Experiment 7: Exercise with Counters (Lab Project)

Lab Report

ELL201

Aditya Jaiswal

2023EE10512

October 29, 2024

Table Number 18

Tuesday Batch

Aditya Jaiswal 2023EE10512

Garv Patidar 2023EE31206

Samarth Tiwari 2023EE10550

Amit Kumar Meena 2023EE10524

Date of Submission: November 12, 2024

Teaching Assistant: Mr. Abhijeet Kumar

Aim

To design and implement a Synchronous 4-bit Gray-Code Counter using SR Flip-Flops. The counter should cycle through all 16 Gray-Code states, ensuring only one-bit changes between two consecutive states.

Devices Used

1. CPLD (Complex Programmable Logic Device) Board
2. JTAG Cable

Software Used

1. Intel Quartz II
2. JTAG Shell
3. EDA Playground Online Site for Waveforms

SR Flip-Flop

An SR (Set-Reset) Flip-Flop is a type of sequential logic circuit and a fundamental building block in digital electronics. It has two inputs, typically labelled S (Set) and R (Reset), and two outputs, Q and Q' (the inverse of Q). The SR Flip-Flop's state (whether Q is 0 or 1) depends on the inputs, which control the outputs as follows:

1. S = 1, R = 0: Sets the output Q to 1 (Q = 1), regardless of the previous state.
2. S = 0, R = 1: Resets the output Q to 0 (Q = 0).
3. S = 0, R = 0: Maintains the previous state, keeping Q as it was.
4. S = 1, R = 1: This is an invalid condition in most SR Flip-Flops as it creates ambiguity. The outputs can become unpredictable, so this input combination is typically avoided.

Procedure

- State Table Creation: Prepare a state table showing the sequence in Gray-Code format, where each state transition only differs by one bit.

- Flip-Flop Requirement Calculation: Calculate that four SR Flip-Flops are required to represent the 4-bit counter.

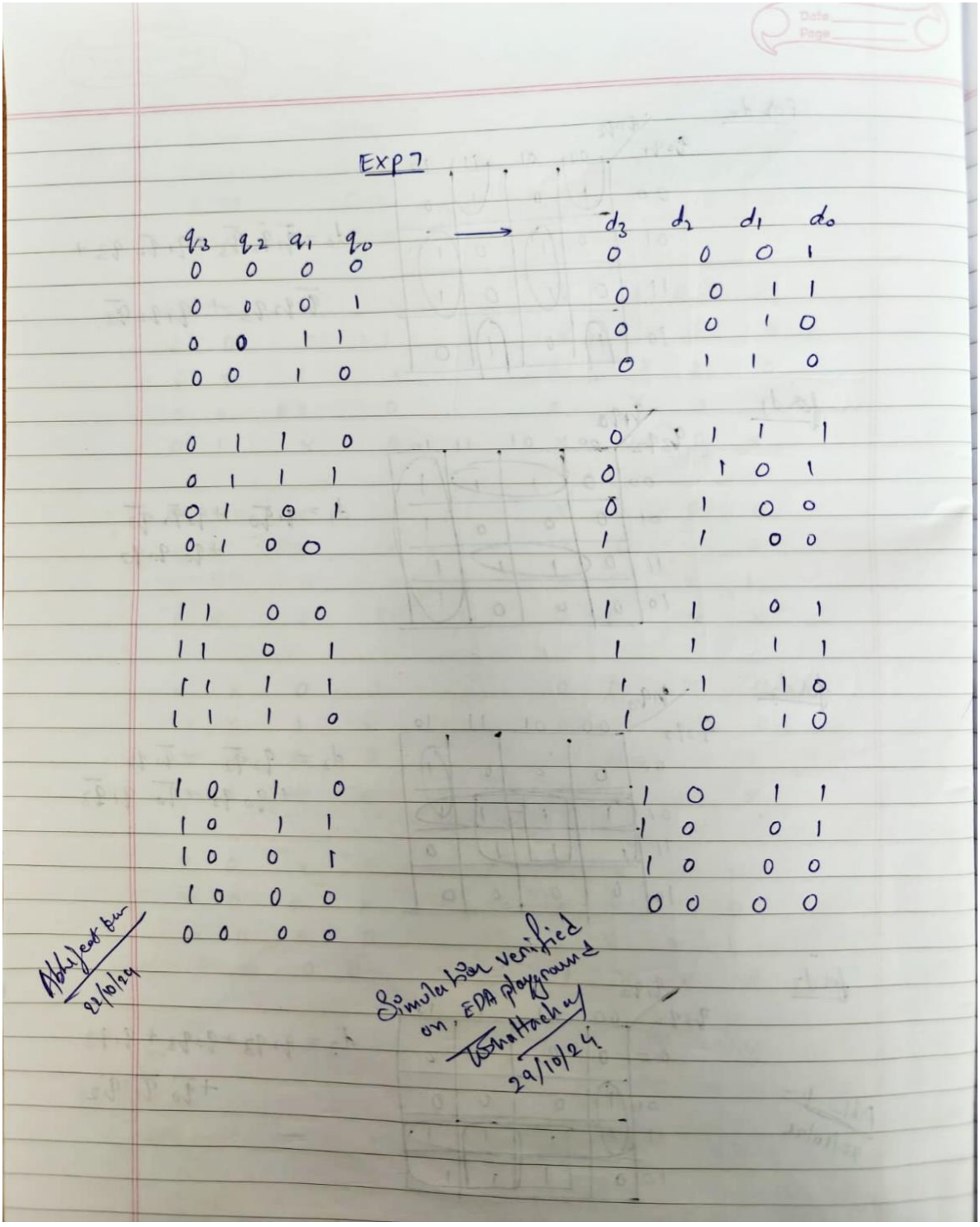
- Input Assignment for SR Flip-Flops: Assign input values to each SR Flip-Flop based on the state transitions.

