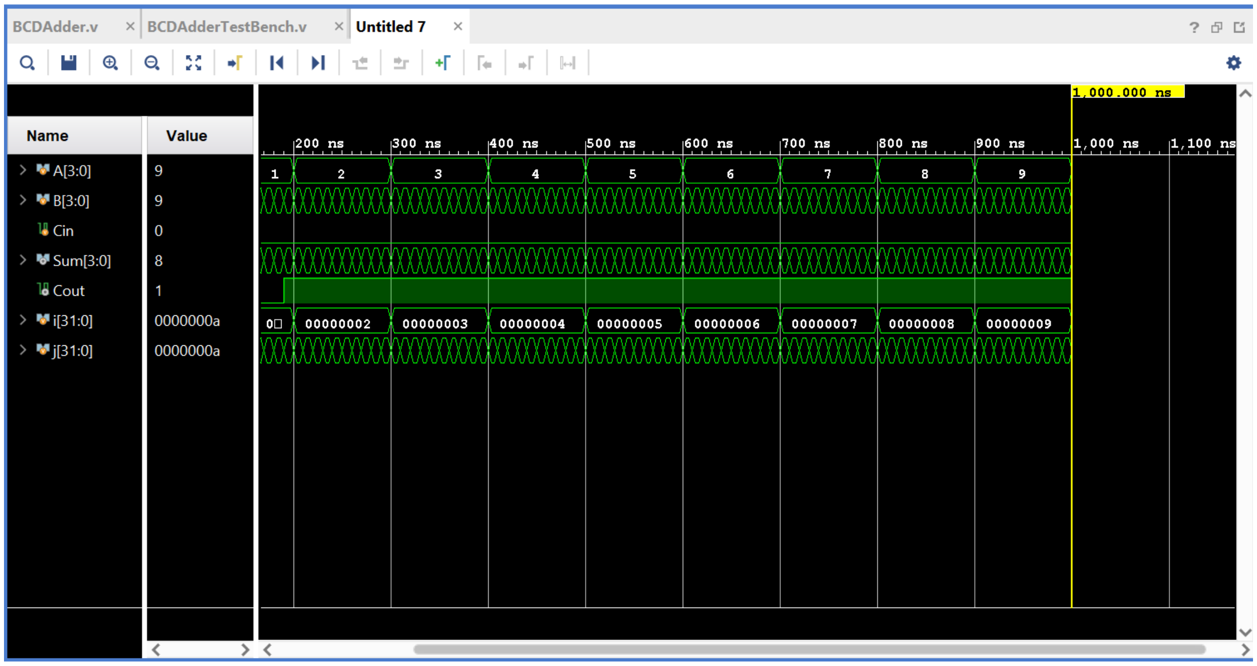


Binary Sum						BCD Sum						Decimal
K	Z_8	Z_4	Z_2	Z_1		C	S_8	S_4	S_2	S_1		
0	0	0	0	0	S	0	0	0	0	0		0
0	0	0	0	1	A	0	0	0	0	1		1
0	0	0	1	0	M	0	0	0	1	0		2
. . . .	. . . .	. . . .	. . . .	. . . .	E C O	. . . .	. . . .	. . . .	. . . .	. . . .		. . . .
0	1	0	0	0	D	0	1	0	0	0		8
0	1	0	0	1	E	0	1	0	0	1		9
10 to 19 Binary and BCD codes are not the same												
0	1	0	1	0		1	0	0	0	0		10
0	1	0	1	1		1	0	0	0	1		11
0	1	1	0	0		1	0	0	1	0		12
0	1	1	0	1		1	0	0	1	1		13
0	1	1	1	0		1	0	1	0	0		14
0	1	1	1	1		1	0	1	0	1		15
1	0	0	0	0		1	0	1	1	0		16
1	0	0	0	1		1	0	1	1	1		17
1	0	0	1	0		1	1	0	0	0		18
1	0	0	1	1		1	1	0	0	1		19

Code

```
`timescale 1ns / 1ps


module BCDAdder(
input [3:0] A,B,
input Cin,
output reg [3:0] sum,
output reg Cout
);
reg [4:0] sum1;
reg [3:0] sum2;

always @(A,B, Cin)
begin
    sum1 = A+B+Cin;
    if(sum1 > 9)
        begin
            sum1 = sum1+6;
            assign Cout = 1;
            sum = sum1;
        end
    else
        begin
            Cout = 0;
            sum = sum1;
        end
end

endmodule
```

Simulation



Test-Bench

```
`timescale 1ns / 1ps


module BCDAdderTestBench();
reg[3:0]A,B;
reg Cin;
wire [3:0]Sum;
wire Cout;
integer i,j;
BCDAdder B11(A,B,Cin,Sum,Cout);
initial
begin
A = 0;
B = 0;
Cin=1'b0;

for(i=0;i<10;i=i+1) begin
for(j=0;j<10;j=j+1) begin
A = i;
B = j;
#10;
end end
end
```

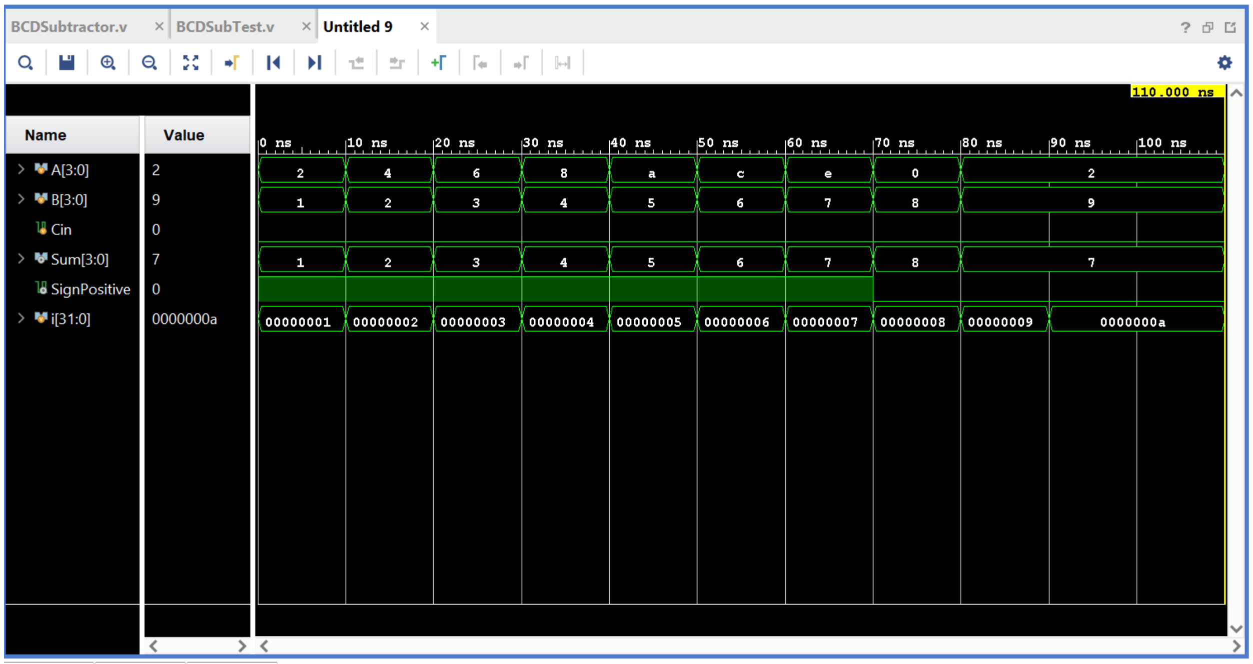
```
//initial # $finish;
endmodule
```

BCD Subtractor

Code

```
`timescale 1ns / 1ps
module BCDSubtractor(
    input [3:0]A,B,
    input Cin,
    output reg [3:0]out,
    output reg Postive
    );

    always @(A,B)
    if(A>=B)
    begin
    out=A-B;
    Postive = 1;
    end
    else
    begin
    out= B-A;
    Postive = 0;
    end
endmodule
```

