Experiment #2 – FPGA Realization of Radix-4 Multiplier

Sana Sari Navaei 810199435
Maryam Jafarabadi 810199549

1.RTL Design and Simulation

1.

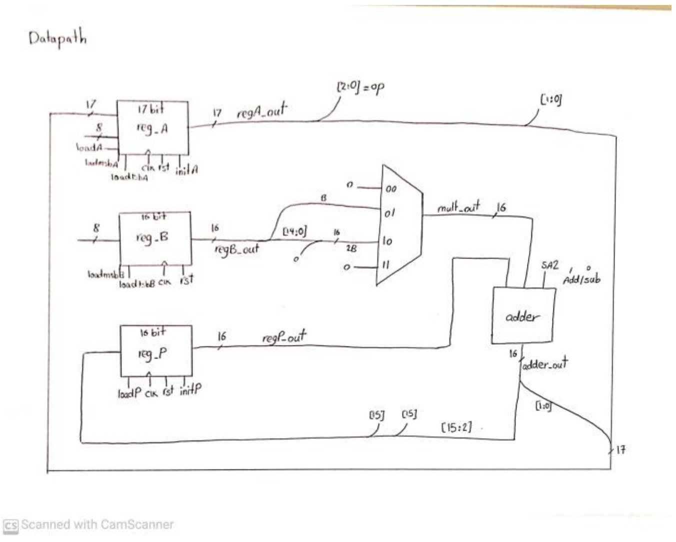


Fig. 1 Datapath of design

2.

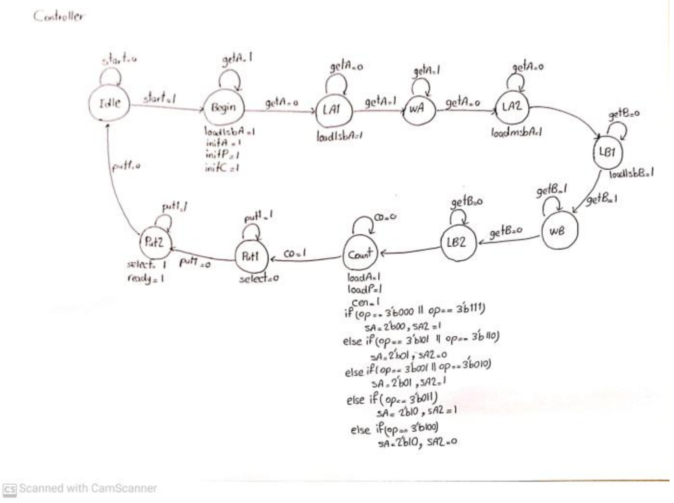
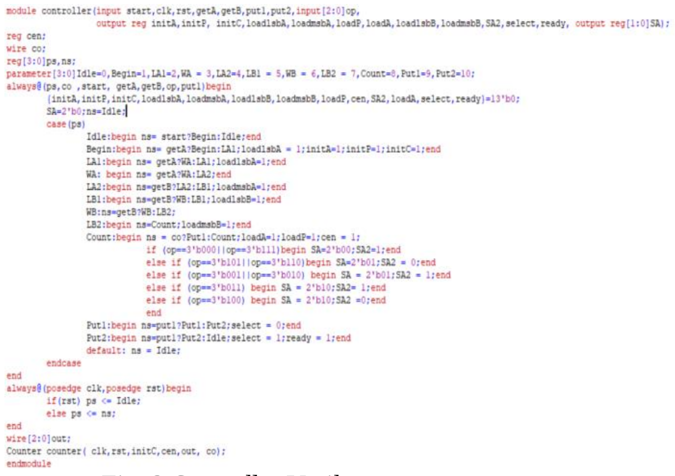