- Karnaugh Map Simplification: Use Karnaugh Maps to derive the minimised Boolean expressions for the S (Set) and R (Reset) inputs of each Flip-Flop based on the current state outputs.

- Verilog/VHDL Coding: Write code in Verilog or VHDL to define the SR Flip-Flops and construct the Gray-Code counter logic using these Flip-Flops as components.

- Simulation: Simulate the counter design in the software to confirm that it covers all 16 states in a cyclic Gray-Code sequence. Testing: Verify that each state transition differs by only one bit, as expected for Gray-Code counting.

Observation Table



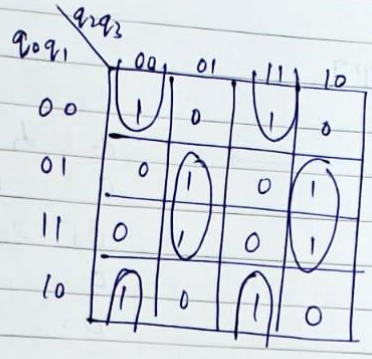
EXP 7

q_3	q_2	q_1	q_0	d_3	d_2	d_1	d_0
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	1
0	0	1	1	0	0	1	0
0	0	1	0	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	0	1	0	1
0	1	0	1	0	1	0	0
0	1	0	0	1	1	0	0
1	1	0	0	1	1	0	1
1	1	0	1	1	1	1	1
1	1	1	1	1	1	1	0
1	1	1	0	1	0	1	0
1	0	1	0	1	0	1	1
1	0	1	1	1	0	0	1
1	0	0	1	1	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0				

Abhijeet Bar 22/10/24

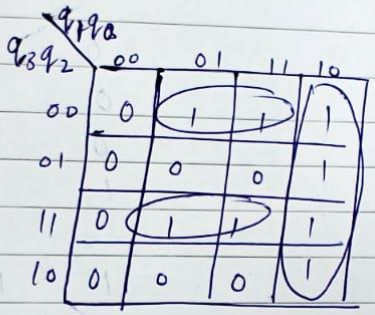
Simulation verified on EDA playground Vishnu Hachu 29/10/24

