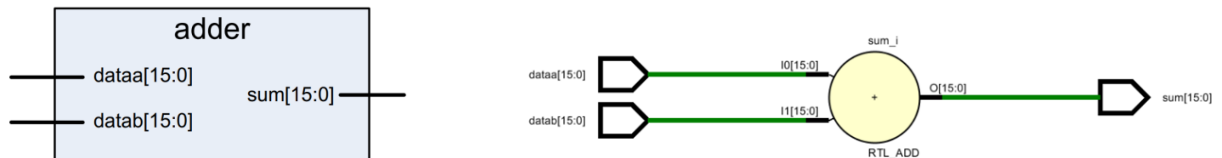


Sequential 8x8 multiplier

Build an 8x8 multiplier. The input to the multiplier consists of two 8-bit multiplicands (dataa and datab) and the output from the multiplier is 16-bit product (product8x8_out). Additional outputs are a done bit (done_flag) and seven signals to drive a 7 segment display (seg_a, seg_b, seg_c, seg_d, seg_e, seg_f & seg_g).

1- 16-bit adder :



Verilog code to perform addition and testbench :

```

1  module adder(
2      input [15:0] dataa, datab,
3      output [15:0] sum);
4  assign sum = dataa + datab;
5  endmodule
6
7  module adder_TB();
8  reg [15:0] dataa ;
9  reg [15:0] datab;
10 wire [15:0] sum;
11 adder adder1(dataa , datab, sum);
12
13
14 initial
15 begin
16     $monitor("dataa = %0d  datab = %0d  Sum= %0d", dataa , datab , sum);
17
18     dataa = 0; datab = 0;
19     #5
20     dataa = 1;
21     datab = 2;
22
23 end
24 endmodule
25

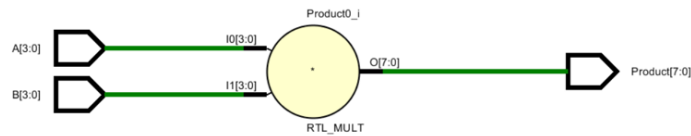
```

```

Transcript
# vsim adder_TB
# Project file D:/verilog/mini_project.mpf is write protected, data cannot be saved.
# Unable to save project.
# ** Warning: (vsim-14) Failed to open "D:/verilog/mini_project.mpf" specified by the MODELSEI
# No such file or directory. (errno = ENOENT)
# ** Error: (vsim-7) Failed to open ini file "D:/verilog/mini_project.mpf" in read mode.
# Invalid argument. (errno = EINVAL)
# Loading work.adder_TB
# Loading work.adder
VSIM 269> run 50
# dataa = 0  datab = 0  Sum= 0
# dataa = 1  datab = 2  Sum= 3
VSIM 270>

```

2- 4x4 multiplier :



Verilog code to perform addition and testbench :

```

1  module mult4x4(
2      input  [3:0] A,
3      input  [3:0] B,
4      output reg [7:0] Product
5  );
6
7      always @(A, B) begin
8          Product = A * B;
9      end
10 endmodule
11 module tb_mul_4x4; //make module to test the encoder
12
13     reg [3:0] A; //reg as we assign values to it in the inital block
14     reg [3:0] B;
15     wire [7:0] Product; //wire as it it the output of the module
16
17     mul_4x4 uut_mul (
18         .A(A),
19         .B(B),
20         .Product(Product)
21     );
22
23     initial begin
24         $monitor("A = %b, B = %b, Product = %d", A, B, Product);
25         A = 2; // Testcase 1
26         B = 3;
27         #10;
28
29         A = 7; // Testcase 2
30         B = 5;
31         #10;
32         // Add more testcases here
33     end
34
35 endmodule

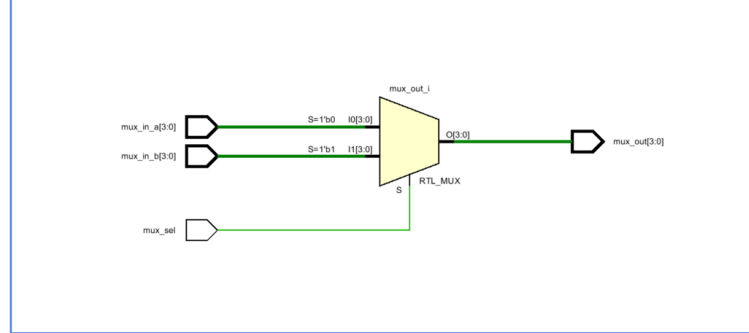
```

Group 1

Mohamed Mohamed tarek Mohamed salah Abdalla Elgohary waleed Emad

```
# Loading work.mul_4x4
VSIM 271> run 200
# A = 0010, B = 0011, Product = 6
# A = 0111, B = 0101, Product = 35
VSIM 272> |
```

3-1 multiplexer:



