ASSİGNMENT 3

Bizden 3 shift regester,1 control fsm, 1 d-filp-flop ve 1 full adder bulunan bir serial adder yapmamız istendi. İlk başta girilecek sayılar için 2 adet shift ragester tasarlayıp a ve b girişlerini bağladım, ardından 1'er bit olacak şekilde bir basamak sağ kaydırıp full adder'a full adder da toplma işlemini yaparken elimizde olacak olan eldeleri de tekrar full ader'ın içine gönderdim ardından toplama işlemini gerçekleştirip birer bit gönderecek şekilde sonuncu shift regester'a bağladım sonra da bütün bunları kontrol edebileceğim bir control fsm tasarladım bütün modülleri birbirine bağladım ve son olarak hepsini ayrı bir modülde bağlayarak kodumu çalıştırdım.

```
module serial_adder(
input wire clk,rstn,start,initial_cin,
input [7:0]a,[7:0]b,
output [8:0]sum_out_main

    );
   reg cin;
wire LSB_A,LSB_B,enable,load,rst,sout,cout;

always @ (posedge clk) begin
    if (load)
    cin <= initial_cin ;

    else
        cin<=cout;
end

fsm             s1        (.clk(clk),.start(start), .rstn(rstn), .enable(enable), .load(load),.rst(rst));

shift_reg       s2        (.clk(clk),.load(load) ,.enable(enable) ,.in(a), .LSB(LSB_A));

shift_reg       s3        (.clk(clk),.load(load) ,.enable(enable),.in(b),.LSB(LSB_B));

full_adder      s4        (.ain(LSB_A), .bin(LSB_B), .cin(cin),.cout(cout), .sumout(sout));


shift_reg_sum   s6        (.clk(clk),.enable(enable),.rst(rst),.sum(sum_out_main),.sout(sout));

endmodule
```

```
module shift_reg_sum(

input wire  clk ,
input wire enable,
input wire rst,
input wire  sout,
output [8:0]sum


);


reg [8:0] shift;
assign sum = shift;

always @ (posedge clk or posedge rst) begin

 if (rst)begin
       shift <= 9'b0;
    end
    else begin

        if(enable) begin
           shift <= {sout,shift[8:1]};
        end
     end
end

endmodule
```

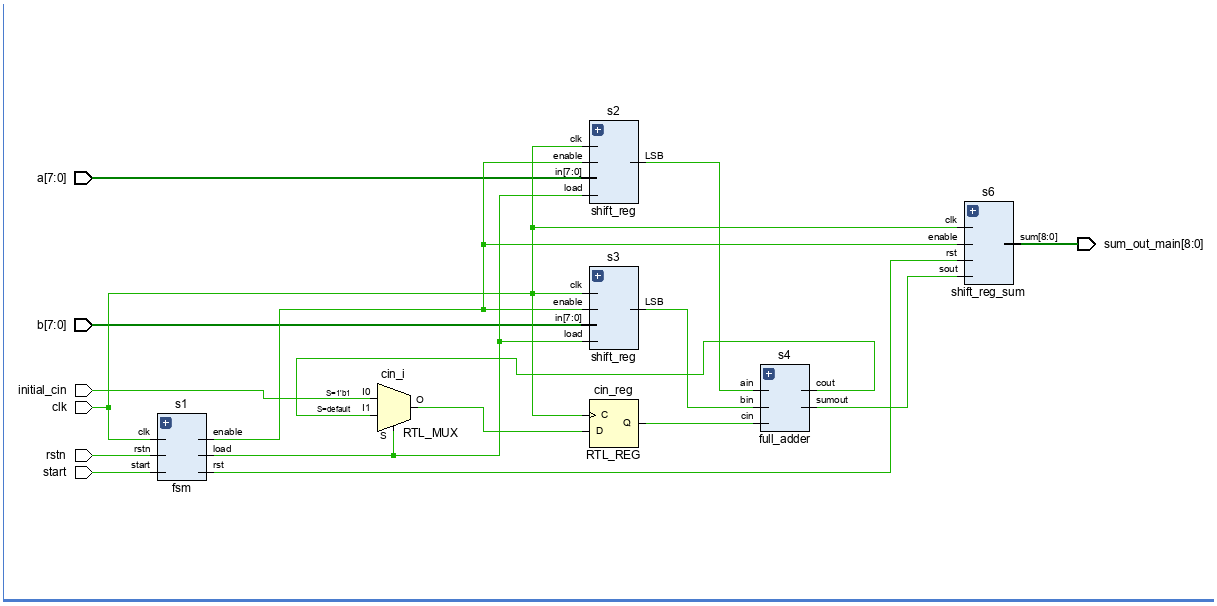
serial_adder.v | fsm.v | full_adder.v | shift_reg_sum.v | s