For d_0



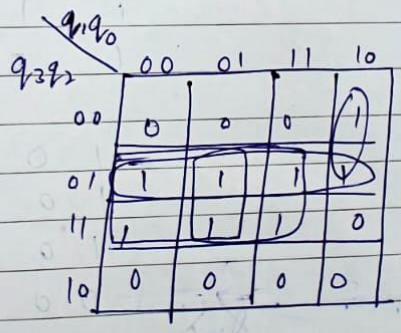
$d_0 = \bar{q}_1\bar{q}_2\bar{q}_3 + q_1\bar{q}_2q_3 + \bar{q}_1q_2q_3 + q_1q_2\bar{q}_3$

for d_1



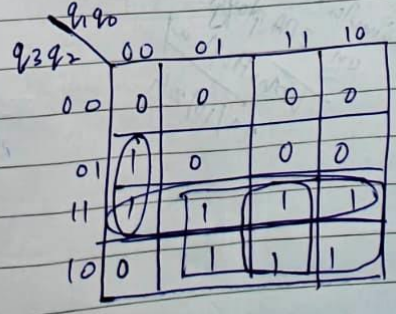
$d_1 = q_1\bar{q}_0 + q_0\bar{q}_2\bar{q}_3 + q_0q_2q_3$

for d_2



$d_2 = q_2\bar{q}_3 + \bar{q}_1q_2 + q_0q_2 + \bar{q}_0q_1\bar{q}_3$

for d_3



$d_3 = q_2q_3 + q_0q_3 + q_1q_3 + \bar{q}_0\bar{q}_1q_2$

Abhi 1:00
22/10/24

$0 \to 1$	$S=1$	$R=0$
$1 \to 0$	$S=0$	$R=1$
$0 \to 0$	$S=0$	$R=X$
$1 \to 1$	$S=X$	$R=0$

S_3	S_2	S_1	S_0	R_3	R_2	R_1	R_0
0	0	0	1	X	X	X	0
0	0	1	X	X	X	0	0
0	0	X	0	X	X	0	1
0	1	X	0	X	0	0	X
0	X	X	1	X	0	0	0
0	X	0	X	X	0	1	0
0	X	0	0	X	0	X	1
1	X	0	0	0	0	X	X
X	X	0	1	0	0	X	0
X	X	1	X	0	0	X	0
X	X	X	0	0	0	0	1
X	0	X	0	0	1	0	X
X	0	X	1	0	X	0	0
X	0	0	X	0	X	1	0
X	0	0	0	0	X	X	1
0	0	0	0	1	X	X	X

Abhijeet Das
02/10/24

for S_1

q_3q_2 \ q_1q_0	00	01	11	10
00	0	1	X	X
01	0	0	0	X
11	0	1	X	X
10	0	0	0	X

$S_1 = q_0\bar{q}_2\bar{q}_3 + q_0q_2q_3$

for S_2

q_3q_2 \ q_1q_0	00	01	11	10
00	0	0	0	1
01	X	X	X	X
11	X	X	X	0
10	0	0	0	0

$S_2 = \bar{q}_0q_1\bar{q}_3$

for S_3

q_3q_2 \ q_1q_0	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	X	X	X	X
10	0	X	X	X

$S_3 = \bar{q}_0\bar{q}_1q_2$

for S_0

q_3q_2 \ q_1q_0	00	01	11	10
00	1	X	0	0
01	0	0	X	1
11	1	X	0	0
10	0	0	X	1

$S_0 = \bar{q}_1\bar{q}_2\bar{q}_3 + q_1q_2\bar{q}_3 + \bar{q}_1q_2q_3 + q_1\bar{q}_2q_3$

for R_3

q_3q_2 \ q_1q_0	00	01	11	10
00	X	X	X	X
01	0	X	X	X
11	0	0	0	0
10	1	0	0	0

$R_3 = \bar{q}_0\bar{q}_1\bar{q}_2$

for R_2

q_3q_2 \ q_1q_0	00	01	11	10
00	X	X	X	0
01	0	0	0	0
11	0	0	0	1
10	X	X	X	X

$R_2 = \bar{q}_0q_1q_3$

for R_1

q_3q_2 \ q_1q_0	00	01	11	10
00	X	0	0	0
01	X	X	1	0
11	X	0	0	0
10	X	X	1	0

$R_1 = q_0q_2\bar{q}_3 + q_0\bar{q}_2q_3$
$= q_0(q_2 \oplus q_3)$

Projected Ex.
24/10/24

for R_0

q_3q_2 \ q_1q_0	00	01	11	10
00	0	0	1	X
01	X	1	0	0
11	0	0	1	X
10	X	1	0	0

$R_0 = q_1\bar{q}_2\bar{q}_3 + \bar{q}_1 q_2 \bar{q}_3 + q_1 q_2 q_3 + \bar{q}_1 q_2 \bar{q}_3$