Simulation



Test-Bench

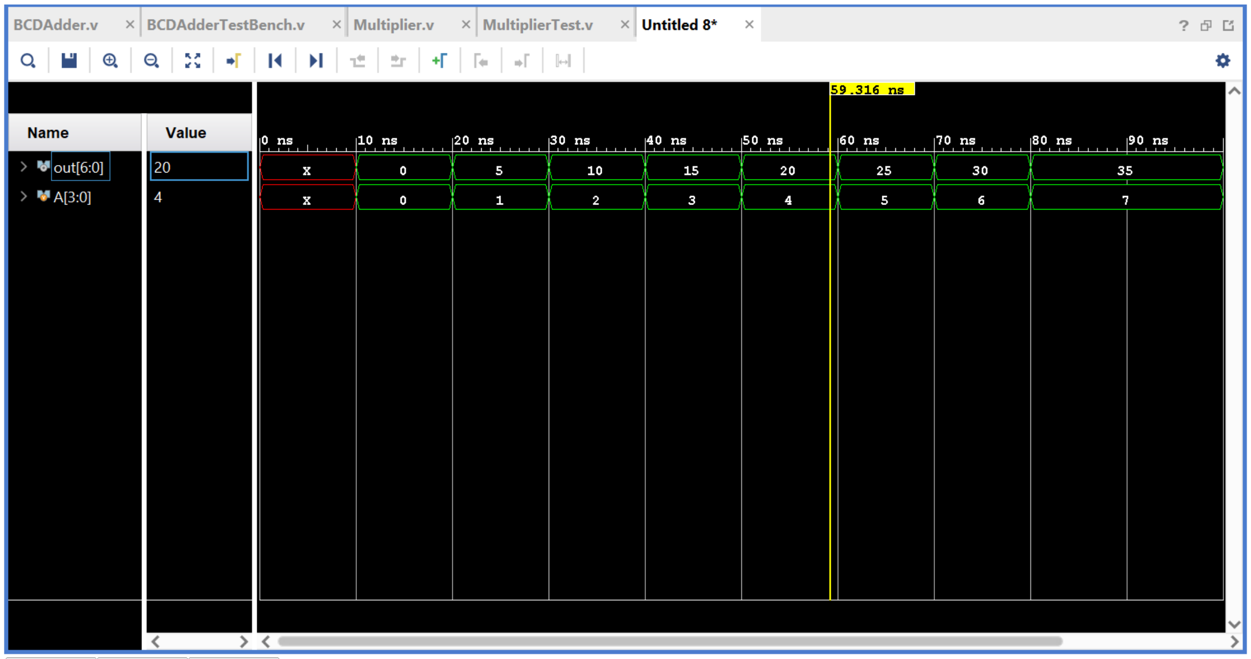
```
`timescale 1ns / 1ps
module BCDSubtractorTest();
    reg[3:0]A,B;
    reg Cin;
    wire [3:0]Sum;
    wire SignPositive;
    integer i;
    BCDSubtractor B11(A,B,Cin,Sum,SignPositive);
    initial
    begin
    for (i = 1; i<10;i=i+1)begin
    A=2*i;
    B=i;
    Cin=0;
    #10;
    end
    end
    initial #110 $finish;
endmodule
```

Multiplier

Code

```
`timescale 1ns / 1ps
module Multiplier(
input [3:0]A,
output [6:0] out
    );

assign out= (A << 2) +A;
endmodule
```

Simulation



Test-Bench

```
`timescale 1ns / 1ps

module MultiplierTest();
wire[6:0]out;
reg[3:0]A;
Multiplier M1(A,out);
initial
begin
```

```
    #10 A=4'b0000;
    #10 A=4'b0001;
    #10 A=4'b0010;
    #10 A=4'b0011;
    #10 A=4'b0100;
    #10 A=4'b0101;
    #10 A =4'b0110;
    #10 A=4'b0111;
end
initial #100 $finish;
endmodule
```