HO CHI MINH CITY NATIONAL UNIVERSITY

BACH KHOA UNIVERSITY SCHOOL OF INDUSTRIAL MANAGEMENT

-----🕮-----

REPORT
LAB 5

Instructor: Pro.Dr. Hoàng Trang

Supporter: Đặng Công Thịnh

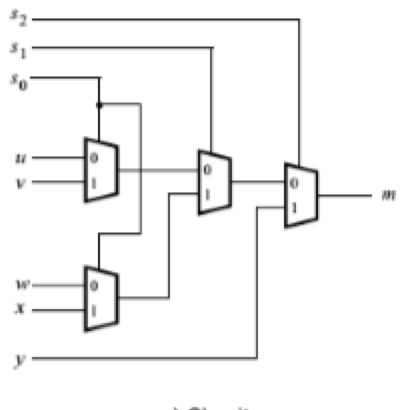
Student name: Trần Văn Trọng

ID code: 1870262

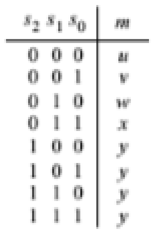
HCM City, Day 22 Month 12 Year 2018

Chapter 1: Exercise 01

Requirement: Develop the following hardware (for every figure) with Verilog HDL.



a) Circuit



s_2 s_1 s_0	m
0 0 0	u
0 0 1	v
0 1 0	w
0 1 1	x
1 0 0	y
1 0 1	y
1 1 0	y
1 1 1	y

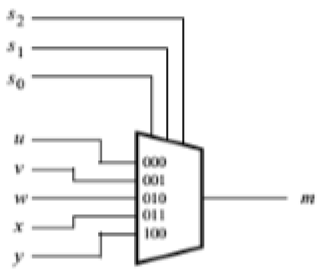
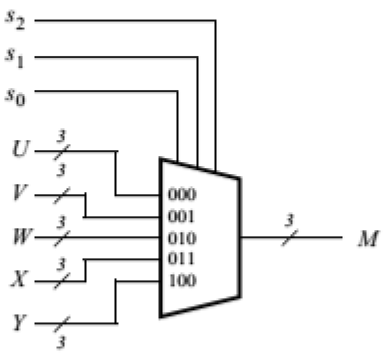


Figure 1.1 Multiplexer 5-1



<u>Solution</u>

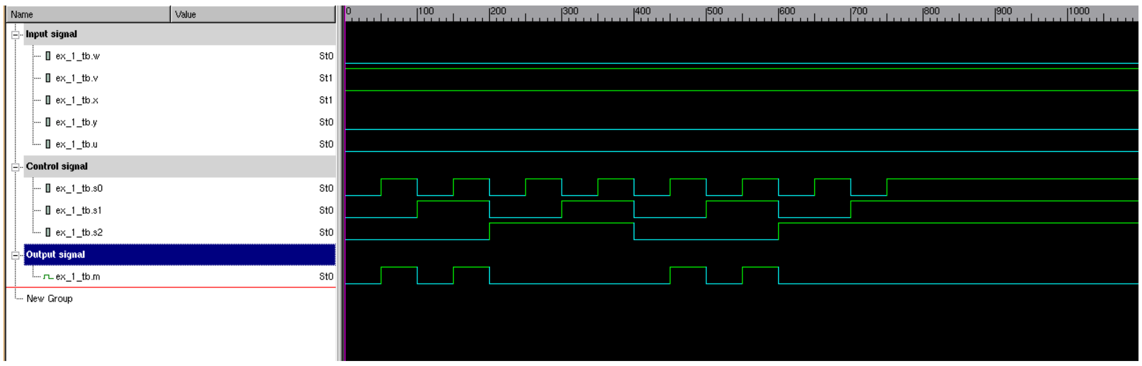
a)

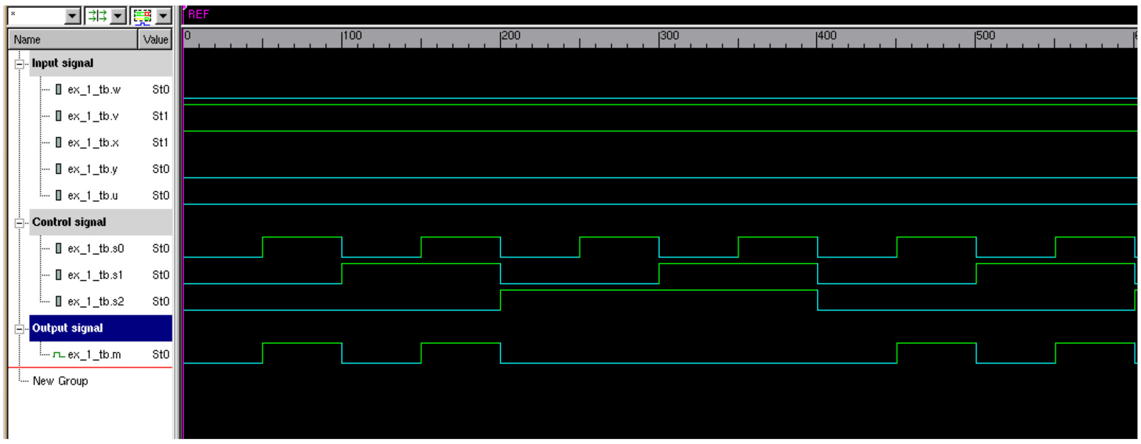
Log file of testbench:

```
{u, v, w, x, y}={0,1,0,1,0}, {(s2, s1, s0)}= 000, m=0
{u, v, w, x, y}={0,1,0,1,0}, {(s2, s1, s0)}= 001, m=1
{u, v, w, x, y}={0,1,0,1,0}, {(s2, s1, s0)}= 010, m=0
{u, v, w, x, y}={0,1,0,1,0}, {(s2, s1, s0)}= 011, m=1
{u, v, w, x, y}={0,1,0,1,0}, {(s2, s1, s0)}= 100, m=0
{u, v, w, x, y}={0,1,0,1,0}, {(s2, s1, s0)}= 101, m=0
{u, v, w, x, y}={0,1,0,1,0}, {(s2, s1, s0)}= 110, m=0
{u, v, w, x, y}={0,1,0,1,0}, {(s2, s1, s0)}= 111, m=0
{u, v, w, x, y}={0,1,0,1,0}, {(s2, s1, s0)}= 000, m=0
```

{u, v, w, x, y}={0,1,0,1,0}, {(s2, s1, s0)}= 001, m=1
{u, v, w, x, y}={0,1,0,1,0}, {(s2, s1, s0)}= 010, m=0
{u, v, w, x, y}={0,1,0,1,0}, {(s2, s1, s0)}= 011, m=1
{u, v, w, x, y}={0,1,0,1,0}, {(s2, s1, s0)}= 100, m=0
{u, v, w, x, y}={0,1,0,1,0}, {(s2, s1, s0)}= 101, m=0
{u, v, w, x, y}={0,1,0,1,0}, {(s2, s1, s0)}= 110, m=0
{u, v, w, x, y}={0,1,0,1,0}, {(s2, s1, s0)}= 111, m=0

Waveform:





Coding:

```
module mux2_1 (u, v, w, x, y, s0, s1, s2, m);

input   u;
input   v;
input   w;
input   x;
input   y;
input   s0;
input   s1;
input   s2;
output  m;

wire temp_0;
wire temp_1;
wire temp_2;

assign temp_0 = (s0==1'b1)?v:u;
assign temp_1 = (s0==1'b1)?x:w;
assign temp_2 = (s1==1'b1)?temp_1:temp_0;
assign m =  (s2==1'b1)?y:temp_2;

endmodule
```

Testbench:

```
`include "ex_1.v"
module ex_1_tb;

reg w;
reg v;
reg x;
reg y;
reg u;
reg s0;
reg s1;
reg s2;
wire m;

mux2_1 mux2_1_00 (.u(u), .v(v), .w(w), .x(x), .y(y), .s2(s2), .s1(s1), .s0(s0), .m(m));

initial begin
$monitor ("{u, v, w, x, y}={%b,%b,%b,%b,%b}, {(s2, s1, s0)}= %b%b%b, m=%b \n",
          u, v, w, x, y, s2, s1, s0, m);
end

initial begin
u=0;
v=1;
w=0;
x=1;
y=0;
s2=0;
s1=0;
s0=0;
#50;

s2=0;
s1=0;
s0=1;
#50;

s2=0;
s1=1;
s0=0;
#50;

s2=0;
s1=1;
s0=1;
#50;

s2=1;
s1=0;
s0=0;
#50;

s2=1;
s1=0;
s0=1;
#50;

s2=1;
s1=1;
s0=0;
#50;
```

```
s2=1;
s1=1;
s0=1;
#50;

s2=0;
s1=0;
s0=0;
#50;

s2=0;
s1=0;
s0=1;
#50;

s2=0;
s1=1;
s0=0;
#50;

s2=0;
s1=1;
s0=1;
#50;

s2=1;
s1=0;
s0=0;
#50;


s2=1;
s1=0;
s0=1;
#50;

s2=1;
s1=1;
s0=0;
#50;

s2=1;
s1=1;
s0=1;
#50;

s2=1;
s1=1;
s0=1;
#50;

s2=1;
s1=1;
s0=1;
#50;

s2=1;
s1=1;
s0=1;
#50;
```

```
s2=1;
s1=1;
s0=1;
#50;

s2=1;
s1=1;
s0=1;
#50;

s2=1;
s1=1;
s0=1;
#50;

$finish;
end

initial begin
   $vcdplusfile("exe_1.vpd");
   $vcdpluson();
end
endmodule
```

b)

Log file of testbench:

```
{u, v, w, x, y}={0,1,0,1,0}, {(s2, s1, s0)}= 000, m=0
{u, v, w, x, y}={0,1,0,1,0}, {(s2, s1, s0)}= 001, m=1
{u, v, w, x, y}={0,1,0,1,0}, {(s2, s1, s0)}= 010, m=0
{u, v, w, x, y}={0,1,0,1,0}, {(s2, s1, s0)}= 011, m=1
{u, v, w, x, y}={0,1,0,1,0}, {(s2, s1, s0)}= 100, m=0
{u, v, w, x, y}={0,1,0,1,0}, {(s2, s1, s0)}= 101, m=0
{u, v, w, x, y}={0,1,0,1,0}, {(s2, s1, s0)}= 110, m=0
{u, v, w, x, y}={0,1,0,1,0}, {(s2, s1, s0)}= 111, m=0
{u, v, w, x, y}={0,1,0,1,0}, {(s2, s1, s0)}= 000, m=0
{u, v, w, x, y}={0,1,0,1,0}, {(s2, s1, s0)}= 001, m=1
{u, v, w, x, y}={0,1,0,1,0}, {(s2, s1, s0)}= 010, m=0
{u, v, w, x, y}={0,1,0,1,0}, {(s2, s1, s0)}= 011, m=1
{u, v, w, x, y}={0,1,0,1,0}, {(s2, s1, s0)}= 100, m=0
{u, v, w, x, y}={0,1,0,1,0}, {(s2, s1, s0)}= 101, m=0
{u, v, w, x, y}={0,1,0,1,0}, {(s2, s1, s0)}= 110, m=0
{u, v, w, x, y}={0,1,0,1,0}, {(s2, s1, s0)}= 111, m=0
```

Coding:

Method 1:

```
module mux5_1 (u, v, w, x, y, s0, s1, s2, m);

input   u;
input   v;
input   w;
input   x;
input   y;
input   s0;
input   s1;
input   s2;
output  m;

wire [2:0] temp;
wire       m;
```

```
assign temp = ({s2, s1, s0});
assign m    = (temp == 3'b000)? u:
              (temp == 3'b001)? v:
              (temp == 3'b010)? w:
              (temp == 3'b011)? x:
              (temp == 3'b100)? y:1'b0;
endmodule
```

Method 2:

```
module mux5_1_1 (u, v, w, x, y, s0, s1, s2, m);

input   u;
input   v;
input   w;
input   x;
input   y;
input   s0;
input   s1;
input   s2;
output  m;

wire [2:0] temp;
reg        m;

assign temp = ({s2, s1, s0});

always @(*) begin
   case (temp)
   3'b000: begin
      m = u;
   end
   3'b001: begin
      m = v;
   end
   3'b010: begin
      m = w;
   end
   3'b011: begin
      m = x;
   end
   3'b100: begin
      m = y;
   end
   default: begin
      m = 1'b0;
   end
   endcase
end
endmodule
```

Method 3:

```
module mux5_1_0 (u, v, w, x, y, s0, s1, s2, m);

input   u;
input   v;
input   w;
input   x;
input   y;
input   s0;
input   s1;
input   s2;
output  m;

wire [2:0] temp;
reg        m;

assign temp = ({s2, s1, s0});

always @(*) begin
   if (temp == 3'b000) begin
      m = u;
   end
   else if (temp == 3'b001) begin
      m = v;
   end
   else if (temp == 3'b010) begin
      m = w;
   end
   else if (temp == 3'b011) begin
      m = x;
   end
   else if (temp == 3'b100) begin
      m = y;
   end
   else begin
      m = 0;
   end
end
endmodule
```

Testbench:

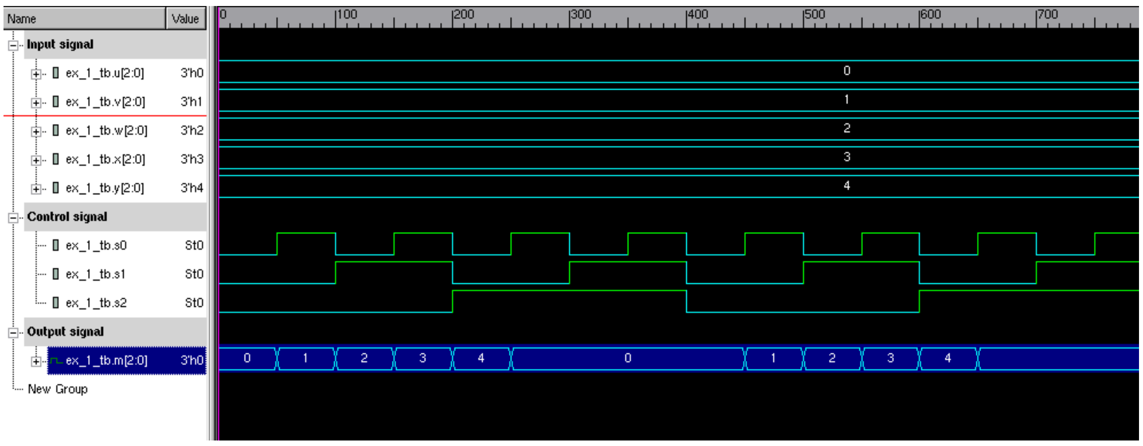
We can re-use testbench from a).

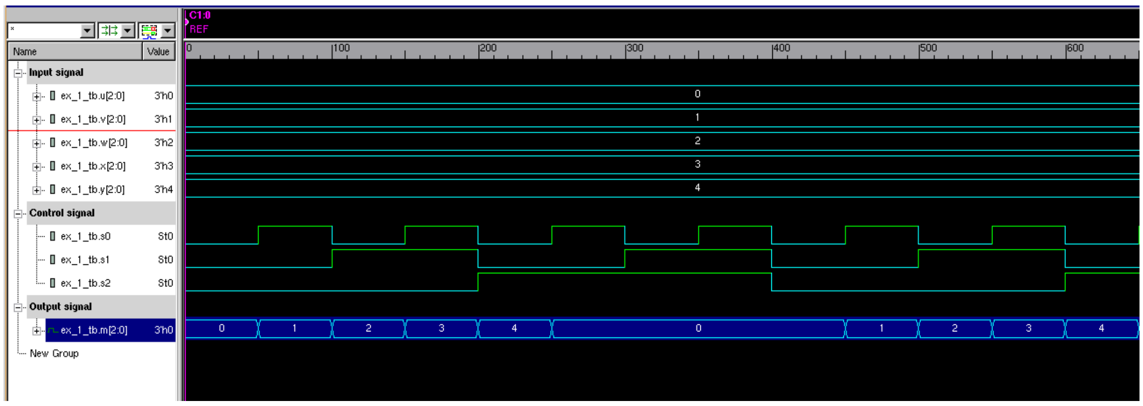
c)

Log file of testbench:

```
{u, v, w, x, y}={000,001,010,011,100}, {(s2, s1, s0)}= 000, m=000
{u, v, w, x, y}={000,001,010,011,100}, {(s2, s1, s0)}= 001, m=001
{u, v, w, x, y}={000,001,010,011,100}, {(s2, s1, s0)}= 010, m=010
{u, v, w, x, y}={000,001,010,011,100}, {(s2, s1, s0)}= 011, m=011
{u, v, w, x, y}={000,001,010,011,100}, {(s2, s1, s0)}= 100, m=100
{u, v, w, x, y}={000,001,010,011,100}, {(s2, s1, s0)}= 101, m=000
{u, v, w, x, y}={000,001,010,011,100}, {(s2, s1, s0)}= 110, m=000
{u, v, w, x, y}={000,001,010,011,100}, {(s2, s1, s0)}= 111, m=000
{u, v, w, x, y}={000,001,010,011,100}, {(s2, s1, s0)}= 000, m=000
{u, v, w, x, y}={000,001,010,011,100}, {(s2, s1, s0)}= 001, m=001
{u, v, w, x, y}={000,001,010,011,100}, {(s2, s1, s0)}= 010, m=010
{u, v, w, x, y}={000,001,010,011,100}, {(s2, s1, s0)}= 011, m=011
{u, v, w, x, y}={000,001,010,011,100}, {(s2, s1, s0)}= 100, m=100
{u, v, w, x, y}={000,001,010,011,100}, {(s2, s1, s0)}= 101, m=000
{u, v, w, x, y}={000,001,010,011,100}, {(s2, s1, s0)}= 110, m=000
{u, v, w, x, y}={000,001,010,011,100}, {(s2, s1, s0)}= 111, m=000
```

Waveform:





Coding:

Method 1:

```
module mux5_1_3bit (u, v, w, x, y, s0, s1, s2, m);

parameter DATA_WIDTH = 3;

input   [DATA_WIDTH - 1:0] u;
input   [DATA_WIDTH - 1:0] v;
input   [DATA_WIDTH - 1:0] w;
input   [DATA_WIDTH - 1:0] x;
input   [DATA_WIDTH - 1:0] y;
input   s0;
input   s1;
input   s2;
output  [DATA_WIDTH - 1:0] m;

wire    [DATA_WIDTH - 1:0] temp;
wire    [DATA_WIDTH - 1:0] m;

assign temp = ({s2, s1, s0});
assign m    = (temp == 3'b000)? u:
            (temp == 3'b001)? v:
            (temp == 3'b010)? w:
            (temp == 3'b011)? x:
            (temp == 3'b100)? y:3'b000;
endmodule
```

Method 2:

```
module mux5_1_3 (u, v, w, x, y, s0, s1, s2, m);
```

```
parameter DATA_WIDTH = 3;

input   [DATA_WIDTH - 1:0] u;
input   [DATA_WIDTH - 1:0] v;
input   [DATA_WIDTH - 1:0] w;
input   [DATA_WIDTH - 1:0] x;
input   [DATA_WIDTH - 1:0] y;
input   s0;
input   s1;
input   s2;
output  [DATA_WIDTH - 1:0] m;

wire    [DATA_WIDTH - 1:0] temp;
reg     [DATA_WIDTH - 1:0] m;
assign temp = ({s2, s1, s0});

always @(*) begin
   if (temp == 3'b000) begin
      m = u;
   end
   else if (temp == 3'b001) begin
      m = v;
   end
   else if (temp == 3'b010) begin
      m = w;
   end
   else if (temp == 3'b011) begin
      m = x;
   end
   else if (temp == 3'b100) begin
      m = y;
   end
   else begin
      m = 3'b000;
   end
end
endmodule
```

Method 3:

```
module mux5_1_4 (u, v, w, x, y, s0, s1, s2, m);

parameter DATA_WIDTH = 3;
input   [DATA_WIDTH - 1:0] u;
input   [DATA_WIDTH - 1:0] v;
input   [DATA_WIDTH - 1:0] w;
input   [DATA_WIDTH - 1:0] x;
input   [DATA_WIDTH - 1:0] y;
input   s0;
input   s1;
input   s2;
output  [DATA_WIDTH - 1:0] m;

wire    [DATA_WIDTH - 1:0] temp;
reg     [DATA_WIDTH - 1:0] m;
assign temp = ({s2, s1, s0});

always @(*) begin
   case (temp)
   3'b000: begin
      m = u;
   end
```

```
  3'b001: begin
    m = v;
  end
  3'b010: begin
    m = w;
  end
  3'b011: begin
    m = x;
  end
  3'b100: begin
    m = y;
  end
  default: begin
    m = 3'b000;
  end
  endcase
end
endmodule
```

Testbench:

```
`include "mux5_1_3bit.v"
module ex_1_tb;

parameter DATA_WIDTH = 3;
reg [DATA_WIDTH - 1:0] w;
reg [DATA_WIDTH - 1:0] v;
reg [DATA_WIDTH - 1:0] x;
reg [DATA_WIDTH - 1:0] y;
reg [DATA_WIDTH - 1:0] u;
reg s0;
reg s1;
reg s2;
wire [DATA_WIDTH - 1:0] m;

mux5_1_3bit mux5_1_3bit_00 (.u(u), .v(v), .w(w), .x(x), .y(y), .s2(s2), .s1(s1), .s0(s0), .m(m));

initial begin
$monitor ("{u, v, w, x, y}={%b,%b,%b,%b,%b}, {(s2, s1, s0)}= %b%b%b, m=%b \n",
        u, v, w, x, y, s2, s1, s0, m);
end

initial begin
u=3'b000;
v=3'b001;
w=3'b010;
x=3'b011;
y=3'b100;
s2=0;
s1=0;
s0=0;

#50;
s2=0;
s1=0;
s0=1;

#50;
s2=0;
s1=1;
s0=0;
```

```
#50;
s2=0;
s1=1;
s0=1;
#50;

s2=1;
s1=0;
s0=0;
#50;

s2=1;
s1=0;
s0=1;
#50;

s2=1;
s1=1;
s0=0;
#50;

s2=1;
s1=1;
s0=1;
#50;

s2=0;
s1=0;
s0=0;
#50;

s2=0;
s1=0;
s0=1;

#50;
s2=0;
s1=1;
s0=0;

#50;
s2=0;
s1=1;
s0=1;

#50;
s2=1;
s1=0;
s0=0;
#50;

s2=1;
s1=0;
s0=1;
#50;

s2=1;
s1=1;
s0=0;
#50;
```

```
s2=1;
s1=1;
s0=1;
#50;

s2=1;
s1=1;
s0=1;
#50;

s2=1;
s1=1;
s0=1;
#50;

s2=1;
s1=1;
s0=1;
#50;

s2=1;
s1=1;
s0=1;
#50;

s2=1;
s1=1;
s0=1;
#50;

s2=1;
s1=1;
s0=1;
#50;

$finish;
end
initial begin
   $vcdplusfile("exe_1.vpd");
   $vcdpluson();
end
endmodule
```

Chapter 2: Exercise 02

Requirement: Develop the following hardware with Verilog HDL.

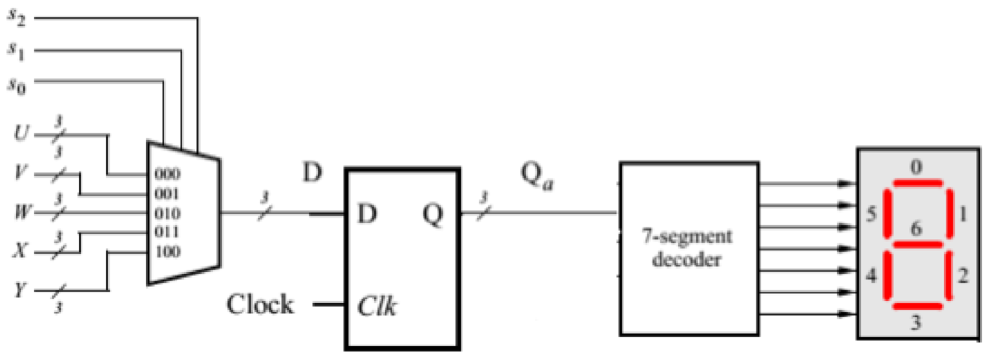


Figure 2.1 Ex_02 Design

With this figure, We have two solutions to solve it. The first solution is that we can use "always" structure and the second is that we can use "assign" statement. Refer to two following methods.

<u>Solution</u>

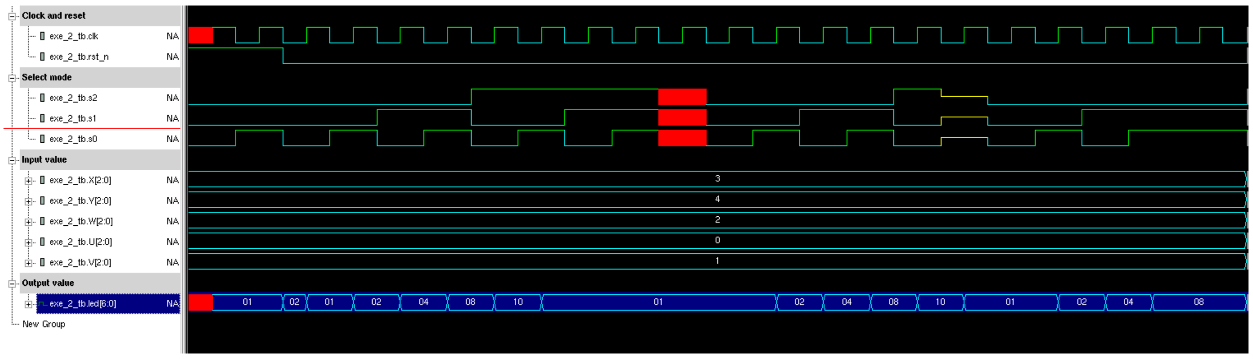
Method 1: Coding with "always" structure.

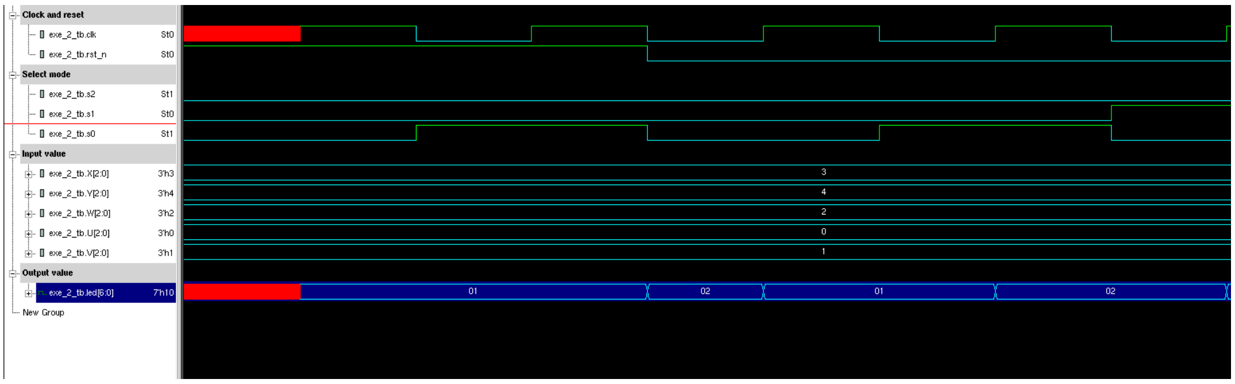
Log file of testbench:

```
rst_n=1, clk=x, s2s1s0=000, {X,Y,W,U,V}={011,100,010,000,001}, led=xxxxxxx
rst_n=1, clk=1, s2s1s0=000, {X,Y,W,U,V}={011,100,010,000,001}, led=0000001
rst_n=1, clk=0, s2s1s0=001, {X,Y,W,U,V}={011,100,010,000,001}, led=0000001
rst_n=1, clk=1, s2s1s0=001, {X,Y,W,U,V}={011,100,010,000,001}, led=0000001
rst_n=0, clk=0, s2s1s0=000, {X,Y,W,U,V}={011,100,010,000,001}, led=0000010
rst_n=0, clk=1, s2s1s0=000, {X,Y,W,U,V}={011,100,010,000,001}, led=0000001
rst_n=0, clk=0, s2s1s0=001, {X,Y,W,U,V}={011,100,010,000,001}, led=0000001
rst_n=0, clk=1, s2s1s0=001, {X,Y,W,U,V}={011,100,010,000,001}, led=0000010
rst_n=0, clk=0, s2s1s0=010, {X,Y,W,U,V}={011,100,010,000,001}, led=0000010
rst_n=0, clk=1, s2s1s0=010, {X,Y,W,U,V}={011,100,010,000,001}, led=0000100
rst_n=0, clk=0, s2s1s0=011, {X,Y,W,U,V}={011,100,010,000,001}, led=0000100
rst_n=0, clk=1, s2s1s0=011, {X,Y,W,U,V}={011,100,010,000,001}, led=0001000
rst_n=0, clk=0, s2s1s0=100, {X,Y,W,U,V}={011,100,010,000,001}, led=0001000
rst_n=0, clk=1, s2s1s0=100, {X,Y,W,U,V}={011,100,010,000,001}, led=0010000
rst_n=0, clk=0, s2s1s0=101, {X,Y,W,U,V}={011,100,010,000,001}, led=0010000
rst_n=0, clk=1, s2s1s0=101, {X,Y,W,U,V}={011,100,010,000,001}, led=0000001
rst_n=0, clk=0, s2s1s0=110, {X,Y,W,U,V}={011,100,010,000,001}, led=0000001
rst_n=0, clk=1, s2s1s0=110, {X,Y,W,U,V}={011,100,010,000,001}, led=0000001
rst_n=0, clk=0, s2s1s0=111, {X,Y,W,U,V}={011,100,010,000,001}, led=0000001
rst_n=0, clk=1, s2s1s0=111, {X,Y,W,U,V}={011,100,010,000,001}, led=0000001
rst_n=0, clk=0, s2s1s0=xxx, {X,Y,W,U,V}={011,100,010,000,001}, led=0000001
rst_n=0, clk=1, s2s1s0=xxx, {X,Y,W,U,V}={011,100,010,000,001}, led=0000001
rst_n=0, clk=0, s2s1s0=000, {X,Y,W,U,V}={011,100,010,000,001}, led=0000001
rst_n=0, clk=1, s2s1s0=000, {X,Y,W,U,V}={011,100,010,000,001}, led=0000001
rst_n=0, clk=0, s2s1s0=001, {X,Y,W,U,V}={011,100,010,000,001}, led=0000001
rst_n=0, clk=1, s2s1s0=001, {X,Y,W,U,V}={011,100,010,000,001}, led=0000010
rst_n=0, clk=0, s2s1s0=010, {X,Y,W,U,V}={011,100,010,000,001}, led=0000010
```

```
rst_n=0, clk=1, s2s1s0=010, {X,Y,W,U,V}={011,100,010,000,001}, led=0000100
rst_n=0, clk=0, s2s1s0=011, {X,Y,W,U,V}={011,100,010,000,001}, led=0000100
rst_n=0, clk=1, s2s1s0=011, {X,Y,W,U,V}={011,100,010,000,001}, led=0001000
rst_n=0, clk=0, s2s1s0=100, {X,Y,W,U,V}={011,100,010,000,001}, led=0001000
rst_n=0, clk=1, s2s1s0=100, {X,Y,W,U,V}={011,100,010,000,001}, led=0010000
rst_n=0, clk=0, s2s1s0=zzz, {X,Y,W,U,V}={011,100,010,000,001}, led=0010000
rst_n=0, clk=1, s2s1s0=zzz, {X,Y,W,U,V}={011,100,010,000,001}, led=0000001
rst_n=0, clk=0, s2s1s0=000, {X,Y,W,U,V}={011,100,010,000,001}, led=0000001
rst_n=0, clk=1, s2s1s0=000, {X,Y,W,U,V}={011,100,010,000,001}, led=0000001
rst_n=0, clk=0, s2s1s0=001, {X,Y,W,U,V}={011,100,010,000,001}, led=0000001
rst_n=0, clk=1, s2s1s0=001, {X,Y,W,U,V}={011,100,010,000,001}, led=0000010
rst_n=0, clk=0, s2s1s0=010, {X,Y,W,U,V}={011,100,010,000,001}, led=0000010
rst_n=0, clk=1, s2s1s0=010, {X,Y,W,U,V}={011,100,010,000,001}, led=0000100
rst_n=0, clk=0, s2s1s0=011, {X,Y,W,U,V}={011,100,010,000,001}, led=0000100
rst_n=0, clk=1, s2s1s0=011, {X,Y,W,U,V}={011,100,010,000,001}, led=0001000
rst_n=0, clk=0, s2s1s0=011, {X,Y,W,U,V}={011,100,010,000,001}, led=0001000
rst_n=0, clk=1, s2s1s0=011, {X,Y,W,U,V}={011,100,010,000,001}, led=0001000
rst_n=0, clk=0, s2s1s0=011, {X,Y,W,U,V}={011,100,010,000,001}, led=0001000
```

Waveform:





Coding:

```
module led_7 (clk, rst_n, s0, s1, s2, X, Y, W, U, V, led);

parameter DATA_IN_WIDTH = 3;
parameter DATA_OUT_WIDTH = 7;
input clk;
input rst_n;
input  s0;
input  s1;
input  s2;
input  [DATA_IN_WIDTH - 1:0]  X;
input  [DATA_IN_WIDTH - 1:0]  Y;
input  [DATA_IN_WIDTH - 1:0]  W;
input  [DATA_IN_WIDTH - 1:0]  U;
input  [DATA_IN_WIDTH - 1:0]  V;
```

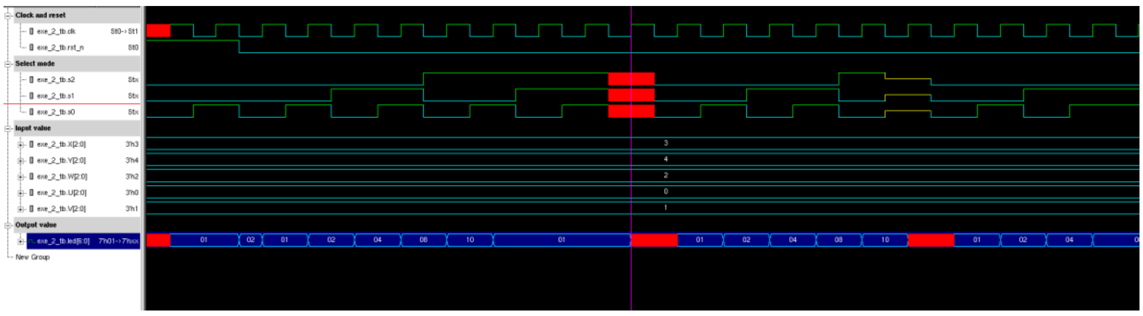
```
output [DATA_OUT_WIDTH - 1:0] led;

reg    [DATA_IN_WIDTH - 1:0]  D;
reg    [DATA_OUT_WIDTH - 1:0] Q;
reg    [DATA_OUT_WIDTH - 1:0] led;

always @(*) begin
   case ({s2,s1,s0})
      3'b000: begin
         D = U;
      end
      3'b001: begin
         D = V;
      end
      3'b010: begin
         D = W;
      end
      3'b011: begin
         D = X;
      end
      3'b100: begin
         D = Y;
      end
      default: begin
         D = 3'b000;
      end
   endcase
end

always @(posedge clk or negedge rst_n) begin
    if (rst_n) begin
      Q <= 0;
    end
    else begin
      Q <= D;
    end
end

always @(*) begin
   case (Q)
      3'b000: begin
         led = 7'b0000001;
      end
      3'b001: begin
         led = 7'b0000010;
      end
      3'b010: begin
         led = 7'b0000100;
      end
      3'b011: begin
         led = 7'b0001000;
      end
      3'b100: begin
         led = 7'b0010000;
      end
      3'b101: begin
         led = 7'b0100000;
      end
      3'b110: begin
         led = 7'b1000000;
      end
    default: begin
      led = 7'b0000000;
    end
  endcase
end
endmodule
```

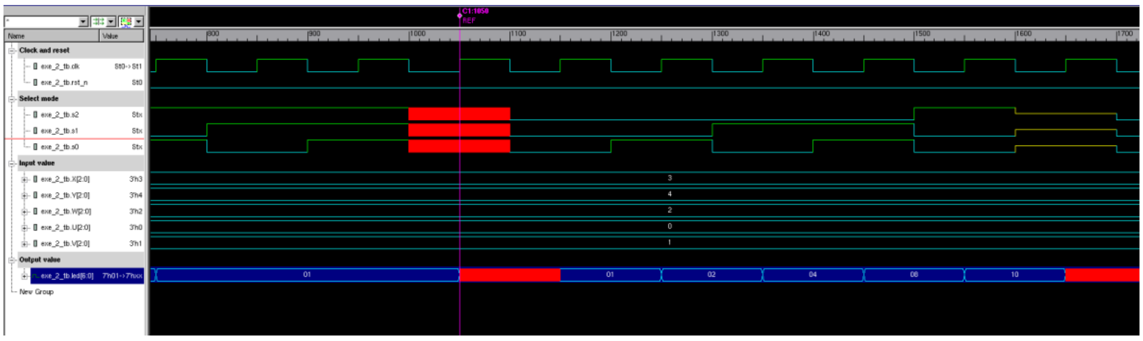
Method 2: Coding with "assign" statement.

Log file of testbench:

```
rst_n=1, clk=x, s2s1s0=000, {X,Y,W,U,V}={011,100,010,000,001}, led=xxxxxxx
rst_n=1, clk=1, s2s1s0=000, {X,Y,W,U,V}={011,100,010,000,001}, led=0000001
rst_n=1, clk=0, s2s1s0=001, {X,Y,W,U,V}={011,100,010,000,001}, led=0000001
rst_n=1, clk=1, s2s1s0=001, {X,Y,W,U,V}={011,100,010,000,001}, led=0000001
rst_n=0, clk=0, s2s1s0=000, {X,Y,W,U,V}={011,100,010,000,001}, led=0000010
rst_n=0, clk=1, s2s1s0=000, {X,Y,W,U,V}={011,100,010,000,001}, led=0000001
rst_n=0, clk=0, s2s1s0=001, {X,Y,W,U,V}={011,100,010,000,001}, led=0000001
rst_n=0, clk=1, s2s1s0=001, {X,Y,W,U,V}={011,100,010,000,001}, led=0000010
rst_n=0, clk=0, s2s1s0=010, {X,Y,W,U,V}={011,100,010,000,001}, led=0000010
rst_n=0, clk=1, s2s1s0=010, {X,Y,W,U,V}={011,100,010,000,001}, led=0000100
rst_n=0, clk=0, s2s1s0=011, {X,Y,W,U,V}={011,100,010,000,001}, led=0000100
rst_n=0, clk=1, s2s1s0=011, {X,Y,W,U,V}={011,100,010,000,001}, led=0001000
rst_n=0, clk=0, s2s1s0=100, {X,Y,W,U,V}={011,100,010,000,001}, led=0001000
rst_n=0, clk=1, s2s1s0=100, {X,Y,W,U,V}={011,100,010,000,001}, led=0010000
rst_n=0, clk=0, s2s1s0=101, {X,Y,W,U,V}={011,100,010,000,001}, led=0010000
rst_n=0, clk=1, s2s1s0=101, {X,Y,W,U,V}={011,100,010,000,001}, led=0000001
rst_n=0, clk=0, s2s1s0=110, {X,Y,W,U,V}={011,100,010,000,001}, led=0000001
rst_n=0, clk=1, s2s1s0=110, {X,Y,W,U,V}={011,100,010,000,001}, led=0000001
rst_n=0, clk=0, s2s1s0=111, {X,Y,W,U,V}={011,100,010,000,001}, led=0000001
rst_n=0, clk=1, s2s1s0=111, {X,Y,W,U,V}={011,100,010,000,001}, led=0000001
rst_n=0, clk=0, s2s1s0=xxx, {X,Y,W,U,V}={011,100,010,000,001}, led=0000001
rst_n=0, clk=1, s2s1s0=xxx, {X,Y,W,U,V}={011,100,010,000,001}, led=xxxxxxx
rst_n=0, clk=0, s2s1s0=000, {X,Y,W,U,V}={011,100,010,000,001}, led=xxxxxxx
rst_n=0, clk=1, s2s1s0=000, {X,Y,W,U,V}={011,100,010,000,001}, led=0000001
rst_n=0, clk=0, s2s1s0=001, {X,Y,W,U,V}={011,100,010,000,001}, led=0000001
rst_n=0, clk=1, s2s1s0=001, {X,Y,W,U,V}={011,100,010,000,001}, led=0000010
rst_n=0, clk=0, s2s1s0=010, {X,Y,W,U,V}={011,100,010,000,001}, led=0000010
rst_n=0, clk=1, s2s1s0=010, {X,Y,W,U,V}={011,100,010,000,001}, led=0000100
rst_n=0, clk=0, s2s1s0=011, {X,Y,W,U,V}={011,100,010,000,001}, led=0000100
rst_n=0, clk=1, s2s1s0=011, {X,Y,W,U,V}={011,100,010,000,001}, led=0001000
rst_n=0, clk=0, s2s1s0=100, {X,Y,W,U,V}={011,100,010,000,001}, led=0001000
rst_n=0, clk=1, s2s1s0=100, {X,Y,W,U,V}={011,100,010,000,001}, led=0010000
rst_n=0, clk=0, s2s1s0=zzz, {X,Y,W,U,V}={011,100,010,000,001}, led=0010000
rst_n=0, clk=1, s2s1s0=zzz, {X,Y,W,U,V}={011,100,010,000,001}, led=xxxxxxx
rst_n=0, clk=0, s2s1s0=000, {X,Y,W,U,V}={011,100,010,000,001}, led=xxxxxxx
rst_n=0, clk=1, s2s1s0=000, {X,Y,W,U,V}={011,100,010,000,001}, led=0000001
rst_n=0, clk=0, s2s1s0=001, {X,Y,W,U,V}={011,100,010,000,001}, led=0000001
rst_n=0, clk=1, s2s1s0=001, {X,Y,W,U,V}={011,100,010,000,001}, led=0000010
rst_n=0, clk=0, s2s1s0=010, {X,Y,W,U,V}={011,100,010,000,001}, led=0000010
rst_n=0, clk=1, s2s1s0=010, {X,Y,W,U,V}={011,100,010,000,001}, led=0000100
rst_n=0, clk=0, s2s1s0=011, {X,Y,W,U,V}={011,100,010,000,001}, led=0000100
rst_n=0, clk=1, s2s1s0=011, {X,Y,W,U,V}={011,100,010,000,001}, led=0001000
rst_n=0, clk=0, s2s1s0=011, {X,Y,W,U,V}={011,100,010,000,001}, led=0001000
rst_n=0, clk=1, s2s1s0=011, {X,Y,W,U,V}={011,100,010,000,001}, led=0001000
rst_n=0, clk=0, s2s1s0=011, {X,Y,W,U,V}={011,100,010,000,001}, led=0001000
```

Waveform:





Coding:

```
module led_7 (clk, rst_n, s0, s1, s2, X, Y, W, U, V, led);

//Define parameter
parameter DATA_IN_WIDTH  = 3;
parameter DATA_OUT_WIDTH = 7;

// Define input and output
input  clk;
input  rst_n;
input  s0;
input  s1;
input  s2;
input  [DATA_IN_WIDTH  - 1:0]  X;
input  [DATA_IN_WIDTH  - 1:0]  Y;
input  [DATA_IN_WIDTH  - 1:0]  W;
input  [DATA_IN_WIDTH  - 1:0]  U;
input  [DATA_IN_WIDTH  - 1:0]  V;
output [DATA_OUT_WIDTH - 1:0]  led;
wire   [DATA_IN_WIDTH  - 1:0]  D;
wire   [DATA_IN_WIDTH  - 1:0]  temp;
reg    [DATA_OUT_WIDTH - 1:0]  Q;

assign temp = ({s2, s1, s0});
assign D = (temp == 3'b000)? U:
           (temp == 3'b001)? V:
           (temp == 3'b010)? W:
           (temp == 3'b011)? X:
           (temp == 3'b100)? Y:3'b000;
always @ (posedge clk or negedge rst_n) begin
   if (rst_n) begin
      Q <= 0;
   end
   else begin
      Q <= D;
   end
end
```

```
assign led = (Q == 3'b000)? 7'b0000001:
            (Q == 3'b001)? 7'b0000010:
            (Q == 3'b010)? 7'b0000100:
            (Q == 3'b011)? 7'b0001000:
            (Q == 3'b100)? 7'b0010000:
            (Q == 3'b101)? 7'b0100000:
            (Q == 3'b110)? 7'b1000000:7'b0000000;
endmodule
```

Testbench for two methods:

```
module exe_2_tb;

parameter DATA_IN_WIDTH = 3;
parameter DATA_OUT_WIDTH = 7;
parameter CYCLE = 50;
reg  clk;
reg  rst_n;
reg  s0;
reg  s1;
reg  s2;
reg  [DATA_IN_WIDTH - 1:0]  X;
reg  [DATA_IN_WIDTH - 1:0]  Y;
reg  [DATA_IN_WIDTH - 1:0]  W;
reg  [DATA_IN_WIDTH - 1:0]  U;
reg  [DATA_IN_WIDTH - 1:0]  V;
wire [DATA_OUT_WIDTH - 1:0] led;

always begin
   #CYCLE;
   clk=1;
   #CYCLE;
   clk=0;
end

led_7 led_7_00 (.clk(clk), .rst_n(rst_n), .s0(s0), .s1(s1), .s2(s2), .X(X), .Y(Y), .W(W), .U(U), .V(V),
.led(led));

initial begin

$monitor ("rst_n=%b, clk=%b, s2s1s0=%b%b%b, {X,Y,W,U,V}={%b,%b,%b,%b,%b}, led=%b \n",
          rst_n, clk, s2, s1, s0, X, Y, W, U, V, led);
end

initial begin

   rst_n=1;
   s2=0;
   s1=0;
   s0=0;
   U=3'b000;
   V=3'b001;
   W=3'b010;
   X=3'b011;
   Y=3'b100;

   #100;
   s2=0;
   s1=0;
   s0=1;
```

```
#100;
rst_n=0;
s2=0;
s1=0;
s0=0;

#100;
s2=0;
s1=0;
s0=1;

#100;
s2=0;
s1=1;
s0=0;

#100;
s2=0;
s1=1;
s0=1;
#100;
s2=1;
s1=0;
s0=0;

#100;
s2=1;
s1=0;
s0=1;

#100;
s2=1;
s1=1;
s0=0;

#100;
s2=1;
s1=1;
s0=1;

#100;
s2=1'bx;
s1=1'bx;
s0=1'bx;

#100;
s2=0;
s1=0;
s0=0;

#100;
s2=0;
s1=0;
s0=1;

#100;
s2=0;
s1=1;
s0=0;

#100;
```

```
   s2=0;
   s1=1;
   s0=1;

   #100;
   s2=1;
   s1=0;
   s0=0;

   #100;
   s2=1'bz;
   s1=1'bz;
   s0=1'bz;

   #100;
   s2=0;
   s1=0;
   s0=0;

   #100;
   s2=0;
   s1=0;
   s0=1;

   #100;
   s2=0;
   s1=1;
   s0=0;

   #100;
   rst_n=2;
   s2=0;
   s1=1;
   s0=1;

   #250;
   $finish;
end
initial begin
   $vcdplusfile("exe_2.vpd");
   $vcdpluson();
end
endmodule
```

Chapter 3: Exercise 03

Requirement: Develop the following hardware with Verilog HDL.

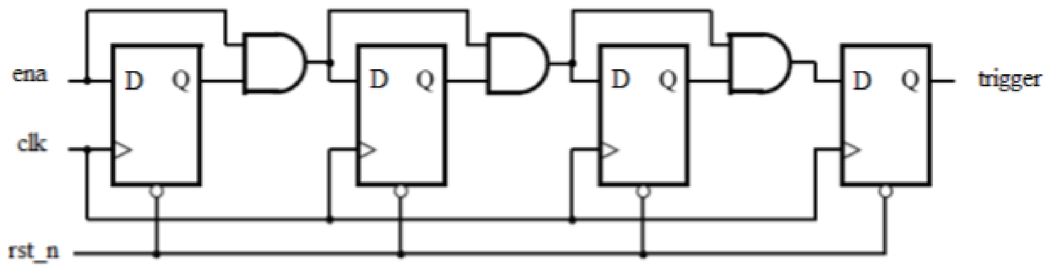


Figure 3.1 Ex_03 Design

Input: ena, clk, rst_n

Output: trigger

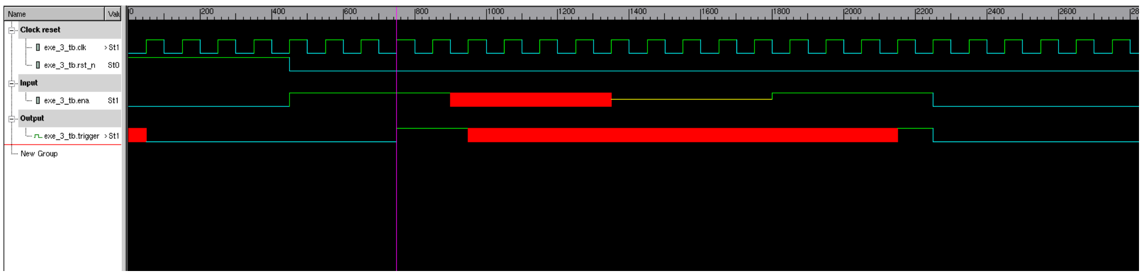
<u>Solution</u>

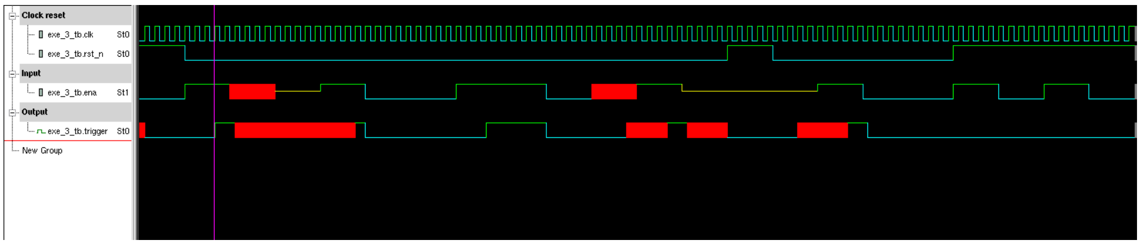
Log file of testbench:

```
time:                    0, ena:0, rst_n:1, clk:0, trigger:x
time:                   50, ena:0, rst_n:1, clk:1, trigger:0
time:                  100, ena:0, rst_n:1, clk:0, trigger:0
time:                  150, ena:0, rst_n:1, clk:1, trigger:0
time:                  200, ena:0, rst_n:1, clk:0, trigger:0
time:                  250, ena:0, rst_n:1, clk:1, trigger:0
time:                  300, ena:0, rst_n:1, clk:0, trigger:0
time:                  350, ena:0, rst_n:1, clk:1, trigger:0
time:                  400, ena:0, rst_n:1, clk:0, trigger:0
time:                  450, ena:1, rst_n:0, clk:1, trigger:0
time:                  500, ena:1, rst_n:0, clk:0, trigger:0
time:                  550, ena:1, rst_n:0, clk:1, trigger:0
time:                  600, ena:1, rst_n:0, clk:0, trigger:0
time:                  650, ena:1, rst_n:0, clk:1, trigger:0
time:                  700, ena:1, rst_n:0, clk:0, trigger:0
time:                  750, ena:1, rst_n:0, clk:1, trigger:1
time:                  800, ena:1, rst_n:0, clk:0, trigger:1
time:                  850, ena:1, rst_n:0, clk:1, trigger:1
time:                  900, ena:x, rst_n:0, clk:0, trigger:1
time:                  950, ena:x, rst_n:0, clk:1, trigger:x
time:                 1000, ena:x, rst_n:0, clk:0, trigger:x
time:                 1050, ena:x, rst_n:0, clk:1, trigger:x
time:                 1100, ena:x, rst_n:0, clk:0, trigger:x
time:                 1150, ena:x, rst_n:0, clk:1, trigger:x
time:                 1200, ena:x, rst_n:0, clk:0, trigger:x
time:                 1250, ena:x, rst_n:0, clk:1, trigger:x
time:                 1300, ena:x, rst_n:0, clk:0, trigger:x
time:                 1350, ena:z, rst_n:0, clk:1, trigger:x
time:                 1400, ena:z, rst_n:0, clk:0, trigger:x
time:                 1450, ena:z, rst_n:0, clk:1, trigger:x
time:                 1500, ena:z, rst_n:0, clk:0, trigger:x
time:                 1550, ena:z, rst_n:0, clk:1, trigger:x
time:                 1600, ena:z, rst_n:0, clk:0, trigger:x
time:                 1650, ena:z, rst_n:0, clk:1, trigger:x
```

```
time:          1700, ena:z, rst_n:0, clk:0, trigger:x
time:          1750, ena:z, rst_n:0, clk:1, trigger:x
time:          1800, ena:1, rst_n:0, clk:0, trigger:x
time:          1850, ena:1, rst_n:0, clk:1, trigger:x
time:          1900, ena:1, rst_n:0, clk:0, trigger:x
time:          1950, ena:1, rst_n:0, clk:1, trigger:x
time:          2000, ena:1, rst_n:0, clk:0, trigger:x
time:          2050, ena:1, rst_n:0, clk:1, trigger:x
time:          2100, ena:1, rst_n:0, clk:0, trigger:x
time:          2150, ena:1, rst_n:0, clk:1, trigger:1
time:          2200, ena:1, rst_n:0, clk:0, trigger:1
time:          2250, ena:0, rst_n:0, clk:1, trigger:0
time:          2300, ena:0, rst_n:0, clk:0, trigger:0
time:          2350, ena:0, rst_n:0, clk:1, trigger:0
time:          2400, ena:0, rst_n:0, clk:0, trigger:0
time:          2450, ena:0, rst_n:0, clk:1, trigger:0
time:          2500, ena:0, rst_n:0, clk:0, trigger:0
time:          2550, ena:0, rst_n:0, clk:1, trigger:0
time:          2600, ena:0, rst_n:0, clk:0, trigger:0
time:          2650, ena:0, rst_n:0, clk:1, trigger:0
time:          2700, ena:0, rst_n:0, clk:0, trigger:0
time:          2750, ena:0, rst_n:0, clk:1, trigger:0
time:          2800, ena:0, rst_n:0, clk:0, trigger:0
time:          2850, ena:0, rst_n:0, clk:1, trigger:0
time:          2900, ena:0, rst_n:0, clk:0, trigger:0
time:          2950, ena:0, rst_n:0, clk:1, trigger:0
time:          3000, ena:0, rst_n:0, clk:0, trigger:0
time:          3050, ena:0, rst_n:0, clk:1, trigger:0
time:          3100, ena:0, rst_n:0, clk:0, trigger:0
time:          3150, ena:1, rst_n:0, clk:1, trigger:0
time:          3200, ena:1, rst_n:0, clk:0, trigger:0
time:          3250, ena:1, rst_n:0, clk:1, trigger:0
time:          3300, ena:1, rst_n:0, clk:0, trigger:0
time:          3350, ena:1, rst_n:0, clk:1, trigger:0
time:          3400, ena:1, rst_n:0, clk:0, trigger:0
time:          3450, ena:1, rst_n:0, clk:1, trigger:1
time:          3500, ena:1, rst_n:0, clk:0, trigger:1
time:          3550, ena:1, rst_n:0, clk:1, trigger:1
time:          3600, ena:1, rst_n:0, clk:0, trigger:1
time:          3650, ena:1, rst_n:0, clk:1, trigger:1
time:          3700, ena:1, rst_n:0, clk:0, trigger:1
time:          3750, ena:1, rst_n:0, clk:1, trigger:1
time:          3800, ena:1, rst_n:0, clk:0, trigger:1
time:          3850, ena:1, rst_n:0, clk:1, trigger:1
time:          3900, ena:1, rst_n:0, clk:0, trigger:1
time:          3950, ena:1, rst_n:0, clk:1, trigger:1
time:          4000, ena:1, rst_n:0, clk:0, trigger:1
……………………
```

Waveform:





Coding:

```
module trigger_signal (clk, rst_n, ena, trigger);

// Define variable
input ena;
input clk;
input rst_n;
output trigger;

wire trigger_0;
wire trigger_1;
wire trigger_2;
reg  trigger;

dff_and dff_and00 (.ena(ena), .clk(clk), .rst_n(rst_n), .trigger(trigger_0));
dff_and dff_and01 (.ena(trigger_0), .clk(clk), .rst_n(rst_n), .trigger(trigger_1));
dff_and dff_and10 (.ena(trigger_1), .clk(clk), .rst_n(rst_n), .trigger(trigger_2));

always @(posedge clk or negedge rst_n) begin
  if (rst_n) begin
    trigger <= 0;
  end
  else begin
    trigger <= trigger_2;
  end
end
endmodule

module dff_and (clk, rst_n, ena, trigger);

input ena;
input clk;
input rst_n;
output trigger;
reg Q;

always @(posedge clk or negedge rst_n) begin
  if (rst_n) begin
    Q <= 0;
  end
  else begin
    Q <= ena;
  end
end

assign trigger = Q&ena;
endmodule
```

Testbench:

```
`include "exercise_03.v"
module exe_3_tb;

parameter CYCLE = 50;
reg ena;
reg clk;
reg rst_n;
wire  trigger;

always begin
clk=0;
#CYCLE;
clk=1;
#CYCLE;
end

trigger_signal trigger_signal_00 (.ena(ena), .clk(clk), .rst_n(rst_n), .trigger(trigger));

initial begin
$monitor("time:%d, ena:%b, rst_n:%b, clk:%b, trigger:%b \n",
        $time, ena, rst_n, clk, trigger);
end

initial begin
   ena=1'b0;
   rst_n=1;
   #450;
   ena=1'b1;
   rst_n=0;
   #450;
   ena=1'bx;
   rst_n=0;
   #450;
   ena=1'bz;
   rst_n=0;
   #450;
   ena=1'b1;
   rst_n=0;
   #450;
   ena=1'b0;
   rst_n=0;
   #450;
   ena=1'b0;
   rst_n=0;
   #450;
   ena=1'b1;
   rst_n=0;
   #450;
   ena=1'b1;
   rst_n=0;
   #450;
   ena=1'b0;
   rst_n=0;
   #450;
   ena=1'bx;
   rst_n=0;
   #450;
   ena=1'b1;
   rst_n=0;
```

```
  #450;
  ena=1'bz;
  rst_n=0;
  #450;
  ena=1'bz;
  rst_n=1;
  #450;
  ena=1'bz;
  rst_n=0;
  #450;
  ena=1'b1;
  rst_n=0;
  #450;
  ena=1'b0;
  rst_n=0;
  #450;
  ena=1'b0;
  rst_n=0;
  #450;
  ena=1'b1;
  rst_n=1;
  #450;
  ena=1'b0;
  rst_n=1;
  #450;
  ena=1'b1;
  rst_n=1;
  #450;
  ena=1'b0;
  rst_n=1;
  #450;
  $finish;
end
initial begin
  $vcdplusfile("exe_3.vpd");
  $vcdpluson();
end
endmodule
```

Chapter 4: Exercise 04

Requirement: Develop the following hardware (4 bit Addition) with Verilog HDL.

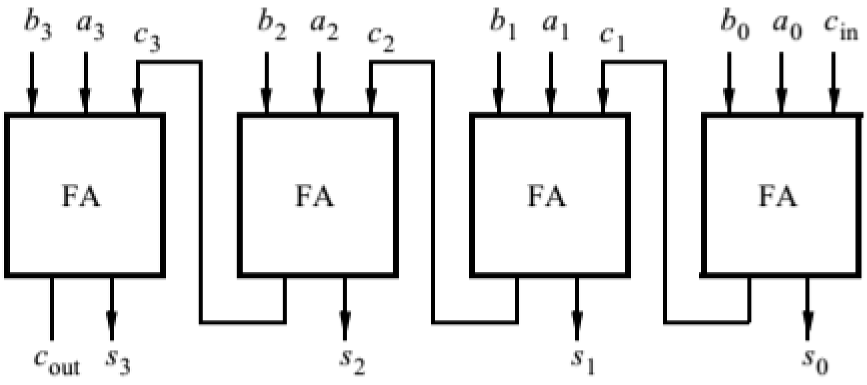


Figure 4.1 Ex_04 Design

With this requirement, we have two methods.

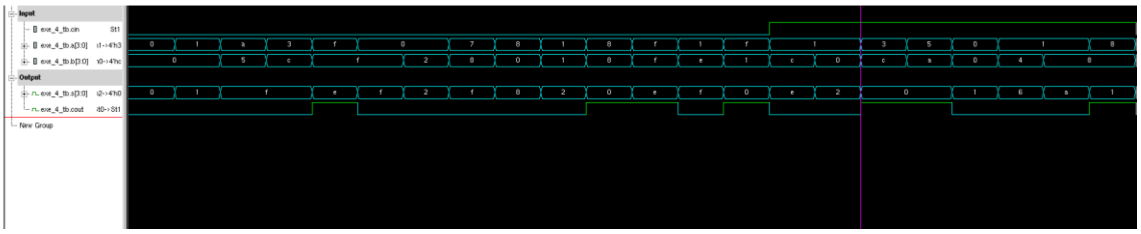
<u>Solution</u>

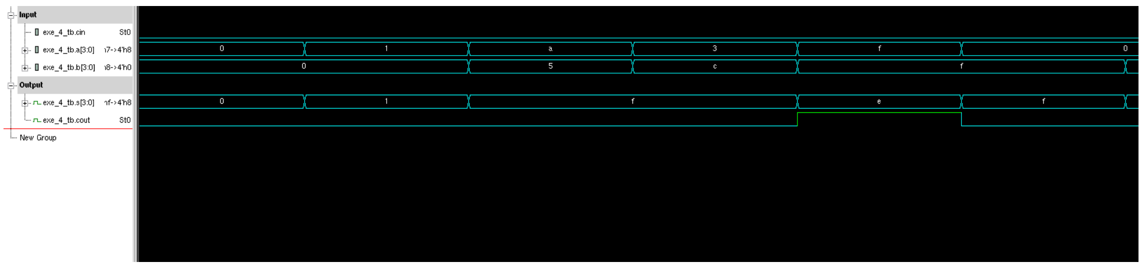
Method 1:

Log file of testbench:

cin=0, a=0000, b=0000, s=0000, cout=0
cin=0, a=0001, b=0000, s=0001, cout=0
cin=0, a=1010, b=0101, s=1111, cout=0
cin=0, a=0011, b=1100, s=1111, cout=0
cin=0, a=1111, b=1111, s=1110, cout=1
cin=0, a=0000, b=1111, s=1111, cout=0
cin=0, a=0000, b=0010, s=0010, cout=0
cin=0, a=0111, b=1000, s=1111, cout=0
cin=0, a=1000, b=0000, s=1000, cout=0
cin=0, a=0001, b=0001, s=0010, cout=0
cin=0, a=1000, b=1000, s=0000, cout=1
cin=0, a=1111, b=1111, s=1110, cout=1
cin=0, a=0001, b=1110, s=1111, cout=0
cin=0, a=1111, b=0001, s=0000, cout=1
cin=1, a=0001, b=1100, s=1110, cout=0
cin=1, a=0001, b=0000, s=0010, cout=0
cin=1, a=0011, b=1100, s=0000, cout=1
cin=1, a=0101, b=1010, s=0000, cout=1
cin=1, a=0000, b=0000, s=0001, cout=0
cin=1, a=0001, b=0100, s=0110, cout=0
cin=1, a=0001, b=1000, s=1010, cout=0
cin=1, a=1000, b=1000, s=0001, cout=1

Waveform:





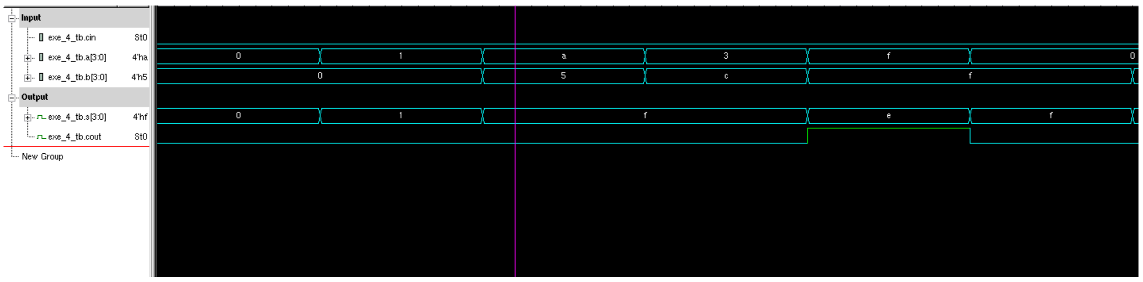
Coding:

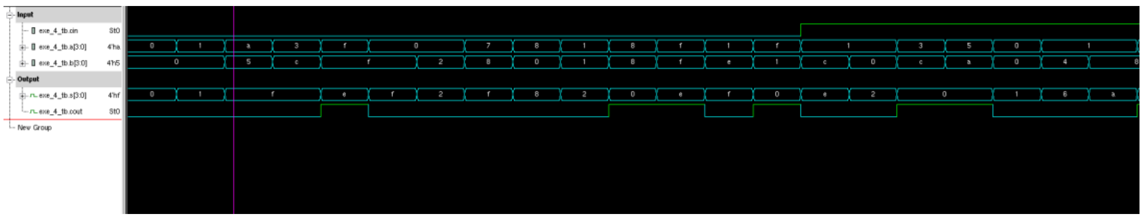
```
module add_4bit (cin, a, b, cout, s)
parameter  DATA_WIDTH = 4;

input   cin;
input   [DATA_WIDTH - 1:0] a;
input   [DATA_WIDTH - 1:0] b;
output  cout;
output  [DATA_WIDTH - 1:0] s;

wire c1;
wire c2;
wire c3;

add_1bit add_1bit_00 (.a(a[0]), .b(b[0]), .cin(cin), .s(s[0]), .cout(c1));
add_1bit add_1bit_01 (.a(a[1]), .b(b[1]), .cin(c1), .s(s[1]), .cout(c2));
add_1bit add_1bit_10 (.a(a[2]), .b(b[2]), .cin(c2), .s(s[2]), .cout(c3));
add_1bit add_1bit_11 (.a(a[3]), .b(b[3]), .cin(c3), .s(s[3]), .cout(cout));
endmodule

module add_1bit (a, b, cin, s, cout);

//Define variable
input  a;
input  b;
input  cin;
output s;
output cout;

reg    s;
wire   cout;

always @(*) begin
  case ({a, b, cin})
  3'b000: begin
    s = 0;
  end
  3'b001: begin
    s = 1;
  end
  3'b010: begin
    s = 1;
  end
  3'b011: begin
    s = 0;
  end
  3'b100: begin
    s = 1;
  end
  3'b101: begin
    s = 0;
  end
  3'b110: begin
    s = 0;
  end
  3'b111: begin
    s = 1;
  end
  default: begin
    s = 0;
  end
  endcase
end
assign cout = (a&b) + (b&cin) + (cin&a);
endmodule
```

Method 2:
Log file of test bench:

```
cin=0, a=0000, b=0000, s=0000, cout=0
cin=0, a=0001, b=0000, s=0001, cout=0
cin=0, a=1010, b=0101, s=1111, cout=0
cin=0, a=0011, b=1100, s=1111, cout=0
cin=0, a=1111, b=1111, s=1110, cout=1
cin=0, a=0000, b=1111, s=1111, cout=0
cin=0, a=0000, b=0010, s=0010, cout=0
cin=0, a=0111, b=1000, s=1111, cout=0
cin=0, a=1000, b=0000, s=1000, cout=0
cin=0, a=0001, b=0001, s=0010, cout=0
cin=0, a=1000, b=1000, s=0000, cout=1
cin=0, a=1111, b=1111, s=1110, cout=1
cin=0, a=0001, b=1110, s=1111, cout=0
cin=0, a=1111, b=0001, s=0000, cout=1
cin=1, a=0001, b=1100, s=1110, cout=0
cin=1, a=0001, b=0000, s=0010, cout=0
cin=1, a=0011, b=1100, s=0000, cout=1
cin=1, a=0101, b=1010, s=0000, cout=1
cin=1, a=0000, b=0000, s=0001, cout=0
cin=1, a=0001, b=0100, s=0110, cout=0
cin=1, a=0001, b=1000, s=1010, cout=0
cin=1, a=1000, b=1000, s=0001, cout=1
```

Waveform:





Coding:

```
module add_4bit (cin, a, b, cout, s);
```

```
parameter DATA_WIDTH = 4;
input   cin;
input  [DATA_WIDTH - 1:0]  a;
input  [DATA_WIDTH - 1:0]  b;
output cout;
output [DATA_WIDTH - 1:0]  s;

wire c1;
wire c2;
wire c3;
wire [DATA_WIDTH - 1:0]  s;
wire cout;

assign s[0]   = (!a[0]&b[0]&!cin)|(a[0]&!b[0]&!cin)|(!a[0]&!b[0]&cin)|(a[0]&b[0]&cin);
assign c1   = (a[0]&b[0]) + (b[0]&cin) + (cin&a[0]);
assign s[1]   = (!a[1]&b[1]&!c1)|(a[1]&!b[1]&!c1)|(!a[1]&!b[1]&c1)|(a[1]&b[1]&c1);
assign c2   = (a[1]&b[1]) + (b[1]&c1) + (c1&a[1]);
assign s[2]   = (!a[2]&b[2]&!c2)|(a[2]&!b[2]&!c2)|(!a[2]&!b[2]&c2)|(a[2]&b[2]&c2);
assign c3   = (a[2]&b[2]) + (b[2]&c2) + (c2&a[2]);
assign s[3]   = (!a[3]&b[3]&!c3)|(a[3]&!b[3]&!c3)|(!a[3]&!b[3]&c3)|(a[3]&b[3]&c3);
assign cout = (a[3]&b[3]) + (b[3]&c3) + (c3&a[3]);
endmodule
```

Testbench for two methods:

```
`include "exercise_04_other.v"
module exe_4_tb;

parameter DATA_WIDTH = 4;
parameter CYCLE      = 10;
reg   [DATA_WIDTH - 1:0] a;
reg   [DATA_WIDTH - 1:0] b;
wire  [DATA_WIDTH - 1:0] s;
wire  cout;
reg   cin;

add_4bit add_4bit_00 (.cin(cin), .a(a), .b(b), .s(s), .cout(cout));

initial begin
$monitor("cin=%b, a=%b, b=%b, s=%b, cout=%b \n",
        cin, a, b, s, cout);
end

initial begin
   cin=1'b0;
   a=4'b0000;
   b=4'b0000;
   #CYCLE;

   cin=1'b0;
   a=4'b0001;
   b=4'b0000;
   #CYCLE;

   cin=1'b0;
   a=4'b1010;
   b=4'b0101;
   #CYCLE;

   cin=1'b0;
   a=4'b0011;
```

```
b=4'b1100;
#CYCLE;

cin=1'b0;
a=4'b1111;
b=4'b1111;
#CYCLE;

cin=1'b0;
a=4'b0000;
b=4'b1111;
#CYCLE;

cin=1'b0;
a=4'b0000;
b=4'b0010;
#CYCLE;

cin=1'b0;
a=4'b0111;
b=4'b1000;
#CYCLE;

cin=1'b0;
a=4'b1000;
b=4'b0000;
#CYCLE;

cin=1'b0;
a=4'b0001;
b=4'b0001;
#CYCLE;

cin=1'b0;
a=4'b1000;
b=4'b1000;
#CYCLE;

cin=1'b0;
a=4'b1111;
b=4'b1111;
#CYCLE;

cin=1'b0;
a=4'b0001;
b=4'b1110;
#CYCLE;

cin=1'b0;
a=4'b1111;
b=4'b0001;
#CYCLE;

cin=1'b1;
a=4'b0001;
b=4'b1100;
#CYCLE;

cin=1'b1;
a=4'b0001;
b=4'b0000;
```

```
  #CYCLE;

  cin=1'b1;
  a=4'b0011;
  b=4'b1100;
  #CYCLE;

  cin=1'b1;
  a=4'b0101;
  b=4'b1010;
  #CYCLE;

  cin=1'b1;
  a=4'b0000;
  b=4'b0000;
  #CYCLE;

  cin=1'b1;
  a=4'b0001;
  b=4'b0100;
  #CYCLE;

  cin=1'b1;
  a=4'b0001;
  b=4'b1000;
  #CYCLE;

  cin=1'b1;
  a=4'b1000;
  b=4'b1000;
  #CYCLE;
  $finish;
end

initial begin
  $vcdplusfile("exe_4_tb.vpd");
  $vcdpluson();
end
endmodule
```

Chapter 5: Exercise 05

Requirement: Develop the following hardware (4 bit Multiplier) with Verilog HDL.

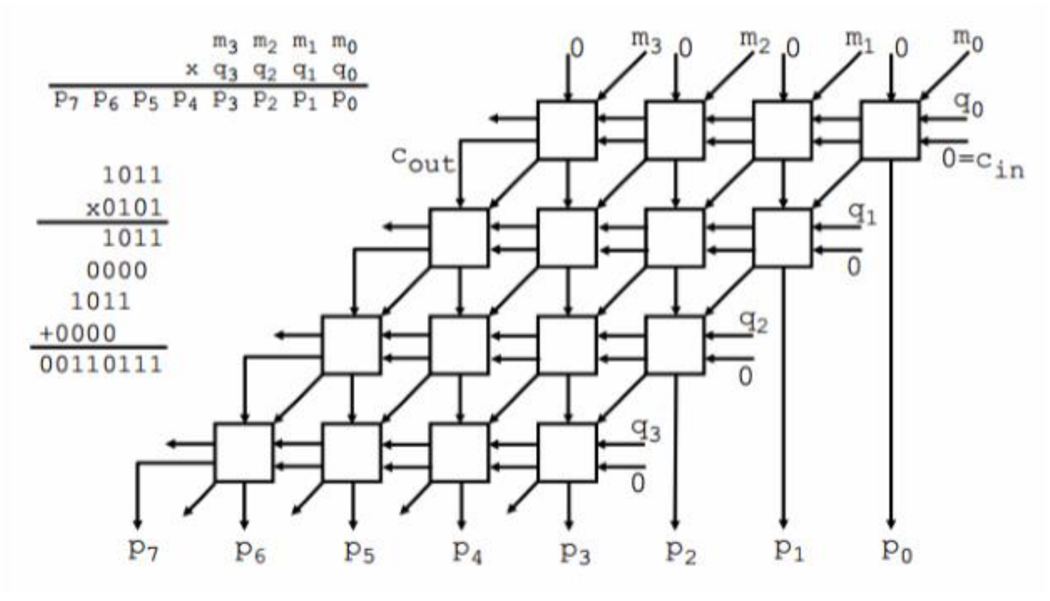


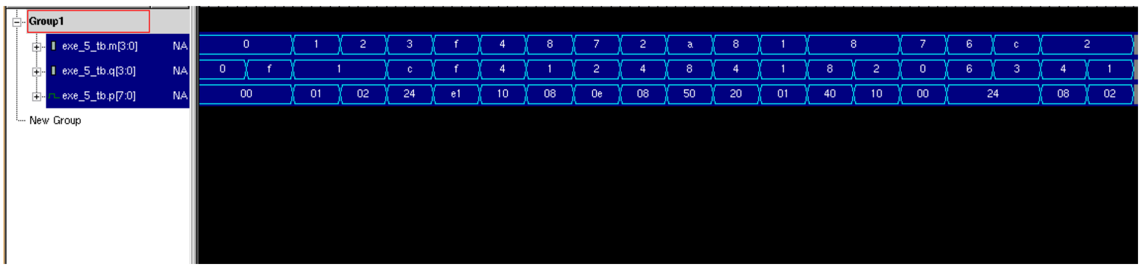
Figure 5.1 Ex_05 Design

<u>Solution</u>

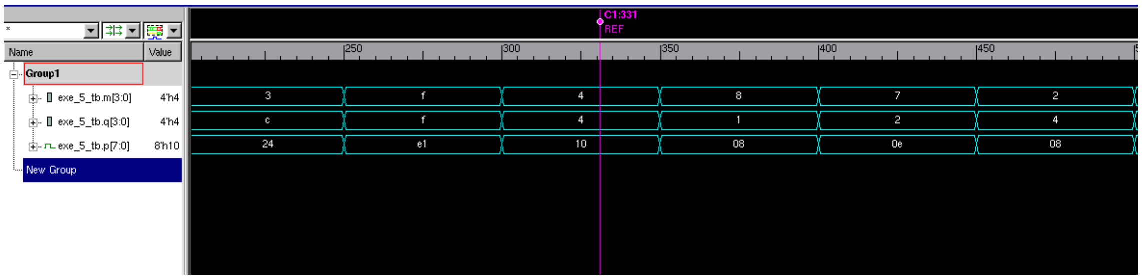
Log file of testbench:

```
m=0000, q=0000, p=00000000
m=0000, q=1111, p=00000000
m=0001, q=0001, p=00000001
m=0010, q=0001, p=00000010
m=0011, q=1100, p=00100100
m=1111, q=1111, p=11100001
m=0100, q=0100, p=00010000
m=1000, q=0001, p=00001000
m=0111, q=0010, p=00001110
m=0010, q=0100, p=00001000
m=1010, q=1000, p=01010000
m=1000, q=0100, p=00100000
m=0001, q=0001, p=00000001
m=1000, q=1000, p=01000000
m=1000, q=0010, p=00010000
m=0111, q=0000, p=00000000
m=0110, q=0110, p=00100100
m=1100, q=0011, p=00100100
m=0010, q=0100, p=00001000
m=0010, q=0001, p=00000010
```

Waveform:



Group1
exe_5_tb.m[3:0] NA
exe_5_tb.q[3:0] NA
exe_5_tb.p[7:0] NA
New Group
0 1 2 3 f 4 8 7 2 a 8 1 8 7 6 c 2
0 f 1 c f 4 1 2 4 8 4 1 8 2 0 6 3 4 1
00 01 02 24 e1 10 08 0e 08 50 20 01 40 10 00 24 08 02



Coding:

```
module mul_4bits(m, q, p);
// inputs and outputs
input   [3:0] m, q;
output [7:0] p;
wire a1,a2,a3,a4,b1,b2,b3,b4,c1,c2,c3,c4,d1,d2,d3,d4;
wire o2,o3,o4,o6,o7,o8,o10,o11,o12;
wire r1,r2,r3,r4,r5,r6,r7,r8,r9,r10,r11,r12,r13,r14,r15;
assign a1=m[0]&q[0];
assign a2=m[1]&q[0];
assign a3=m[2]&q[0];
assign a4=m[3]&q[0];
assign b1=m[0]&q[1];
assign b2=m[1]&q[1];
assign b3=m[2]&q[1];
assign b4=m[3]&q[1];
assign c1=m[0]&q[2];
assign c2=m[1]&q[2];
assign c3=m[2]&q[2];
assign c4=m[3]&q[2];
assign d1=m[0]&q[3];
assign d2=m[1]&q[3];
assign d3=m[2]&q[3];
assign d4=m[3]&q[3];

adder FA1(.a(a1),.b(1'b0),.cin(1'b0),.s(p[0]),.cout(r1));
adder FA2(.a(a2),.b(1'b0),.cin(r1),.s(o2),.cout(r2));
adder FA3(.a(a3),.b(1'b0),.cin(r2),.s(o3),.cout(r3));
adder FA4(.a(a4),.b(1'b0),.cin(r3),.s(o4),.cout(r4));
adder FA5(.a(b1),.b(o2),.cin(1'b0),.s(p[1]),.cout(r5));
adder FA6(.a(b2),.b(o3),.cin(r5),.s(o6),.cout(r6));
adder FA7(.a(b3),.b(o4),.cin(r6),.s(o7),.cout(r7));
adder FA8(.a(b4),.b(r4),.cin(r7),.s(o8),.cout(r8));
adder FA9(.a(c1),.b(o6),.cin(1'b0),.s(p[2]),.cout(r9));
adder FA10(.a(c2),.b(o7),.cin(r9),.s(o10),.cout(r10));
adder FA11(.a(c3),.b(o8),.cin(r10),.s(o11),.cout(r11));
adder FA12(.a(c4),.b(r8),.cin(r11),.s(o12),.cout(r12));
adder FA13(.a(d1),.b(o10),.cin(1'b0),.s(p[3]),.cout(r13));
adder FA14(.a(d2),.b(o11),.cin(r13),.s(p[4]),.cout(r14));
adder FA15(.a(d3),.b(o12),.cin(r14),.s(p[5]),.cout(r15));
adder FA16(.a(d4),.b(r12),.cin(r15),.s(p[6]),.cout(p[7]));

endmodule

module adder(a, b, cin, s, cout);
// inputs and outputs
input   a, b, cin;
output s,cout;
wire p,g;
assign p=a^b;
assign g=a&b;
assign s=p^cin;
assign cout=g|(p&cin);
endmodule
```

Test bench :

```
`include "exercise_05.v"
module exe_5_tb;

parameter DATA_IN_WIDTH = 4;
parameter DATA_OUT_WIDTH = 8;
reg [DATA_IN_WIDTH - 1:0] m;
reg [DATA_IN_WIDTH - 1:0] q;
wire [DATA_OUT_WIDTH - 1:0] p;

mul_4bits mul_4bits_00 (.m(m), .q(q), .p(p));

initial begin
$monitor ("m=%b, q=%b, p=%b \n",
          m, q, p);
end

initial begin
m=4'b0000;
q=4'b0000;
#50;
m=4'b0000;
q=4'b1111;
#50;
m=4'b0001;
q=4'b0001;
#50;
m=4'b0010;
q=4'b0001;
#50;
m=4'b0011;
q=4'b1100;
#50;
m=4'b1111;
q=4'b1111;
#50;
m=4'b0100;
q=4'b0100;
#50;
m=4'b1000;
q=4'b0001;
#50;
m=4'b0111;
q=4'b0010;
#50;
m=4'b0010;
q=4'b0100;
#50;
m=4'b1010;
q=4'b1000;
#50;
m=4'b1000;
q=4'b0100;
#50;
m=4'b0001;
q=4'b0001;
#50;
```

```
m=4'b1000;
q=4'b1000;
#50;
m=4'b1000;
q=4'b0010;
#50;
m=4'b0111;
q=4'b0000;
#50;
m=4'b0110;
q=4'b0110;
#50;
m=4'b1100;
q=4'b0011;
#50;
m=4'b0010;
q=4'b0100;
#50;
m=4'b0010;
q=4'b0001;
#50;
$finish;
end

initial begin
   $vcdplusfile("exe_5.vpd");
   $vcdpluson();
end
endmodule
```

Chapter 6: Exercise 06

Requirement: Develop the following hardware (Simple State Machine) with Verilog HDL

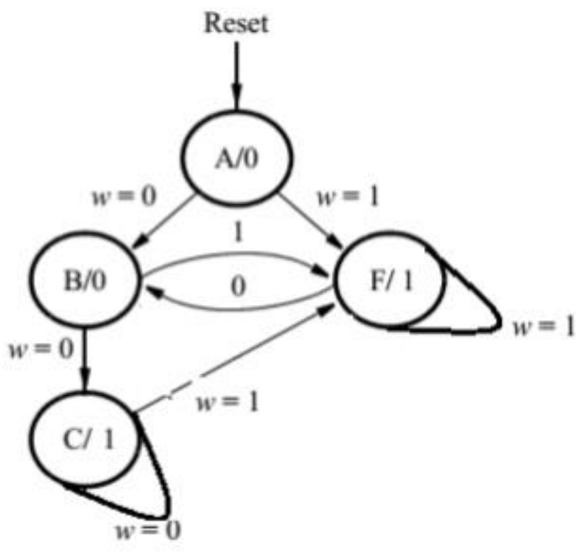


Figure 6.1 Ex_06 Design

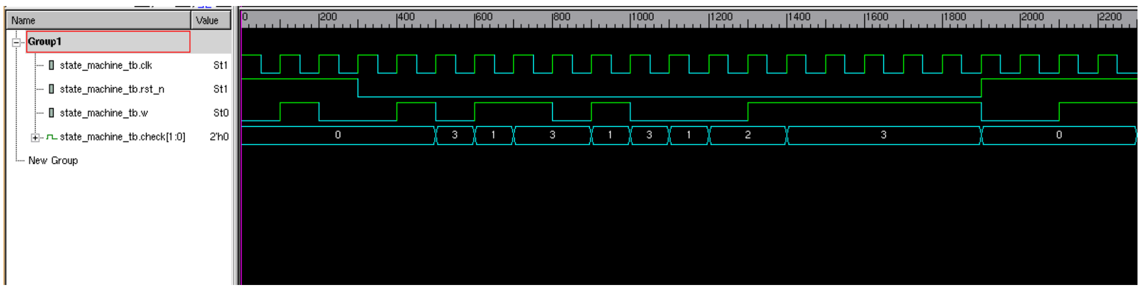
Solution

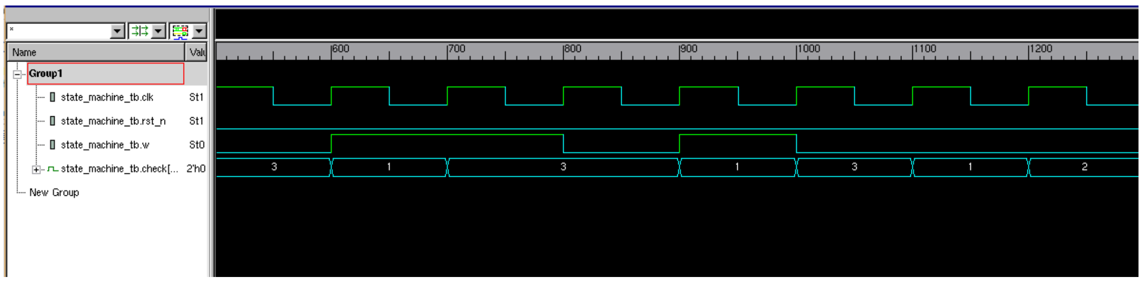
Log file of testbench:

```
time=                    0, reset=1, clk=1, w=0, check=00
time=                   50, reset=1, clk=0, w=0, check=00
time=                  100, reset=1, clk=1, w=1, check=00
time=                  150, reset=1, clk=0, w=1, check=00
time=                  200, reset=1, clk=1, w=0, check=00
time=                  250, reset=1, clk=0, w=0, check=00
time=                  300, reset=0, clk=1, w=0, check=00
time=                  350, reset=0, clk=0, w=0, check=00
time=                  400, reset=0, clk=1, w=1, check=00
time=                  450, reset=0, clk=0, w=1, check=00
time=                  500, reset=0, clk=1, w=0, check=11
time=                  550, reset=0, clk=0, w=0, check=11
time=                  600, reset=0, clk=1, w=1, check=01
time=                  650, reset=0, clk=0, w=1, check=01
time=                  700, reset=0, clk=1, w=1, check=11
time=                  750, reset=0, clk=0, w=1, check=11
time=                  800, reset=0, clk=1, w=0, check=11
time=                  850, reset=0, clk=0, w=0, check=11
time=                  900, reset=0, clk=1, w=1, check=01
time=                  950, reset=0, clk=0, w=1, check=01
time=                 1000, reset=0, clk=1, w=0, check=11
time=                 1050, reset=0, clk=0, w=0, check=11
time=                 1100, reset=0, clk=1, w=0, check=01
time=                 1150, reset=0, clk=0, w=0, check=01
time=                 1200, reset=0, clk=1, w=0, check=10
time=                 1250, reset=0, clk=0, w=0, check=10
time=                 1300, reset=0, clk=1, w=1, check=10
time=                 1350, reset=0, clk=0, w=1, check=10
time=                 1400, reset=0, clk=1, w=1, check=11
time=                 1450, reset=0, clk=0, w=1, check=11
time=                 1500, reset=0, clk=1, w=1, check=11
time=                 1550, reset=0, clk=0, w=1, check=11
```

```
time=          1600, reset=0, clk=1, w=1, check=11
time=          1650, reset=0, clk=0, w=1, check=11
time=          1700, reset=0, clk=1, w=1, check=11
time=          1750, reset=0, clk=0, w=1, check=11
time=          1800, reset=0, clk=1, w=1, check=11
time=          1850, reset=0, clk=0, w=1, check=11
time=          1900, reset=1, clk=1, w=0, check=00
time=          1950, reset=1, clk=0, w=0, check=00
time=          2000, reset=1, clk=1, w=0, check=00
time=          2050, reset=1, clk=0, w=0, check=00
time=          2100, reset=1, clk=1, w=1, check=00
time=          2150, reset=1, clk=0, w=1, check=00
time=          2200, reset=1, clk=1, w=1, check=00
time=          2250, reset=1, clk=0, w=1, check=00
```

Waveform:





Coding:

```
module state_machine (clk, rst_n, w, check);

parameter A = 2'b00;
parameter B = 2'b01;
parameter C = 2'b10;
parameter F = 2'b11;
parameter DATA_WIDTH = 2;

input clk;
input rst_n;
input w;
output [DATA_WIDTH - 1:0] check;

reg [DATA_WIDTH - 1:0] state;
reg [DATA_WIDTH - 1:0] next_state;
reg [DATA_WIDTH - 1:0] check;

always @(state or w) begin
  next_state = 2'b00;
  case (state)
  A: begin
    if (w) begin
      next_state = F;
    end
```

```
      else begin
        next_state = A;
      end
    end
    B: begin
      if (w) begin
        next_state = F;
      end
      else begin
        next_state = C;
      end
    end
    C: begin
      if (w) begin
        next_state = F;
      end
      else begin
        next_state = C;
      end
    end
    F: begin
      if (w) begin
        next_state = F;
      end
      else begin
        next_state = B;
      end
    end
    default: begin
        next_state = A;
    end
    endcase
end

always @(posedge clk) begin
    if (rst_n) begin
      state <= A;
    end
    else begin
      state <= next_state;
    end
end

always @(posedge clk) begin
    if (rst_n) begin
      check <= 2'b00;
    end
    else begin
      case (state)
      A: begin
        check <= 2'b00;
      end
      B: begin
        check <= 2'b01;
      end
      C: begin
        check <= 2'b10;
      end
      F: begin
        check <= 2'b11;
      end
    default: begin
      check <= 2'b00;
    end
    endcase
end
end
endmodule
```

Testbench:

```
`include "exercise_06.v"
module state_machine_tb;

parameter DATA_WIDTH = 2;
parameter CLK_CYCLE = 50;

reg  clk;
reg  rst_n;
reg  w;
wire  [DATA_WIDTH - 1:0] check;

always begin
clk = 1;
#CLK_CYCLE;
clk = 0;
#CLK_CYCLE;
end

state_machine state_machine_00 (.clk(clk), .rst_n(rst_n), .w(w), .check(check));

initial begin
  $monitor ("time=%d, reset=%b, clk=%b, w=%b, check=%b \n",
  $time, rst_n, clk, w, check);
end

initial begin
rst_n = 1;
w = 1'b0;
#100;
rst_n = 1;
w = 1'b1;
#100;
rst_n = 1;
w = 1'b0;
#100;
rst_n = 0;
w = 1'b0;
#100;
rst_n = 0;
w = 1'b1;
#100;
rst_n = 0;
w = 1'b0;
#100;
rst_n = 0;
w = 1'b1;
#100;
rst_n = 0;
w = 1'b1;
#100;
rst_n = 0;
w = 1'b0;
```

```
#100;
rst_n = 0;
w = 1'b1;
#100;
rst_n = 0;
w = 1'b0;
#100;
rst_n = 0;
w = 1'b0;
#100;
rst_n = 0;
w = 1'b0;
#100;
rst_n = 0;
w = 1'b1;
#100;
rst_n = 0;
w = 1'b1;
#100;
rst_n = 0;
w = 1'b1;
#100;
rst_n = 0;
w = 1'b1;
#100;
rst_n = 0;
w = 1'b1;
#100;
rst_n = 0;
w = 1'b1;
#100;
rst_n = 1;
w = 1'b0;
#100;
rst_n = 1;
w = 1'b0;
#100;
rst_n = 1;
w = 1'b1;
#200;
$finish;
end

initial begin
$vcdplusfile("exe_6.vpd");
$vcdpluson();
end
endmodule
```