

ECS 409/609

Assignment 3

Sequential Based Assignment

Solutions

Kunwar Arpit Singh

Submission Deadline: September 14c, 2025

For code repository of this assignment:  **Verilog Assignments**

1. Problem 1

Problem statement

Implement a verilog module of an SR latch with asynchronous enable and reset.

Verilog source (Sequential)

```
1 module sr_latch(input S, input R, input en, input reset, output reg Q, output
  reg Q_n);
2   always @(S, R, en, reset) begin
3       if (reset) begin
4           Q <= 1'b0;
5           Q_n <= 1'b1;
6       end
7       else if (en) begin
8           case ({S, R})
9               2'b00: begin
10                  Q <= Q;
11                  Q_n <= Q_n;
12              end
13              2'b01: begin
14                  Q <= 1'b0;
15                  Q_n <= 1'b1;
16              end
17              2'b10: begin
18                  Q <= 1'b1;
19                  Q_n <= 1'b0;
20              end
21              2'b11: begin
22                  Q <= 1'bx;
23                  Q_n <= 1'bx;
24              end
25          endcase
26      end
```

```

27     end
28 endmodule

```

Testbench

```

1 module sr_latch_tb;
2     reg S;
3     reg R;
4     reg en;
5     reg reset;
6     wire Q;
7     wire Q_n;
8     reg [8*24-1:0] name;
9     sr_latch dut (.S(S), .R(R), .en(en), .reset(reset), .Q(Q), .Q_n(Q_n));
10
11     initial begin
12         name = "_Kunwar_Arpit_Singh_22185";
13         $display("Kunwar_Arpit_Singh");
14         $dumpfile("assign3_problem1_sr_latch.vcd");
15         $dumpvars(1, sr_latch_tb);
16         $display("Kunwar_Arpit_Singh");
17
18         $display("|_reset_|_en_|_S_|_R_|_Q_|_Q_n_|");
19         $display("-----");
20
21         reset = 1'b1; S = 1'b0; R = 1'b0; en = 1'b0; #10;
22         $display("|_%%b%%b_|_%%b_|_%%b_|_%%b_|_%%b_|_%%b_|",reset, en, S, R, Q,
23             Q_n);
24         reset = 1'b0; S = 1'b0; R = 1'b0; en = 1'b0; #10;
25         $display("|_%%b%%b_|_%%b_|_%%b_|_%%b_|_%%b_|_%%b_|",reset, en, S, R, Q,
26             Q_n);
27         reset = 1'b0; S = 1'b0; R = 1'b0; en = 1'b1; #10;
28         $display("|_%%b%%b_|_%%b_|_%%b_|_%%b_|_%%b_|_%%b_|",reset, en, S, R, Q,
29             Q_n);
30         reset = 1'b0; S = 1'b0; R = 1'b1; en = 1'b1; #10;
31         $display("|_%%b%%b_|_%%b_|_%%b_|_%%b_|_%%b_|_%%b_|",reset, en, S, R, Q,
32             Q_n);
33         reset = 1'b0; S = 1'b1; R = 1'b0; en = 1'b1; #10;
34         $display("|_%%b%%b_|_%%b_|_%%b_|_%%b_|_%%b_|_%%b_|",reset, en, S, R, Q,
35             Q_n);
36         reset = 1'b0; S = 1'b0; R = 1'b0; en = 1'b1; #10;
37         $display("|_%%b%%b_|_%%b_|_%%b_|_%%b_|_%%b_|_%%b_|",reset, en, S, R, Q,
38             Q_n);
39         reset = 1'b0; S = 1'b1; R = 1'b1; en = 1'b1; #10;
40         $display("|_%%b%%b_|_%%b_|_%%b_|_%%b_|_%%b_|_%%b_|",reset, en, S, R, Q,
41             Q_n);
42         $finish;

```

```

38     end
39 endmodule

```

Output in Terminal

```

1 Kunwar Arpit Singh
2 VCD info: dumpfile assign3_problem1_sr_latch.vcd opened for output.
3 Kunwar Arpit Singh
4 | reset | en | S | R | Q | Q_n |
5 -----
6 | 1 | 0 | 0 | 0 | 0 | 1 |
7 | 0 | 0 | 0 | 0 | 0 | 1 |
8 | 0 | 1 | 0 | 0 | 0 | 1 |
9 | 0 | 1 | 0 | 1 | 0 | 1 |
10 | 0 | 1 | 0 | 0 | 0 | 1 |
11 | 0 | 1 | 1 | 0 | 1 | 0 |
12 | 0 | 1 | 0 | 0 | 1 | 0 |
13 | 0 | 1 | 1 | 1 | x | x |

```

Output Screenshots

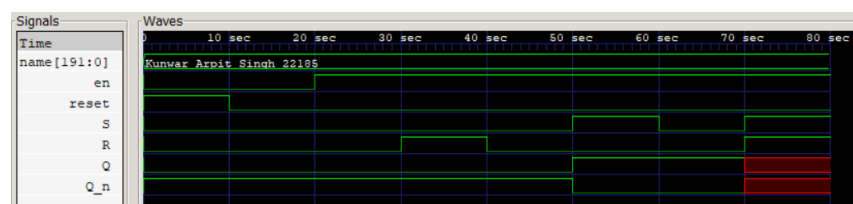
Terminal output (SR Latch)

```

Kunwar Arpit Singh
VCD info: dumpfile assign3_problem1_sr_latch.vcd opened for output.
Kunwar Arpit Singh
| reset | en | S | R | Q | Q_n |
-----
| 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | x | x |

```

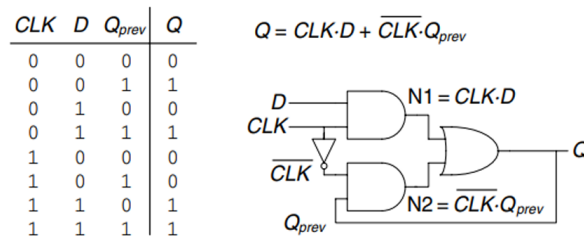
GTKWave waveform (SR Latch)



2. Problem 2

Problem statement

Implement the Verilog module of an improved version of D latch as shown in figure 1. Specify delays of 1 ns to each gate. With your simulator, show that the latch operates correctly.



Verilog source (Sequential)

```
1 module d_latch_improved(input D, input CLK, output reg Q);
2     wire CLK_n;
3     wire D_and_CLK;
4     wire Qprev_and_CLK_n;
5
6     not #1 U1(CLK_n, CLK);
7     and #1 U2(D_and_CLK, D, CLK);
8     and #1 U3(Qprev_and_CLK_n, Q, CLK_n);
9
10    always @(*) begin
11        #1 Q = D_and_CLK | Qprev_and_CLK_n;
12    end
13    initial Q = 0;
14 endmodule
```

Testbench

```
1 module d_latch_improved_tb;
2     reg D, CLK;
3     wire Q;
4     reg [8*24-1:0] name;
5
6     d_latch_improved dut (.D(D), .CLK(CLK), .Q(Q) );
7
8     initial begin
9         name = "KunwarArpitSingh22185";
10        $dumpfile("assign3_problem2_d_latch.vcd");
11        $dumpvars(1, d_latch_improved_tb);
12        $display("KunwarArpitSingh");
13        CLK = 1; D = 0; #10;
14        CLK = 0; #10;
```

```

15
16     CLK = 1; D = 1; #10;
17     CLK = 0; #10;
18
19     CLK = 1; D = 0; #10;
20     CLK = 0; D = 1; #10;
21
22     CLK = 1; D = 1; #10;
23     CLK = 0; #10;
24
25     CLK = 1; D = 0; #10;
26
27     CLK = 0; D = 1; #10;
28     CLK = 1; D = 0; #10;
29
30     CLK = 0; D = 0; #10;
31     CLK = 1; D = 1; #10;
32
33     CLK = 0; D = 1; #10;
34     CLK = 1; D = 1; #10;
35     $finish;
36 end
37
38 initial begin
39     $display("|_CLK_|_D_|_Q_|");
40     $monitor("|_%b_|_%b_|_%b_|", CLK, D, Q);
41 end
42 endmodule

```

Output in Terminal

```

1 Kunwar Arpit Singh
2 VCD info: dumpfile assign2_problem6_wallace_tree_multiplier_4_bit.vcd opened
  for output.
3 Kunwar Arpit Singh
4 | A(B) | A(D) | B(B) | B(D) | Product(B) | Product(D) |
5 | 0001 | 1 | 0001 | 1 | 00000001 | 1 |
6 | 0101 | 5 | 1010 | 10 | 00110010 | 50 |
7 | 1111 | 15 | 1111 | 15 | 11100001 | 225 |
8 | 1001 | 9 | 0111 | 7 | 00111111 | 63 |
9 | 1100 | 12 | 0011 | 3 | 00100100 | 36 |
10 | 0100 | 4 | 1111 | 15 | 00111100 | 60 |
11 | 0000 | 0 | 1111 | 15 | 00000000 | 0 |

```

Output Screenshots

Terminal output (D Latch)

```
VCD info: dumpfile assign3_problem2_d_latch.vcd opened for output.
Kunwar Arpit Singh
| CLK | D | Q |
| 1 | 0 | 0 |
| 0 | 0 | 0 |
| 1 | 1 | 0 |
| 1 | 1 | 1 |
| 0 | 1 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 1 | 0 |
| 1 | 1 | 1 |
| 0 | 1 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 0 | 0 | 0 |
| 1 | 1 | 0 |
| 1 | 1 | 1 |
| 0 | 1 | 1 |
| 0 | 1 | 0 |
| 1 | 1 | 0 |
| 1 | 1 | 1 |
```

GTKWave waveform (D Latch)



3. Problem 3

Problem statement

Design a verilog module of counter that counts from 7 to 0 (e.g., 7, 6, 5, 4, 3, 2, 1, 0).

Verilog source (Sequential)

```
1 module binary_counter(input CLK, input RESET, output reg [2:0] COUNT);
2     always @(posedge CLK, posedge RESET) begin
3         if (RESET) begin
4             COUNT <= 3'b111;
5         end
6         else begin
7             if (COUNT == 3'b000) begin
8                 COUNT <= 3'b111;
9             end else begin
10                COUNT <= COUNT - 1;
11            end
12        end
13    end
14 endmodule
```

Testbench

```
1 module binary_counter_tb;
2     reg CLK;
3     reg RESET;
4     wire [2:0] COUNT;
5     reg [8*24-1:0] name;
6
7     binary_counter dut (.CLK(CLK), .RESET(RESET), .COUNT(COUNT));
8
9     initial begin
10         CLK = 0;
11         forever #5 CLK = ~CLK;
12     end
13
14     initial begin
15         name = "_Kunwar_Arpit_Singh_22185";
16
17         $dumpfile("assign3_problem3_counter.vcd");
18         $dumpvars(1, binary_counter_tb);
19
20         $display("Kunwar_Arpit_Singh");
21         $display("|_Reset_|_Clock_|_Count_|");
22         $display("-----");
23
24         $monitor("|_%b_|_%b_|_%b_|(%d)|", RESET, CLK, COUNT, COUNT);
25         RESET = 1; #15;
```

```

26         RESET = 0; #60;
27         $finish;
28     end
29 endmodule

```

Output in Terminal

```

1 VCD info: dumpfile assign3_problem3_counter.vcd opened for output.
2 Kunwar Arpit Singh
3 | Reset | Clock | Count |
4 -----
5 | 1 | 0 | 111 (7) |
6 | 1 | 1 | 111 (7) |
7 | 1 | 0 | 111 (7) |
8 | 0 | 1 | 110 (6) |
9 | 0 | 0 | 110 (6) |
10 | 0 | 1 | 101 (5) |
11 | 0 | 0 | 101 (5) |
12 | 0 | 1 | 100 (4) |
13 | 0 | 0 | 100 (4) |
14 | 0 | 1 | 011 (3) |
15 | 0 | 0 | 011 (3) |
16 | 0 | 1 | 010 (2) |
17 | 0 | 0 | 010 (2) |
18 | 0 | 1 | 001 (1) |
19 | 0 | 0 | 001 (1) |
20 | 0 | 1 | 000 (0) |

```

Output Screenshots

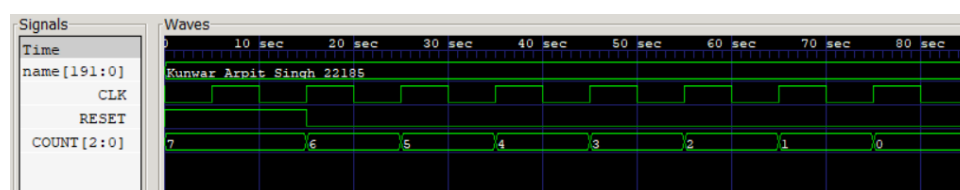
Terminal output (Counter)

```

VCD info: dumpfile assign3_problem3_counter.vcd opened for output.
Kunwar Arpit Singh
| Reset | Clock | Count |
-----
| 1 | 0 | 111 (7) |
| 1 | 1 | 111 (7) |
| 1 | 0 | 111 (7) |
| 0 | 1 | 110 (6) |
| 0 | 0 | 110 (6) |
| 0 | 1 | 101 (5) |
| 0 | 0 | 101 (5) |
| 0 | 1 | 100 (4) |
| 0 | 0 | 100 (4) |
| 0 | 1 | 011 (3) |
| 0 | 0 | 011 (3) |
| 0 | 1 | 010 (2) |
| 0 | 0 | 010 (2) |
| 0 | 1 | 001 (1) |
| 0 | 0 | 001 (1) |
| 0 | 1 | 000 (0) |

```

GTKWave waveform (Counter)



4. Problem 4

Problem statement

Implement a verilog module of UP/DOWN modulo 8 Gray Code Counter as shown in table 2, adding an Input UP. If UP = 1, the counter advances sequentially to the next number. Otherwise, UP = 0, the counter stays with the old value.

Verilog source (Sequential)

```
1 module gray_counter(input CLK, input RESET, input UP, output reg [2:0] GRAYout
2 );
3     always @(posedge CLK or posedge RESET) begin
4         if (RESET) begin
5             GRAYout <= 3'b000;
6         end else begin
7             if (UP) begin
8                 case (GRAYout)
9                     3'b000: GRAYout <= 3'b001;
10                    3'b001: GRAYout <= 3'b011;
11                    3'b011: GRAYout <= 3'b010;
12                    3'b010: GRAYout <= 3'b110;
13                    3'b110: GRAYout <= 3'b111;
14                    3'b111: GRAYout <= 3'b101;
15                    3'b101: GRAYout <= 3'b100;
16                    3'b100: GRAYout <= 3'b000;
17                    default: GRAYout <= 3'b000;
18                endcase
19            end
20        end
21    end
22 endmodule
```

Testbench

```
1 module gray_counter_tb;
2     reg CLK;
3     reg RESET;
4     reg UP;
5     wire [2:0] GRAYout;
6
7     gray_counter dut (.CLK(CLK), .RESET(RESET), .UP(UP), .GRAYout(GRAYout));
8
9     initial begin
10         CLK = 0;
11         forever #5 CLK = ~CLK;
12     end
13
```

```

14  initial begin
15      $dumpfile("assign3_problem4_gray_code_counter.vcd");
16      $dumpvars(1, gray_counter_tb);
17
18      $display("Kunwar_Arpit_Singh");
19      $display("|_Reset_|_UP_|_Gray_Output_|");
20      $display("-----");
21      $monitor("|_%%b_|_%%b_|_%%b_(%d)|", RESET, UP, GRAYout, GRAYout
22              );
23
24      RESET = 1; UP = 0; #20;
25      RESET = 0; #5;
26
27      UP = 1; #70;
28
29      UP = 0; #40;
30
31      UP = 1; #30;
32      $finish;
33  end
endmodule

```

Output in Terminal

```

1  VCD info: dumpfile assign3_problem4_gray_code_counter.vcd opened for output.
2  Kunwar Arpit Singh
3  | Reset | UP | Gray Output |
4  -----
5  | 1 | 0 | 000 (0) |
6  | 0 | 0 | 000 (0) |
7  | 0 | 1 | 001 (1) |
8  | 0 | 1 | 011 (3) |
9  | 0 | 1 | 010 (2) |
10 | 0 | 1 | 110 (6) |
11 | 0 | 1 | 111 (7) |
12 | 0 | 1 | 101 (5) |
13 | 0 | 1 | 100 (4) |
14 | 0 | 0 | 100 (4) |
15 | 0 | 1 | 000 (0) |
16 | 0 | 1 | 001 (1) |
17 | 0 | 1 | 011 (3) |
18 | 0 | 1 | 010 (2) |

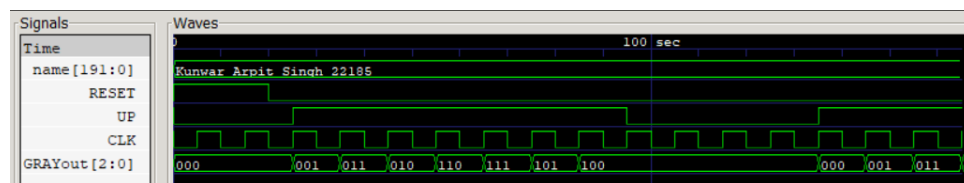
```

Output Screenshots

Terminal output (Gray Code Counter)

```
VCD info: dumpfile assign3_problem4_gray_code_counter.vcd opened for output.
Kunwar Arpit Singh
| Reset | UP | Gray Output |
-----
| 1 | 0 | 000 (0) |
| 0 | 0 | 000 (0) |
| 0 | 1 | 001 (1) |
| 0 | 1 | 011 (3) |
| 0 | 1 | 010 (2) |
| 0 | 1 | 110 (6) |
| 0 | 1 | 111 (7) |
| 0 | 1 | 101 (5) |
| 0 | 1 | 100 (4) |
| 0 | 0 | 100 (4) |
| 0 | 1 | 000 (0) |
| 0 | 1 | 001 (1) |
| 0 | 1 | 011 (3) |
| 0 | 1 | 010 (2) |
```

GTKWave waveform (Gray Code Counter)



5. Problem 5

Problem statement

Implement a verilog module of N-bit bidirectional shift registers using D ip op. (Hint: Use Parameterized Module to implement N).

Number	Gray code		
0	0	0	0
1	0	0	1
2	0	1	1
3	0	1	0
4	1	1	0
5	1	1	1
6	1	0	1
7	1	0	0

Verilog source (Sequential)

```
1 module shift_register #(parameter N = 8)(input clk, reset, input shift_en,
2   load_en, input direction, input serial_in, input [N-1:0] parallel_in,
3   output reg [N-1:0] parallel_out);
4
5   always @(posedge clk, posedge reset) begin
6       if (reset) begin
7           parallel_out <= {N{1'b0}};
8       end else if (load_en) begin
9           parallel_out <= parallel_in;
10      end else if (shift_en) begin
11          if (direction) begin
12              parallel_out <= {serial_in, parallel_out[N-1:1]};
13          end else begin
14              parallel_out <= {parallel_out[N-2:0], serial_in};
15          end
16      end
17  end
```

```
17 endmodule
```

Testbench

```
1 module shift_register_tb;
2     localparam N = 8;
3     reg clk;
4     reg reset;
5     reg shift_en, load_en, direction, serial_in;
6     reg [N-1:0] parallel_in;
7     wire [N-1:0] parallel_out;
8     reg [8*24-1:0] name;
9     shift_register #(N(N)) dut (.clk(clk), .reset(reset), .shift_en(shift_en),
10     .load_en(load_en), .direction(direction), .serial_in(serial_in), .
11     parallel_in(parallel_in), .parallel_out(parallel_out)
12 );
13
14 initial begin
15     clk = 0;
16     forever #5 clk = ~clk;
17 end
18
19 initial begin
20     name = "KunwarArpitSingh22185";
21     $dumpfile("assign3_problem5_bidirectional_shift_register.vcd");
22     $dumpvars(1, shift_register_tb);
23     $display("---N-BitShiftRegisterSimulation(N=%0d)---", N);
24     $display("KunwarArpitSingh");
25     $display("|Reset|Load_en|Shift_en|Direction|Serial_in|Parallel_out|");
26     reset = 1; #20; reset = 0;
27     $monitor("%b|b|b|b|b|b|b", reset, load_en, shift_en,
28     direction, serial_in, parallel_out);
29
30     load_en = 1; parallel_in = 8'b10110101;
31     #10;
32     load_en = 0;
33     #10;
34
35     shift_en = 1; direction = 1;
36     serial_in = 1;
37     #10;
38     serial_in = 0;
39     #10;
40     serial_in = 1;
41     #10;
42
43     direction = 0;
44     serial_in = 0;
45     #10;
```

```

43     serial_in = 1;
44     #10;
45     serial_in = 0;
46     #10;
47     shift_en = 0;
48
49     $finish;
50 end
51 endmodule

```

Output in Terminal

```

1 VCD info: dumpfile assign3_problem5_bidirectional_shift_register.vcd opened
  for output.
2 --- N-Bit Shift Register Simulation (N=8) ---
3 Kunwar Arpit Singh
4 | Reset | Load_en | Shift_en | Direction | Serial_in | Parallel_out |
5 0 | 1 | x | x | x | 00000000
6 0 | 1 | x | x | x | 10110101
7 0 | 0 | x | x | x | 10110101
8 0 | 0 | 1 | 1 | 1 | 10110101
9 0 | 0 | 1 | 1 | 1 | 11011010
10 0 | 0 | 1 | 1 | 0 | 11011010
11 0 | 0 | 1 | 1 | 0 | 01101101
12 0 | 0 | 1 | 1 | 1 | 01101101
13 0 | 0 | 1 | 1 | 1 | 10110110
14 0 | 0 | 1 | 0 | 0 | 10110110
15 0 | 0 | 1 | 0 | 0 | 01101100
16 0 | 0 | 1 | 0 | 1 | 01101100
17 0 | 0 | 1 | 0 | 1 | 11011001
18 0 | 0 | 1 | 0 | 0 | 11011001
19 0 | 0 | 1 | 0 | 0 | 10110010
20 0 | 0 | 0 | 0 | 0 | 10110010

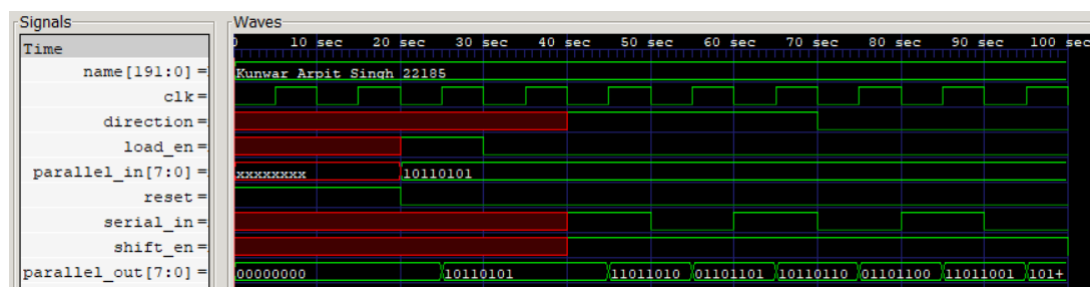
```

Output Screenshots

Terminal output (Bidirectional Shift Register)

```
VCD info: dumpfile assign3_problem5_bidirectional_shift_register.vcd opened for output.
--- N-Bit Shift Register Simulation (N=8) ---
Kunwar Arpit Singh
| Reset | Load_en | Shift_en | Direction | Serial_in | Parallel_out |
0 | 1 | x | x | x | 00000000
0 | 1 | x | x | x | 10110101
0 | 0 | x | x | x | 10110101
0 | 0 | 1 | 1 | 1 | 10110101
0 | 0 | 1 | 1 | 1 | 11011010
0 | 0 | 1 | 1 | 0 | 11011010
0 | 0 | 1 | 1 | 0 | 01101101
0 | 0 | 1 | 1 | 1 | 01101101
0 | 0 | 1 | 1 | 1 | 10110110
0 | 0 | 1 | 0 | 0 | 10110110
0 | 0 | 1 | 0 | 0 | 01101100
0 | 0 | 1 | 0 | 1 | 01101100
0 | 0 | 1 | 0 | 1 | 11011001
0 | 0 | 1 | 0 | 0 | 11011001
0 | 0 | 1 | 0 | 0 | 10110010
0 | 0 | 0 | 0 | 0 | 10110010
```

GTKWave waveform (Bidirectional Shift Register)



6. Problem 6

Problem statement

Implement a single port memory of address space 128 and the addressability of 16-bit in Verilog. The memory has four inputs: clock, write enable, write address register, and read address register, and two outputs: read and write data register.

Verilog source (Behavioral)

```
1 module simple_dual_port_ram #(parameter DATA_WIDTH = 16, parameter ADDR_WIDTH
  = 7)( input clk, input we, input [ADDR_WIDTH-1:0] waddr, input [ADDR_WIDTH
  -1:0] raddr, input [DATA_WIDTH-1:0] wdata, output reg [DATA_WIDTH-1:0]
  rdata);
2
3     localparam DEPTH = 1 << ADDR_WIDTH;
4
5     reg [DATA_WIDTH-1:0] memory [0:DEPTH-1];
6
7     always @(posedge clk) begin
8         if (we) begin
9             memory[waddr] <= wdata;
10        end
11        rdata <= memory[raddr];
12    end
13
14 endmodule
```

Testbench

```
1 module simple_dual_port_ram_tb;
2
3     localparam DATA_WIDTH = 16;
4     localparam ADDR_WIDTH = 7;
5
6     reg clk;
7     reg we;
8     reg [ADDR_WIDTH-1:0] waddr;
9     reg [ADDR_WIDTH-1:0] raddr;
10    reg [DATA_WIDTH-1:0] wdata;
11    wire [DATA_WIDTH-1:0] rdata;
12    reg [8*24-1:0] name;
13    simple_dual_port_ram #(.DATA_WIDTH(DATA_WIDTH), .ADDR_WIDTH(ADDR_WIDTH))
        dut (.clk(clk), .we(we), .waddr(waddr), .raddr(raddr), .wdata(wdata), .
        rdata(rdata));
14
15    initial begin
16        clk = 0;
17        forever #5 clk = ~clk; end
```



```

18
19 initial begin
20     name = "Kunwar_Arpit_Singh_22185";
21     $dumpfile("assign3_problem5_bidirectional_shift_register.vcd");
22     $dumpvars(1, simple_dual_port_ram_tb);
23     $display("Kunwar_Arpit_Singh");
24     $display("|_WE_|_WAddr_|_WData_|_RAddr_|_RData");
25     $monitor("|_b_|_d_|_h_|_d_|_h", we, waddr, wdata, raddr, rdata);
26
27     we = 0;
28     waddr = 0;
29     raddr = 0;
30     wdata = 0;
31     #10;
32
33     we = 1;
34     waddr = 10;
35     wdata = 16'hABCD;
36     #10;
37     we = 0;
38     #10;
39
40     we = 1;
41     waddr = 42;
42     wdata = 16'hDEAD;
43     #10;
44     we = 0;
45     #10;
46
47     raddr = 10;
48     #10;
49     raddr = 42;
50     #10;
51     raddr = 5;
52     #10;
53     we = 1;
54     waddr = 127;
55     wdata = 16'hFOOD;
56     raddr = 10;
57     #10;
58     we = 0;
59     #10;
60
61     raddr = 127;
62     #10;
63
64     $finish;
65 end
66
67 endmodule

```

Output in Terminal

```

1 VCD info: dumpfile assign3_problem5_bidirectional_shift_register.vcd opened
  for output.
2 Kunwar Arpit Singh
3 | WE | WAddr | WData | RAddr | RData
4 | 0 | 0 | 0000 | 0 | xxxx
5 | 1 | 10 | abcd | 0 | xxxx
6 | 0 | 10 | abcd | 0 | xxxx
7 | 1 | 42 | dead | 0 | xxxx
8 | 0 | 42 | dead | 0 | xxxx
9 | 0 | 42 | dead | 10 | xxxx
10 | 0 | 42 | dead | 10 | abcd
11 | 0 | 42 | dead | 42 | abcd
12 | 0 | 42 | dead | 42 | dead
13 | 0 | 42 | dead | 5 | dead
14 | 0 | 42 | dead | 5 | xxxx
15 | 1 | 127 | f00d | 10 | xxxx
16 | 1 | 127 | f00d | 10 | abcd
17 | 0 | 127 | f00d | 10 | abcd
18 | 0 | 127 | f00d | 127 | abcd
19 | 0 | 127 | f00d | 127 | f00d

```

Output Screenshots

Terminal output (Single Port RAM)

```

VCD info: dumpfile assign3_problem5_bidirectional_shift_register.vcd opened for output.
Kunwar Arpit Singh
| WE | WAddr | WData | RAddr | RData
| 0 | 0 | 0000 | 0 | xxxx
| 1 | 10 | abcd | 0 | xxxx
| 0 | 10 | abcd | 0 | xxxx
| 1 | 42 | dead | 0 | xxxx
| 0 | 42 | dead | 0 | xxxx
| 0 | 42 | dead | 10 | xxxx
| 0 | 42 | dead | 10 | abcd
| 0 | 42 | dead | 42 | abcd
| 0 | 42 | dead | 42 | dead
| 0 | 42 | dead | 5 | dead
| 0 | 42 | dead | 5 | xxxx
| 1 | 127 | f00d | 10 | xxxx
| 1 | 127 | f00d | 10 | abcd
| 0 | 127 | f00d | 10 | abcd
| 0 | 127 | f00d | 127 | abcd
| 0 | 127 | f00d | 127 | f00d

```

GTKWave waveform (Single Port RAM)