Verilog code to perform addition and testbench :

```
2  module mux4( input wire [3:0] mux_in_a ,
3  input wire [3:0] mux_in_b ,
4  input mux_sel ,
5  output reg[3:0]mux_out
6  );
7  always @(*)
8  begin
9      case (mux_sel )
10     1'b0 : mux_out <= mux_in_a ;
11     1'b1 : mux_out <= mux_in_b ;
12
13     endcase
14 end
15
16 endmodule
17
18 module mux_TB();
19 wire [3:0] out;
20 reg [3:0] in_a,in_b;
21 reg sel;
22
23 mux mux1 (in_a,in_b,sel,out);
24
25
26 initial
27 begin
28     $monitor("in_a=%d in_b=%d sel=%d out=%d",in_a,in_b,sel,out);
29
30     in_a=1; in_b=2; sel=0;
31     #20
32     in_a=1; in_b=2; sel=1;
33
34 end
35 endmodule
36
```

Group 1

Mohamed Mohamed tarek Mohamed salah Abdalla Elgohary waleed Emad

```
# Loading work.mux
VSIM 273> run 100
# in_a= 1 in_b= 2 sel=0 out= 1
# in_a= 1 in_b= 2 sel=1 out= 2
VSIM 274> ]
```

4-Shifter



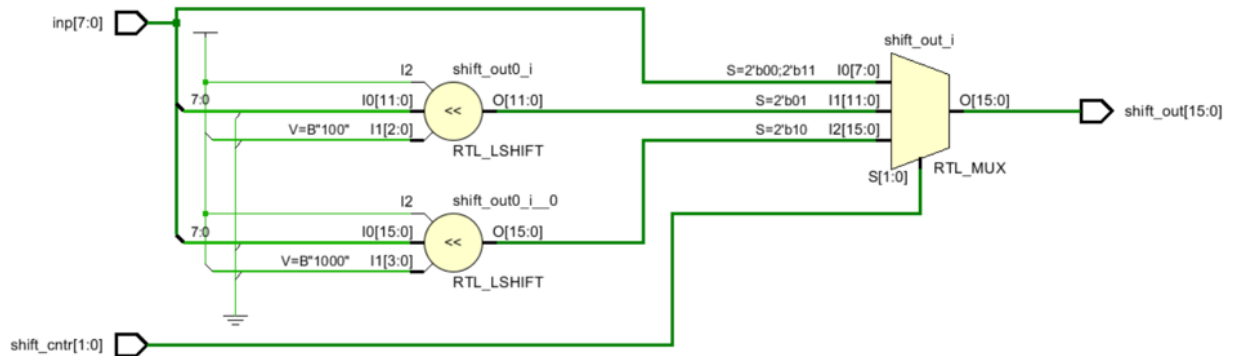
Verilog code to perform addition and testbench :

```
1  module shifter(  
2      input [7:0] inp,  
3      input [1:0] shift_cntrl,  
4      output reg [15:0] shift_out);  
5  always@(*) begin  
6  
7      case(shift_cntrl)  
8          0,3 : shift_out = inp;  
9          1 : shift_out = inp << 4;  
10         2 : shift_out = inp << 8;  
11         default : shift_out = inp;  
12     endcase  
13 end  
14 endmodule  
15  
16 module r_TB();  
17 reg [7:0] inp;  
18 reg [1:0] shift_cntrl;  
19 wire [15:0] shift_out;  
20  
21 shifter s(inp,shift_cntrl,shift_out);  
22 initial  
23 begin  
24     $monitor("shift_cntrl=%d , in_data= %b ==>>> shift_out=%b",shift_cntrl,inp,shift_out);  
25     inp = 7;  
26     shift_cntrl = 0;  
27     #10  
28     shift_cntrl = 1;  
29     #10  
30     shift_cntrl = 2;  
31     #10  
32     shift_cntrl = 3;  
33 end  
34 endmodule  
35
```

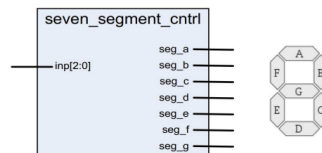
```

# coding: utf-8
VSIM 275> run 100
# shift_cntr=0 , in_data= 00000111 ==>>> shift_out=0000000000000111
# shift_cntr=1 , in_data= 00000111 ==>>> shift_out=0000000001110000
# shift_cntr=2 , in_data= 00000111 ==>>> shift_out=0000011100000000
# shift_cntr=3 , in_data= 00000111 ==>>> shift_out=0000000000000111
VSIM 276> ]

```



7-segment display encoder:



- Write Verilog code to perform encoder as following table:

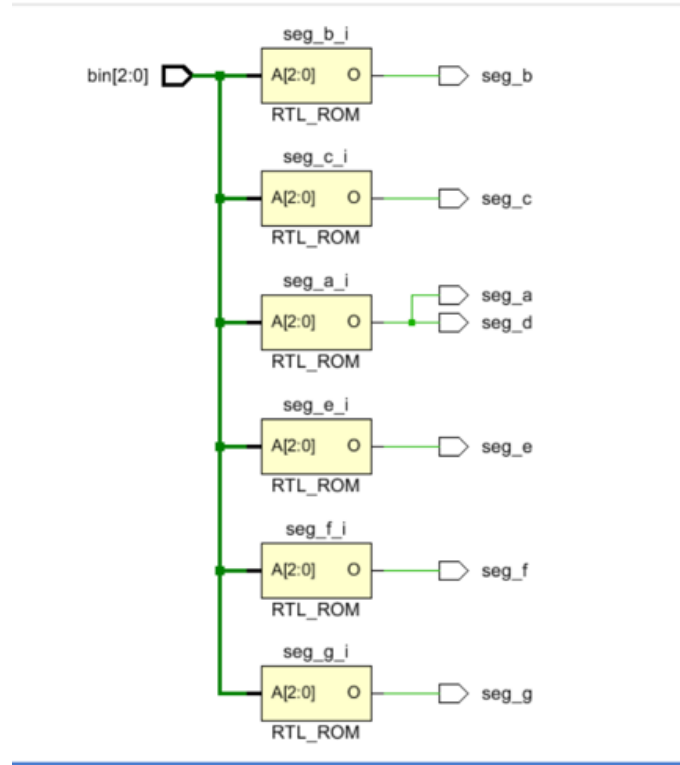
```

1 module seven_segment_encoder(
2     input [2:0] bin,
3     output reg seg_a,
4     output reg seg_b,
5     output reg seg_c,
6     output reg seg_d,
7     output reg seg_e,
8     output reg seg_f,
9     output reg seg_g
10 );
11
12 always @(bin)
13 begin
14     case(bin)
15         3'b000: begin seg_a = 1'b1; seg_b = 1'b1; seg_c = 1'b1; seg_d = 1'b1; seg_e = 1'b1; seg_f = 1'b1; seg_g = 1'b0; end // 0
16         3'b001: begin seg_a = 1'b0; seg_b = 1'b1; seg_c = 1'b1; seg_d = 1'b1; seg_e = 1'b0; seg_f = 1'b0; seg_g = 1'b0; end // 1
17         3'b010: begin seg_a = 1'b1; seg_b = 1'b1; seg_c = 1'b0; seg_d = 1'b1; seg_e = 1'b1; seg_f = 1'b0; seg_g = 1'b1; end // 2
18         3'b011: begin seg_a = 1'b1; seg_b = 1'b1; seg_c = 1'b1; seg_d = 1'b1; seg_e = 1'b0; seg_f = 1'b0; seg_g = 1'b1; end // 3
19         default: begin seg_a = 1'b1; seg_b = 1'b0; seg_c = 1'b0; seg_d = 1'b1; seg_e = 1'b1; seg_f = 1'b1; seg_g = 1'b1; end // E
20     endcase
21 end
22
23 endmodule

```

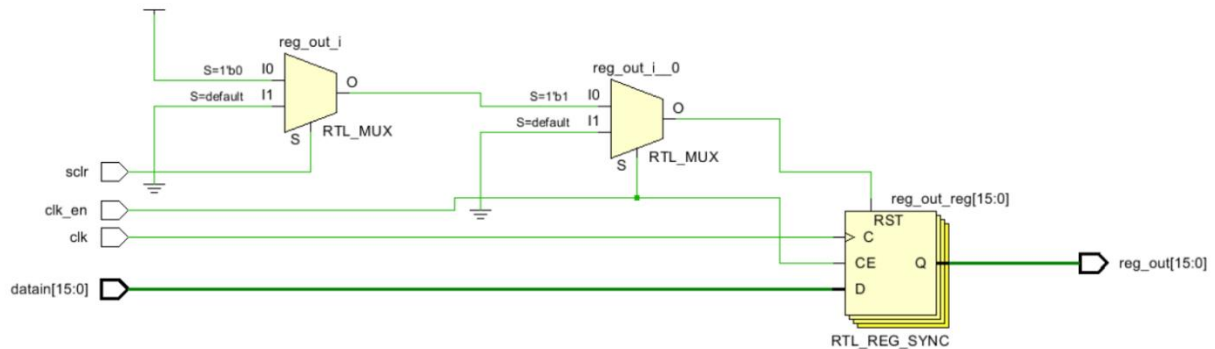
```
24 module tb_encoder; //make module to test the encoder
25     reg [2:0] bin; //reg as we assign values to it in the initial block
26     wire seg_a; //wire as it is the output of the module
27     wire seg_b;
28     wire seg_c;
29     wire seg_d;
30     wire seg_e;
31     wire seg_g;
32     wire seg_f;
33
34
35     seven_segment_encoder uut_encoder ( //make an instant of the module
36         .bin(bin),
37         .seg_a(seg_a),
38         .seg_b(seg_b),
39         .seg_c(seg_c),
40         .seg_d(seg_d),
41         .seg_e(seg_e),
42         .seg_g(seg_g),
43         .seg_f(seg_f)
44     );
45
46     initial begin
47         $monitor("bin = %b, seg = %b", bin, seg); //display the output
48
49         bin = 3'b000; // Test for 0
50         #10; //a delay of 10 time units
51
52
53         bin = 3'b001; // Test for 1
54         #10;
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
```

```
46     initial begin
47         $monitor("bin = %b, seg = %b", bin, seg); //display the output
48
49         bin = 3'b000; // Test for 0
50         #10; //a delay of 10 time units
51
52
53         bin = 3'b001; // Test for 1
54         #10;
55
56         bin = 3'b010; // Test for 2
57         #10;
58
59         bin = 3'b011; // Test for 3
60         #10;
61         // Test for other values
62         bin = 3'b100;
63         #10;
64
65         bin = 3'b101;
66         #10;
67
68         bin = 3'b110;
69         #10;
70
71         bin = 3'b111;
72         #10;
73
74         $finish;
75     end
76
77
78 endmodule
```



```
Transcript
# Invalid argument. (errno = EINVAL)
# Loading work.tb_encoder
# Loading work.seven_segment_encoder
VSIM 282> run 100
# bin = 000, seg_a = 1
# bin = 001, seg_a = 0
# bin = 010, seg_a = 1
# bin = 011, seg_a = 1
# bin = 100, seg_a = 1
# bin = 101, seg_a = 1
# bin = 110, seg_a = 1
# bin = 111, seg_a = 1
# ** Note: $finish      : D:/verilog/FF_tested.v(76)
#   Time: 80 ns  Iteration: 0  Instance: /tb_encoder
```

reg16



```

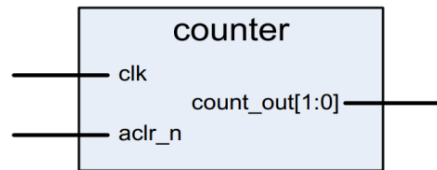
1  module reg16(
2      input clk,sclr,clk_en,
3      input [15:0] datain,
4      output reg [15:0] reg_out);
5  always@(posedge clk) begin
6      if(clk_en) begin
7          if(!sclr)
8              reg_out <= 0;
9          else
10             reg_out <= datain;
11         end
12     end
13 endmodule
14 module TB();
15     reg clk ,sclr,clk_en;
16     reg [15:0] datain;
17     wire [15:0] reg_out;
18     reg16 r(clk,sclr,clk_en,datain,reg_out);
19     always
20         #5 clk=~clk;
21     initial
22         $monitor("sclr=%b clk_en=%b datain=%d reg_out=%d",sclr,clk_en,datain,reg_out);
23     initial begin
24         clk = 0; clk_en=0; sclr=1; datain=2;
25         #10
26         clk_en=1;
27         #10
28         sclr=0;
29         #80;
30     end
31 end
32 endmodule
33

```

```

# Invalid argument. (errno = EINVAL)
# Loading work.TB
# Loading work.reg16
VSIM 289> run 100
# sclr=1 clk_en=0 datain= 2 reg_out= x
# sclr=1 clk_en=1 datain= 2 reg_out= x
# sclr=1 clk_en=1 datain= 2 reg_out= 2
# sclr=0 clk_en=1 datain= 2 reg_out= 2
# sclr=0 clk_en=1 datain= 2 reg_out= 0
VSIM 290>

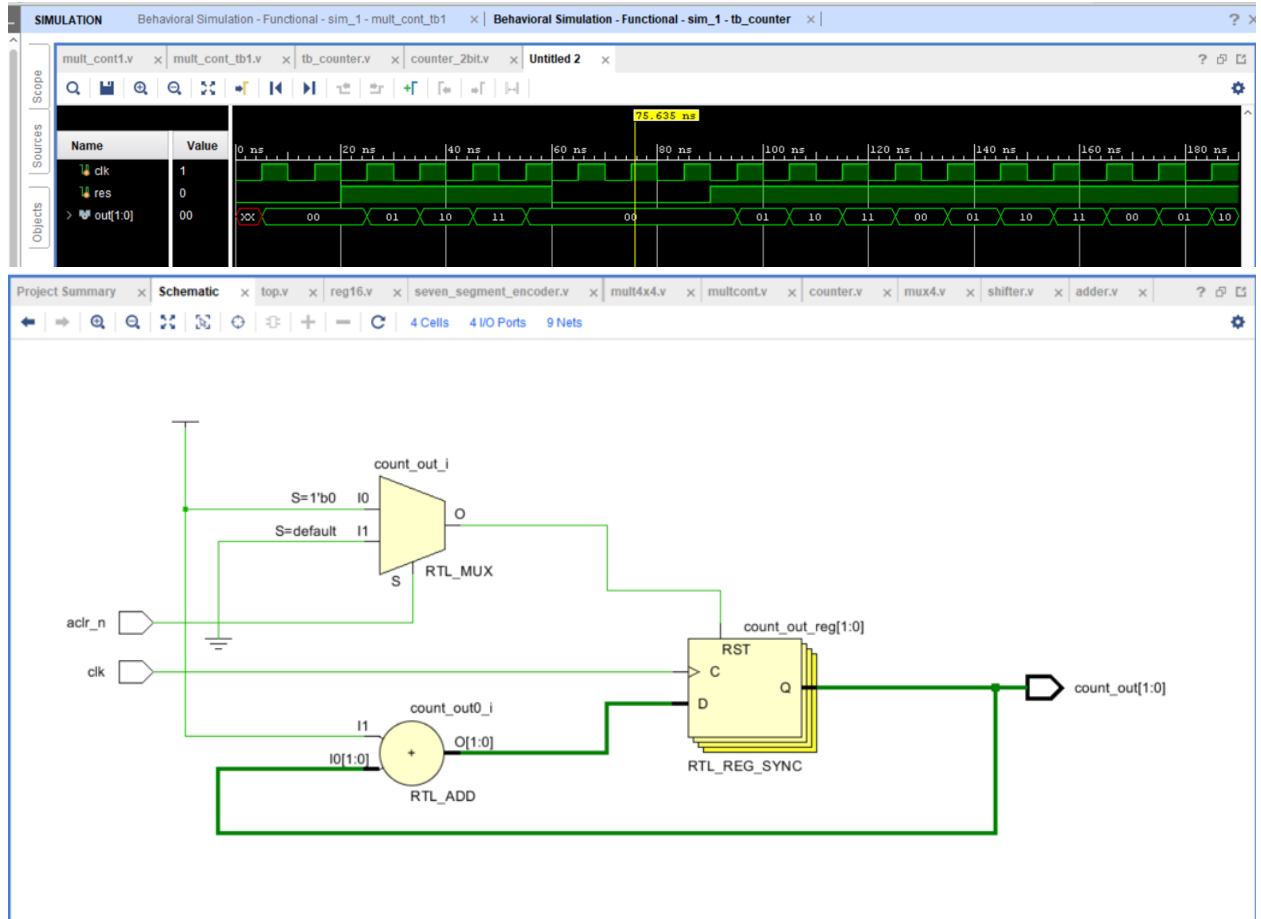
```


2-bit Counter with asynchronous control:

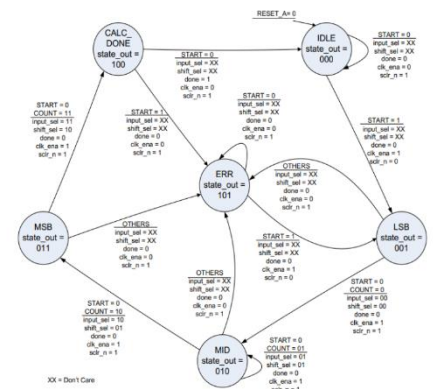
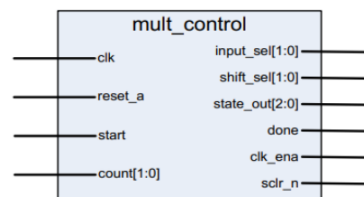
```
1  module counter(  
2      input clk,aclr_n,  
3      output reg [1:0] count_out  
4  );  
5  
6      always@(posedge clk)  
7      begin  
8          count_out <=0;  
9          if(!aclr_n)  
10             count_out <=0;  
11         else count_out <= count_out+1 ;  
12     end  
13  
14 endmodule  
15 module tb_counter;  
16 reg clk;  
17 reg res;  
18 wire [1:0] out;  
19 counter_2bit c0(  
20     .clk(clk),  
21     .aclr_n(res),  
22     .count_out(out)  
23 );  
24 //Makes the clk changes its state every 5ns  
25 always #5 clk = ~clk;  
26 initial begin  
27     clk <=0;  
28     res <=0;  
29  
30     #20 res <=1;  
31     #40 res <=0;  
32     #30 res <=1;  
33     #100 $finish;  
34 end  
35 endmodule  
36
```

Group 1

Mohamed Mohamed tarek Mohamed salah Abdalla Elgohary waleed Emad



Multiplier controller



The outputs of mult_control state machine control the other blocks in the design

Group 1

Mohamed Mohamed tarek Mohamed salah Abdalla Elgohary waleed Emad

```

21 module multcont(
22     input clk , reset_a, start,
23     input [1:0] count,
24     output reg [1:0] input_sel , shift_sel,
25     output [2:0] state_out,
26     output reg done , clk_ena, sclr_n);
27     output reg done , clk_ena, sclr_n);
28     reg [2:0] current_state, next_state;
29
30
31     parameter IDLE = 3'b000;
32     parameter LSB = 3'b001;
33     parameter MID = 3'b010;
34     parameter MSB = 3'b011;
35     parameter CALC_DONE=3'b100;
36     parameter ERR =3'b101;
37
38     assign state_out = current_state;
39
40     always@(*) begin
41         if(reset_a==0)
42             next_state=IDLE;
43         else
44             begin
45                 case(current_state)
46                     IDLE: begin
47                         if(start==0)
48                             begin
49                                 input_sel=0;
50                                 shift_sel=0;
51                                 done=0;
52                                 clk_ena=0;
53                                 sclr_n=1;
54                                 next_state=IDLE;
55                             end
56                         else
57                             begin
58                                 input_sel=0;
59                                 shift_sel=0;
60                                 done=0;
61                                 clk_ena=1;
62                                 sclr_n=0;
63                                 next_state = LSB;
64                             end
65                         end
66                     LSB: begin
67                         if((start==0) && (count==0))
68                             begin
69                                 input_sel=0;
70                                 shift_sel=0;
71                                 done=0;
72                                 clk_ena=1;
73                                 sclr_n=1;
74                                 next_state=MID;
75                             end
76                         else
77                             begin
78                                 input_sel=0;
79                                 shift_sel=0;
80                                 done=0;
81                                 clk_ena=0;
82                                 sclr_n=1;
83                                 next_state=ERR;
84                             end
85                         end
86                     MID: begin
87                         if((start==0)&&(count==2'b01))
88                             begin
89                                 input_sel=2'b01;
90                                 shift_sel=2'b01;
91                                 done=0;
92                                 clk_ena=1;
93                                 sclr_n=0;
94                                 next_state= MID;
95                             end
96                         else if((start==0) && (count==2'b10))
97                             begin
98                                 input_sel=2'b10;
99                                 shift_sel=2'b01;
100                                done=0;
101                                clk_ena=1;
102                                sclr_n=1;
103                                next_state=MSB;
104                            end
105                        else
106                            begin
107                                input_sel=0;
108                                shift_sel=0;
109                                done=0;
110                                clk_ena=0;
111                                sclr_n=1;
112                                next_state=ERR;
113                            end
114                        end
115                    end;
116                MSB: begin
117                    if((start==0) && (count==2'b11))
118                        begin
119                            input_sel=2'b11;
120                            shift_sel=2'b10;
121                            done=0;
122                            clk_ena=1;
123                            sclr_n=1;
124                            next_state= CALC_DONE;
125                        end
126                    else
127                        begin
128                            input_sel=0;
129                            shift_sel=0;
130                            done=0;
131                            clk_ena=0;
132                            sclr_n=1;
133                            next_state=ERR;
134                        end
135                    end
136                end;
137                CALC_DONE: begin
138                    if(start==1)
139                        begin
140                            input_sel=0;
141                            shift_sel=0;
142                            done=0;
143                            clk_ena=0;
144                            sclr_n=1;
145                            next_state=ERR;
146                        end
147                    else
148                        begin
149                            input_sel=0;
150                            shift_sel=0;
151                            done=1;
152                        end
153                    end
154                end;
155            end
156        end
157    end