$= \bar{q}_1 (q_2 \oplus q_3) + q_1 (\overline{q_2 \oplus q_3})$

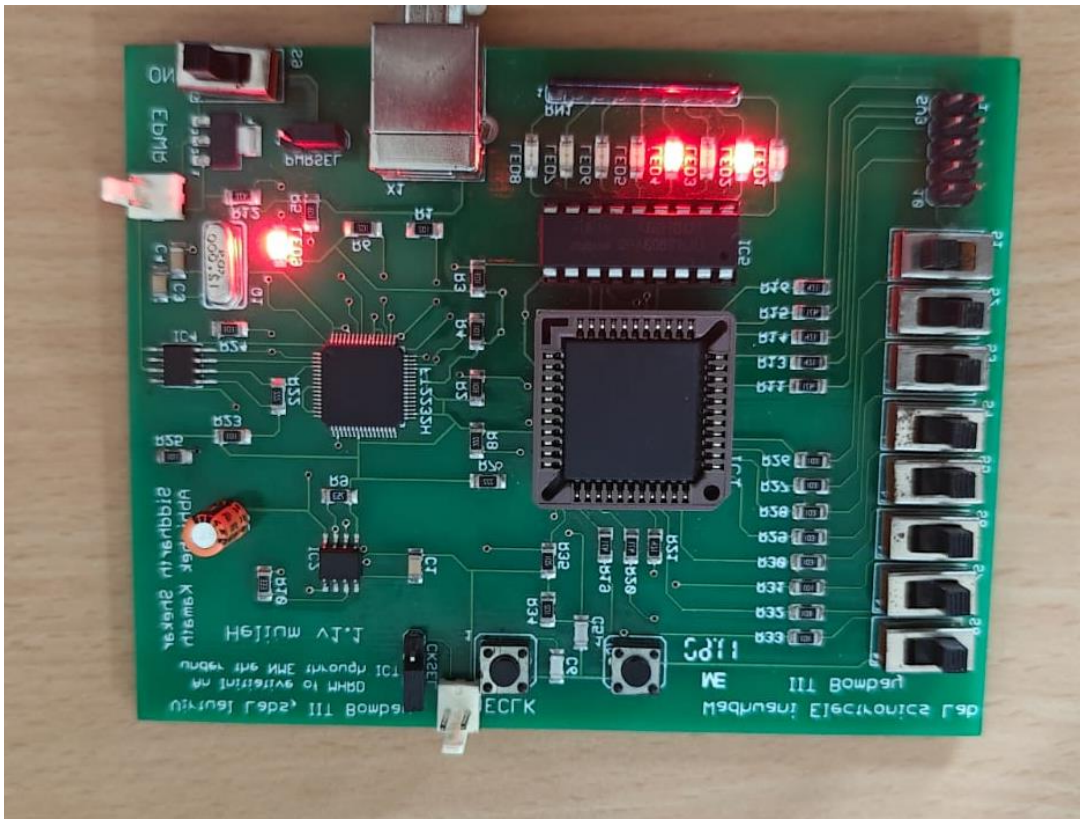
$= q_1 \oplus q_2 \oplus q_3$

Observation Table

Led 4 (27)	Led 3 (26)	Led 2 (25)	Led 1 (Pin 24)
q_3	q_2	q_1	q_0
0	0	0	0
0	0	0	1
0	0	1	1
0	0	1	0
0	1	1	0
0	1	1	1
0	1	0	1
0	1	0	0
1	1	0	0
1	1	0	1
1	1	1	1
1	1	1	0
1	0	1	0
1	0	1	1
1	0	0	1
1	0	0	0

Abhijeet U
22/10/24

CPLD Board:



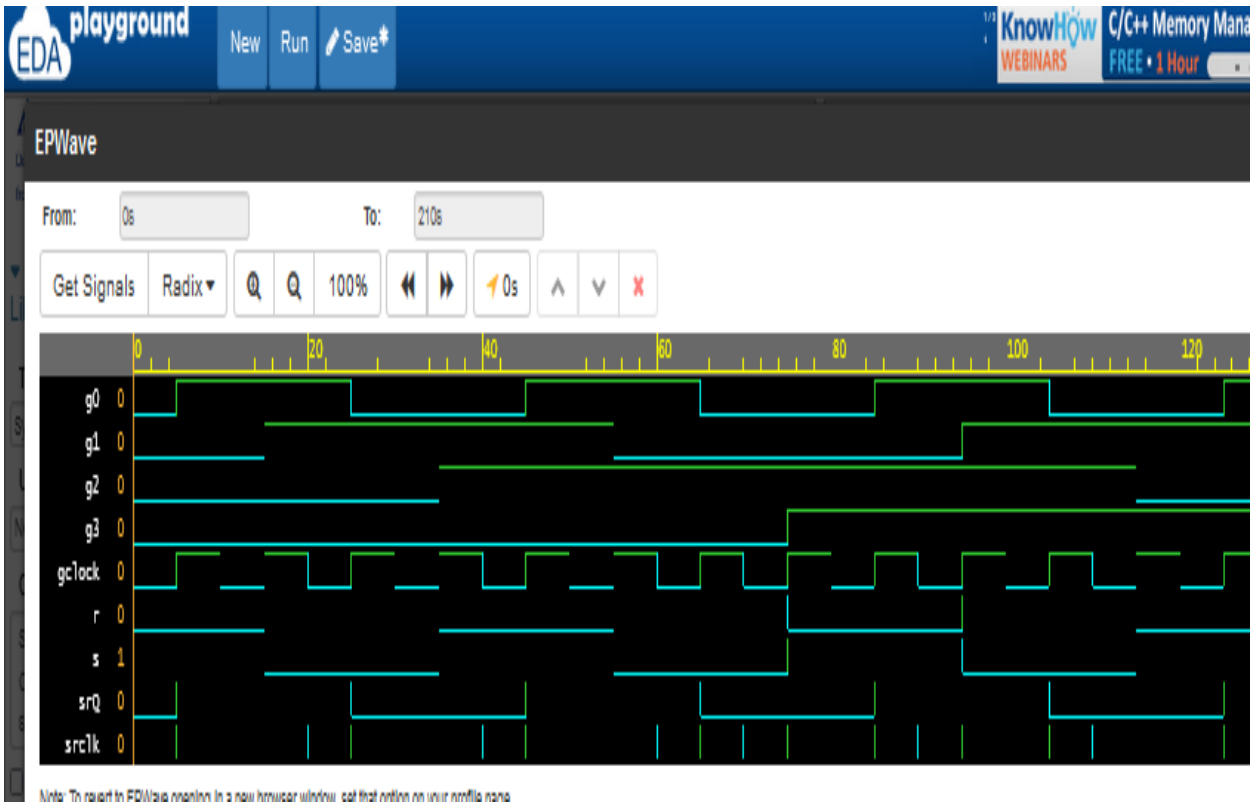
EDA Playground Interface:

```
module tb_gcount;

    // Inputs
    reg gclock;

    // Outputs
    wire g0, g1, g2, g3;

    // Instantiate the gcount module
    gcount uut (
        .gclock(gclock),
        .g0(g0),
        .g1(g1),
        .g2(g2),
        .g3(g3)
    );

    // Clock generation
    initial begin
        gclock = 0;
        forever #5 gclock = ~gclock; // 10 time units clock period
    end

    // Test procedure
    initial begin
        // Open VCD file for waveform generation
        $dumpfile("gcount_waveform.vcd");
        $dumpvars(0, tb_gcount); // Dump all variables in this
module

        // Monitor the outputs
        $monitor("Time: %0t | g0: %b, g1: %b, g2: %b, g3: %b",
$time, g0, g1, g2, g3);

        // Initial state
        #10; // Wait for 10 time units

        // Extend simulation time
        #200; // Additional simulation time for state changes
```

```
module gcount(input gclock, output g0, output g1,output g2, output g3);
sr_flip_flop sr0(~(g1^g2^g3),g1^g2^g3,gclock,g0);
sr_flip_flop sr1((g0)&(~(g2^g3)),g0&(g2^g3),gclock,g1);
sr_flip_flop sr2((~g0)&(g1)&(~g3),(~g0)&(g1)&(g3),gclock,g2);
sr_flip_flop sr3((~g0)&(~g1)&(g2),(~g0)&(~g1)&(~g2),gclock,g3);
endmodule
module sr_flip_flop(
    input s,
    input r,
    input srclk,
    output reg srQ
);
    initial srQ = 0; // Initialize output to 0
    always @(posedge srclk) begin
        if (s && ~r)
            srQ <= 1;
        else if (~s && r)
            srQ <= 0;
    end
endmodule
```


EPWave

Waveform on EDA Playground:



Design Code

```
module gcount(input gclock, output g0, output g1,output g2, output g3);
 sr_flip_flop sr0(~(g1^g2^g3),g1^g2^g3,gclock,g0);
 sr_flip_flop sr1((g0)&(~(g2^g3)),g0&(g2^g3),gclock,g1);
 sr_flip_flop sr2((~g0)&(g1)&(~g3),(~g0)&(g1)&(g3),gclock,g2);
 sr_flip_flop sr3((~g0)&(~g1)&(g2),(~g0)&(~g1)&(~g2),gclock,g3);
endmodule
module sr_flip_flop(
   input s,
   input r,
   input srclk,
   output reg srQ
);
   initial srQ = 0; // Initialize output to 0
   always @(posedge srclk) begin
      if (s && ~r)
         srQ <= 1;
      else if (~s && r)
         srQ <= 0;
   end
endmodule
```

Testbench Code

```
module tb_gcount;

   // Inputs
   reg gclock;

   // Outputs
   wire g0, g1, g2, g3;

   // Instantiate the gcount module
   gcount uut (
      .gclock(gclock),
      .g0(g0),
      .g1(g1),
      .g2(g2),
      .g3(g3)
   );
```

```
    // Clock generation
    initial begin
        gclock = 0;
        forever #5 gclock = ~gclock; // 10 time units clock period
    end

    // Test procedure
    initial begin
        // Open VCD file for waveform generation
        $dumpfile("gcount_waveform.vcd");
        $dumpvars(0, tb_gcount); // Dump all variables in this module

        // Monitor the outputs
        $monitor("Time: %0t | g0: %b, g1: %b, g2: %b, g3: %b", $time, g0, g1, g2, g3);

        // Initial state
        #10; // Wait for 10 time units

        // Extend simulation time
        #200; // Additional simulation time for state changes

        // Finish simulation
        $finish;
    end

endmodule
```

Inference

The Synchronous 4-bit Gray-Code Counter was successfully designed and simulated. The counter transitions through all 16 states with only a single-bit change between consecutive states, as expected. This design is suitable for applications that require minimized bit transitions to reduce switching noise or power consumption.

Sources of Error

- **Incorrect State Table**: Any error in the state table could lead to incorrect counter transitions.
- **Karnaugh Map Errors**: Mistakes in simplifying the Boolean expressions using Karnaugh Maps could result in incorrect SR input values.
- **Coding Errors**: Errors in Verilog code, such as incorrect Flip-Flop instantiation or incorrect logic implementation, may cause the counter not to function as expected.
- **Simulation Limitations**: If the simulation software has limited support for SR Flip-Flops or does not accurately model them, the results may be unreliable.

Precautions

- **Verify State Table**: Double-check the Gray-Code sequence to ensure the state table is correct.
- **Accurate Karnaugh Mapping**: Carefully simplify Boolean expressions using Karnaugh Maps to avoid logic errors.
- **Code Verification**: Thoroughly review and debug the HDL code before simulation.
- **Simulation Settings**: Ensure the simulation environment is correctly configured for edge-triggered behavior, as SR Flip-Flops should not function as latches.