CS201P Assignment 5 (LAB)

B20218 Narmit Kumar

1st Question

4 * 4 Booth Multiplier code:

Booth_4bit.v *

```
1  `timescale 1ns / 1ps
2  module Booth_4bit(
3  input clk,en,
4  input [3:0] A,B,
5  output reg [7:0] Prod,
6  output reg done
7  );
8
9  reg [3:0] cnt;
10 reg [8:0] temp;
11 wire [3:0] negB;
12
13 assign negB = ~B + 1'b1 ;
14
15 always @ (posedge clk)
16 begin
17 if(~en)
18     begin
19     cnt=4'd0;
20     temp=9'd0;
21     done <=1'b0;
22     end
23 else
24     begin
25
26     if(cnt==4'd0)
27         begin
28         temp <= {4'd0,A,1'b0};
29         cnt <= cnt+1;
30         end
31     else if(cnt> 4'd0 && cnt <4'd5)
32         begin
33         if(temp[1:0] == 2'b00 || temp[1:0] == 2'b11 )
34         begin
35         temp = {temp[8],temp[8:1]};
36         cnt = cnt+1'b1;
37         end
```

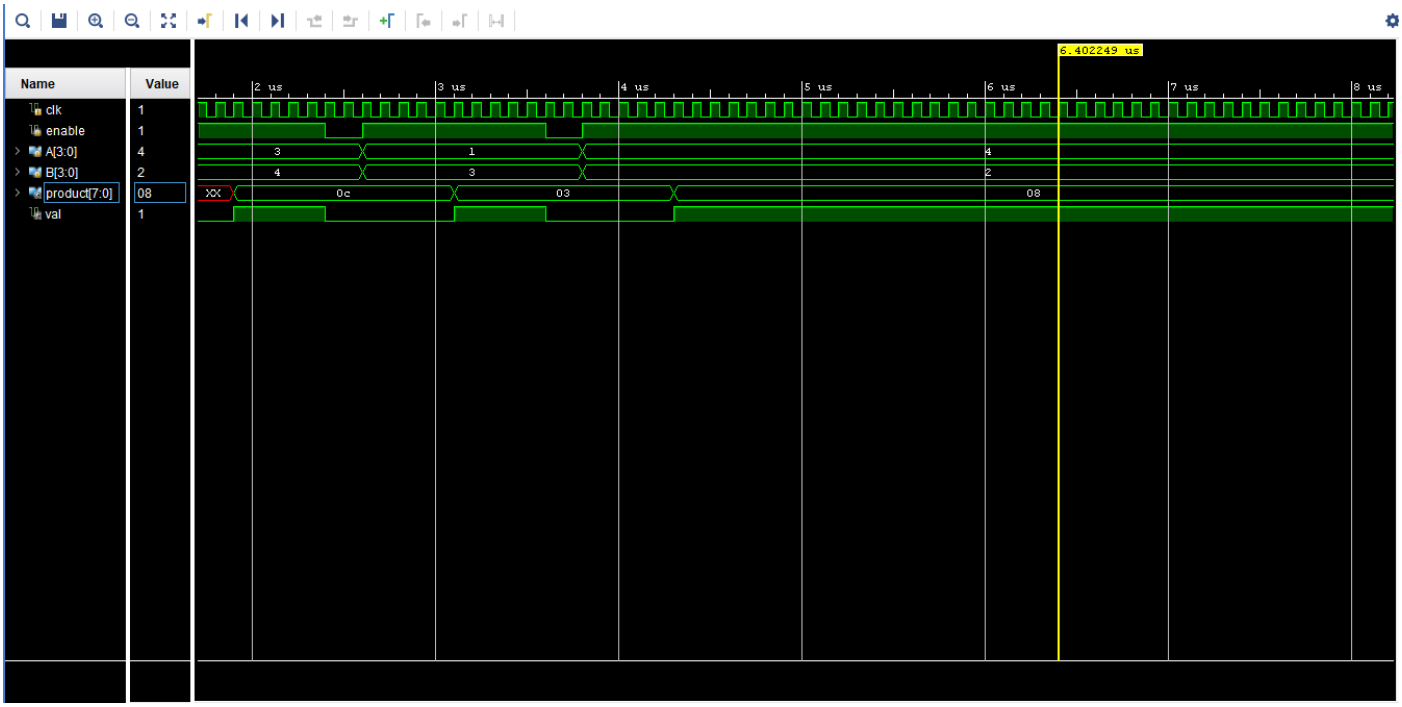
```
        cnt = cnt+1'b1;
        end
        else if(temp[1:0] == 2'b01)
        begin
        temp[8:5] = temp[8:5] + B;
        temp = {temp[8],temp[8:1]};
        cnt = cnt+1'b1;
        end
        else if(temp[1:0] == 2'b10)
        begin
        temp[8:5] = temp[8:5] + negB;
        temp = {temp[8],temp[8:1]};
        cnt = cnt+1'b1;
        end
        end
    else
        begin
        Prod = temp[8:1];
        done =1'b1;
        end
    end
end

endmodule
```

4 * 4 Booth Multiplier testbench:

TB_4bit.v *

```
`timescale 1ns / 1ps
module TB_4bit(
    input clk
    );
reg enable;
reg [3:0] A,B;
wire [7:0] product;
wire val;
Booth_4bit Booth_4bitdut(
    .clk(clk),
    .en(enable),
    .A(A),
    .B(B),
    .Prod(product),
    .done(val)
);
initial
begin
enable = 1'b0; #200
enable = 1'b1;
A = 4'b0010;
B = 4'b0010; #1000
enable = 1'b0; #200
enable = 1'b1;
A = 4'b0011;
B = 4'b0100; #1000
enable = 1'b0; #200
enable = 1'b1;
A = 4'b0001;
B = 4'b0011; #1000
enable = 1'b0;#200
enable = 1'b1;
A = 4'b0100;
B = 4'b0010; #1000
$stop;
end
endmodule
```

Output :



16 * 16 Booth Multiplier code:

Booth_16bit.v

```
1  `timescale 1ns / 1ps
2  module Booth_mul(
3  input clk,en,
4  input [15:0] A,B,
5  output reg [31:0] Prod,
6  output reg done
7  );
8  reg [15:0] cnt;
9  reg [32:0] temp;
10 wire [15:0] negB;
11 assign negB = ~B + 1'b1 ;
12
13 always @ (posedge clk)
14 begin
15 if(~en)
16     begin
17     cnt=16'd0;
18     temp=33'd0;
19     done <=1'b0;
20     end
21 else
22     begin
23
24     if(cnt==16'd0)
25         begin
26         temp <= {16'd0,A,1'b0};
27         cnt <= cnt+1;
28         end
29     else if(cnt> 16'd0 && cnt <16'd17)
30         begin
31         if(temp[1:0] == 2'b00 || temp[1:0] == 2'b11 )
32         begin
33         temp = {temp[32],temp[32:1]};
34         cnt = cnt+1'b1;
35         end
36         else if(temp[1:0] == 2'b01)
37         begin
```