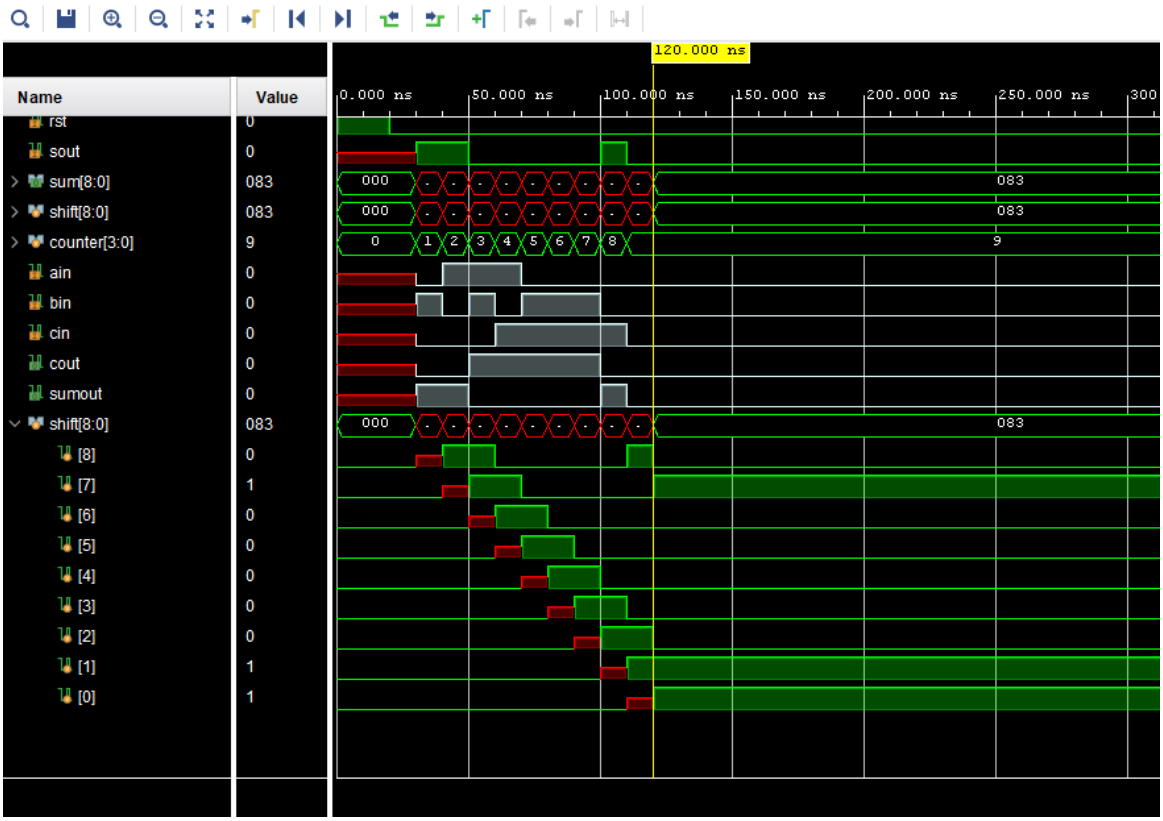
C:/Users/Can/Desktop/homework_3/serial_adder/serial_adder.srcs/sources_1/nev

```
4       input rstn,
5       output reg rst,
6       output reg enable,
7       output reg load
8       );
9        reg [3:0] counter;
10      always@(posedge clk or posedge rstn) begin
11          if (rstn) begin
12              load <=0;
13              counter <=0;
14              enable <= 0;
15              rst<=1;
16          end
17          else begin
18              if (start) begin
19              load <= 1;
20              enable <= 1;
21              rst <= 0;
22           end
23           else if (load) begin
24              load<=0;
25           end
26           if (counter > 4'b1000) begin
27              enable <= 0;
28          end
29          else if(enable)begin
30          counter <= counter+1;
31        end
32        end
33        end
34
35      endmodule
36
```

```
module full_adder(
input wire ain,
input wire bin,
input wire cin,
output wire cout,
output wire sumout
    );

 assign cout = ((ain&bin) |((ain^bin)&cin));
 assign sumout = ((ain^bin)^cin);



endmodule
```

```
module serial_adder_tb();
reg clk=0,rstn,start=0,initial_cin=0;
reg[7:0]a;
reg[7:0]b;
wire [8:0] sum_out_main;
serial_adder dut (.clk(clk), .rstn(rstn) , .start(start), .a(a), .b(b), .sum_out_main(sum_out_main),.initial_cin(initial_cin));

always begin
    clk= ~clk;
    #5;
end

initial begin
a=8'd14;
b=8'd117;
    rstn = 1;
    #20;
    rstn = 0;
    start=1;
    #10
    start=0;

end


endmodule
```


a[7:0]
b[7:0]
initial_cin
clk
rstn
start
s1
fsm
clk
rstn
start
enable
load
rst
s2
shift_reg
clk
enable
in[7:0]
load
LSB
s3
shift_reg
clk
enable
in[7:0]
load
LSB
cin_i
S=1'b1 I0
S=default I1
O
S
RTL_MUX
cin_reg
RTL_REG
C
D
Q
s4
full_adder
ain
bin
cin
cout
sumout
s6
shift_reg_sum
clk
enable
rst
sout
sum[8:0]
sum_out_main[8:0]


120.000 ns
0.000 ns
50.000 ns
100.000 ns
150.000 ns
200.000 ns
250.000 ns
300
Name
Value
rst
0
sout
0
sum[8:0]
083
shift[8:0]
083
counter[3:0]
9
ain
0
bin
0
cin
0
cout
0
sumout
0
shift[8:0]
083
[8]
0
[7]
1
[6]
0
[5]
0
[4]
0
[3]
0
[2]
0
[1]
1
[0]
1
000
083
0
1
2
3
4
5
6
7
8
9