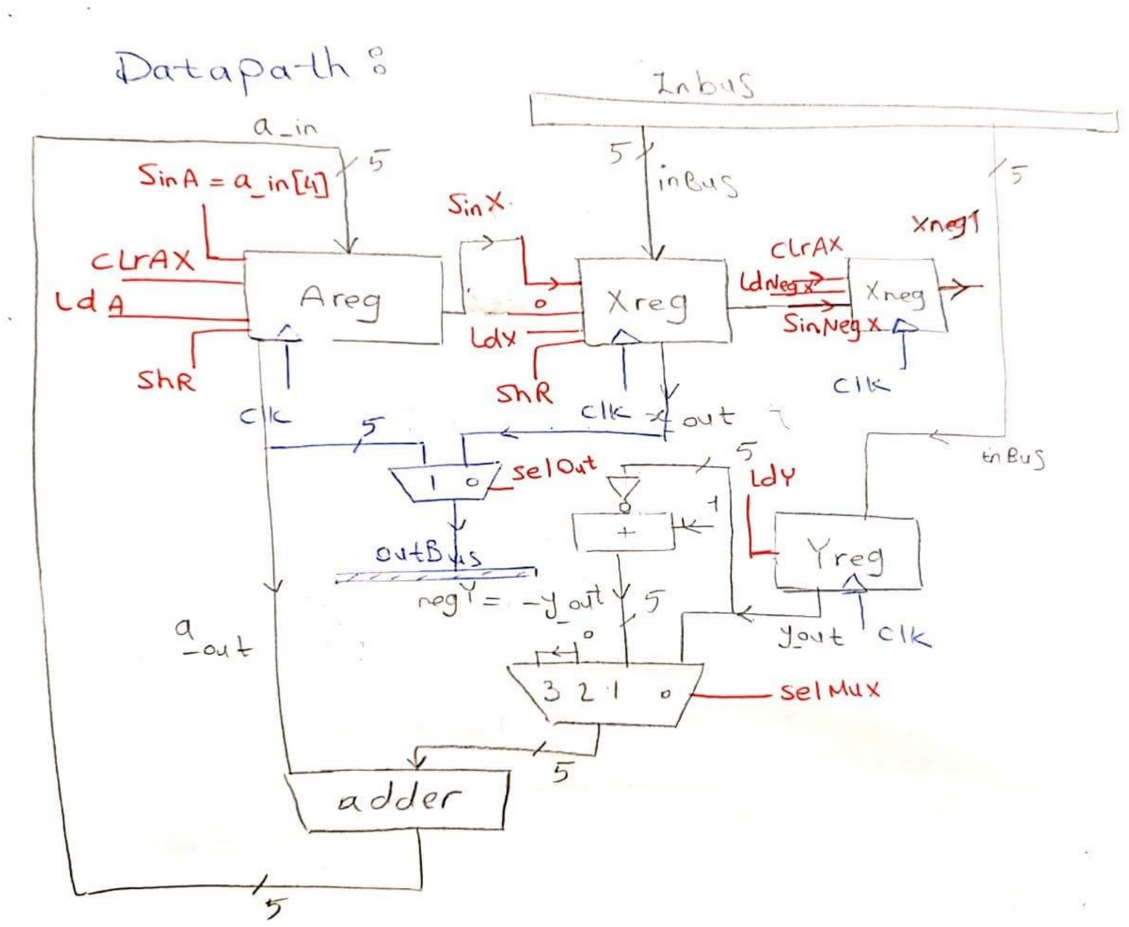
Computer Assignment #1 (Booth Multiplier)

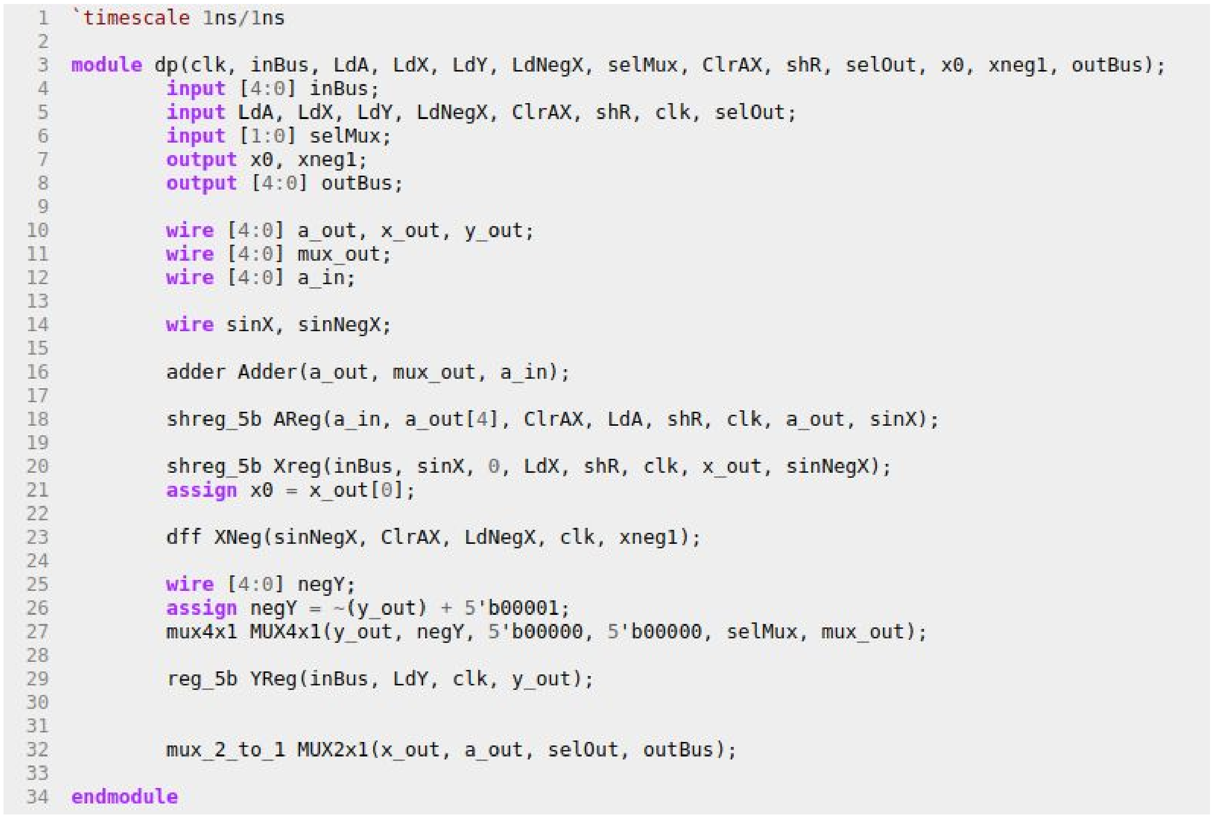
Helia Hoseini (810197491)
Daneshvar Amrollahi (810197685)

Datapath:

Block Diagram:



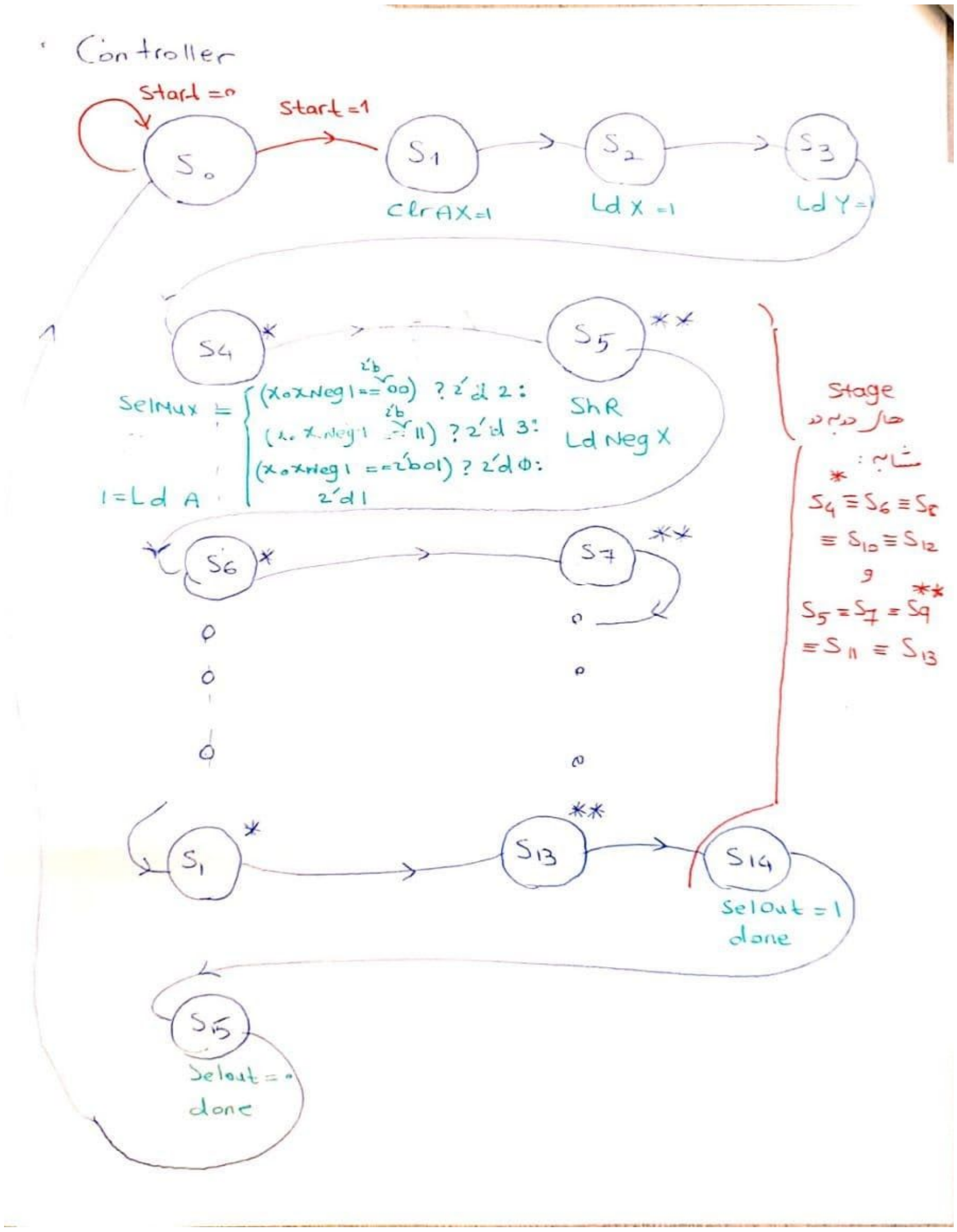
Verilog Code:



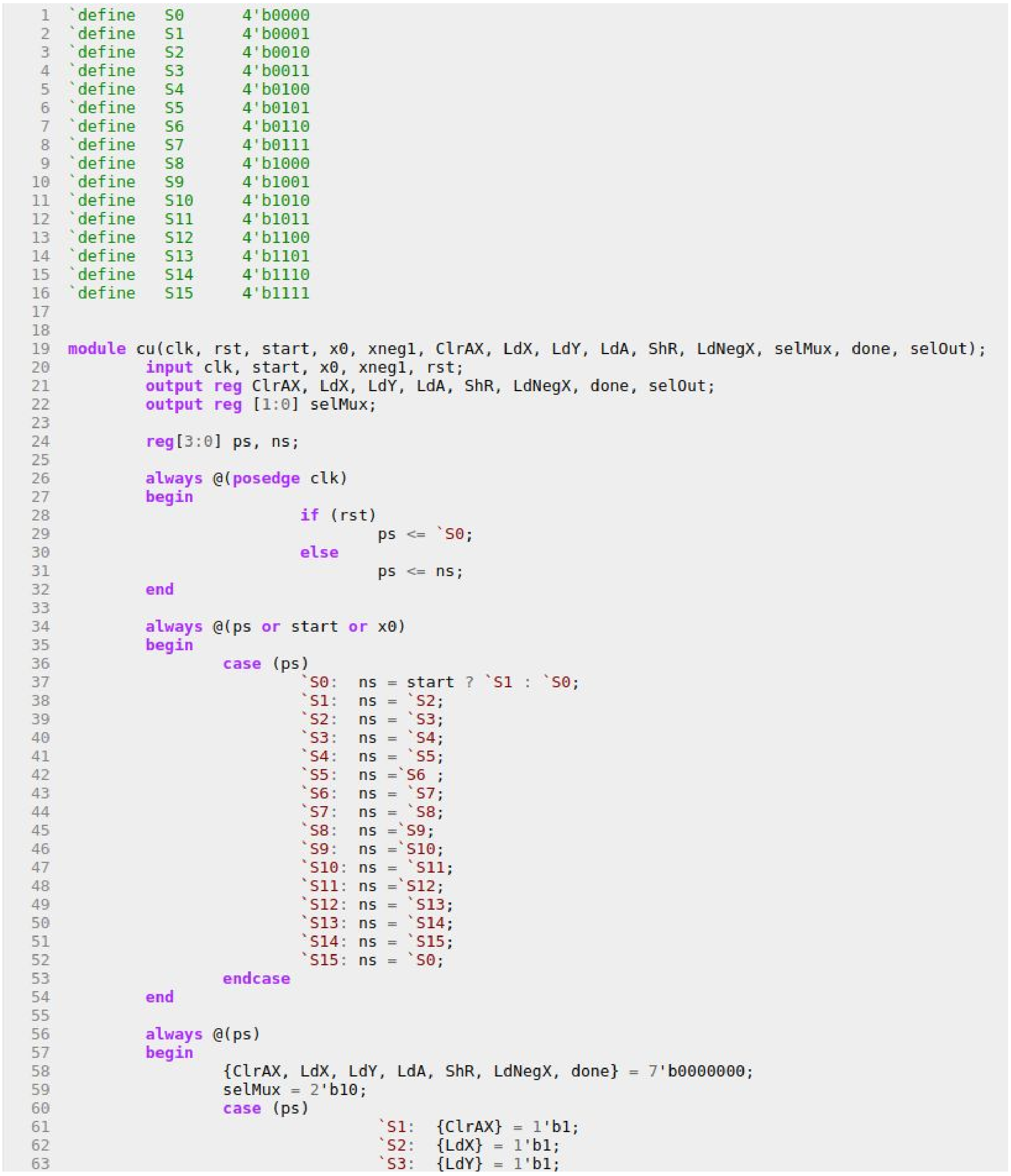
```
`timescale 1ns/1ns

module dp(clk, inBus, LdA, LdX, LdY, LdNegX, selMux, ClrAX, shR, selOut, x0, xneg1, outBus);
        input [4:0] inBus;
        input LdA, LdX, LdY, LdNegX, ClrAX, shR, clk, selOut;
        input [1:0] selMux;
        output x0, xneg1;
        output [4:0] outBus;

        wire [4:0] a_out, x_out, y_out;
        wire [4:0] mux_out;
        wire [4:0] a_in;

        wire sinX, sinNegX;

        adder Adder(a_out, mux_out, a_in);

        shreg_5b AReg(a_in, a_out[4], ClrAX, LdA, shR, clk, a_out, sinX);

        shreg_5b Xreg(inBus, sinX, 0, LdX, shR, clk, x_out, sinNegX);
        assign x0 = x_out[0];

        dff XNeg(sinNegX, ClrAX, LdNegX, clk, xneg1);

        wire [4:0] negY;
        assign negY = ~(y_out) + 5'b00001;
        mux4x1 MUX4x1(y_out, negY, 5'b00000, 5'b00000, selMux, mux_out);

        reg_5b YReg(inBus, LdY, clk, y_out);


        mux_2_to_1 MUX2x1(x_out, a_out, selOut, outBus);

endmodule
```

Controller

State Machine



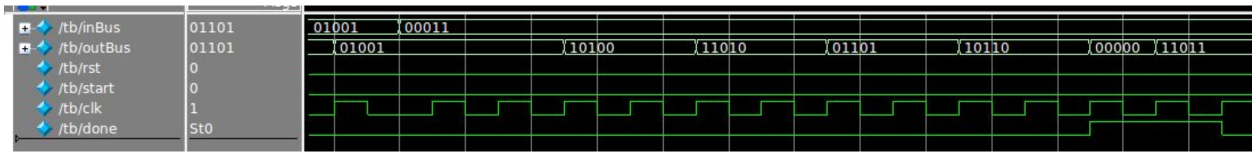
Verilog Code



```verilog
`define   S0      4'b0000
`define   S1      4'b0001
`define   S2      4'b0010
`define   S3      4'b0011
`define   S4      4'b0100
`define   S5      4'b0101
`define   S6      4'b0110
`define   S7      4'b0111
`define   S8      4'b1000
`define   S9      4'b1001
`define   S10     4'b1010
`define   S11     4'b1011
`define   S12     4'b1100
`define   S13     4'b1101
`define   S14     4'b1110
`define   S15     4'b1111


module cu(clk, rst, start, x0, xneg1, ClrAX, LdX, LdY, LdA, ShR, LdNegX, selMux, done, selOut);
        input clk, start, x0, xneg1, rst;
        output reg ClrAX, LdX, LdY, LdA, ShR, LdNegX, done, selOut;
        output reg [1:0] selMux;

        reg[3:0] ps, ns;

        always @(posedge clk)
        begin
                        if (rst)
                                ps <= `S0;
                        else
                                ps <= ns;
        end

        always @(ps or start or x0)
        begin
                case (ps)
                        `S0:  ns = start ? `S1 : `S0;
                        `S1:  ns = `S2;
                        `S2:  ns = `S3;
                        `S3:  ns = `S4;
                        `S4:  ns = `S5;
                        `S5:  ns =`S6 ;
                        `S6:  ns = `S7;
                        `S7:  ns = `S8;
                        `S8:  ns =`S9;
                        `S9:  ns =`S10;
                        `S10: ns = `S11;
                        `S11: ns =`S12;
                        `S12: ns = `S13;
                        `S13: ns = `S14;
                        `S14: ns = `S15;
                        `S15: ns = `S0;
                endcase
        end

        always @(ps)
        begin
                {ClrAX, LdX, LdY, LdA, ShR, LdNegX, done} = 7'b0000000;
                selMux = 2'b10;
                case (ps)
                                `S1:  {ClrAX} = 1'b1;
                                `S2:  {LdX} = 1'b1;
                                `S3:  {LdY} = 1'b1;

                                `S4:  {selMux, LdA} = {{x0, xneg1} == 2'b00 ? 2'd2:
                                                                        {x0, xneg1} == 2'b11 ? 2'd3:
                                                                        {x0, xneg1} == 2'b01 ? 2'd0:
                                                                        2'd1, 1'b1};
                                `S5:  {ShR, LdNegX} = 2'b11;



                                `S6:  {selMux, LdA} = {{x0, xneg1} == 2'b00 ? 2'd2:
                                                                        {x0, xneg1} == 2'b11 ? 2'd3:
                                                                        {x0, xneg1} == 2'b01 ? 2'd0:
                                                                        2'd1, 1'b1};

                                `S7:  {ShR, LdNegX} = 2'b11;


                                `S8:  {selMux, LdA} = {{x0, xneg1} == 2'b00 ? 2'd2:
                                                                        {x0, xneg1} == 2'b11 ? 2'd3:
                                                                        {x0, xneg1} == 2'b01 ? 2'd0:
                                                                        2'd1, 1'b1};

                                `S9:  {ShR, LdNegX} = 2'b11;

                                `S10:  {selMux, LdA} = {{x0, xneg1} == 2'b00 ? 2'd2:
                                                                        {x0, xneg1} == 2'b11 ? 2'd3:
                                                                        {x0, xneg1} == 2'b01 ? 2'd0:
                                                                        2'd1, 1'b1};

                                `S11:  {ShR, LdNegX} = 2'b11;

                                `S12:  {selMux, LdA} = {{x0, xneg1} == 2'b00 ? 2'd2:
                                                                        {x0, xneg1} == 2'b11 ? 2'd3:
                                                                        {x0, xneg1} == 2'b01 ? 2'd0:
                                                                        2'd1, 1'b1};

                                `S13:  {ShR, LdNegX} = 2'b11;

                                `S14: {selOut, done} = 2'b11;
                                `S15: {selOut, done} = 2'b01;
                endcase
        end

endmodule
```

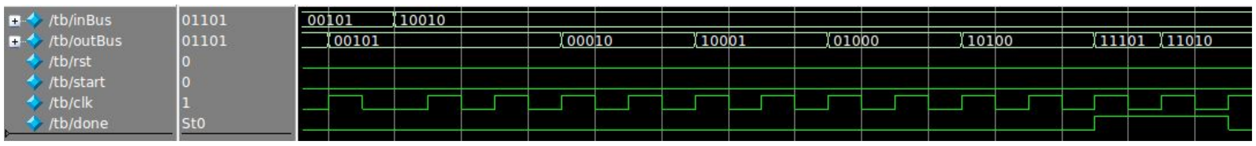
Testing

When the circuit receives a pulse on *start* signal, it inputs the multiplicand (y) and multiplier (x) during 2 clock cycles from inBus[4:0]. When the multiplication is process is finished, the *done* signal becomes high and the results is sent on outBus[4:0] during 2 clock pulses (one for each 5 bit of the result which is 10 bits)



$y = (01001)_2 = 9$, $x = (00011)_2 = 3$

$$y \times x \;=\; 27 \;=\; 0000011011$$



$y = (00101)_2 = 5$, $x = (10010)_2 = -14$

$y \times x \;=\; -70 \;=\; 1110111010$ (with sign extension)

```
`timescale 1ns/1ns;

module tb();
        reg [4:0] inBus;
        wire [4:0] outBus;
        reg rst, start, clk;
        wire done;

        booth5b MUT(inBus, clk, rst, start, outBus, done);

        initial begin
        repeat(5) begin
                rst = 1; #10;
                clk = 1; #10;
                rst = 0; #10;
                clk = 0; #10;

                start = 1; #10;
                clk = 1; #10;
                clk = 0; #10;

                start = 0;
                clk = 1; #10;
                clk = 0; #10;

                inBus = $random % 64; #10;
                clk = 1; #10;
                clk = 0; #10;

                inBus = $random % 64; #10;
                clk = 1; #10;
                clk = 0; #10;
                clk = 1; #10;
                clk = 0; #10;

                repeat(5) begin
                        clk = 1; #10;
                        clk = 0; #10;

                        clk = 1; #10;
                        clk = 0; #10;
                end
                clk = 1; #10;
                clk = 0; #10;
                clk = 1; #10;
        end

        $stop;
        end

endmodule
```

Verilog Code for test bench

This test bench is testing the *booth5b* module with 5 pair of random signed 5 bit inputs.