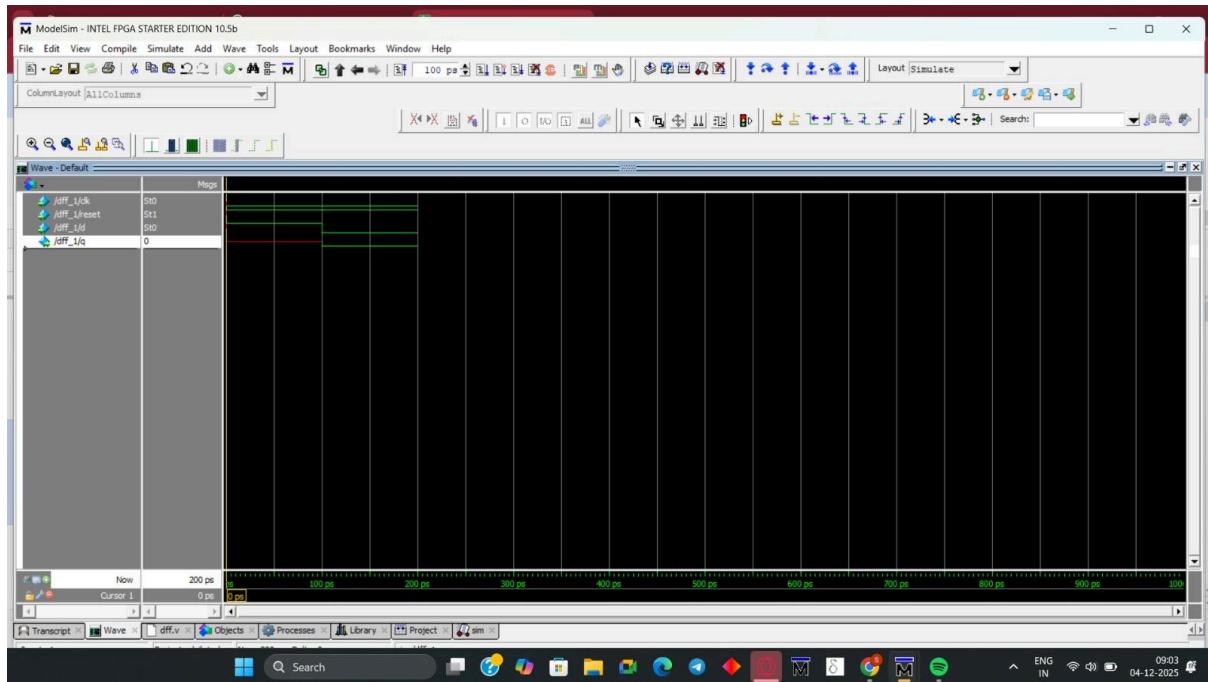


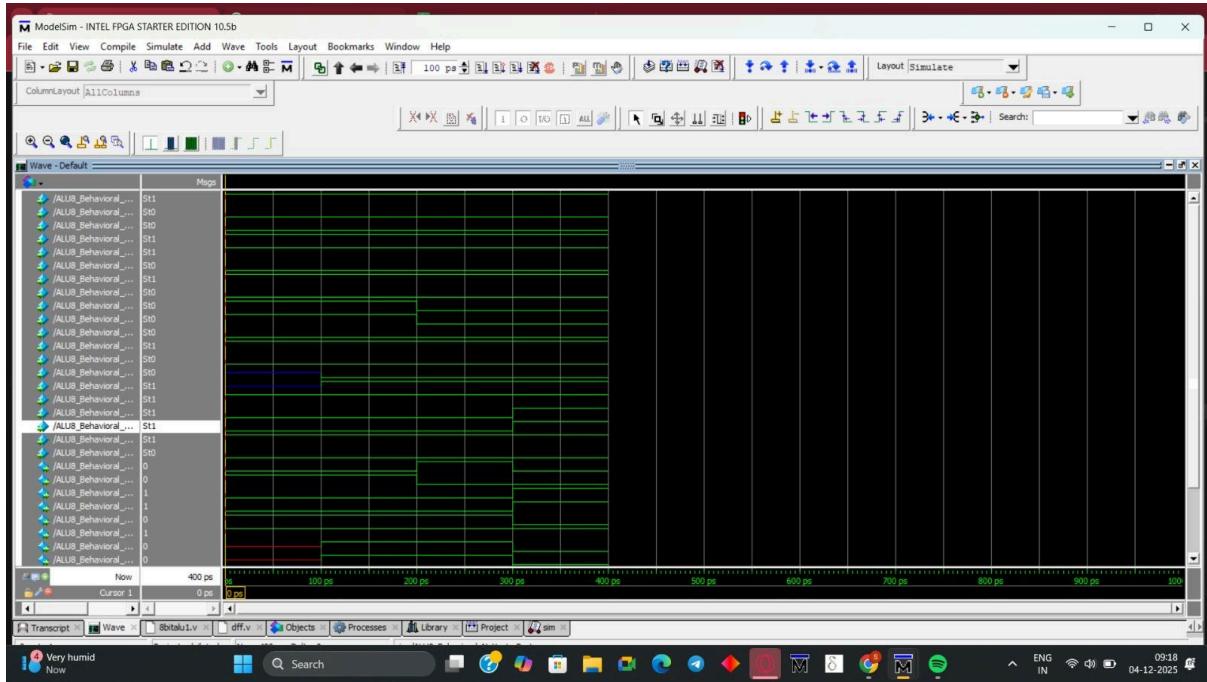
## Dff



The screenshot shows the ModelSim simulation interface. The top window is titled "Wave - Default" and displays waveforms for four signals: dff\_1/clk, dff\_1/reset, dff\_1/d, and dff\_1/q. The dff\_1/clk signal is a square wave at 100 ps intervals. The dff\_1/reset signal is a pulse at 100 ps. The dff\_1/d signal is a constant value of 0. The dff\_1/q signal shows the state changing from 0 to 1 at the rising edge of the clock. The bottom window shows the source code for the DFF module.

```
1 moduledff_1(
2     input clk,
3     input reset,
4     input d,
5     output reg q
6 );
7 begin
8     always @ (posedge clk) begin
9         if (reset)
10             q <= 0;
11         else
12             q <= d;
13     end
14 endmodule
```

8 bit alu



```

ModelSim - INTEL FPGA STARTER EDITION 10.5b
File Edit View Compile Simulate Add Source Tools Layout Bookmarks Window Help
ColumnLayout [AllColumns]
Wave | I O IO [1] All | Layout | Simulate |
Search: | Wave | Objects | Processes | Library | Project | sim |

```

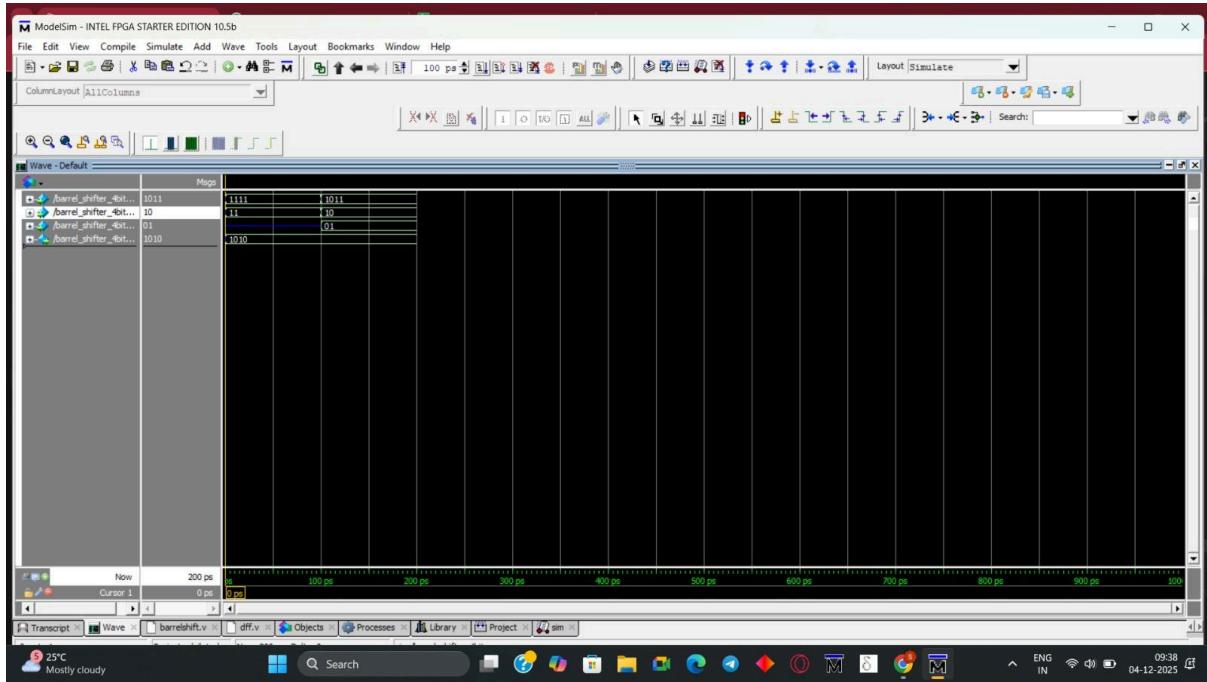
```

C:\intelFPGA\18.1\Bitalu.v Default

1 module ALU8_Behavioral_NoVectorPorts(
2     input A0, A1, A2, A3, A4, A5, A6, A7,
3     input B0, B1, B2, B3, B4, B5, B6, B7,
4     input S0, S1, S2, S3,
5     output reg OUT0, OUT1, OUT2, OUT3, OUT4, OUT5, OUT6, OUT7,
6     output reg CarryOut
7 );
8     req [7:0] A;
9     req [7:0] B;
10    req [7:0] R;
11    req [3:0] Sel;
12    req [8:0] tmp;
13
14    always @(*) begin
15        A = (A7, A6, A5, A4, A3, A2, A1, A0);
16        B = (B7, B6, B5, B4, B3, B2, B1, B0);
17        Sel = (S3, S2, S1, S0);
18        R = 8'b00000000;
19        CarryOut = 0;
20        case (Sel)
21            4'b0000: begin
22                tmp = A + B;
23                R = tmp[7:0];
24                CarryOut = tmp[8];
25            end
26            4'b0001: begin
27                tmp = A - B;
28                R = tmp[7:0];
29                CarryOut = tmp[8];
30            end
31            4'b0010: R = A & B;
32            4'b0011: R = A | B;
33            4'b0100: R = A ^ B;
34            4'b1111: R = 8'b11111111;
35        endcase
36    end
37 endmodule

```

Barrel shift



```

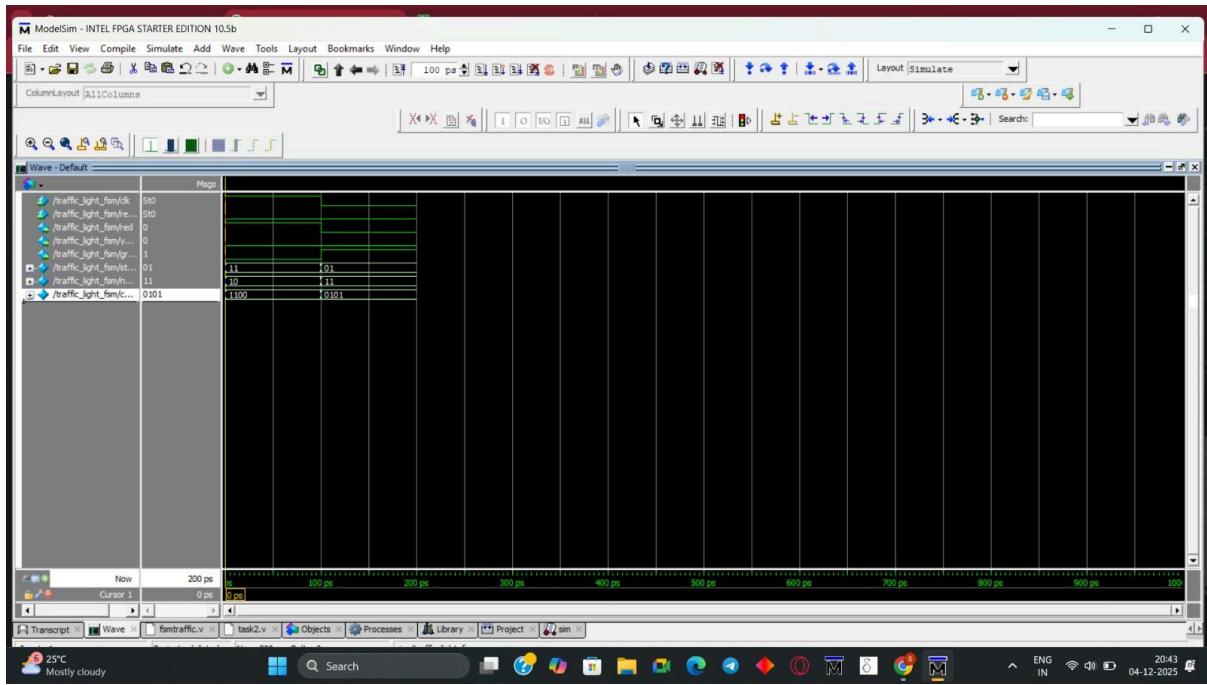
ModelSim - INTEL FPGA STARTER EDITION 10.5b
File Edit View Compile Simulate Add Wave Tools Layout Bookmarks Window Help
ColumnLayout [AllColumns]
[ X X X X X | I O VIO ALL | Processes | Library | Project | sim | Layout | Simulate ]
[ Wave | Transcript | Wave | barrelshift.v | diff.v | Objects | Processes | Library | Project | sim | ]
C:\intelFPGA\18.1\barrelshift.v - Default
Ln# 1 module barrel_shifter_4bit(
2   input [3:0] data,
3   input [1:0] shift,
4   input [1:0] mode,
5   output [3:0] out
6 );
7
8   assign out =
9     (mode == 2'b00) ? (data << shift) :
10    (mode == 2'b01) ? (data >> shift) :
11    (mode == 2'b10) ? ((data << shift) | (data >> (4-shift))) :
12    (mode == 2'b11) ? ((data >> shift) | (data << (4-shift))) :
13    4'b0000;
14
15 endmodule
16

```

Traffic light fsm

The screenshot shows the ModelSim - INTEL FPGA STARTER EDITION 10.5b software interface. The top menu bar includes File, Edit, View, Compile, Simulate, Add Source, Tools, Layout, Bookmarks, Window, and Help. The toolbar contains various icons for file operations like Open, Save, Import, Export, and simulation controls. The main window displays an HDL code editor for a traffic light system named "fmltraffic.v". The code defines a module with inputs clk, reset, and outputs req red, req yellow, and req green. It includes parameters for traffic light times (RED = 2'b00, GREEN = 2'b01, YELLOW = 2'b10) and logic for state transitions and a counter. Below the code editor is a transcript window showing simulation logs. The bottom taskbar includes icons for Transcript, Wave, fmltraffic.v, task2.v, Objects, Processes, Library, Project, and a search bar. The system tray shows the date (04-12-2025), time (2042), and various system icons.

```
in# module traffic_light_fsm (
    input clk,
    input reset,
    output reg red,
    output reg yellow,
    output reg green
);
parameter RED = 2'b00;
parameter GREEN = 2'b01;
parameter YELLOW = 2'b10;
reg [1:0] state, next_state;
parameter RED_TIME = 5;
parameter GREEN_TIME = 5;
parameter YELLOW_TIME = 3;
reg [3:0] counter;
always @(posedge clk or posedge reset) begin
    if (reset) begin
        state <= RED;
        counter <= RED;
    end
    else begin
        state <= next_state;
    end
end
always @(posedge clk or posedge reset) begin
    if (reset)
        counter <= 0;
    else
        counter <= counter + 1;
end
```



## Verilog sequence detector