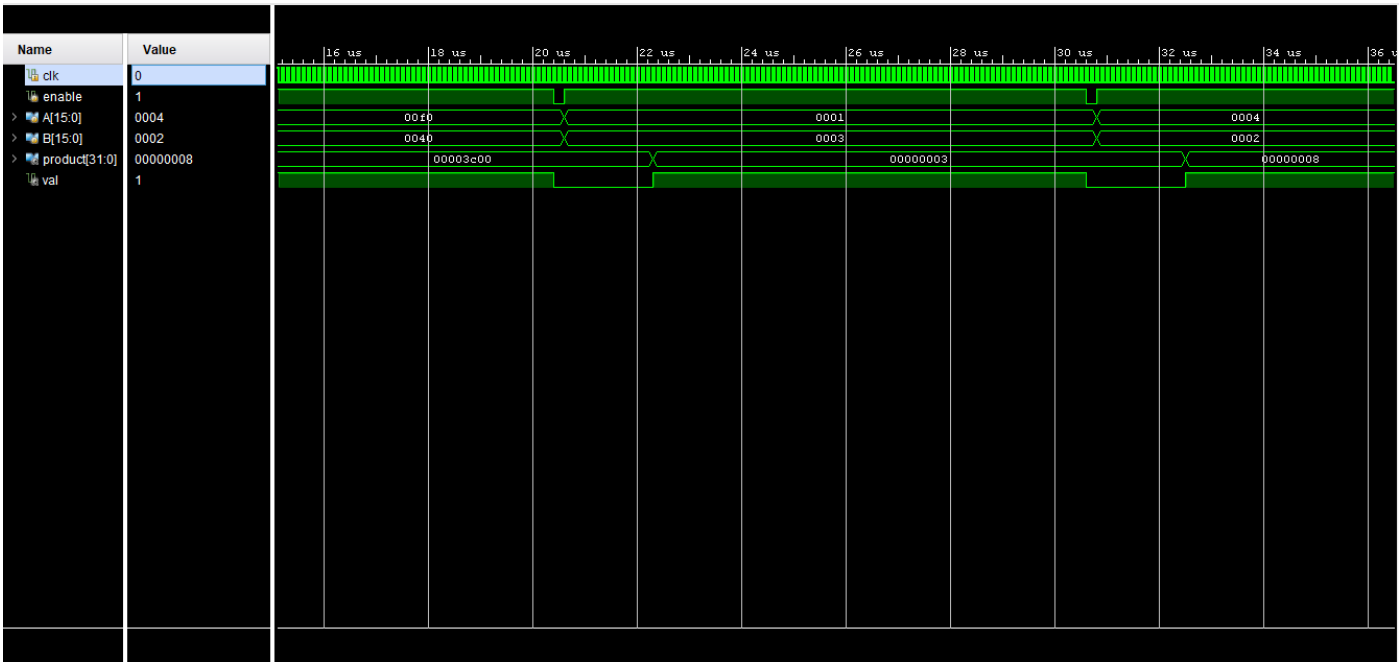
```
37          begin
38          temp[32:17] = temp[32:17] + B;
39          temp = {temp[32],temp[32:1]};
40          cnt = cnt+1'b1;
41          end
42          else if(temp[1:0] == 2'b10)
43          begin
44          temp[32:17] = temp[32:17] + negB;
45          temp = {temp[32],temp[32:1]};
46          cnt = cnt+1'b1;
47          end
48          end
49      else
50          begin
51          Prod = temp[32:1];
52          done =1'b1;
53          end
54      end
55  end
56
57  endmodule
58
59
```

16 * 16 Booth Multiplier testbench:

TB_16bit.v

```
43  initial
44  begin
45
46  enable = 1'b0;
47  #200
48  enable = 1'b1;
49  A = 16'b0100000000000100;
50  B = 16'b0010;
51  #10000
52
53  enable = 1'b0;
54  #200
55  enable = 1'b1;
56  A = 8'b11110000;
57  B = 8'b01000000;
58  #10000
59
60  enable = 1'b0;
61  #200
62  enable = 1'b1;
63  A = 4'b0001;
64  B = 4'b0011;
65  #10000
66
67  enable = 1'b0;
68  #200
69  enable = 1'b1;
70  A = 4'b0100;
71  B = 4'b0010;
72  #10000
73  $stop;
74
75
76  end
77
78  endmodule
79
```

Output:



3rd Question

Pattern Generator code:

pattern_gen.v *

```
1  `timescale 1ns / 1ps
2  module CKT(clk,en,gen,in,Y);
3  input clk, en, gen, in;
4  output reg [3:0] Y = 4'b0000;
5  reg in_prev ;
6  always@(posedge clk)
7  begin
8  if (en)
9      begin
10         if (in)
11             begin
12             if (!in_prev)
13             Y <= 4'b0000;
14             else
15             Y <= Y + 4'b0010;
16             end
17         else
18             begin
19             if (in_prev)
20             Y <= 4'b0000;
21             else
22                 begin
23                     case(Y)
24                     4'b0000 : Y <= 4'b0001;
25                     4'b1111 : Y <= 4'b0000;
26                     default : Y <= Y + 4'b0010;
27                     endcase
28                 end
29             end
30     end
31 else
32     Y<= 4'bXXXX;
33 in_prev <= in;
34 end
35
36 endmodule
```

Testbench:

tb_ckt.v

```
1  `timescale 1ns / 1ps
2
3  module tb_ckt;
4
5  reg clk, en, gen ,in;
6  wire[3:0] Y;
7
8  CKT uut(
9  .clk(clk), .en(en), .gen(gen) , .in(in), .Y(Y)
10 );
11
12 initial clk = 0;
13 always #20 clk = ~clk;
14
15 initial begin
16 en = 1;gen = 1; in = 1; #1000000
17 in = 0 ;#10000;
18
19 end
20 endmodule
21
22
```

Output: