Verilog HDL

Assignment 6
FSM

By: Magdy Ahmed Abbas Abdelhamid

Question 1

SEQUENCE_010_MOORE

Verilog Code:

```
module Sequence_010_Moore (x, clk, rst, y, count);
parameter STORE = 2'b00;
parameter IDLE = 2'b01;
parameter ZERO = 2'b10;
parameter ONE = 2'b11;
input x, clk, rst;
output reg y;
output reg [9: 0] count;
reg [1: 0] cs, ns;
// Next state Logic
always @(cs or x) begin
  case (cs)
    STORE :
      if (x)
        ns = IDLE;
      else
        ns = ZERO;
    IDLE :
      if (x)
        ns = IDLE;
      else
        ns = ZERO;
    ZERO :
      if (x)
        ns = ONE;
      else
        ns = ZERO;
    ONE :
      if (x)
        ns = IDLE;
      else
        ns = STORE;
    default : ns = IDLE;
  endcase
end
```

```
// State Memory
always @(posedge clk or posedge rst) begin
  if (rst) begin
    cs <= IDLE;
    count <= 0;
  end
  else
    cs <= ns;
end

// Output Logic
always @(cs) begin
  case (cs)
    STORE : begin
              y = 1;
              count = count + 1;
            end
    IDLE, ZERO, ONE : y = 0;
  endcase
end

endmodule
```

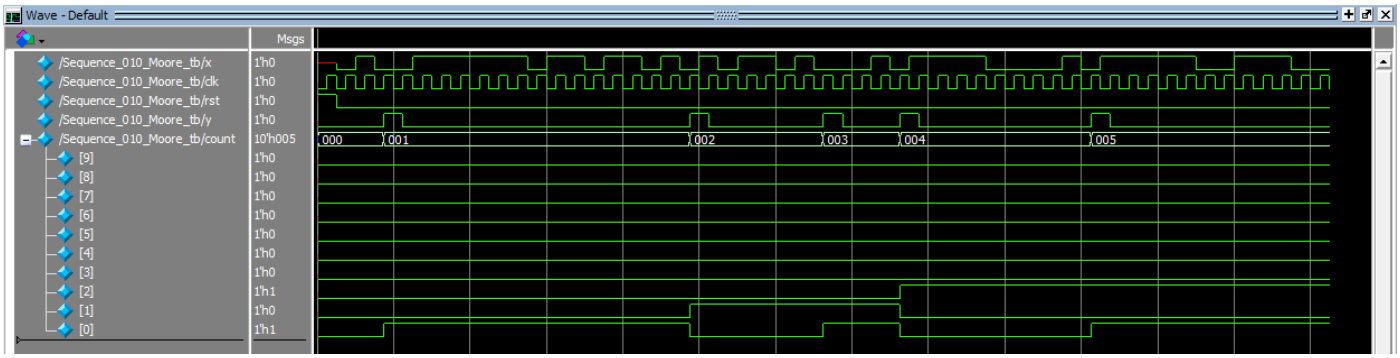
Testbench Code:

```
module Sequence_010_Moore_tb ();

parameter STORE = 2'b00;
parameter IDLE = 2'b01;
parameter ZERO = 2'b10;
parameter ONE = 2'b11;

reg x, clk, rst;
wire y;
wire [9: 0] count;

Sequence_010_Moore #(STORE, IDLE, ZERO, ONE) s1(x, clk, rst, y, count);

initial begin
  clk = 0;
  forever
    #25 clk = ~clk;
end

initial begin
  rst = 1;
  #50;
  rst = 0;
  x = 0;
  #50;
  x = 1;
  #50;
  x = 0;
  #50;

  if ((y != 1) && (count != 1)) begin
    $display("The FSM Design is Wrong! ");
    $stop;
  end

  repeat (50) begin
    @(negedge clk);
    x = $random;
  end
  #2 $stop;
end

initial
  $monitor ("time = %0t, x = %b, clk = %b, rst = %b, y = %b, count = %b", $time, x, clk, rst, y, count);

endmodule
```

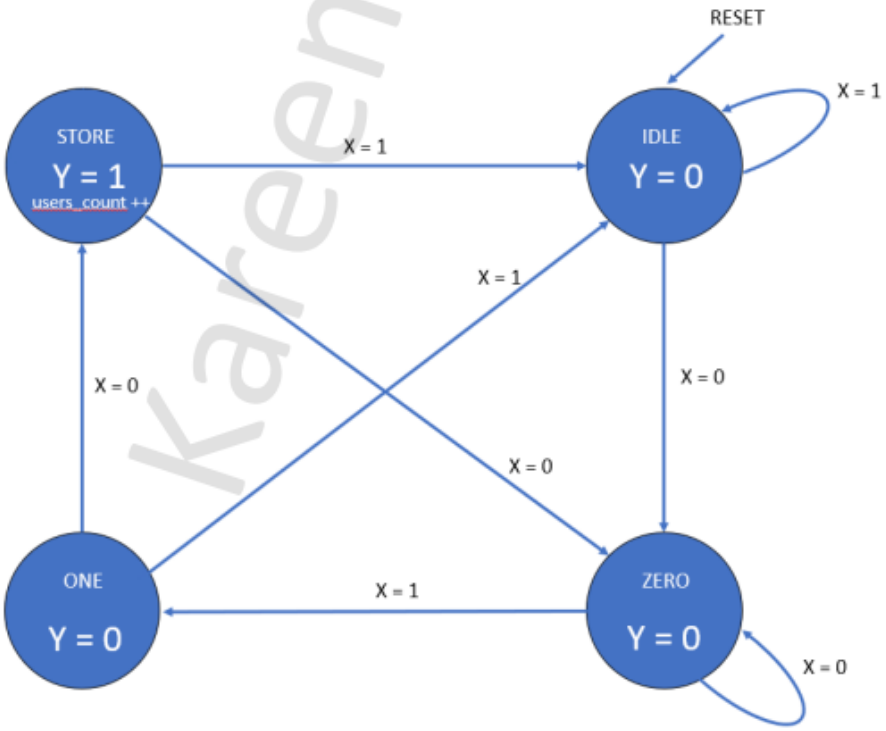
Do File for (Sequence_010_Moore) Question 1

```
vlib work
vlog Sequence_010_Moore.v Sequence_010_Moore_tb.v
vsim -voptargs=+acc work.Sequence_010_Moore_tb
add wave *
run -all
```

Waveform:

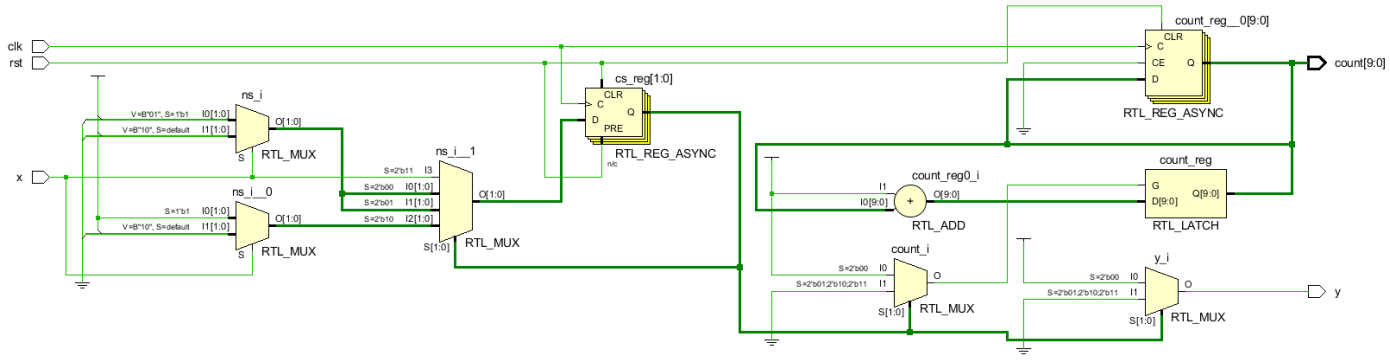


Name	Type	Size	Description
x	Input	1 bit	Input sequence
clk			Clock
rst			Active high asynchronous reset
y	Output	1 bit	Output that is HIGH when the sequence 010 is detected
count		10 bits	Outputs the number of time the pattern was detected

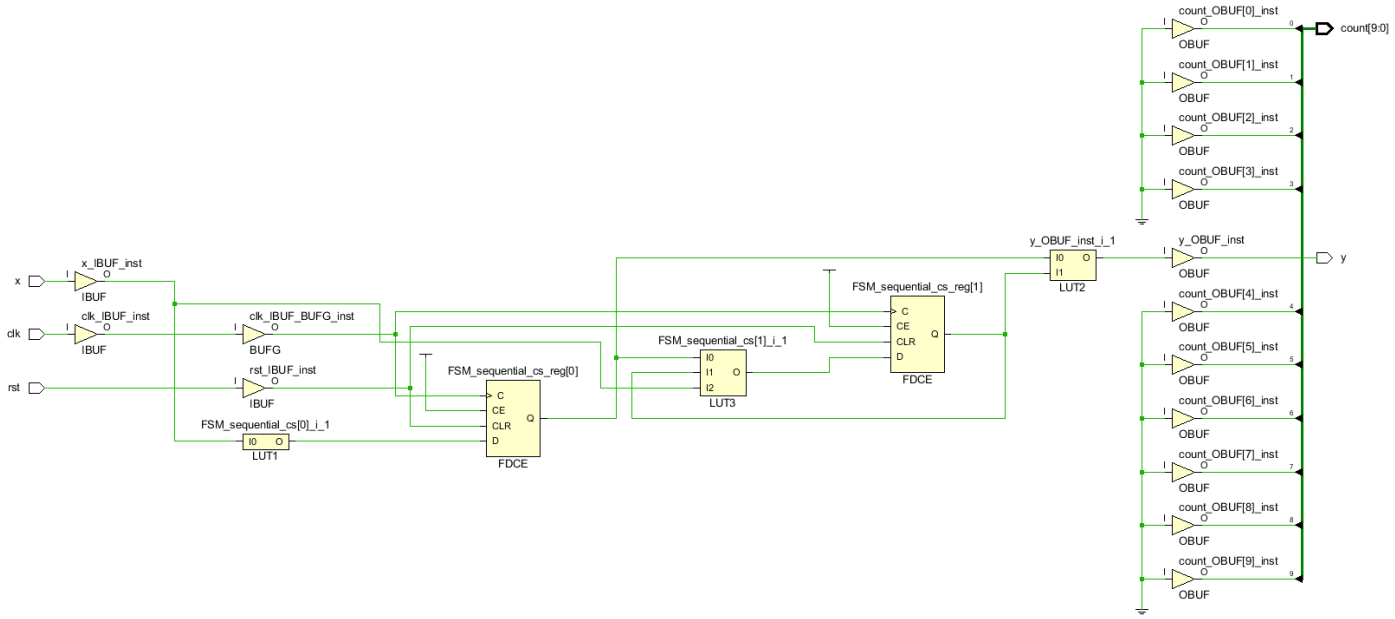


```
# time = 1100, x = 1, clk = 0, rst = 0, y = 0, count = 0000000010
# time = 1125, x = 1, clk = 1, rst = 0, y = 0, count = 0000000010
# time = 1150, x = 1, clk = 0, rst = 0, y = 0, count = 0000000010
# time = 1175, x = 1, clk = 1, rst = 0, y = 0, count = 0000000010
# time = 1200, x = 0, clk = 0, rst = 0, y = 0, count = 0000000010
# time = 1225, x = 0, clk = 1, rst = 0, y = 0, count = 0000000010
# time = 1250, x = 1, clk = 0, rst = 0, y = 0, count = 0000000010
# time = 1275, x = 1, clk = 1, rst = 0, y = 0, count = 0000000010
# time = 1300, x = 0, clk = 0, rst = 0, y = 0, count = 0000000010
# time = 1325, x = 0, clk = 1, rst = 0, y = 1, count = 0000000011
# time = 1350, x = 0, clk = 0, rst = 0, y = 1, count = 0000000011
# time = 1375, x = 0, clk = 1, rst = 0, y = 0, count = 0000000011
# time = 1400, x = 0, clk = 0, rst = 0, y = 0, count = 0000000011
# time = 1425, x = 0, clk = 1, rst = 0, y = 0, count = 0000000011
# time = 1450, x = 1, clk = 0, rst = 0, y = 0, count = 0000000011
# time = 1475, x = 1, clk = 1, rst = 0, y = 0, count = 0000000011
# time = 1500, x = 0, clk = 0, rst = 0, y = 0, count = 0000000011
# time = 1525, x = 0, clk = 1, rst = 0, y = 1, count = 0000000100
# time = 1550, x = 1, clk = 0, rst = 0, y = 1, count = 0000000100
# time = 1575, x = 1, clk = 1, rst = 0, y = 0, count = 0000000100
# time = 1600, x = 1, clk = 0, rst = 0, y = 0, count = 0000000100
# time = 1625, x = 1, clk = 1, rst = 0, y = 0, count = 0000000100
# time = 1650, x = 1, clk = 0, rst = 0, y = 0, count = 0000000100
# time = 1675, x = 1, clk = 1, rst = 0, y = 0, count = 0000000100
# time = 1700, x = 1, clk = 0, rst = 0, y = 0, count = 0000000100
# time = 1725, x = 1, clk = 1, rst = 0, y = 0, count = 0000000100
# time = 1750, x = 0, clk = 0, rst = 0, y = 0, count = 0000000100
# time = 1775, x = 0, clk = 1, rst = 0, y = 0, count = 0000000100
# time = 1800, x = 0, clk = 0, rst = 0, y = 0, count = 0000000100
# time = 1825, x = 0, clk = 1, rst = 0, y = 0, count = 0000000100
# time = 1850, x = 0, clk = 0, rst = 0, y = 0, count = 0000000100
# time = 1875, x = 0, clk = 1, rst = 0, y = 0, count = 0000000100
# time = 1900, x = 0, clk = 0, rst = 0, y = 0, count = 0000000100
# time = 1925, x = 0, clk = 1, rst = 0, y = 0, count = 0000000100
# time = 1950, x = 1, clk = 0, rst = 0, y = 0, count = 0000000100
# time = 1975, x = 1, clk = 1, rst = 0, y = 0, count = 0000000100
# time = 2000, x = 0, clk = 0, rst = 0, y = 0, count = 0000000100
# time = 2025, x = 0, clk = 1, rst = 0, y = 1, count = 0000000101
# time = 2050, x = 1, clk = 0, rst = 0, y = 1, count = 0000000101
# time = 2075, x = 1, clk = 1, rst = 0, y = 0, count = 0000000101
# time = 2100, x = 1, clk = 0, rst = 0, y = 0, count = 0000000101
# time = 2125, x = 1, clk = 1, rst = 0, y = 0, count = 0000000101
# time = 2150, x = 1, clk = 0, rst = 0, y = 0, count = 0000000101
# time = 2175, x = 1, clk = 1, rst = 0, y = 0, count = 0000000101
# time = 2200, x = 1, clk = 0, rst = 0, y = 0, count = 0000000101
# time = 2225, x = 1, clk = 1, rst = 0, y = 0, count = 0000000101
# time = 2250, x = 1, clk = 0, rst = 0, y = 0, count = 0000000101
# time = 2275, x = 1, clk = 1, rst = 0, y = 0, count = 0000000101
# time = 2300, x = 0, clk = 0, rst = 0, y = 0, count = 0000000101
```

Schematic after the Elaboration:



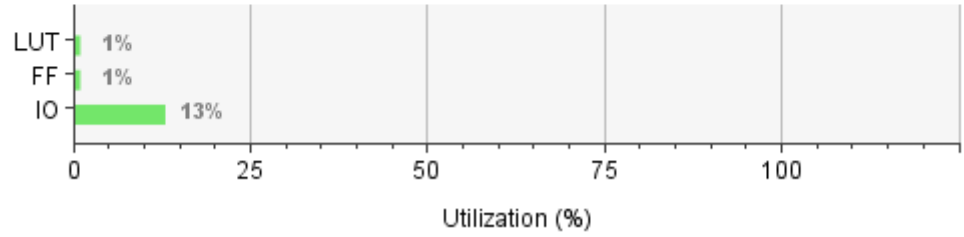
Schematic after the Synthesis:



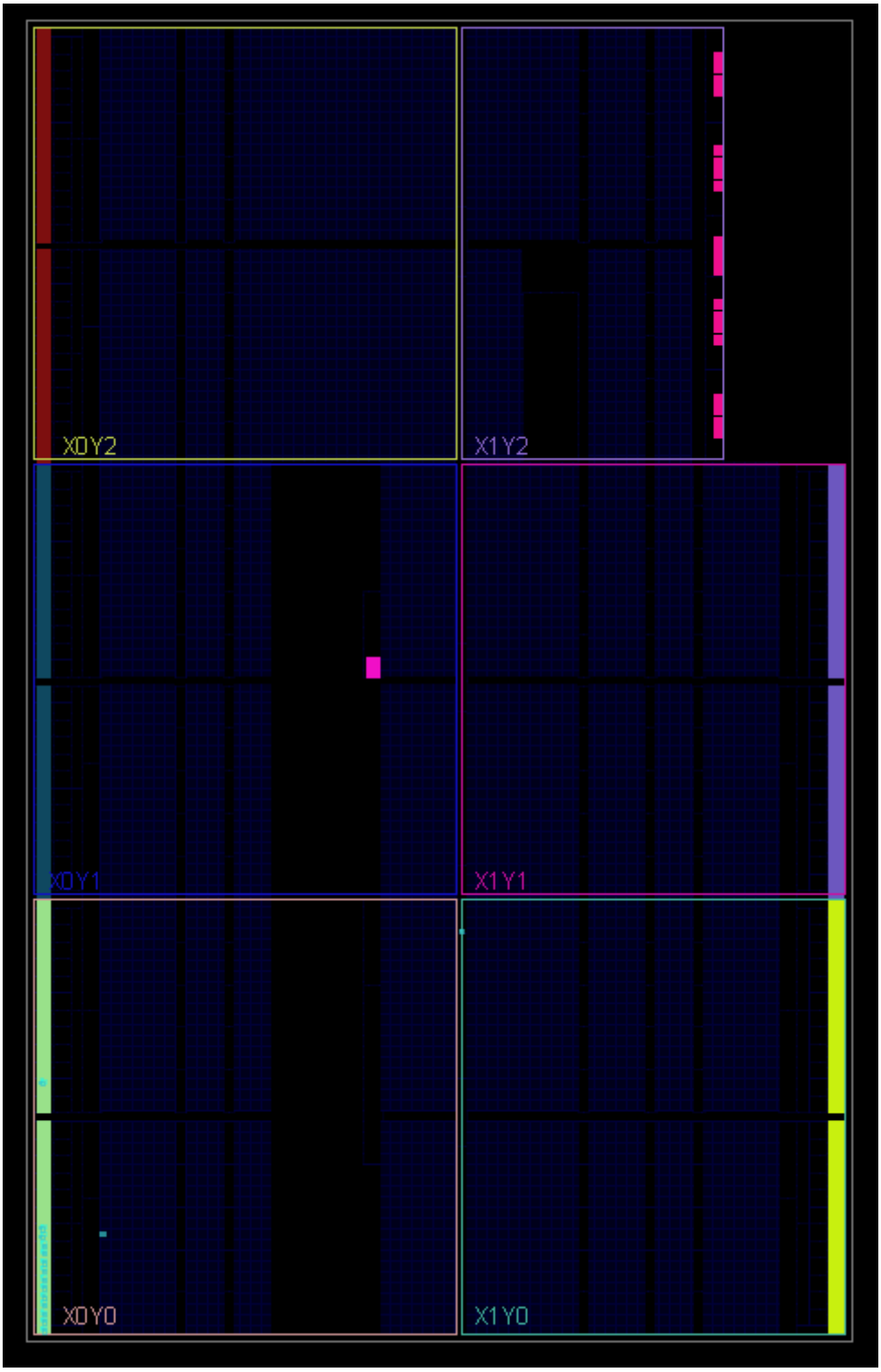
Utilization Report:

Summary

Resource	Utilization	Available	Utilization %
LUT	3	20800	0.01
FF	2	41600	0.00
IO	14	106	13.21



Device after implementation:



Question 2

TESLA - CAR

Verilog Code:

```
module Tesla_Car (speed_limit, car_speed, leading_distance, clk, rst, unlock_doors, accel

parameter MIN_DISTANCE = 7'd40; // 40 meters
parameter ACCELERATE = 2'b00;
parameter STOP = 2'b01;
parameter DECELERATE = 2'b10;
input [7:0] speed_limit, car_speed;
input [6:0] leading_distance;
input clk, rst;
output reg unlock_doors, accelerate_car;
reg [1:0] ns, cs;
// Next State Logic
always @(*) begin
  case (cs)
    ACCELERATE :
        if ((leading_distance >= MIN_DISTANCE) && (car_speed < speed_limit))
          ns = ACCELERATE;
        else if ((leading_distance < MIN_DISTANCE) || (car_speed > speed_limit))
            ns = DECELERATE;
    STOP :
        if (leading_distance < MIN_DISTANCE)
          ns = STOP;
        else if (leading_distance >= MIN_DISTANCE)
          ns = ACCELERATE;
    DECELERATE :
        if ((leading_distance < MIN_DISTANCE) || (car_speed > speed_limit))
          ns = DECELERATE;
        else if ((leading_distance >= MIN_DISTANCE) && (car_speed
            < speed_limit))
            ns = ACCELERATE;
        else if (car_speed == 0)
            ns = STOP;
    default : ns = STOP;
  endcase
end
```

```
// State Memory
always @(posedge clk or posedge rst) begin
  if (rst)
    cs <= STOP;
  else
    cs <= ns;
end

// Output Logic
always @(cs) begin
  case (cs)
    ACCELERATE : begin
                   unlock_doors = 0;
                   accelerate_car = 1;
                 end
    STOP : begin
             unlock_doors = 1;
             accelerate_car = 0;
           end
    DECELERATE : begin
                   unlock_doors = 0;
                   accelerate_car = 0;
                 end
  endcase
end

endmodule
```

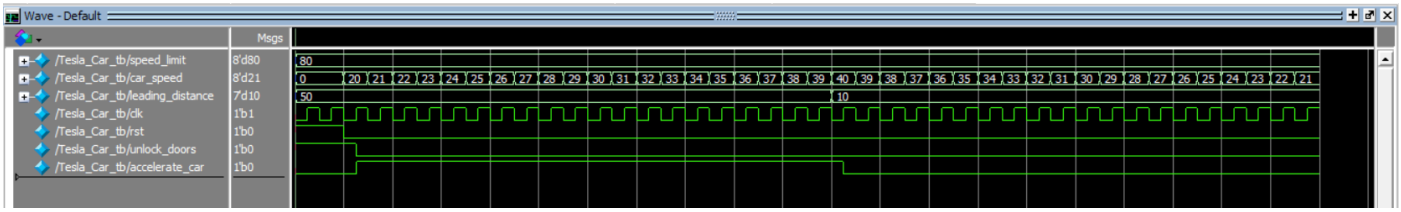
Testbench Code:

```
module Tesla_Car_tb ();
parameter MIN_DISTANCE = 7'd40; // 40 meters
parameter ACCELERATE = 2'b00;
parameter STOP = 2'b01;
parameter DECELERATE = 2'b10;
reg [7: 0] speed_limit, car_speed;
reg [6: 0] leading_distance;
reg clk, rst;
wire unlock_doors, accelerate_car;
Tesla_Car #(MIN_DISTANCE, ACCELERATE, STOP, DECELERATE)
T1(speed_limit, car_speed, leading_distance, clk, rst, unlock_doors, accelerate_car);

initial begin
  clk = 0;
  forever
    #25 clk = ~clk;
end
initial begin
  rst = 1;
  speed_limit = 8'd80;
  leading_distance = 7'd50;
  car_speed = 0;
  #100;
  // Test Acceleration and Deceleration With Speed Limit
  car_speed = 8'd20;
  rst = 0;
  #1000;
  // Test the Safety For Car Crashes with Leading Vehicle / Object
  leading_distance = 7'd10;
  #1000;
  $stop;
end
// Change Speed of the Car During the Current State
always @(negedge clk) begin
  if (~rst) begin
    if (unlock_doors == 0) begin
      if (accelerate_car)
        car_speed = car_speed + 1;
      else
        car_speed = car_speed - 1;
    end
  end
end
initial
$monitor ("clk = %b, rst = %b, speed_limit = %d m/sec, car_speed = %d m/sec, leading_distance
                = %d m, unlock_doors = %b, accelerate_car = %b"
                                    , clk, rst, speed_limit, car_speed, leading_distance, unlock_doors, accelerate_car);

endmodule
```

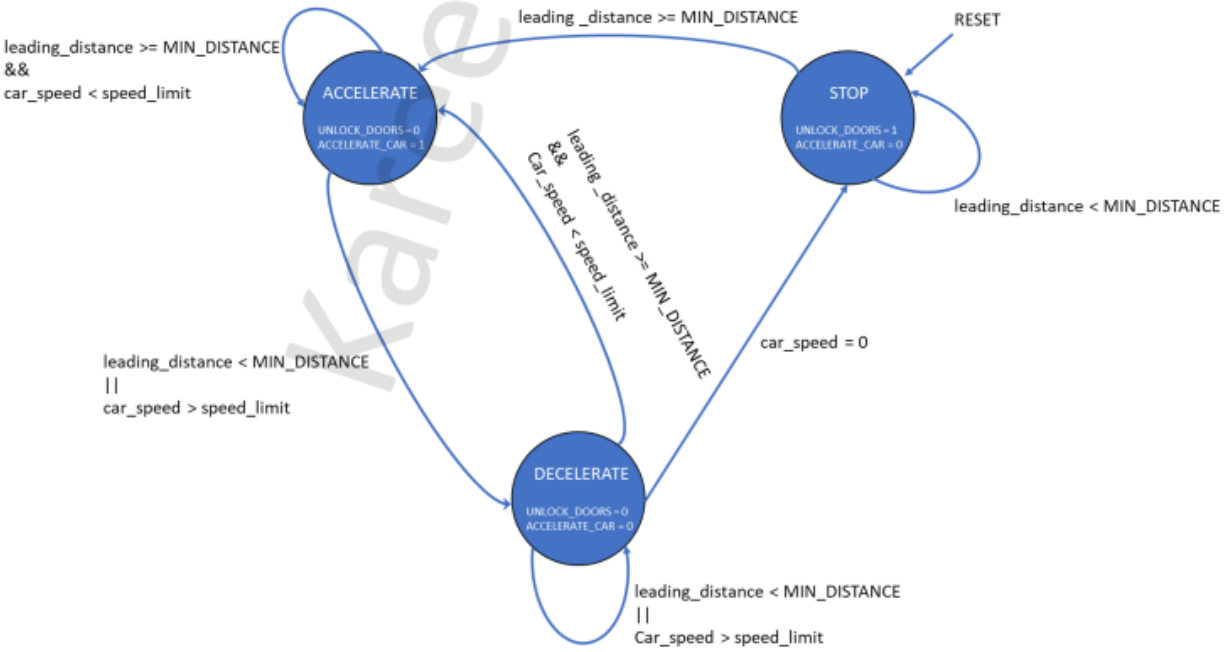
Do File for (Tesla_Car) Question 2

```
vlib work
vlog Tesla_Car.v Tesla_Car_tb.v
vsim -voptargs=+acc work.Tesla_Car_tb
add wave *
run -all
```

Waveform:

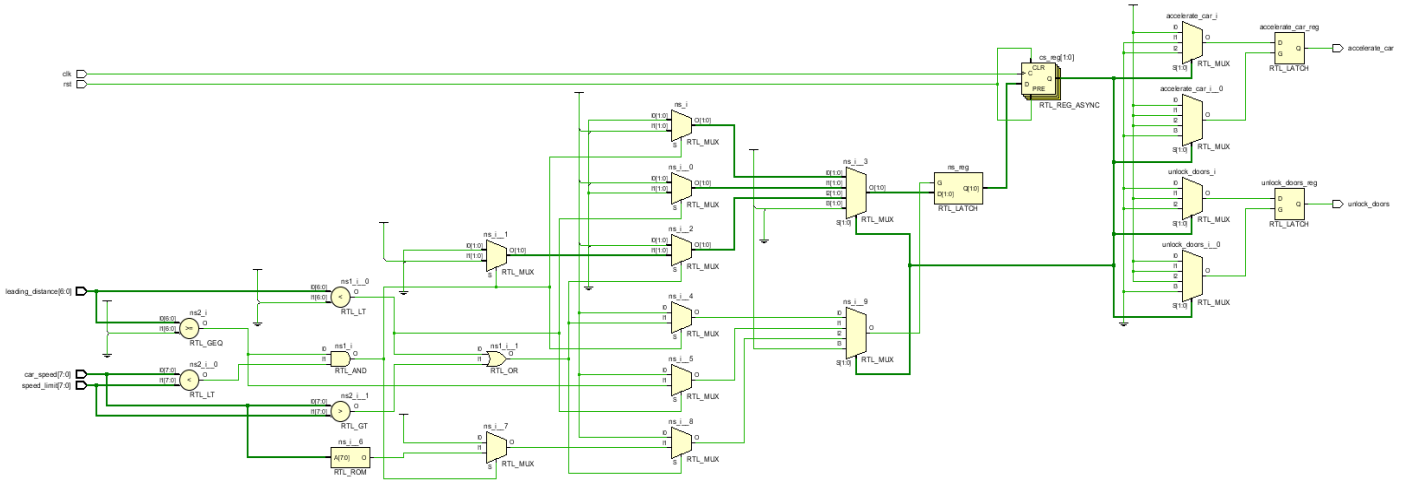


Name	Type	Size	Description
speed_limit	Input	8 bits	Allowable speed limit of the highway
car_speed		8 bits	Current car speed
leading_distance		7 bits	Distance between the car and the vehicle/object in front of it
clk		1 bit	Clock
rst		1 bit	Active high asynchronous reset
unlock_doors	Output	1 bit	Signal that unlock the car doors when HIGH
accelerate_car			Signal that control the flow of the fuel to the engine to accelerate the car when HIGH

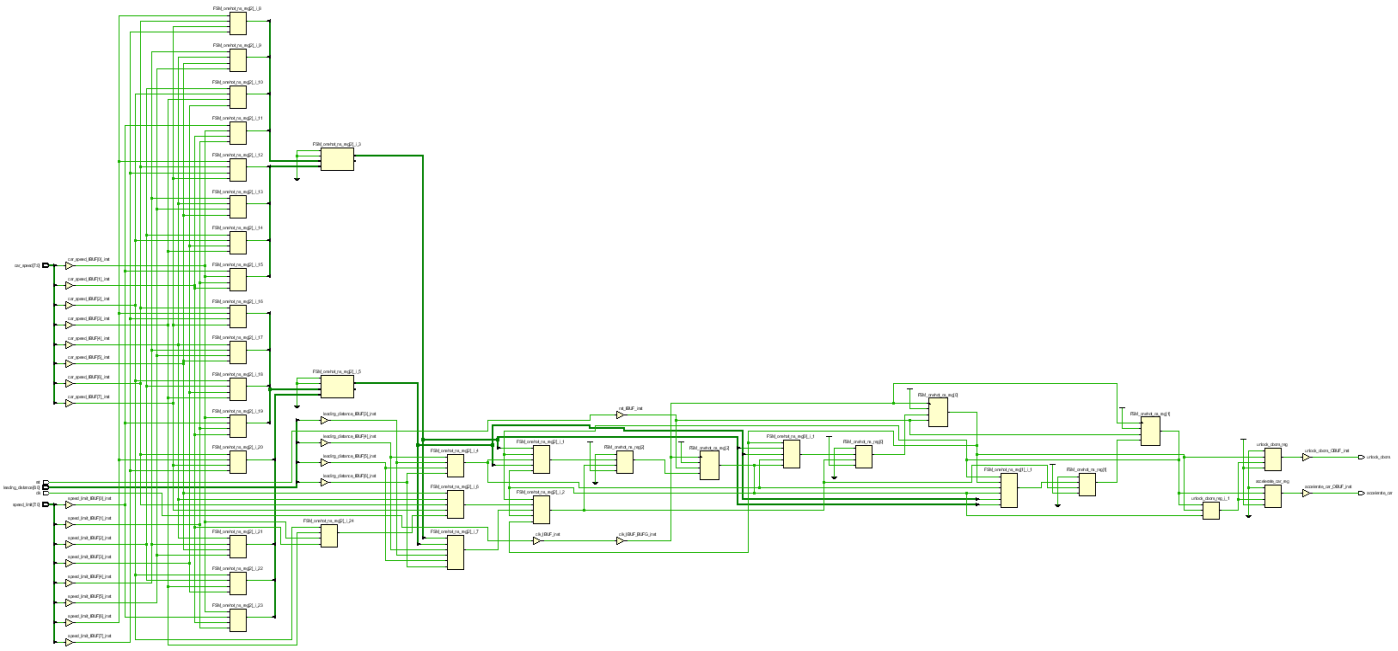


```
# clk = 0, rst = 1, speed_limit =  80 m/sec, car_speed =   0 m/sec, leading_distance =  50 m, unlock_doors = 1, accelerate_car = 0
# clk = 1, rst = 1, speed_limit =  80 m/sec, car_speed =   0 m/sec, leading_distance =  50 m, unlock_doors = 1, accelerate_car = 0
# clk = 0, rst = 1, speed_limit =  80 m/sec, car_speed =   0 m/sec, leading_distance =  50 m, unlock_doors = 1, accelerate_car = 0
# clk = 1, rst = 1, speed_limit =  80 m/sec, car_speed =   0 m/sec, leading_distance =  50 m, unlock_doors = 1, accelerate_car = 0
# clk = 0, rst = 0, speed_limit =  80 m/sec, car_speed =  20 m/sec, leading_distance =  50 m, unlock_doors = 1, accelerate_car = 0
# clk = 1, rst = 0, speed_limit =  80 m/sec, car_speed =  20 m/sec, leading_distance =  50 m, unlock_doors = 0, accelerate_car = 1
# clk = 0, rst = 0, speed_limit =  80 m/sec, car_speed =  21 m/sec, leading_distance =  50 m, unlock_doors = 0, accelerate_car = 1
# clk = 1, rst = 0, speed_limit =  80 m/sec, car_speed =  21 m/sec, leading_distance =  50 m, unlock_doors = 0, accelerate_car = 1
# clk = 0, rst = 0, speed_limit =  80 m/sec, car_speed =  22 m/sec, leading_distance =  50 m, unlock_doors = 0, accelerate_car = 1
# clk = 1, rst = 0, speed_limit =  80 m/sec, car_speed =  22 m/sec, leading_distance =  50 m, unlock_doors = 0, accelerate_car = 1
# clk = 0, rst = 0, speed_limit =  80 m/sec, car_speed =  23 m/sec, leading_distance =  50 m, unlock_doors = 0, accelerate_car = 1
# clk = 1, rst = 0, speed_limit =  80 m/sec, car_speed =  23 m/sec, leading_distance =  50 m, unlock_doors = 0, accelerate_car = 1
# clk = 0, rst = 0, speed_limit =  80 m/sec, car_speed =  24 m/sec, leading_distance =  50 m, unlock_doors = 0, accelerate_car = 1
# clk = 1, rst = 0, speed_limit =  80 m/sec, car_speed =  24 m/sec, leading_distance =  50 m, unlock_doors = 0, accelerate_car = 1
# clk = 0, rst = 0, speed_limit =  80 m/sec, car_speed =  25 m/sec, leading_distance =  50 m, unlock_doors = 0, accelerate_car = 1
# clk = 1, rst = 0, speed_limit =  80 m/sec, car_speed =  25 m/sec, leading_distance =  50 m, unlock_doors = 0, accelerate_car = 1
# clk = 0, rst = 0, speed_limit =  80 m/sec, car_speed =  26 m/sec, leading_distance =  50 m, unlock_doors = 0, accelerate_car = 1
# clk = 1, rst = 0, speed_limit =  80 m/sec, car_speed =  26 m/sec, leading_distance =  50 m, unlock_doors = 0, accelerate_car = 1
# clk = 0, rst = 0, speed_limit =  80 m/sec, car_speed =  27 m/sec, leading_distance =  50 m, unlock_doors = 0, accelerate_car = 1
# clk = 1, rst = 0, speed_limit =  80 m/sec, car_speed =  27 m/sec, leading_distance =  50 m, unlock_doors = 0, accelerate_car = 1
# clk = 0, rst = 0, speed_limit =  80 m/sec, car_speed =  28 m/sec, leading_distance =  50 m, unlock_doors = 0, accelerate_car = 1
# clk = 1, rst = 0, speed_limit =  80 m/sec, car_speed =  28 m/sec, leading_distance =  50 m, unlock_doors = 0, accelerate_car = 1
# clk = 0, rst = 0, speed_limit =  80 m/sec, car_speed =  29 m/sec, leading_distance =  50 m, unlock_doors = 0, accelerate_car = 1
# clk = 1, rst = 0, speed_limit =  80 m/sec, car_speed =  29 m/sec, leading_distance =  50 m, unlock_doors = 0, accelerate_car = 1
# clk = 0, rst = 0, speed_limit =  80 m/sec, car_speed =  30 m/sec, leading_distance =  50 m, unlock_doors = 0, accelerate_car = 1
# clk = 1, rst = 0, speed_limit =  80 m/sec, car_speed =  30 m/sec, leading_distance =  50 m, unlock_doors = 0, accelerate_car = 1
# clk = 0, rst = 0, speed_limit =  80 m/sec, car_speed =  31 m/sec, leading_distance =  50 m, unlock_doors = 0, accelerate_car = 1
# clk = 1, rst = 0, speed_limit =  80 m/sec, car_speed =  31 m/sec, leading_distance =  50 m, unlock_doors = 0, accelerate_car = 1
# clk = 0, rst = 0, speed_limit =  80 m/sec, car_speed =  32 m/sec, leading_distance =  50 m, unlock_doors = 0, accelerate_car = 1
# clk = 1, rst = 0, speed_limit =  80 m/sec, car_speed =  32 m/sec, leading_distance =  50 m, unlock_doors = 0, accelerate_car = 1
# clk = 0, rst = 0, speed_limit =  80 m/sec, car_speed =  33 m/sec, leading_distance =  50 m, unlock_doors = 0, accelerate_car = 1
# clk = 1, rst = 0, speed_limit =  80 m/sec, car_speed =  33 m/sec, leading_distance =  50 m, unlock_doors = 0, accelerate_car = 1
# clk = 0, rst = 0, speed_limit =  80 m/sec, car_speed =  34 m/sec, leading_distance =  50 m, unlock_doors = 0, accelerate_car = 1
# clk = 1, rst = 0, speed_limit =  80 m/sec, car_speed =  34 m/sec, leading_distance =  50 m, unlock_doors = 0, accelerate_car = 1
# clk = 0, rst = 0, speed_limit =  80 m/sec, car_speed =  35 m/sec, leading_distance =  50 m, unlock_doors = 0, accelerate_car = 1
# clk = 1, rst = 0, speed_limit =  80 m/sec, car_speed =  35 m/sec, leading_distance =  50 m, unlock_doors = 0, accelerate_car = 1
# clk = 0, rst = 0, speed_limit =  80 m/sec, car_speed =  36 m/sec, leading_distance =  50 m, unlock_doors = 0, accelerate_car = 1
# clk = 1, rst = 0, speed_limit =  80 m/sec, car_speed =  36 m/sec, leading_distance =  50 m, unlock_doors = 0, accelerate_car = 1
# clk = 0, rst = 0, speed_limit =  80 m/sec, car_speed =  37 m/sec, leading_distance =  50 m, unlock_doors = 0, accelerate_car = 1
# clk = 1, rst = 0, speed_limit =  80 m/sec, car_speed =  37 m/sec, leading_distance =  50 m, unlock_doors = 0, accelerate_car = 1
# clk = 0, rst = 0, speed_limit =  80 m/sec, car_speed =  38 m/sec, leading_distance =  50 m, unlock_doors = 0, accelerate_car = 1
# clk = 1, rst = 0, speed_limit =  80 m/sec, car_speed =  38 m/sec, leading_distance =  50 m, unlock_doors = 0, accelerate_car = 1
# clk = 0, rst = 0, speed_limit =  80 m/sec, car_speed =  39 m/sec, leading_distance =  50 m, unlock_doors = 0, accelerate_car = 1
# clk = 1, rst = 0, speed_limit =  80 m/sec, car_speed =  39 m/sec, leading_distance =  50 m, unlock_doors = 0, accelerate_car = 1
# clk = 0, rst = 0, speed_limit =  80 m/sec, car_speed =  40 m/sec, leading_distance =  10 m, unlock_doors = 0, accelerate_car = 1
# clk = 1, rst = 0, speed_limit =  80 m/sec, car_speed =  40 m/sec, leading_distance =  10 m, unlock_doors = 0, accelerate_car = 0
# clk = 0, rst = 0, speed_limit =  80 m/sec, car_speed =  39 m/sec, leading_distance =  10 m, unlock_doors = 0, accelerate_car = 0
# clk = 1, rst = 0, speed_limit =  80 m/sec, car_speed =  39 m/sec, leading_distance =  10 m, unlock_doors = 0, accelerate_car = 0
# clk = 0, rst = 0, speed_limit =  80 m/sec, car_speed =  38 m/sec, leading_distance =  10 m, unlock_doors = 0, accelerate_car = 0
# clk = 1, rst = 0, speed_limit =  80 m/sec, car_speed =  38 m/sec, leading_distance =  10 m, unlock_doors = 0, accelerate_car = 0
# clk = 0, rst = 0, speed_limit =  80 m/sec, car_speed =  37 m/sec, leading_distance =  10 m, unlock_doors = 0, accelerate_car = 0
# clk = 1, rst = 0, speed_limit =  80 m/sec, car_speed =  37 m/sec, leading_distance =  10 m, unlock_doors = 0, accelerate_car = 0
# clk = 0, rst = 0, speed_limit =  80 m/sec, car_speed =  36 m/sec, leading_distance =  10 m, unlock_doors = 0, accelerate_car = 0
# clk = 1, rst = 0, speed_limit =  80 m/sec, car_speed =  36 m/sec, leading_distance =  10 m, unlock_doors = 0, accelerate_car = 0
# clk = 0, rst = 0, speed_limit =  80 m/sec, car_speed =  35 m/sec, leading_distance =  10 m, unlock_doors = 0, accelerate_car = 0
# clk = 1, rst = 0, speed_limit =  80 m/sec, car_speed =  35 m/sec, leading_distance =  10 m, unlock_doors = 0, accelerate_car = 0
# clk = 0, rst = 0, speed_limit =  80 m/sec, car_speed =  34 m/sec, leading_distance =  10 m, unlock_doors = 0, accelerate_car = 0
```

Schematic after the Elaboration:



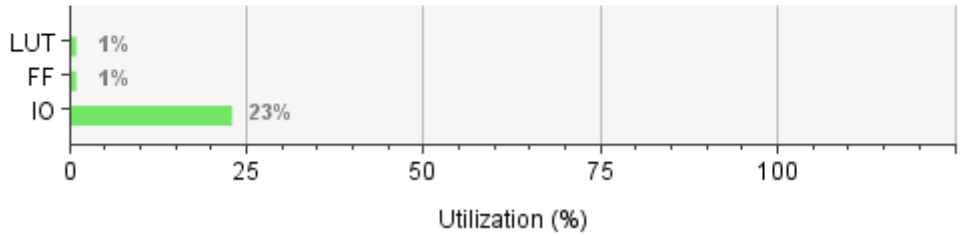
Schematic after the Synthesis:



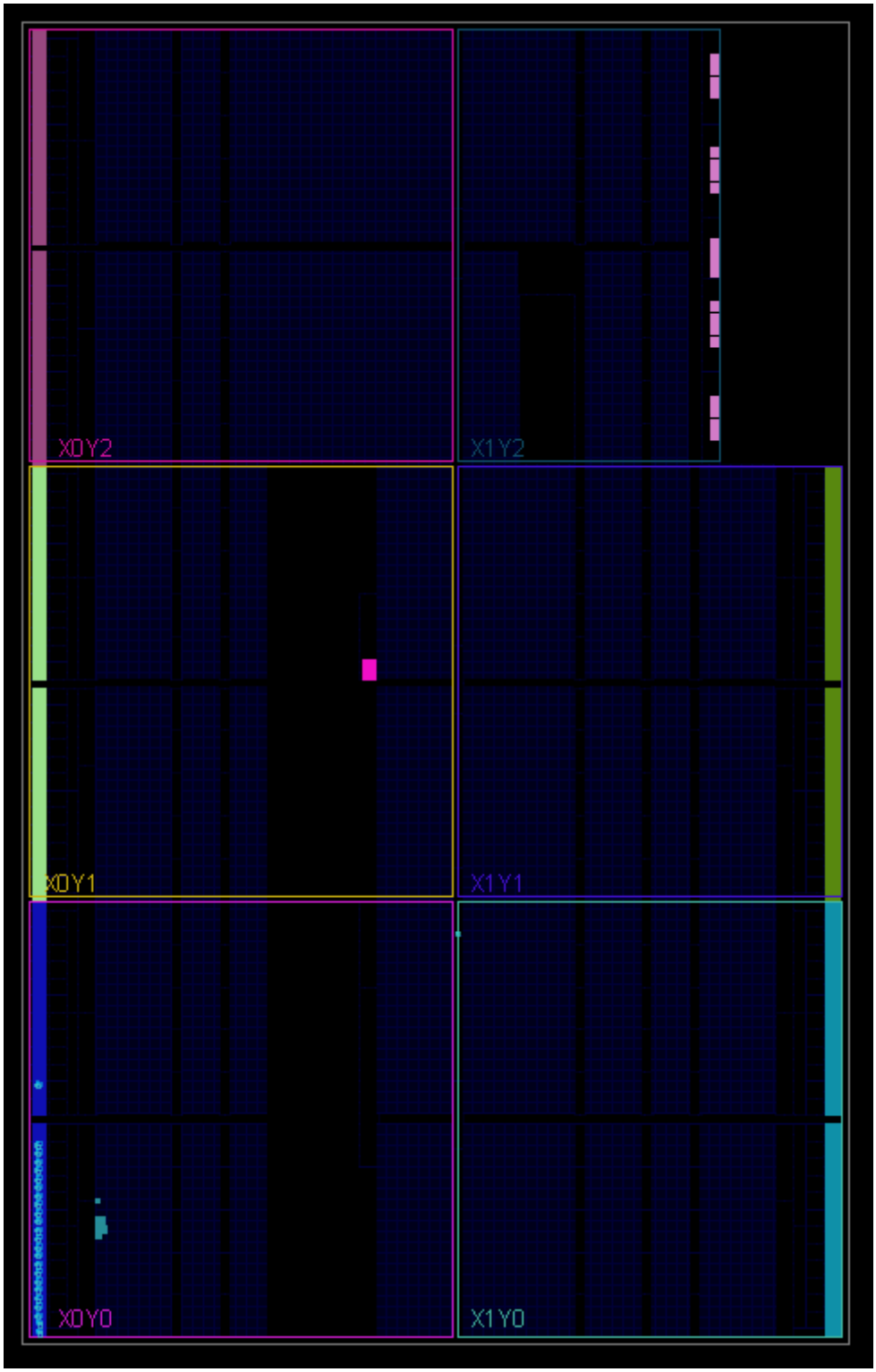
Utilization Report:

Summary

Resource	Utilization	Available	Utilization %
LUT	17	20800	0.08
FF	8	41600	0.02
IO	24	106	22.64



Device after implementation:



Question 3

Gray Code Moore FSM

Verilog Code:

```
module Gray_Code_Moore (clk, rst, y);

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

input clk, rst;
output reg [1: 0] y;

reg [1: 0] cs, ns;

// Next State Logic
always @(cs) begin
  case (cs)
    A : ns = B;
    B : ns = C;
    C : ns = D;
    D : ns = A;
  endcase
end

// State Memory
always @(posedge clk or posedge rst) begin
  if (rst)
    cs <= A;
  else
    cs <= ns;
end

// Output Logic
always @(cs) begin
  case (cs)
    A : y = 2'b00;
    B : y = 2'b01;
    C : y = 2'b11;
    D : y = 2'b10;
  endcase
end
endmodule
```

```
module Gray_Code_Ref (clk, rst, gray_out);

input clk, rst;
output [1: 0] gray_out;

reg [1: 0] binary_cnt;

always @(posedge clk or posedge rst) begin
   if (rst)
      binary_cnt <= 0;
   else begin
      binary_cnt <= binary_cnt + 1;
   end
end

assign gray_out[1] = binary_cnt[1];
assign gray_out[0] = ^binary_cnt;

endmodule
```

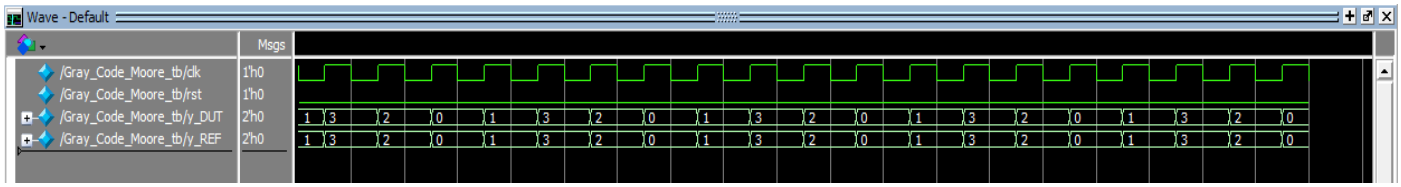
Testbench Code:

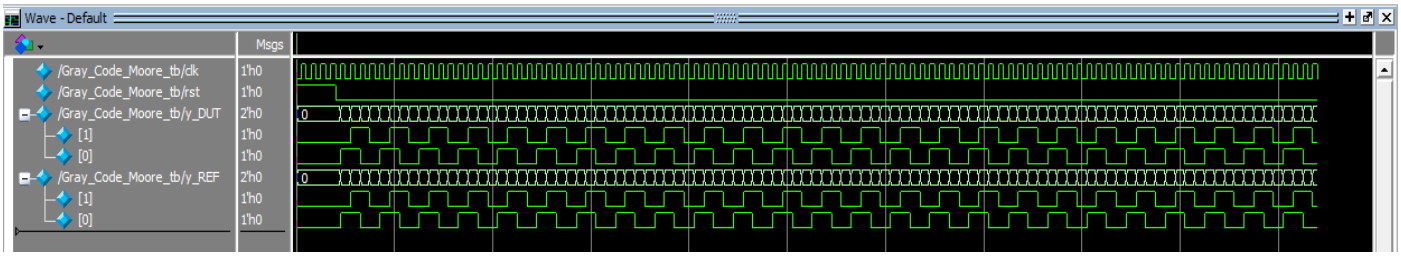
```
module Gray_Code_Moore_tb ();

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

reg clk, rst;
wire [1: 0] y_DUT;
wire [1: 0] y_REF;

Gray_Code_Moore #(A, B, C, D) DUT (clk, rst, y_DUT);
Gray_Code_Ref REF (clk, rst, y_REF);

initial begin
  clk = 0;
  forever
    #25 clk = ~clk;
end

initial begin
  rst = 1;
  repeat (4) @(negedge clk);
  rst = 0;
  repeat (100) begin
    @(negedge clk);
    if (y_DUT != y_REF) begin
      $display("The FSM Design is Wrong! ");
      $stop;
    end
  end
  $stop;
end

initial
$monitor("time = %0t, CLK = %b, RST = %b, Gray_out = %b",
                                                    $time, clk, rst, y_DUT);

endmodule
```

Do File for (Gray_Code_Moore) Question 3

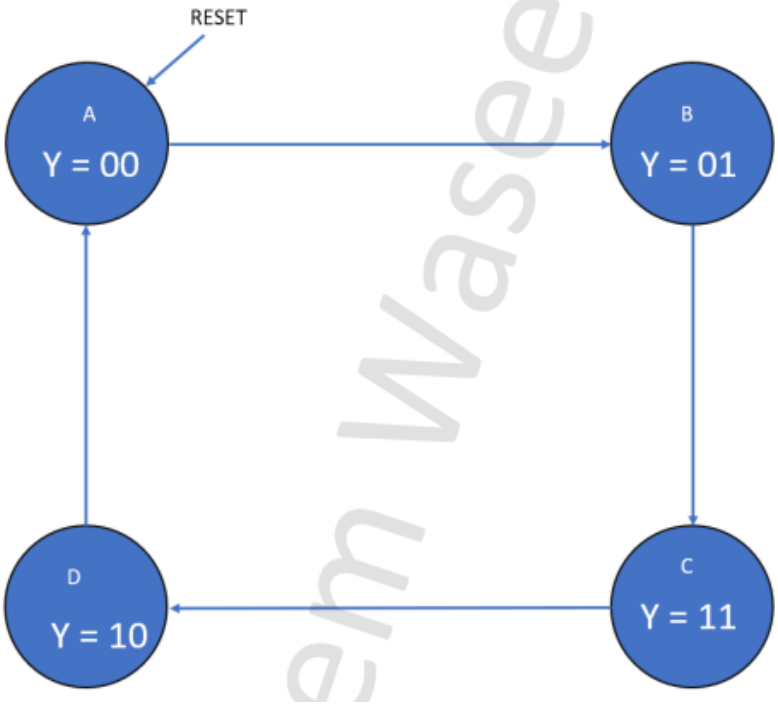
```
vlib work
vlog Gray_Code_Moore. v Gray_Code_Ref. v Gray_Code_Moore_tb. v
vsim  − voptargs = +acc work. Gray_Code_Moore_tb
add wave  *
run  − all
```

Waveform:





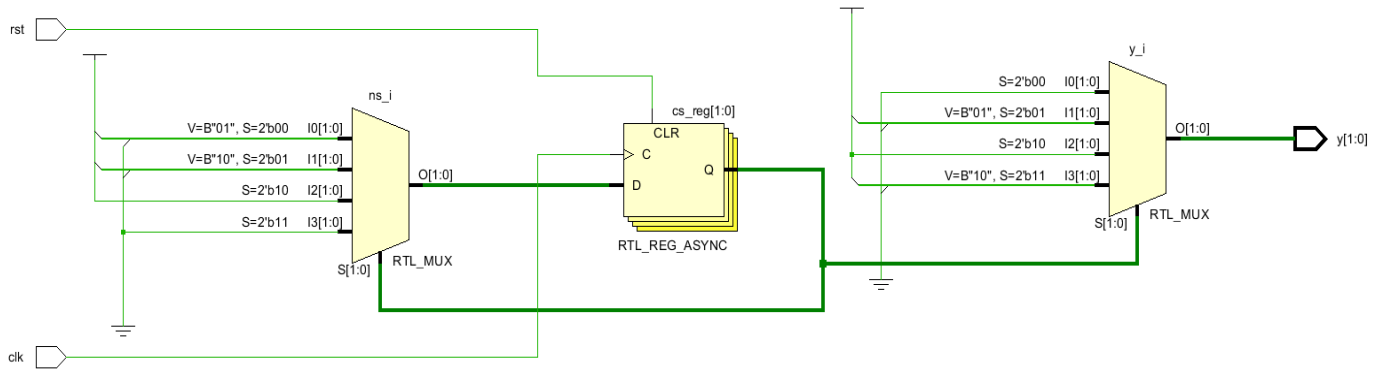

```
# time = 100, CLK = 0, RST = 1, Gray_out = 00
# time = 125, CLK = 1, RST = 1, Gray_out = 00
# time = 150, CLK = 0, RST = 1, Gray_out = 00
# time = 175, CLK = 1, RST = 1, Gray_out = 00
# time = 200, CLK = 0, RST = 0, Gray_out = 00
# time = 225, CLK = 1, RST = 0, Gray_out = 01
# time = 250, CLK = 0, RST = 0, Gray_out = 01
# time = 275, CLK = 1, RST = 0, Gray_out = 11
# time = 300, CLK = 0, RST = 0, Gray_out = 11
# time = 325, CLK = 1, RST = 0, Gray_out = 10
# time = 350, CLK = 0, RST = 0, Gray_out = 10
# time = 375, CLK = 1, RST = 0, Gray_out = 00
# time = 400, CLK = 0, RST = 0, Gray_out = 00
# time = 425, CLK = 1, RST = 0, Gray_out = 01
# time = 450, CLK = 0, RST = 0, Gray_out = 01
# time = 475, CLK = 1, RST = 0, Gray_out = 11
# time = 500, CLK = 0, RST = 0, Gray_out = 11
# time = 525, CLK = 1, RST = 0, Gray_out = 10
# time = 550, CLK = 0, RST = 0, Gray_out = 10
# time = 575, CLK = 1, RST = 0, Gray_out = 00
# time = 600, CLK = 0, RST = 0, Gray_out = 00
# time = 625, CLK = 1, RST = 0, Gray_out = 01
# time = 650, CLK = 0, RST = 0, Gray_out = 01
# time = 675, CLK = 1, RST = 0, Gray_out = 11
# time = 700, CLK = 0, RST = 0, Gray_out = 11
# time = 725, CLK = 1, RST = 0, Gray_out = 10
# time = 750, CLK = 0, RST = 0, Gray_out = 10
# time = 775, CLK = 1, RST = 0, Gray_out = 00
# time = 800, CLK = 0, RST = 0, Gray_out = 00
# time = 825, CLK = 1, RST = 0, Gray_out = 01
# time = 850, CLK = 0, RST = 0, Gray_out = 01
# time = 875, CLK = 1, RST = 0, Gray_out = 11
# time = 900, CLK = 0, RST = 0, Gray_out = 11
# time = 925, CLK = 1, RST = 0, Gray_out = 10
# time = 950, CLK = 0, RST = 0, Gray_out = 10
# time = 975, CLK = 1, RST = 0, Gray_out = 00
# time = 1000, CLK = 0, RST = 0, Gray_out = 00
```



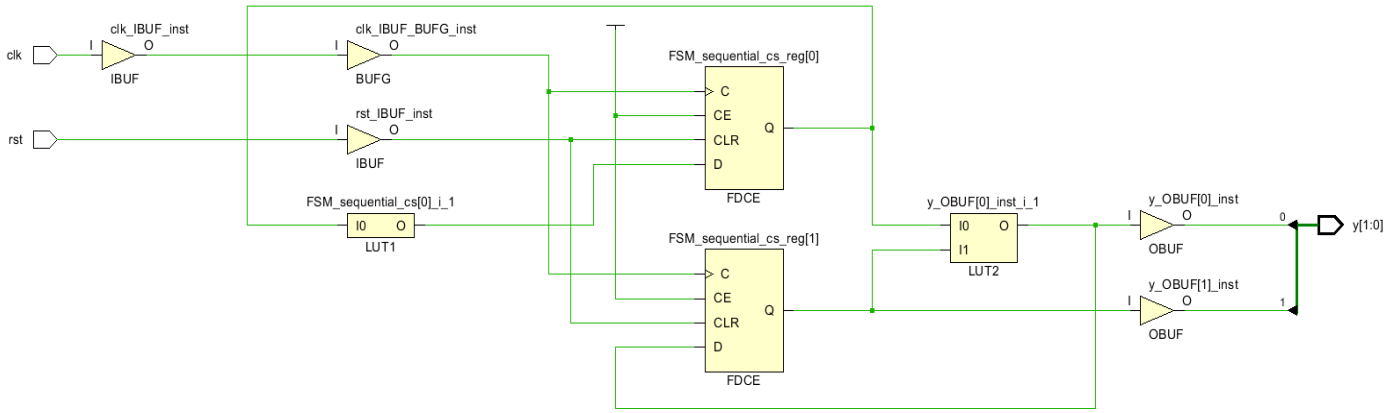
Constraint File:

```
## Clock signal
set_property -dict { PACKAGE_PIN W5   IOSTANDARD LVCMOS33 } [get_ports clk]
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]
## LEDs
set_property -dict { PACKAGE_PIN U16   IOSTANDARD LVCMOS33 } [get_ports {y[0]}]
set_property -dict { PACKAGE_PIN E19   IOSTANDARD LVCMOS33 } [get_ports {y[1]}]
##Buttons
set_property -dict { PACKAGE_PIN U18   IOSTANDARD LVCMOS33 } [get_ports rst]

## Configuration options, can be used for all designs
set_property CONFIG_VOLTAGE 3.3 [current_design]
set_property CFGBVS VCCO [current_design]

## SPI configuration mode options for QSPI boot, can be used for all designs
set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
set_property BITSTREAM.CONFIG.CONFIGRATE 33 [current_design]
set_property CONFIG_MODE SPIx4 [current_design]
```

Schematic after the Elaboration:



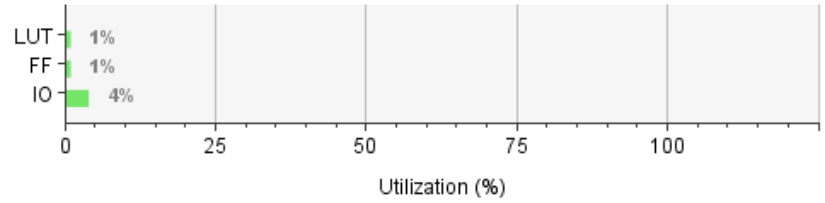
Schematic after the Synthesis:



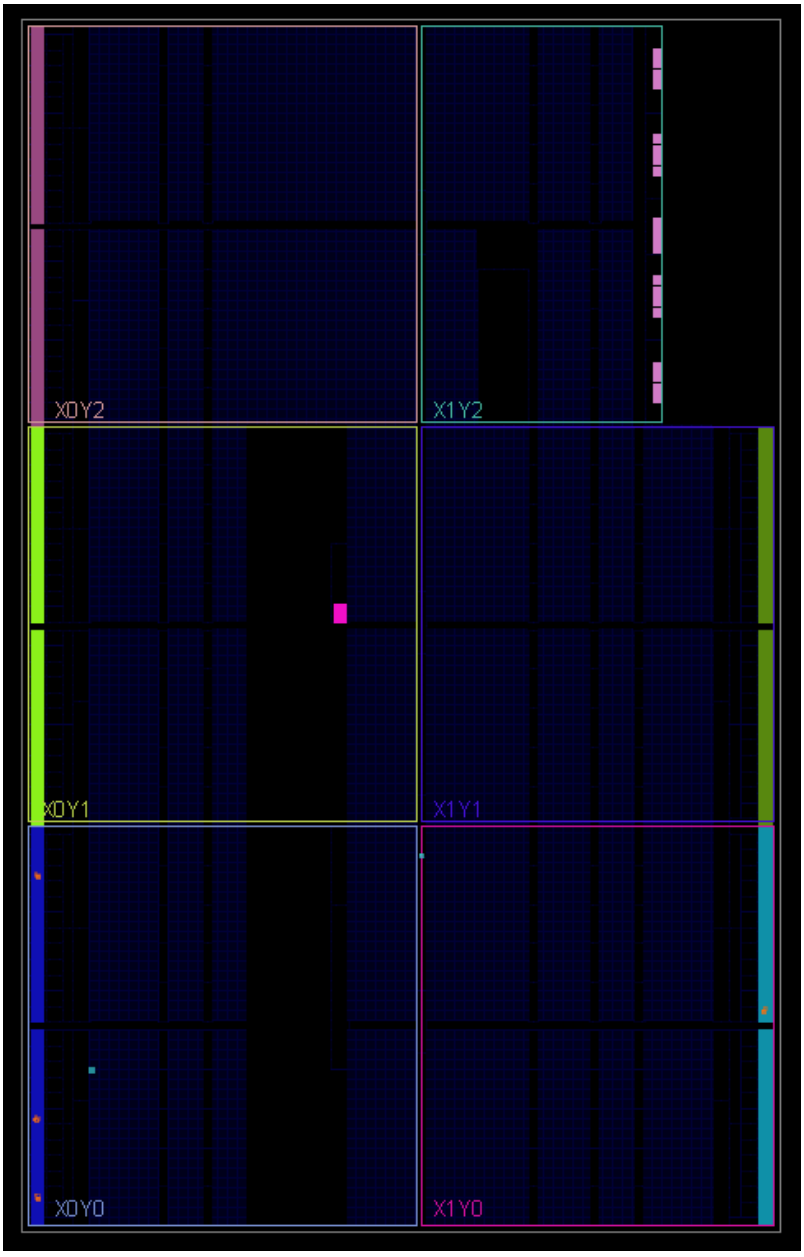
Utilization Report:

Summary

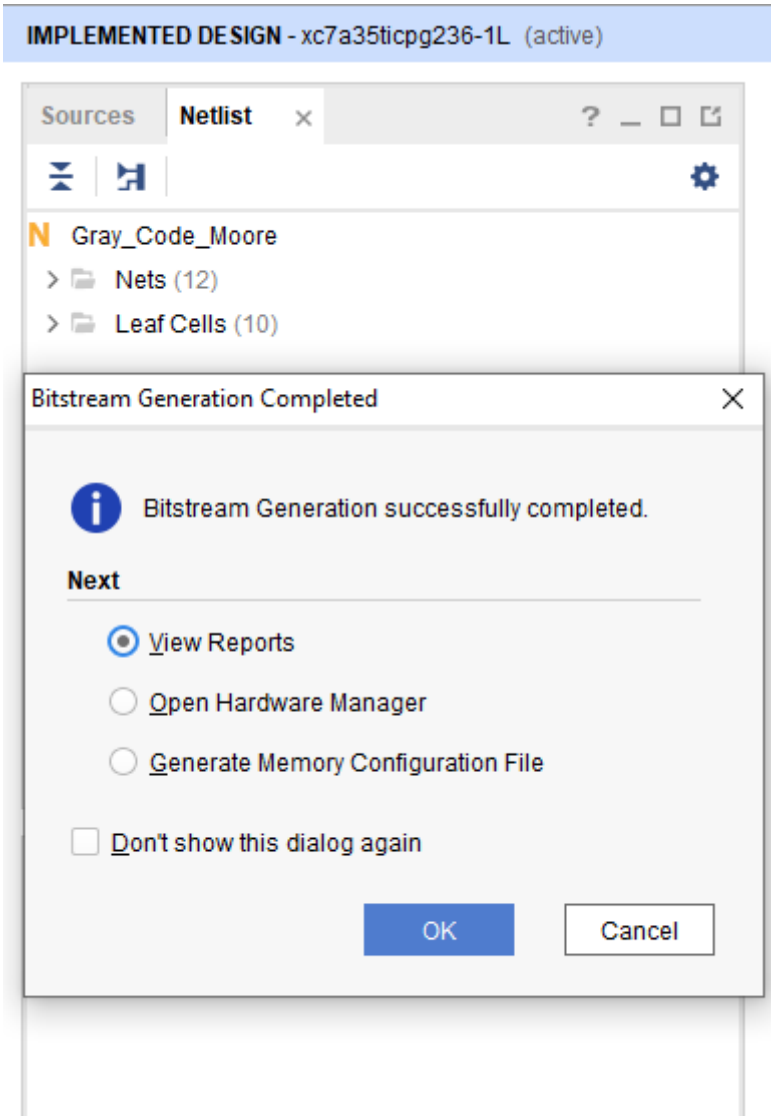
Resource	Utilization	Available	Utilization %
LUT	2	20800	0.01
FF	2	41600	0.00
IO	4	106	3.77



Device after implementation:



Successful Bitstream Generation:

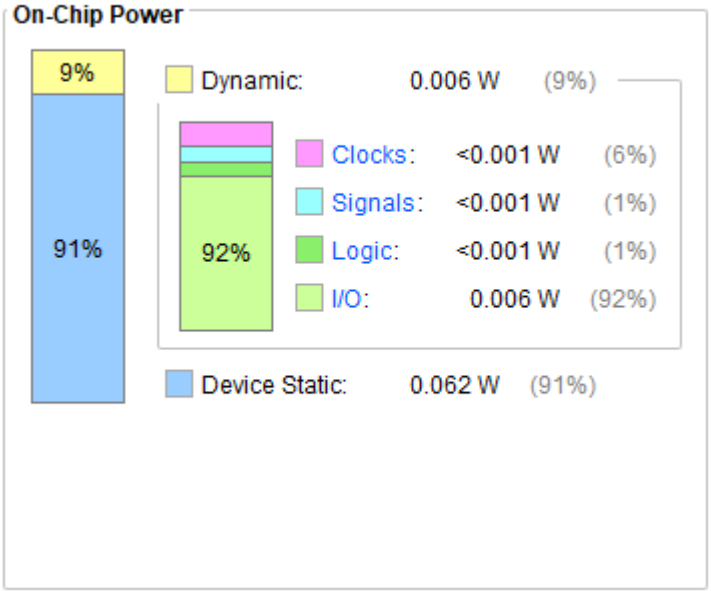


Summary

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**0.068 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**25.3°C**
Thermal Margin:	74.7°C (14.9 W)
Effective ϑJA:	5.0°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Medium

Launch Power Constraint Advisor to find and fix invalid switching activity



Question 4

SEQUENCE DETECTOR MEALY

Verilog Code:

```
module Sequence_Mealy (clk, rst, in, y);
parameter S0 = 2'b00;
parameter S1 = 2'b01;
parameter S2 = 2'b10;
parameter S3 = 2'b11;
input clk, rst, in;
output reg y;
reg [1: 0] cs, ns;          // Next State Logic
always @(*) begin
  case (cs)
    S0 : if (in)
          ns = S1;
        else
          ns = S0;
    S1 : if (in)
          ns = S2;
        else
          ns = S1;
    S2 : if (in)
          ns = S3;
        else
          ns = S2;
    S3 : if (in)
          ns = S1;
        else
          ns = S0;
  endcase
end                              // Memory State
always @(posedge clk or posedge rst) begin
  if (rst)
    cs <= S0;
  else
    cs <= ns;
end                             // Output Logic
always @(*) begin
  if ((cs == S2) && (in == 1))
    y = 1;
  else
    y = 0;
end  endmodule
```

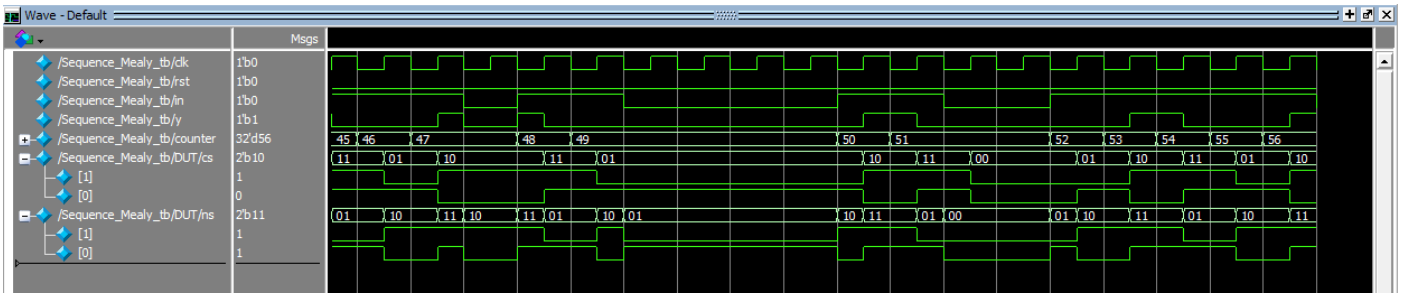
Testbench Code:

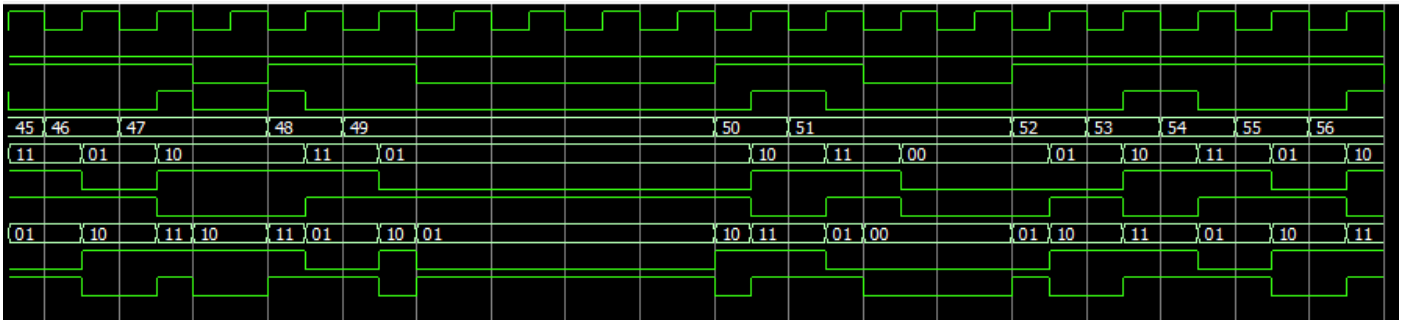
```
module Sequence_Mealy_tb ();

parameter S0 = 2'b00;
parameter S1 = 2'b01;
parameter S2 = 2'b10;
parameter S3 = 2'b11;

reg clk, rst, in;
wire y;

Sequence_Mealy #(S0, S1, S2, S3) DUT(clk, rst, in, y);

integer counter = 0;

initial begin
  clk = 0;
  forever
    #25 clk = ~clk;
end

initial begin
  rst = 1; in = 0;
  repeat (2) @(negedge clk); // Delay for 2 Clock Cycles
  rst = 0;
  repeat (100) begin
    @(negedge clk);
    in = $random;
    if (in == 1)
      counter = counter + 1;
  end
  $stop;
end

always @(y) begin
  if (y) begin
    if ((counter % 3 == 0 && y != 1) || (counter % 3 != 0 && y == 0)) begin
      $display("The FSM Design is Wrong ! ");
      $stop;
    end
  end
end

initial
$monitor("time = %0t, clk = %b, rst = %b, in = %b, y = %b, Counter = %b",
                                                    $time, clk, rst, in, y, counter);

endmodule
```

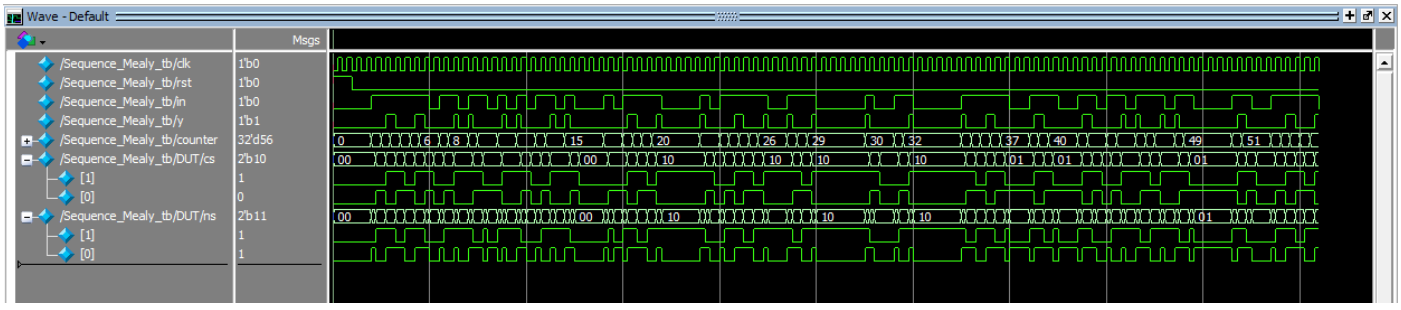
Do File for (Sequence_Mealy) Question 4

```
vlib work
vlog Sequence_Mealy. v Sequence_Mealy_tb. v
vsim  − voptargs = +acc work. Sequence_Mealy_tb
add wave  ∗
run  − all
```

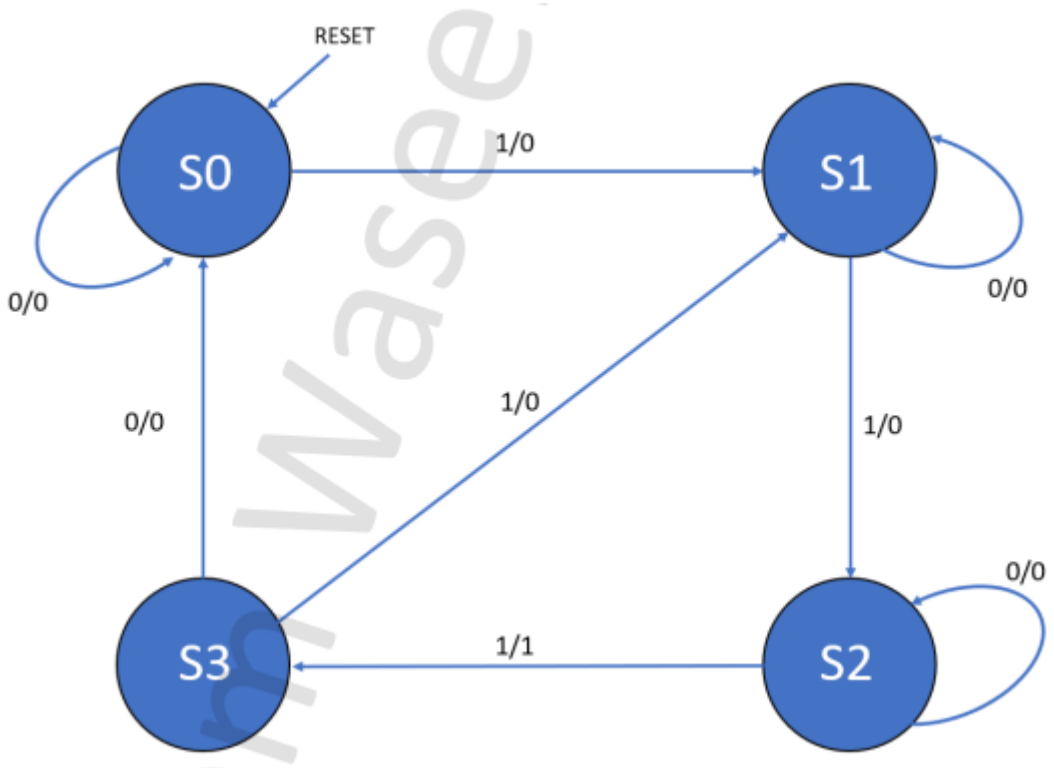
Waveform:







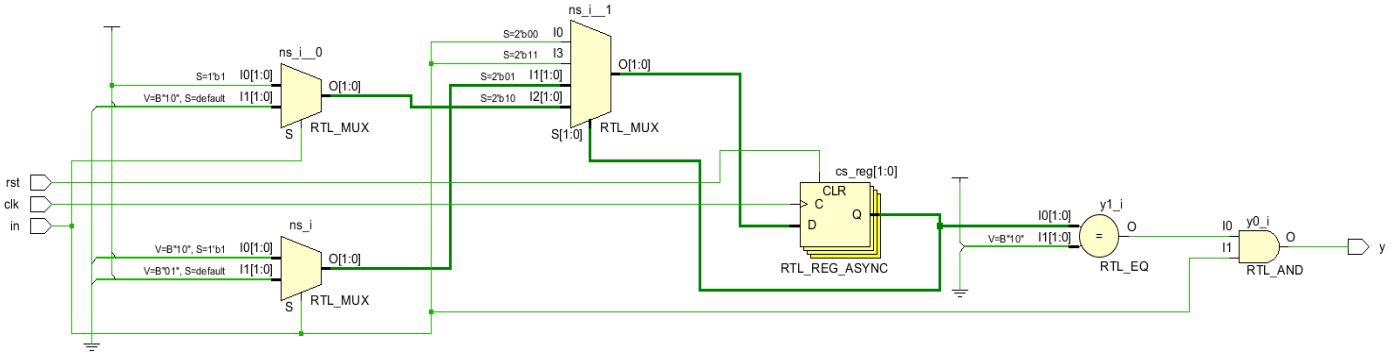
```
# time = 100, clk = 0, rst = 0, in = 0, y = 0, Counter = 00000000000000000000000000000000
# time = 125, clk = 1, rst = 0, in = 0, y = 0, Counter = 00000000000000000000000000000000
# time = 150, clk = 0, rst = 0, in = 0, y = 0, Counter = 00000000000000000000000000000000
# time = 175, clk = 1, rst = 0, in = 0, y = 0, Counter = 00000000000000000000000000000000
# time = 200, clk = 0, rst = 0, in = 1, y = 0, Counter = 00000000000000000000000000000001
# time = 225, clk = 1, rst = 0, in = 1, y = 0, Counter = 00000000000000000000000000000001
# time = 250, clk = 0, rst = 0, in = 1, y = 0, Counter = 00000000000000000000000000000010
# time = 275, clk = 1, rst = 0, in = 1, y = 1, Counter = 00000000000000000000000000000010
# time = 300, clk = 0, rst = 0, in = 1, y = 1, Counter = 00000000000000000000000000000011
# time = 325, clk = 1, rst = 0, in = 1, y = 0, Counter = 00000000000000000000000000000011
# time = 350, clk = 0, rst = 0, in = 1, y = 0, Counter = 00000000000000000000000000000100
# time = 375, clk = 1, rst = 0, in = 1, y = 0, Counter = 00000000000000000000000000000100
# time = 400, clk = 0, rst = 0, in = 1, y = 0, Counter = 00000000000000000000000000000101
# time = 425, clk = 1, rst = 0, in = 1, y = 1, Counter = 00000000000000000000000000000101
# time = 450, clk = 0, rst = 0, in = 1, y = 1, Counter = 00000000000000000000000000000110
# time = 475, clk = 1, rst = 0, in = 1, y = 0, Counter = 00000000000000000000000000000110
# time = 500, clk = 0, rst = 0, in = 0, y = 0, Counter = 00000000000000000000000000000110
# time = 525, clk = 1, rst = 0, in = 0, y = 0, Counter = 00000000000000000000000000000110
# time = 550, clk = 0, rst = 0, in = 1, y = 0, Counter = 00000000000000000000000000000111
# time = 575, clk = 1, rst = 0, in = 1, y = 0, Counter = 00000000000000000000000000000111
# time = 600, clk = 0, rst = 0, in = 1, y = 0, Counter = 00000000000000000000000000001000
# time = 625, clk = 1, rst = 0, in = 1, y = 1, Counter = 00000000000000000000000000001000
# time = 650, clk = 0, rst = 0, in = 0, y = 0, Counter = 00000000000000000000000000001000
# time = 675, clk = 1, rst = 0, in = 0, y = 0, Counter = 00000000000000000000000000001000
# time = 700, clk = 0, rst = 0, in = 1, y = 1, Counter = 00000000000000000000000000001001
# time = 725, clk = 1, rst = 0, in = 1, y = 0, Counter = 00000000000000000000000000001001
# time = 750, clk = 0, rst = 0, in = 1, y = 0, Counter = 00000000000000000000000000001010
# time = 775, clk = 1, rst = 0, in = 1, y = 0, Counter = 00000000000000000000000000001010
# time = 800, clk = 0, rst = 0, in = 0, y = 0, Counter = 00000000000000000000000000001010
# time = 825, clk = 1, rst = 0, in = 0, y = 0, Counter = 00000000000000000000000000001010
# time = 850, clk = 0, rst = 0, in = 1, y = 0, Counter = 00000000000000000000000000001011
# time = 875, clk = 1, rst = 0, in = 1, y = 1, Counter = 00000000000000000000000000001011
# time = 900, clk = 0, rst = 0, in = 0, y = 0, Counter = 00000000000000000000000000001011
# time = 925, clk = 1, rst = 0, in = 0, y = 0, Counter = 00000000000000000000000000001011
# time = 950, clk = 0, rst = 0, in = 1, y = 1, Counter = 00000000000000000000000000001100
# time = 975, clk = 1, rst = 0, in = 1, y = 0, Counter = 00000000000000000000000000001100
# time = 1000, clk = 0, rst = 0, in = 0, y = 0, Counter = 00000000000000000000000000001100
# time = 1025, clk = 1, rst = 0, in = 0, y = 0, Counter = 00000000000000000000000000001100
# time = 1050, clk = 0, rst = 0, in = 1, y = 0, Counter = 00000000000000000000000000001101
# time = 1075, clk = 1, rst = 0, in = 1, y = 0, Counter = 00000000000000000000000000001101
# time = 1100, clk = 0, rst = 0, in = 1, y = 0, Counter = 00000000000000000000000000001110
```



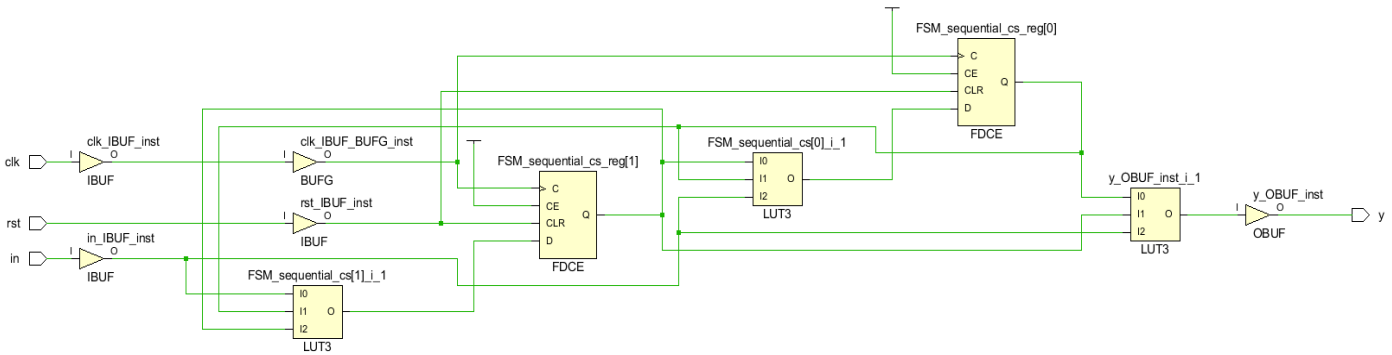
Constraint File:

```
## Clock signal
set_property -dict { PACKAGE_PIN W5   IOSTANDARD LVCMOS33 } [get_ports clk]
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]

## Switches
set_property -dict { PACKAGE_PIN V17   IOSTANDARD LVCMOS33 } [get_ports {in}]

## LEDs
set_property -dict { PACKAGE_PIN U16   IOSTANDARD LVCMOS33 } [get_ports {y}]

##Buttons
set_property -dict { PACKAGE_PIN U18   IOSTANDARD LVCMOS33 } [get_ports rst]

## Configuration options, can be used for all designs
set_property CONFIG_VOLTAGE 3.3 [current_design]
set_property CFGBVS VCCO [current_design]

## SPI configuration mode options for QSPI boot, can be used for all designs
set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
set_property BITSTREAM.CONFIG.CONFIGRATE 33 [current_design]
set_property CONFIG_MODE SPIx4 [current_design]
```

Schematic after the Elaboration:



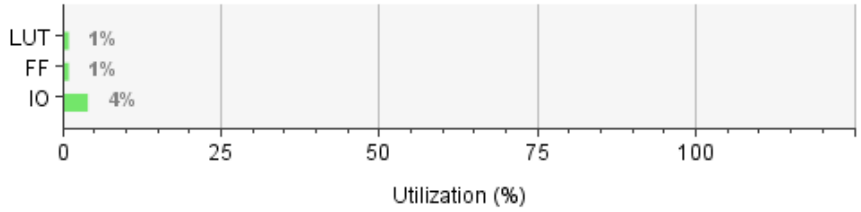
Schematic after the Synthesis:



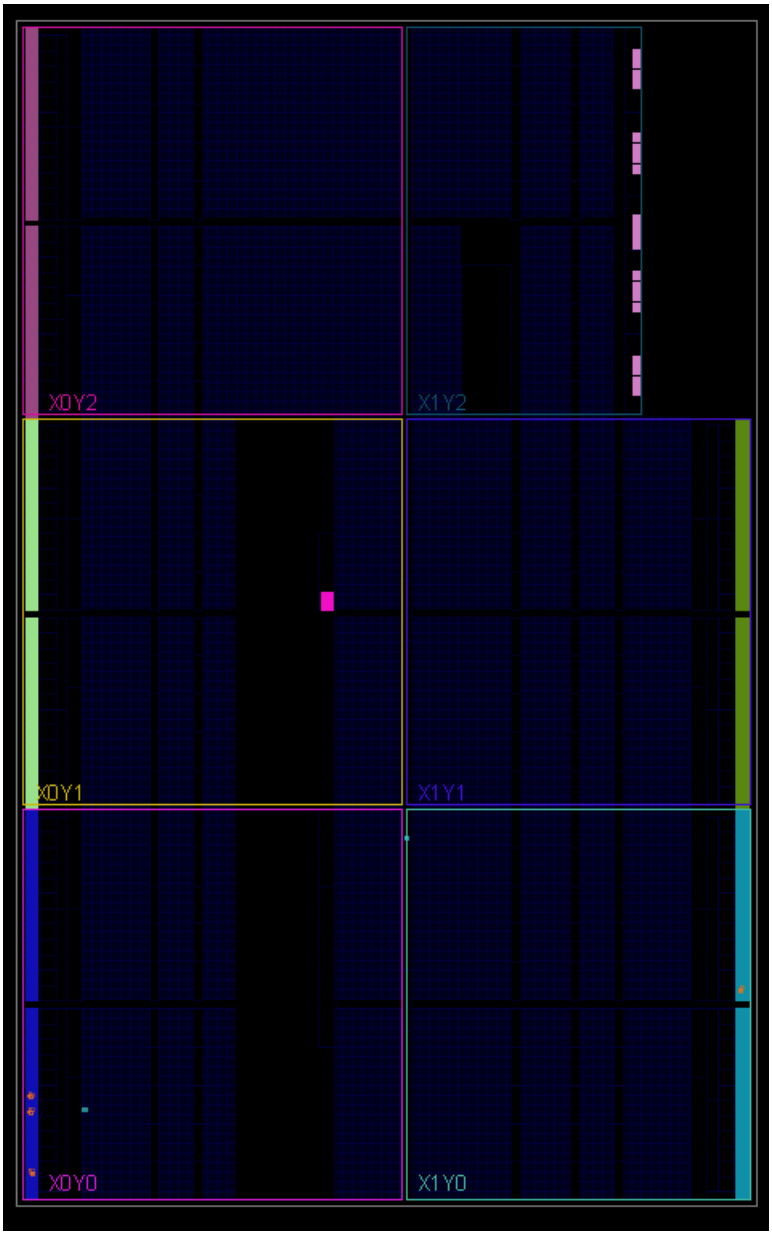
Utilization Report:

Summary

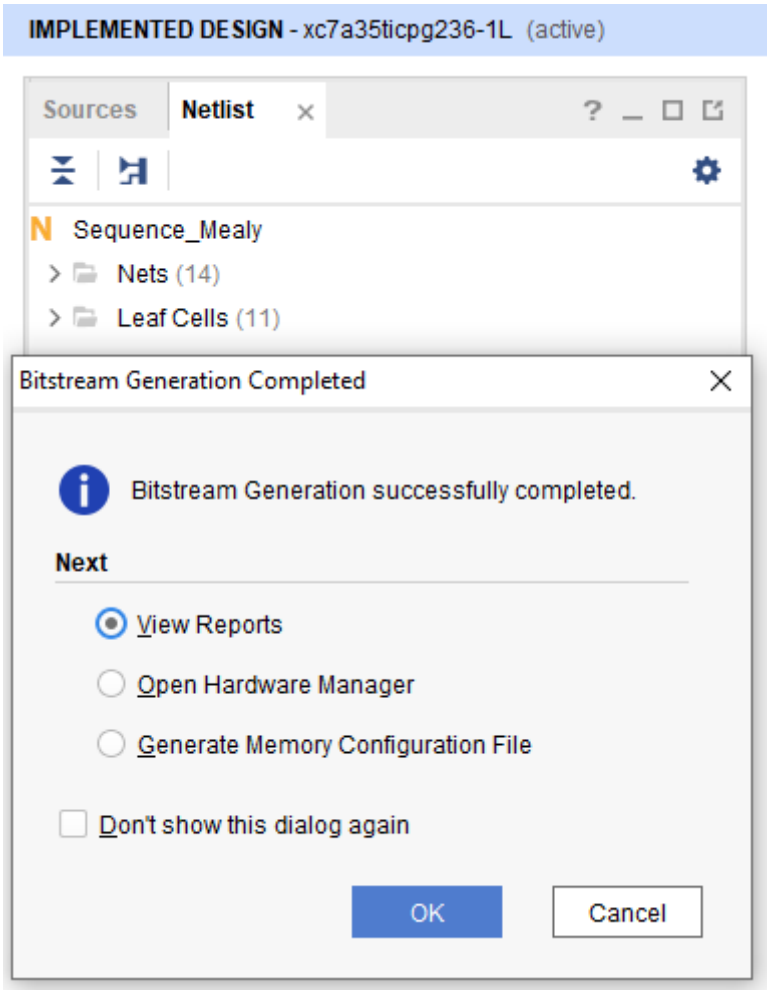
Resource	Utilization	Available	Utilization %
LUT	3	20800	0.01
FF	2	41600	0.00
IO	4	106	3.77



Device after implementation:



Successful Bitstream Generation:

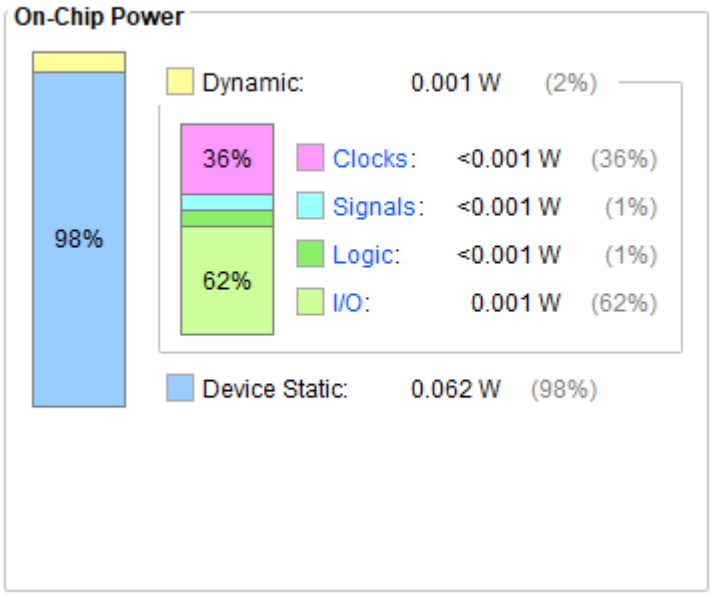


Summary

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**0.063 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**25.3°C**
Thermal Margin:	74.7°C (14.9 W)
Effective ϑJA:	5.0°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Medium

Launch Power Constraint Advisor to find and fix invalid switching activity



Question 5

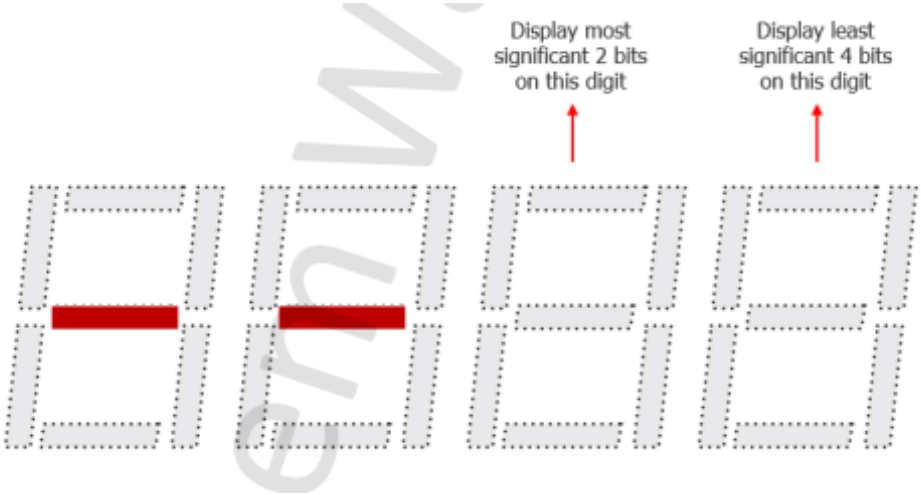
ALSU

Verilog Code:

```
module ALSU (A, B, opcode, cin, serial_in, direction, red_op_A, red_op_B, bypass_A, bypass_B, clk, rst, out, leds);

parameter INPUT_PRIORITY = "A";
parameter FULL_ADDER = "ON";
input clk, rst, cin, serial_in, red_op_A, red_op_B, bypass_A, bypass_B, direction;
input [2: 0] A, B, opcode;
output reg [15: 0] leds;
output reg [5: 0] out;
reg [2: 0] A_FF, B_FF, opcode_FF;
reg cin_FF, serial_in_FF, red_op_A_FF, red_op_B_FF, bypass_A_FF, bypass_B_FF, direction_FF;
wire invalid_red_op, invalid_opcode, invalid;
assign invalid_red_op = (red_op_A_FF | red_op_B_FF) & (opcode_FF[1] | opcode_FF[2]);
assign invalid_opcode = opcode_FF[1] & opcode_FF[2];
assign invalid = invalid_red_op | invalid_opcode;
wire [5: 0] multi_out;
wire [3: 0] adder_out;

generate
  if (FULL_ADDER == "ON")
    c_addsub_0 Adder (
    .A(A_FF),      // input wire [2 : 0] A
    .B(B_FF),      // input wire [2 : 0] B
    .C_IN(cin_FF), // input wire C_IN
    .S(adder_out)      // output wire [3 : 0] S
    );
  else if (FULL_ADDER == "OFF")
    c_addsub_0 Adder (
    .A(A_FF),      // input wire [2 : 0] A
    .B(B_FF),      // input wire [2 : 0] B
    .C_IN(0), // input wire C_IN
    .S(adder_out)      // output wire [3 : 0] S
    );
endgenerate

mult_gen_0 Multiplier (
 .A(A_FF), // input wire [2 : 0] A
 .B(B_FF), // input wire [2 : 0] B
 .P(multi_out) // output wire [5 : 0] P
);
```

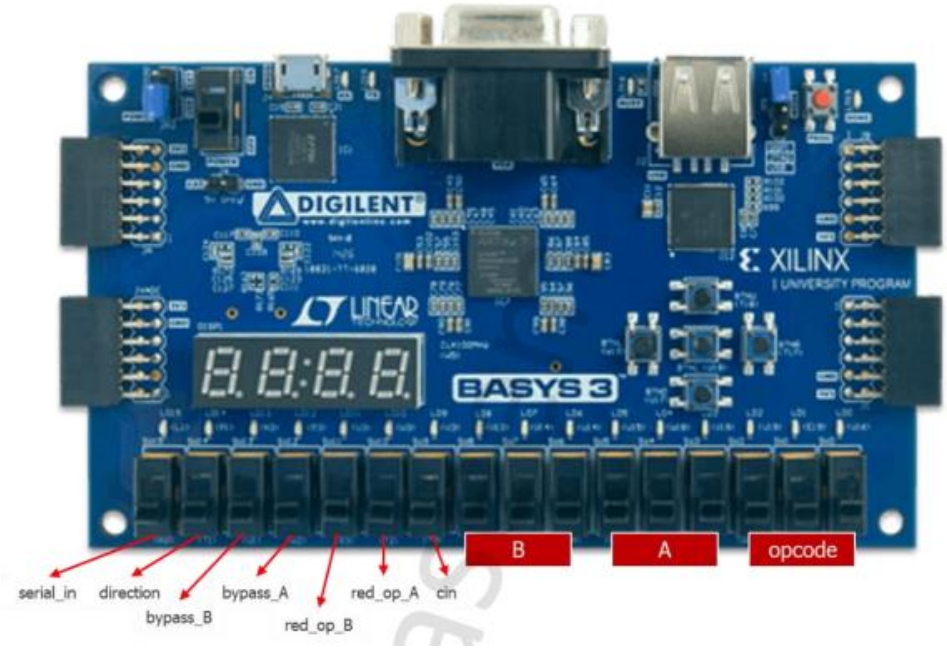
```
always @(posedge clk or posedge rst) begin
  if (rst) begin
    A_FF <= 0; B_FF <= 0; opcode_FF <= 0; cin_FF <= 0; serial_in_FF <= 0; red_op_A_FF <= 0;
    red_op_B_FF <= 0; bypass_A_FF <= 0; bypass_B_FF <= 0; direction_FF <= 0;
  end begin
    A_FF <= A; B_FF <= B; opcode_FF <= opcode; cin_FF <= cin; serial_in_FF <= serial_in; red_op_A_FF
                <= red_op_A;
    red_op_B_FF <= red_op_B; bypass_A_FF <= bypass_A; bypass_B_FF <= bypass_B; direction_FF <
                = direction;
  end
end
always @(posedge clk or posedge rst) begin
  if (rst) begin
    leds <= 0;
  end else begin
    if (invalid)
      leds = ~leds;
    leds <= 0;
  end
end
always @(posedge clk or posedge rst) begin
  if (rst) begin
    out <= 0;
  end
  else begin
    if (bypass_A_FF && bypass_B_FF)
      out <= (INPUT_PRIORITY == "A")? A_FF : B_FF;
    else if (bypass_A_FF)
      out <= A_FF;
    else if (bypass_B_FF)
      out <= B_FF;
    else begin
      if (invalid) begin
        out <= 0;
      end else begin
        case (opcode_FF)
          3'h0 : begin
            if (red_op_A_FF && red_op_B_FF)
              out <= (INPUT_PRIORITY == "A")? &A_FF : &B_FF;
            else if (red_op_A_FF)
              out <= &A_FF;
            else if (red_op_B_FF)
              out <= &B_FF;
            else
              out <= A_FF & B_FF;
          end
```

```
            3'h1 : begin
              if (red_op_A_FF && red_op_B_FF)
                out <= (INPUT_PRIORITY == "A")? ^A_FF : ^B_FF;
              else if (red_op_A_FF)
                out <= ^A_FF;
              else if (red_op_B_FF)
                out <= ^B_FF;
              else
                out <= A_FF ^ B_FF;
            end
            3'h2 : out <= adder_out;
            3'h3 : out <= multi_out;
            3'h4 : begin
              if (direction_FF)
                out <= {out[4:0], serial_in_FF};
              else
                out <= {serial_in_FF, out[5:1]};
            end
            3'h5 : begin
              if (direction_FF)
                out <= {out[4:0], out[5]};
              else
                out <= {out[0], out[5:1]};
            end
          endcase
        end
      end
    end
endmodule
```

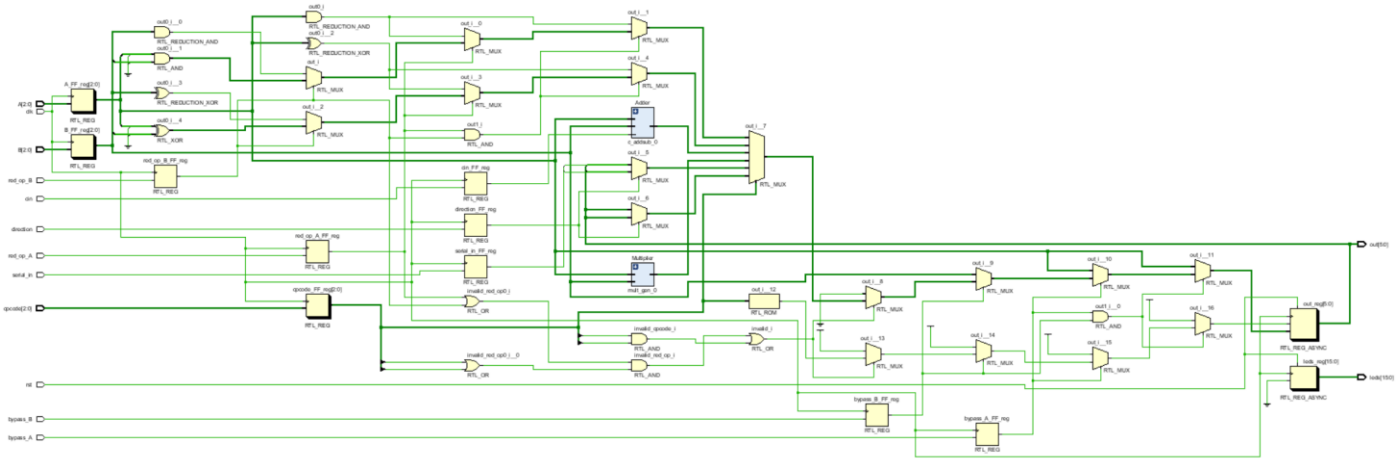


Constraint File:

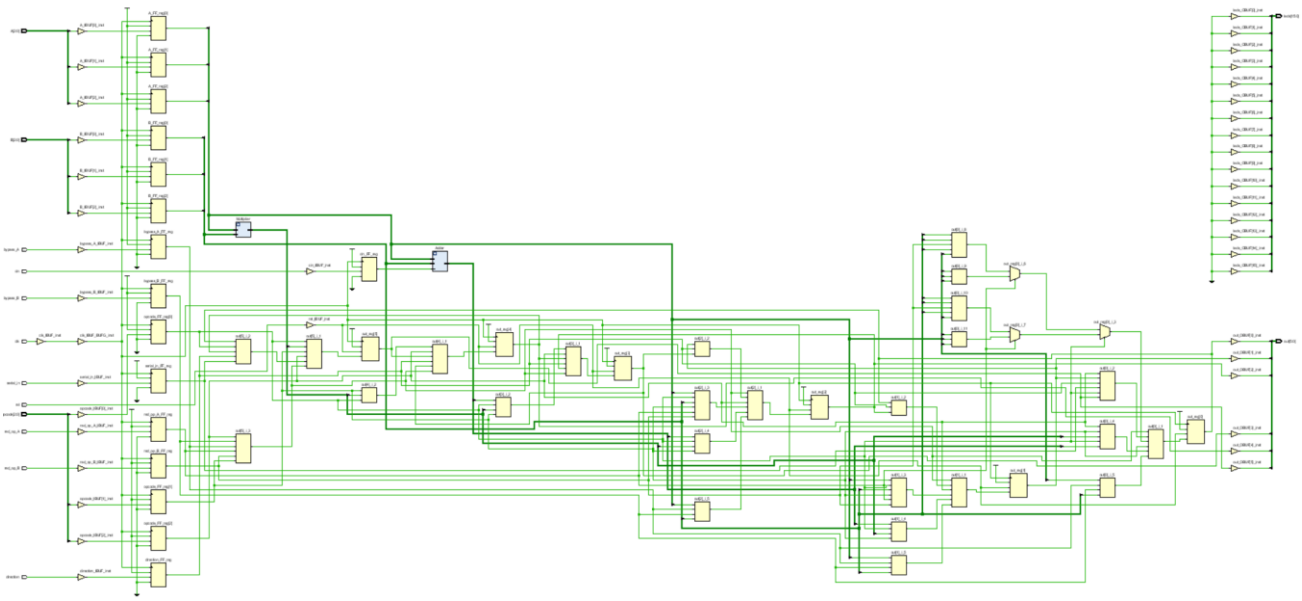
```
## Clock signal
set_property -dict {PACKAGE_PIN W5 IOSTANDARD LVCMOS33} [get_ports clk]
create_clock -period 10.000 -name sys_clk_pin -waveform {0.000 5.000} -add [get_ports clk]
## Switches
set_property -dict {PACKAGE_PIN V17 IOSTANDARD LVCMOS33} [get_ports {opcode[0]}]
set_property -dict {PACKAGE_PIN V16 IOSTANDARD LVCMOS33} [get_ports {opcode[1]}]
set_property -dict {PACKAGE_PIN W16 IOSTANDARD LVCMOS33} [get_ports {opcode[2]}]
set_property -dict {PACKAGE_PIN W17 IOSTANDARD LVCMOS33} [get_ports {A[0]}]
set_property -dict {PACKAGE_PIN W15 IOSTANDARD LVCMOS33} [get_ports {A[1]}]
set_property -dict {PACKAGE_PIN V15 IOSTANDARD LVCMOS33} [get_ports {A[2]}]
set_property -dict {PACKAGE_PIN W14 IOSTANDARD LVCMOS33} [get_ports {B[0]}]
set_property -dict {PACKAGE_PIN W13 IOSTANDARD LVCMOS33} [get_ports {B[1]}]
set_property -dict {PACKAGE_PIN V2 IOSTANDARD LVCMOS33} [get_ports {B[2]}]
set_property -dict {PACKAGE_PIN T3 IOSTANDARD LVCMOS33} [get_ports cin]
set_property -dict {PACKAGE_PIN T2 IOSTANDARD LVCMOS33} [get_ports red_op_A]
set_property -dict {PACKAGE_PIN R3 IOSTANDARD LVCMOS33} [get_ports red_op_B]
set_property -dict {PACKAGE_PIN W2 IOSTANDARD LVCMOS33} [get_ports bypass_A]
set_property -dict {PACKAGE_PIN U1 IOSTANDARD LVCMOS33} [get_ports bypass_B]
set_property -dict {PACKAGE_PIN T1 IOSTANDARD LVCMOS33} [get_ports direction]
set_property -dict {PACKAGE_PIN R2 IOSTANDARD LVCMOS33} [get_ports serial_in]
## LEDs
set_property -dict {PACKAGE_PIN U16 IOSTANDARD LVCMOS33} [get_ports {leds[0]}]
set_property -dict {PACKAGE_PIN E19 IOSTANDARD LVCMOS33} [get_ports {leds[1]}]
set_property -dict {PACKAGE_PIN U19 IOSTANDARD LVCMOS33} [get_ports {leds[2]}]
set_property -dict {PACKAGE_PIN V19 IOSTANDARD LVCMOS33} [get_ports {leds[3]}]
set_property -dict {PACKAGE_PIN W18 IOSTANDARD LVCMOS33} [get_ports {leds[4]}]
set_property -dict {PACKAGE_PIN U15 IOSTANDARD LVCMOS33} [get_ports {leds[5]}]
set_property -dict {PACKAGE_PIN U14 IOSTANDARD LVCMOS33} [get_ports {leds[6]}]
set_property -dict {PACKAGE_PIN V14 IOSTANDARD LVCMOS33} [get_ports {leds[7]}]
set_property -dict {PACKAGE_PIN V13 IOSTANDARD LVCMOS33} [get_ports {leds[8]}]
set_property -dict {PACKAGE_PIN V3 IOSTANDARD LVCMOS33} [get_ports {leds[9]}]
set_property -dict {PACKAGE_PIN W3 IOSTANDARD LVCMOS33} [get_ports {leds[10]}]
set_property -dict {PACKAGE_PIN U3 IOSTANDARD LVCMOS33} [get_ports {leds[11]}]
set_property -dict {PACKAGE_PIN P3 IOSTANDARD LVCMOS33} [get_ports {leds[12]}]
set_property -dict {PACKAGE_PIN N3 IOSTANDARD LVCMOS33} [get_ports {leds[13]}]
set_property -dict {PACKAGE_PIN P1 IOSTANDARD LVCMOS33} [get_ports {leds[14]}]
set_property -dict {PACKAGE_PIN L1 IOSTANDARD LVCMOS33} [get_ports {leds[15]}]
##Buttons
set_property -dict {PACKAGE_PIN U18 IOSTANDARD LVCMOS33} [get_ports rst]
## Configuration options, can be used for all designs
set_property CONFIG_VOLTAGE 3.3 [current_design]
set_property CFGBVS VCCO [current_design]

## SPI configuration mode options for QSPI boot, can be used for all designs
set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
set_property BITSTREAM.CONFIG.CONFIGRATE 33 [current_design]
set_property CONFIG_MODE SPIx4 [current_design]
```



Schematic after the Elaboration:



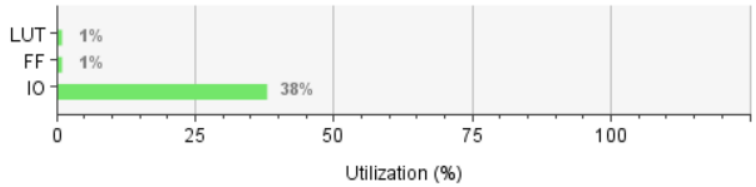
Schematic after the Synthesis:



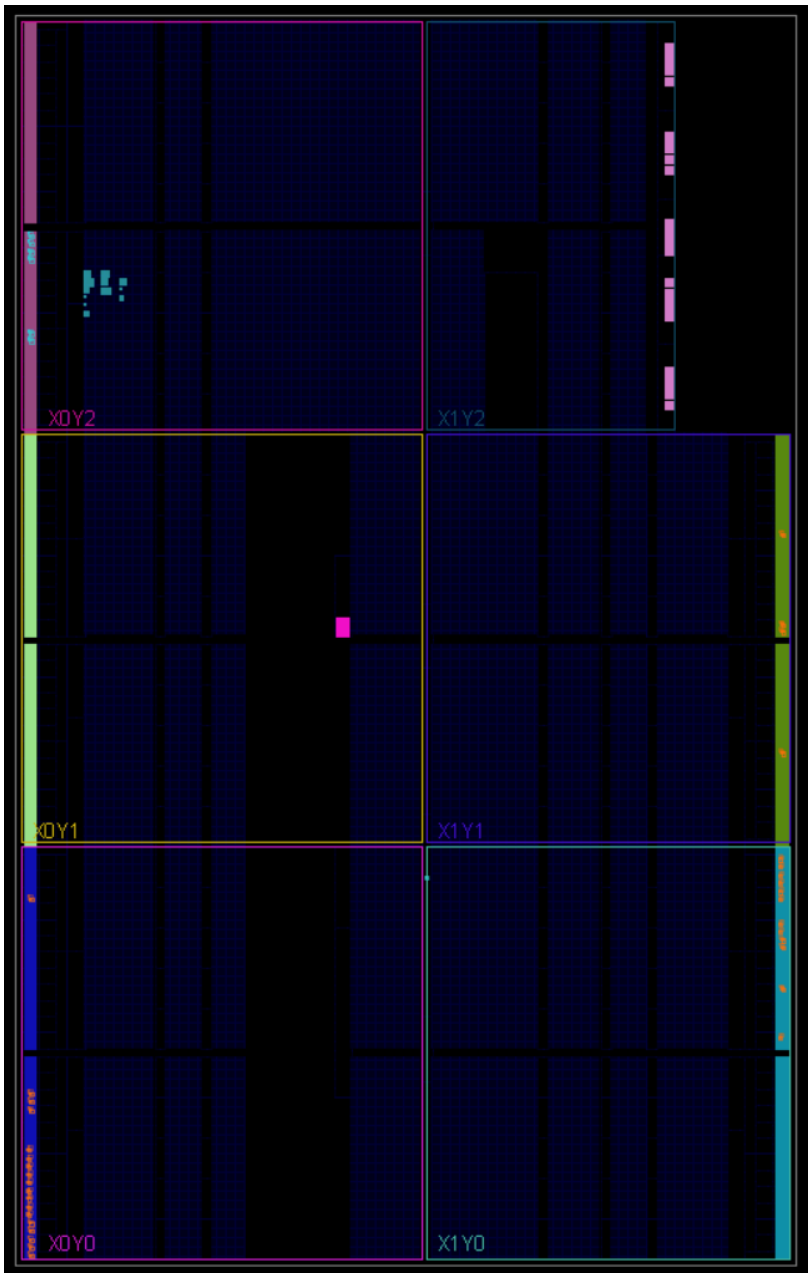
Utilization Report:

Summary

Resource	Utilization	Available	Utilization %
LUT	37	20800	0.18
FF	22	41600	0.05
IO	40	106	37.74



Device after implementation:



Successful Bitstream Generation: