UE22AM251A : SOC DESIGN AND COMPUTER ARCHITECTURE

# Project Report

Nagathejas M S

PES1UG22AM098

Bharateesh L V N

PES1UG22AM088

Malleshappa D Patil

PES1UG22AM090

Mohith B

PES1UG22AM096

*Problem statement* -

Design& implement a four–bit Register with Four D flip–flops and Four 4 × 1 Multiplexers with mode selection inputs s1 and s0. The Register operates according to the values of select lines, if s1s0=00, no change, s1s0=01 complement output, s1s0=10, shift the bits to the right, & s1s0=11, shift the bits to left..

*Iverilog code –*

new.v -

```
module invert(input wire i, output wire o1);
  assign o1 = ~i;
endmodule


module DFlipFlop (
  input wire D,
  input wire CLK,
  input wire RESET,
  output reg Q
);

  always @(posedge CLK or posedge RESET) begin
    if (RESET)
      Q <= 1'b0;
    else
      Q <= D;
  end

endmodule
```

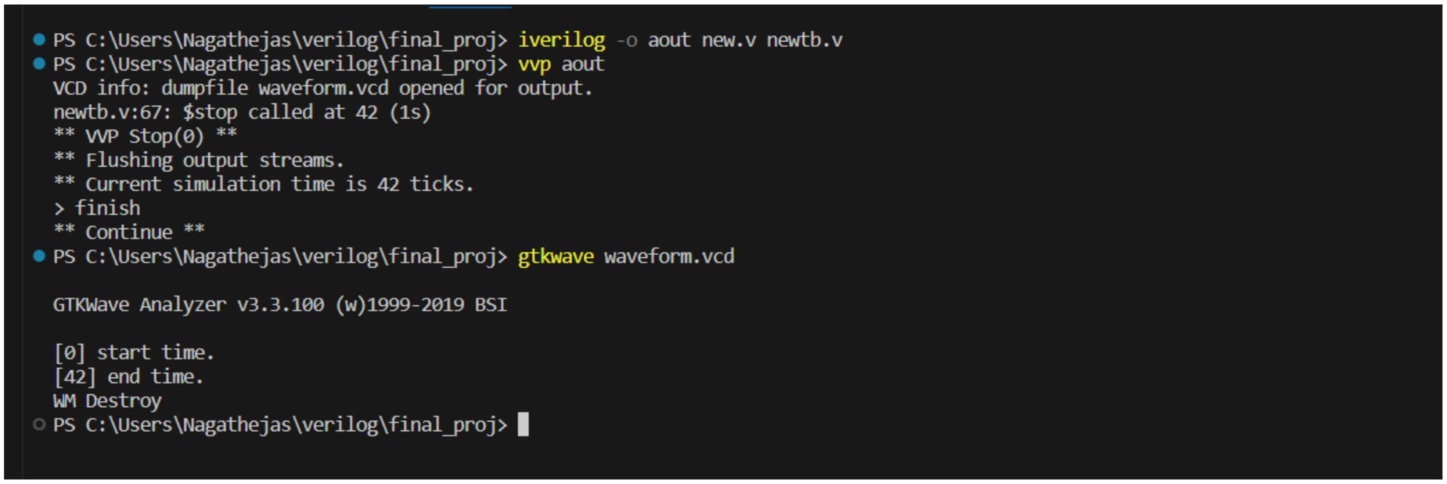
```
module MUX4to1 (
  input wire I0,
  input wire I1,
  input wire I2,
  input wire I3,
  input wire [1:0] S,
  output reg O
);

  always @*
    case (S)
      2'b00: O = I0;
      2'b01: O = I1;
      2'b10: O = I2;
      2'b11: O = I3;
      default: O = 1'b0; // Default value, should not occur
    endcase

endmodule

module register (
  input wire[3:0] D,
  input wire CLK,
  input wire RESET,
  input wire [1:0] S,
  output reg [3:0] Y
);

  wire [3:0] middle;
  wire [3:0] link;

  invert i0(middle[0], link[0]);
  invert i1(middle[1], link[1]);
  invert i2(middle[2], link[2]);
  invert i3(middle[3], link[3]);

  DFlipFlop d0(D[0], CLK, RESET, middle[0]);
  DFlipFlop d1(D[1], CLK, RESET, middle[1]);
  DFlipFlop d2(D[2], CLK, RESET, middle[2]);
  DFlipFlop d3(D[3], CLK, RESET, middle[3]);

  wire [3:0] mux_out;

  MUX4to1 mux0(middle[0], link[0], middle[1], 1'b0, S, mux_out[0]);
  MUX4to1 mux1(middle[1], link[1],  middle[2],middle[0], S, mux_out[1]);
  MUX4to1 mux2(middle[2], link[2],  middle[3],middle[1], S, mux_out[2]);
  MUX4to1 mux3(middle[3], link[3],  1'b0,middle[2], S, mux_out[3]);

  always @*
    Y = mux_out;

endmodule
```

newtb.v -

```
module tb_register;

  reg [3:0] D;
  reg CLK;
  reg RESET;
  reg [1:0] S;
  wire [3:0] Y;

  // Instantiate the register module
  register uut (
    .D(D),
    .CLK(CLK),
    .RESET(RESET),
    .S(S),
    .Y(Y)
  );

  // Clock generation
  initial begin
```

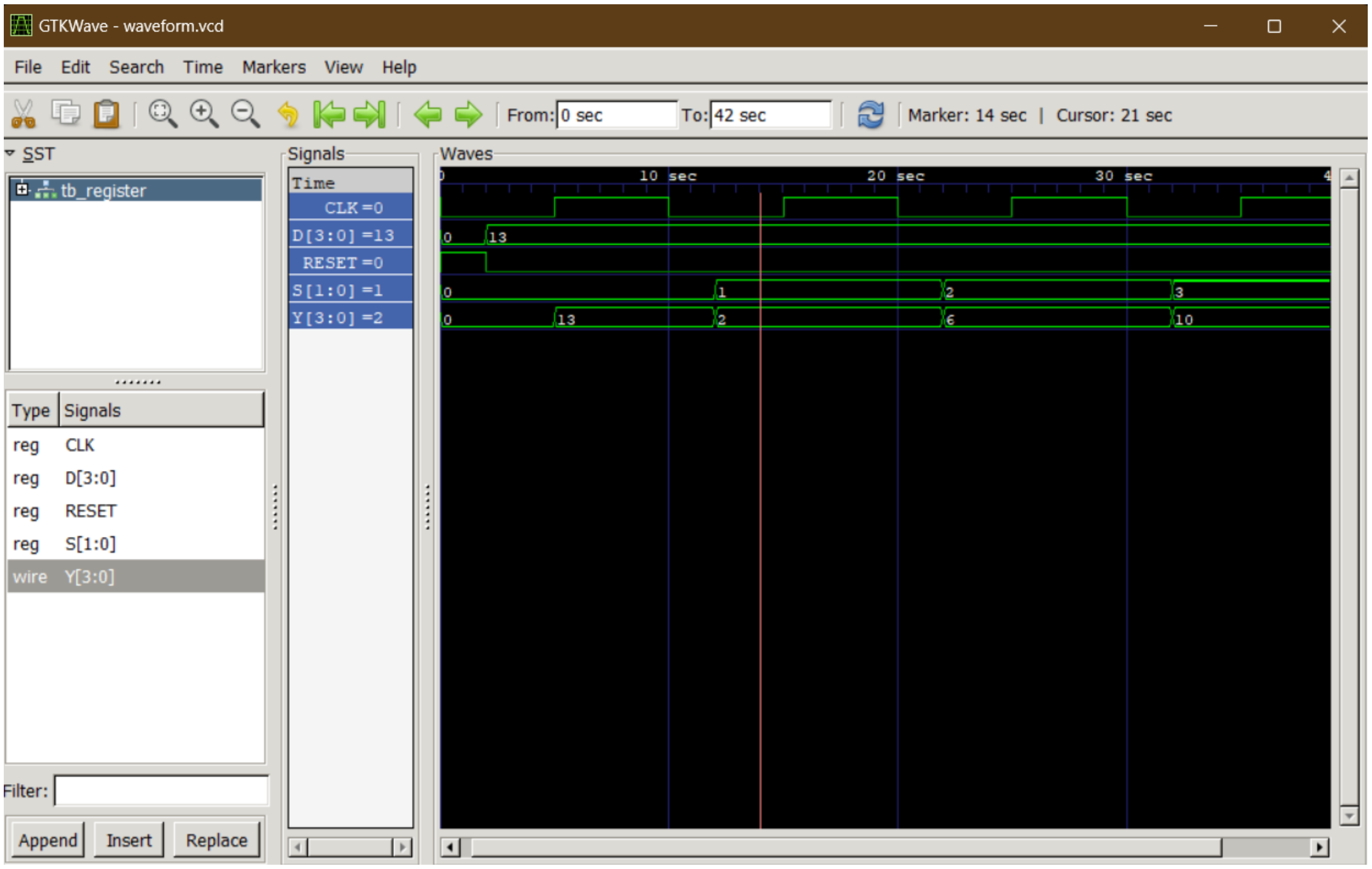
```
    CLK = 0;
    forever #5 CLK = ~CLK;
  end

  // Dump VCD file for GTKWave
  initial begin
    $dumpfile("waveform.vcd");
    $dumpvars(0, tb_register);
  end

  // Test cases
  initial begin

        // Initialize signals
    D = 4'b0000;
    S = 2'b00;
    RESET = 1;

    // Apply reset
    #2 RESET = 0;

    // Case 1: s1s0=00, no change
    D = 4'b1101;
    S = 2'b00;
    RESET = 0;
    #10;

    // Case 2: s1s0=01, complement output
    D = 4'b1101;
    S = 2'b01;
    RESET = 0;
    #10;

    // Case 3: s1s0=10, shift the bits to the right
    D = 4'b1101;
    S = 2'b10;
    RESET = 0;
    #10;
    // Case 4: s1s0=11, shift the bits to the left
    D = 4'b1101;
    S = 2'b11;
    RESET = 0;
    #10;

    // Add more test cases as needed

    $stop; // Stop simulation
  end

endmodule
```

*commands* –



```
● PS C:\Users\Nagathejas\verilog\final_proj> iverilog -o aout new.v newtb.v
● PS C:\Users\Nagathejas\verilog\final_proj> vvp aout
  VCD info: dumpfile waveform.vcd opened for output.
  newtb.v:67: $stop called at 42 (1s)
  ** VVP Stop(0) **
  ** Flushing output streams.
  ** Current simulation time is 42 ticks.
  > finish
  ** Continue **
● PS C:\Users\Nagathejas\verilog\final_proj> gtkwave waveform.vcd

  GTKWave Analyzer v3.3.100 (w)1999-2019 BSI

  [0] start time.
  [42] end time.
  WM Destroy
○ PS C:\Users\Nagathejas\verilog\final_proj> █
```

*gtkwave output -*


GTKWave - waveform.vcd
File Edit Search Time Markers View Help
From: 0 sec
To: 42 sec
Marker: 14 sec | Cursor: 21 sec
SST
tb_register
Signals
Time
CLK =0
D[3:0] =13
RESET =0
S[1:0] =1
Y[3:0] =2
Waves
10 sec
20 sec
30 sec
Type Signals
reg CLK
reg D[3:0]
reg RESET
reg S[1:0]
wire Y[3:0]
Filter:
Append
Insert
Replace