```

Group 1

Mohamed Mohamed tarek Mohamed salah Abdalla Elgohary waleed Emad

```

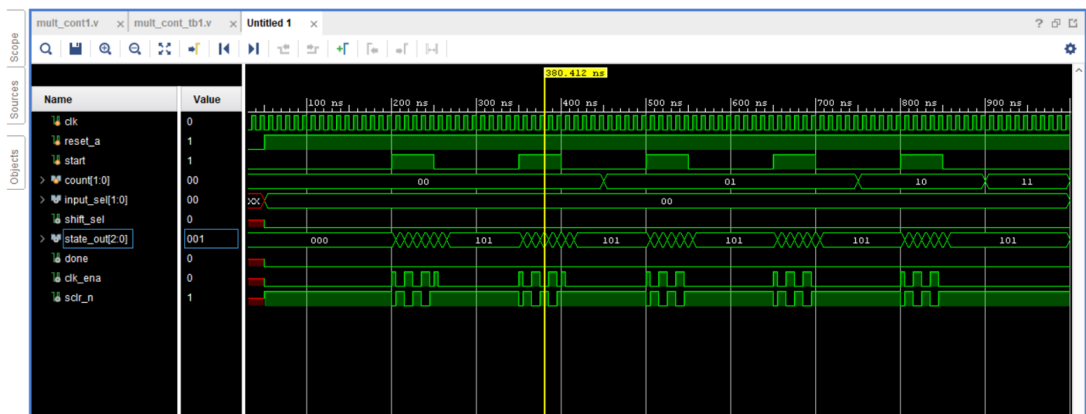
134         end
135     end
136     CALC_DONE: begin
137         if(start==1)
138             begin
139                 input_sel=0;
140                 shift_sel=0;
141                 done=0;
142                 clk_ena=0;
143                 sclr_n=1;
144                 next_state=ERR;
145             end
146         else
147             begin
148                 input_sel=0;
149                 shift_sel=0;
150                 done=1;
151                 clk_ena=0;
152                 sclr_n=1;
153                 next_state=IDLE;
154             end
155         end
156     ERR: begin
157         if(start==1)
158             begin
159                 input_sel=0;
160                 shift_sel=0;
161                 done=0;
162                 clk_ena=1;
163                 sclr_n=0;
164                 next_state=LSB;
165             end
166         else
167             begin
168                 input_sel=0;
169                 shift_sel=0;
170                 done=0;
171                 clk_ena=0;
172                 sclr_n=1;
173                 next_state=ERR;
174             end
175         end
176     endcase
177 end
178 end
179 end
180
181 always@(posedge clk, negedge reset_a)
182 begin
183     if(reset_a==0)
184         current_state <= IDLE;
185     else
186         current_state <= next_state;
187     end
188 end
189
190 endmodule

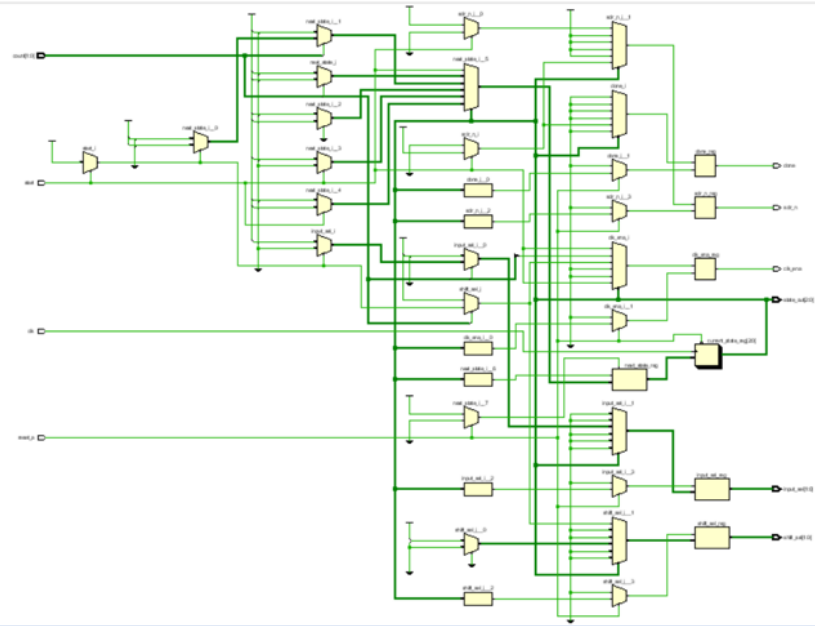
```

```

194 module mult_cont_tbi;
195 reg clk;
196 reg reset_a;
197 reg start;
198 reg [1:0] count;
199
200 // Outputs
201 wire [1:0] input_sel;
202 wire shift_sel;
203 wire [2:0] state_out;
204 wire done;
205 wire clk_ena;
206 wire sclr_n;
207
208 mult_cont1 uut (
209     .clk(clk),
210     .reset_a(reset_a),
211     .start(start),
212     .count(count),
213     .input_sel(input_sel),
214     .shift_sel(shift_sel),
215     .state_out(state_out),
216     .done(done),
217     .clk_ena(clk_ena),
218     .sclr_n(sclr_n)
219 );
220
221 initial begin
222     // Initialize inputs
223     clk = 0;
224     reset_a = 0;
225     start = 0;
226     count = 0;
227     #50 reset_a = 1;
228
229     #50 start = 0;
230     #50 count = 0;
231
232     #50 start = 1;
233     #50 start = 0;
234     #50 count = 0;
235     #50 start = 1;
236     #50 start = 0;
237     #50 count = 1;
238     #50 start = 1;
239     #50 start = 0;
240     #50 count = 2'b01;
241     #50 start = 1;
242     #50 start = 0;
243     #50 count = 2'b10;
244     #50 start = 1;
245     #50 start = 0;
246     #50 count = 2'b11;
247     #50 start = 0;
248     #50 count = 0;
249
250     $finish;
251 end
252
253 // Clock generator
254 always begin
255     #5 clk = ~clk;
256 end
257 end
258 endmodule
259

```





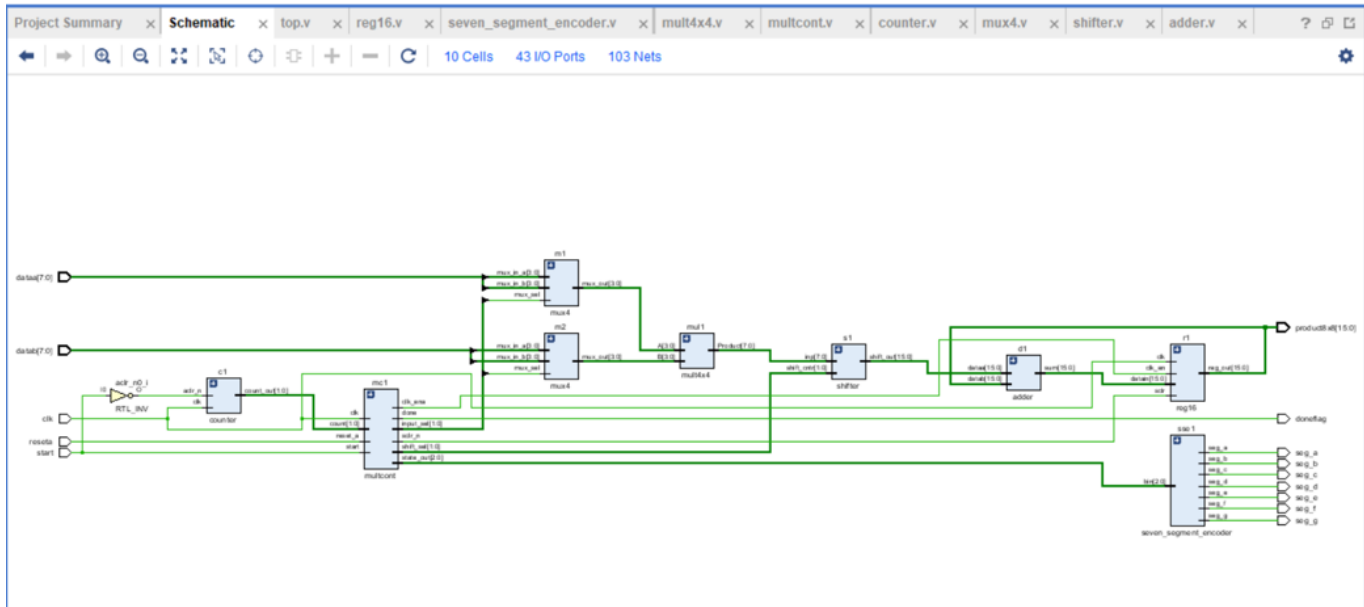
Top module :

```

1  module top (
2  input [7:0] dataa,datab , input start, reseta , clk ,
3  output [15:0] product8x8 , output doneflag ,seg_a, seg_b, seg_c,
4  seg_d, seg_e, seg_f,seg_g);
5
6  wire [3:0] aout,bout;
7  wire [7:0] product;
8  wire [15:0] shift_out , sum;
9  wire [2:0] state_out;
10 wire [1:0] sel ,shift,count;
11 wire clk_ena,sclr_n;
12
13 mux4 m1(dataa[3:0],dataa[7:4],sel[1],aout);
14 mux4 m2(datab[3:0],datab[7:4],sel[0],bout);
15 mult4x4 mul1(aout,bout,product);
16 shifter s1(product, shift,shift_out);
17 adder d1(shift_out,product8x8,sum);
18 reg16 r1(clk ,sclr_n,clk_ena,sum,product8x8);
19 multcont mc1(clk,reseta,start,count,sel,shift,state_out,doneflag,clk_ena,
20 counter c1(clk,(!start),count);
21 seven_segment_encoder sse1 (state_out,seg_a,seg_b,seg_c,seg_d,seg_e,seg_f
22 endmodule
23
24

```

Top schematic



Top Test bench :

```

23
24 module tst();
25   reg [7:0] dataa, datab;
26   reg start, reset_a, clk;
27   wire [15:0] product8x8;
28   wire done_flag, a, b, c, d, e, f, g;
29   top t1(dataa, datab, start, reset_a, clk, product8x8, done_flag, a, b, c, d, e, f, g);
30   initial
31   |   $monitor("time=%0t clk=%b dataa=%0d datab=%0d reset_a=%b start=%b product
32   always
33   |   #5 clk=~clk;
34   initial begin
35     clk=1; reset_a=1; start=0; dataa=12; datab=4;
36     @(negedge clk)
37     reset_a=0;
38     @(negedge clk)
39     reset_a=1; start=1;
40     @(negedge clk)
41     start=0;
42
43   repeat(4)
44   |   @(negedge clk);
45
46   $stop();
47   end
48 endmodule

```

```
# Loading work.mult4x4
# Loading work.shifter
# Loading work.adder
# Loading work.reg16
# Loading work.multcont
# Loading work.counter
# Loading work.seven_segment_encoder
VSIM 293> run 200
# time=0 clk=1 dataa=12 datab=4 reset_a=1 start=0 product8x8=x done_flag=x abcdefg=xxxxxxx
# time=5 clk=0 dataa=12 datab=4 reset_a=0 start=0 product8x8=x done_flag=x abcdefg=1111110
# time=10 clk=1 dataa=12 datab=4 reset_a=0 start=0 product8x8=x done_flag=x abcdefg=1111110
# time=15 clk=0 dataa=12 datab=4 reset_a=1 start=1 product8x8=x done_flag=0 abcdefg=1111110
# time=20 clk=1 dataa=12 datab=4 reset_a=1 start=1 product8x8=0 done_flag=0 abcdefg=0110000
# time=25 clk=0 dataa=12 datab=4 reset_a=1 start=0 product8x8=0 done_flag=0 abcdefg=0110000
# time=30 clk=1 dataa=12 datab=4 reset_a=1 start=0 product8x8=48 done_flag=0 abcdefg=1101101
# time=35 clk=0 dataa=12 datab=4 reset_a=1 start=0 product8x8=48 done_flag=0 abcdefg=1101101
# time=40 clk=1 dataa=12 datab=4 reset_a=1 start=0 product8x8=48 done_flag=0 abcdefg=1101101
# time=45 clk=0 dataa=12 datab=4 reset_a=1 start=0 product8x8=48 done_flag=0 abcdefg=1101101
# time=50 clk=1 dataa=12 datab=4 reset_a=1 start=0 product8x8=48 done_flag=0 abcdefg=1111001
# time=55 clk=0 dataa=12 datab=4 reset_a=1 start=0 product8x8=48 done_flag=0 abcdefg=1111001
# time=60 clk=1 dataa=12 datab=4 reset_a=1 start=0 product8x8=48 done_flag=1 abcdefg=1001111
# Break in Module tst at D:/verilog/top.v line 46
VSIM 294> ]
```