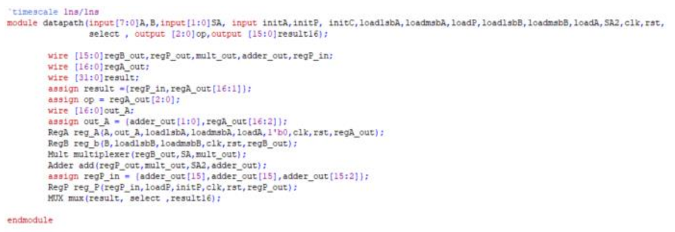
Fig. 2 State diagram of controller

3.



```
module controller(input start,clk,rst,getA,getB,put1,put2,input[2:0]op,
                  output reg initA,initP, initC,loadlsbA,loadmsbA,loadP,loadA,loadlsbB,loadmsbB,SA2,select,ready, output reg[1:0]SA);
reg cen;
wire co;
reg[3:0]ps,ns;
parameter[3:0]Idle=0,Begin=1,LA1=2,WA = 3,LA2=4,LB1 = 5,WB = 6,LB2 = 7,Count=8,Put1=9,Put2=10;
always@(ps,co ,start, getA,getB,op,put1)begin
        {initA,initP,initC,loadlsbA,loadmsbA,loadlsbB,loadmsbB,loadP,cen,SA2,loadA,select,ready}=13'b0;
        SA=2'b0;ns=Idle;
        case(ps)
                Idle:begin ns= start?Begin:Idle;end
                Begin:begin ns= getA?Begin:LA1;loadlsbA = 1;initA=1;initP=1;initC=1;end
                LA1:begin ns= getA?WA:LA1;loadlsbA=1;end
                WA: begin ns= getA?WA:LA2;end
                LA2:begin ns=getB?LA2:LB1;loadmsbA=1;end
                LB1:begin ns=getB?WB:LB1;loadlsbB=1;end
                WB:ns=getB?WB:LB2;
                LB2:begin ns=Count;loadmsbB=1;end
                Count:begin ns = co?Put1:Count;loadA=1;loadP=1;cen = 1;
                        if (op==3'b000||op==3'b111)begin SA=2'b00;SA2=1;end
                        else if (op==3'b101||op==3'b110)begin SA=2'b01;SA2 = 0;end
                        else if (op==3'b001||op==3'b010) begin SA = 2'b01;SA2 = 1;end
                        else if (op==3'b011) begin SA = 2'b10;SA2= 1;end
                        else if (op==3'b100) begin SA = 2'b10;SA2 =0;end
                        end
                Put1:begin ns=put1?Put1:Put2;select = 0;end
                Put2:begin ns=put1?Put2:Idle;select = 1;ready = 1;end
                default: ns = Idle;
        endcase
end
always@(posedge clk,posedge rst)begin
        if(rst) ps <= Idle;
        else ps <= ns;
end
wire[2:0]out;
Counter counter( clk,rst,initC,cen,out, co);
endmodule
```

Fig. 3 Controller Verilog



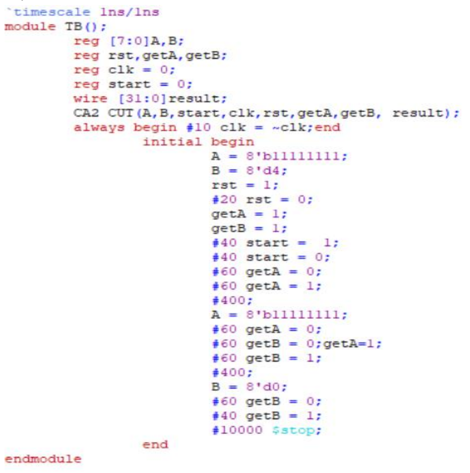
```
`timescale 1ns/1ns
module datapath(input[7:0]A,B,input[1:0]SA, input initA,initP, initC,loadlsbA,loadmsbA,loadP,loadlsbB,loadmsbB,loadA,SA2,clk,rst,
                select , output [2:0]op,output [15:0]result16);

        wire [15:0]regB_out,regP_out,mult_out,adder_out,regP_in;
        wire [16:0]regA_out;
        wire [31:0]result;
        assign result ={regP_in,regA_out[16:1]};
        assign op = regA_out[2:0];
        wire [16:0]out_A;
        assign out_A = {adder_out[1:0],regA_out[16:2]};
        RegA reg_A(A,out_A,loadlsbA,loadmsbA,loadA,1'b0,clk,rst,regA_out);
        RegB reg_b(B,loadlsbB,loadmsbB,clk,rst,regB_out);
        Mult multiplexer(regB_out,SA,mult_out);
        Adder add(regP_out,mult_out,SA2,adder_out);
        assign regP_in = {adder_out[15],adder_out[15],adder_out[15:2]};
        RegP reg_P(regP_in,loadP,initP,clk,rst,regP_out);
        MUX mux(result, select ,result16);

endmodule
```

Fig. 4 Datapath Verilog

4.



```
`timescale 1ns/1ns
module TB();
        reg [7:0]A,B;
        reg rst,getA,getB;
        reg clk = 0;
        reg start = 0;
        wire [31:0]result;
        CA2 CUT(A,B,start,clk,rst,getA,getB, result);
        always begin #10 clk = ~clk;end
                initial begin
                        A = 8'b11111111;
                        B = 8'd4;
                        rst = 1;
                        #20 rst = 0;
                        getA = 1;
                        getB = 1;
                        #40 start =  1;
                        #40 start = 0;
                        #60 getA = 0;
                        #60 getA = 1;
                        #400;
                        A = 8'b11111111;
                        #60 getA = 0;
                        #60 getB = 0;getA=1;
                        #60 getB = 1;
                        #400;
                        B = 8'd0;
                        #60 getB = 0;
                        #40 getB = 1;
                        #10000 $stop;
                end
endmodule
```

Fig. 5 Testbench Verilog

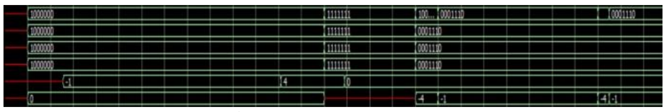


Fig. 6 Waveform result

A = 16’b1111111111111111;
B = 16’b0000000000000010;
A is -1 and B is 4 and the result is -4 as we see.

5.
We have two 16 bit numbers and we want to calculate the result of A * B. First we give the 8 least bit of A and then we give the 8 least bit of B; After that, we give the 8 most bit of A and then we give the 8 most bit of B. The result is -4 and we received it in two times. In the first time we received the 8 least bit of result which is 8’b11111100 and then we received the 8 most bit of result which is 8’b11111111.

2.FPGA Implementation

5.

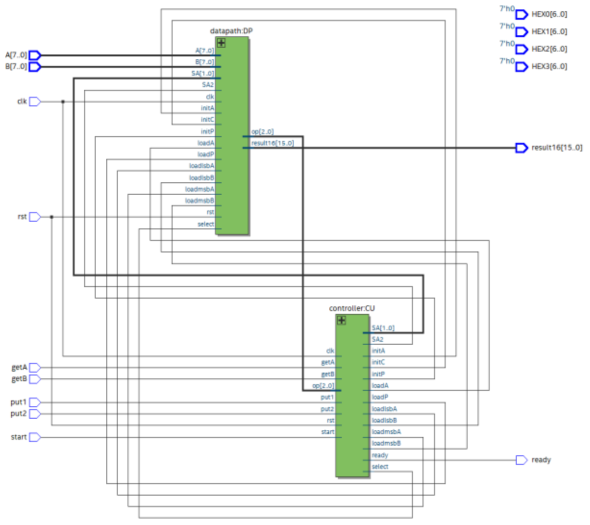


Fig. 7 Synthesized RTL

7.

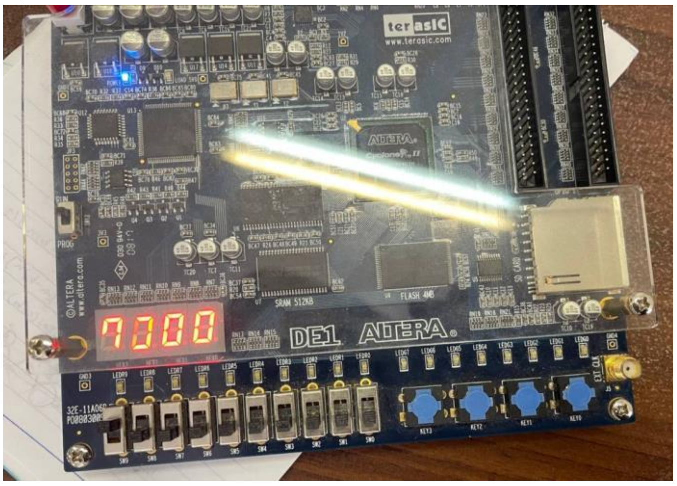


Fig. 8 FPGA output

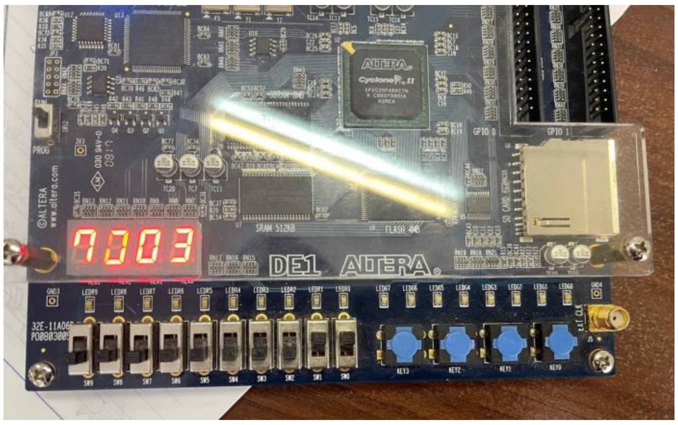


Fig. 9 FPGA output

To see the result in FPGA, at first we enter the reset button. After that, we should enter the start button. Then we get the A in two parts. At the first part, we get the most 8 bit of A which is 8’b00000000 and then we get the 8 least bit of A which is 8’b000000011 . We do the same to get another input. First we get 8’b00000000 and then we get 8’b00000001(we enter the numbers by using switches on FPGA). After getting the inputs, we should press putout button two times because we received the output in 2 parts. In the first part we received the 4 least bit of result in HEX and in the next part we get the 4 most bit of result in HEX.