

Computer Organization

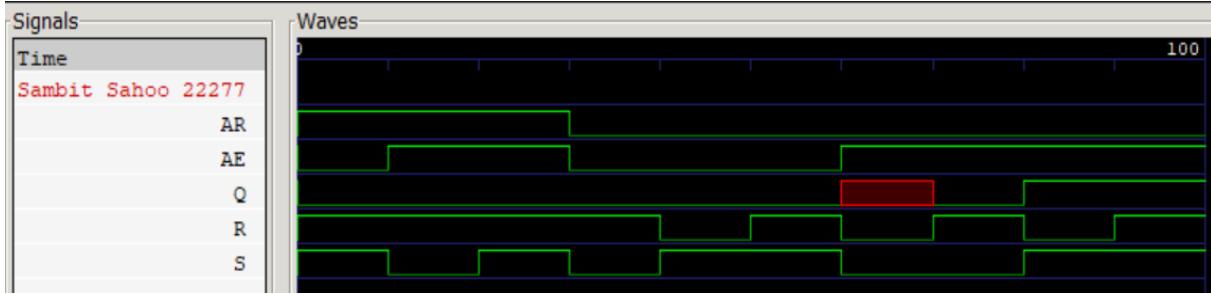
Assignment 3

Sambit Sahoo (22277)

Q1 - SR latch with asynchronous enable and reset

```
Assignment_3 > q1 >  a1.v
1
2 module srslatch_are (input AE, AR, S, R, output reg Q);
3     initial Q = 1'b0;
4     always @(AR, AE, S, R)
5         begin
6             if (AR) Q <= 0;
7             else
8                 begin
9                     if (AE == 1)
10                         begin
11                             if (S == 0 & R == 0) Q <= 1'bx;
12                             else if (S == 0 & R == 1) Q <= 0;
13                             else if (S == 1 & R == 0) Q <= 1;
14                         end
15                 end
16         end
17 endmodule
```

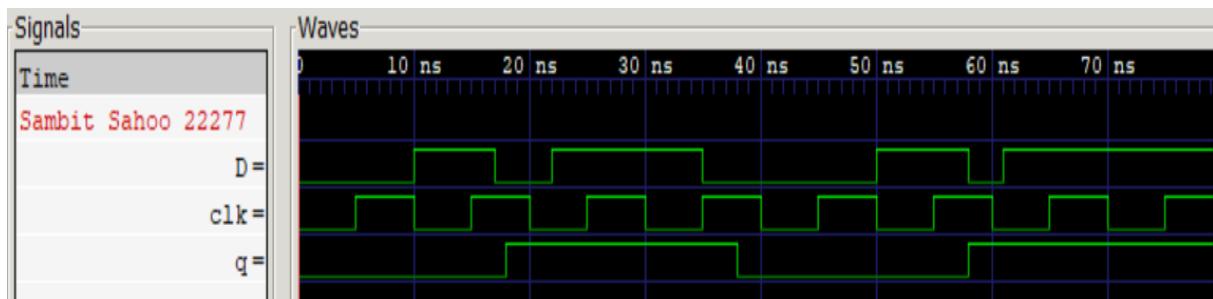
```
Assignment_3 > q1 >  a1.v
19 module a1_tb();
20     reg AE, AR, S, R;
21     wire Q;
22
23     srslatch_are SR (.AE(AE), .AR(AR), .S(S), .R(R), .Q(Q));
24
25     initial begin
26         $dumpfile("a1_tb.vcd");
27         $dumpvars(0,a1_tb);
28
29         AR = 1; AE = 0; S = 1; R = 1; #10;
30         AR = 1; AE = 1; S = 0; R = 1; #10;
31         AR = 1; AE = 1; S = 1; R = 1; #10;
32         AR = 0; AE = 0; S = 0; R = 1; #10;
33         AR = 0; AE = 0; S = 1; R = 0; #10;
34         AR = 0; AE = 0; S = 1; R = 1; #10;
35         AR = 0; AE = 1; S = 0; R = 0; #10;
36         AR = 0; AE = 1; S = 0; R = 1; #10;
37         AR = 0; AE = 1; S = 1; R = 0; #10;
38         AR = 0; AE = 1; S = 1; R = 1; #10;
39
40         $finish;
41     end
42
43 endmodule
```



Q2 - Improved version of D latch with delays of 1 ns to each gate

```
Assignment_3 > q2 > Ξ a2.v
1   `timescale 1ns/1ps
2   module improvedDLatch (input D, input clk, output reg q);
3       initial q = 1'b0;
4       reg nc, w1, w2;
5       always @(clk, D)
6           begin
7               #1 nc <= ~clk;
8               |   w1 <= D & clk;
9               #1 w2 <= nc & q;
10              #1 q <= w1 | w2;
11          end
12      endmodule
```

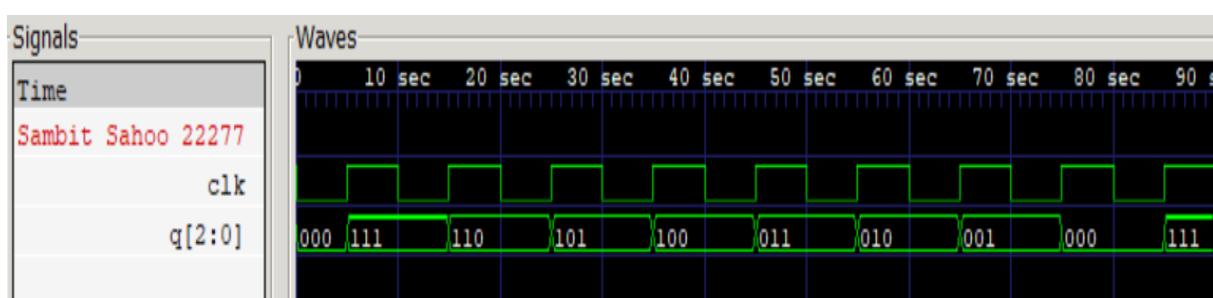
```
Assignment_3 > q2 > Ξ a2.v
14  module a2_tb();
15      reg D, clk;
16      wire q;
17
18      improvedDLatch latch (.D(D), .clk(clk), .q(q));
19
20      initial clk = 0;
21      always begin
22          #5;
23          clk = ~clk;
24      end
25
26      initial begin
27          $dumpfile("a2_tb.vcd");
28          $dumpvars(0,a2_tb);
29
30          D = 0; #10;
31          D = 1; #7;
32          D = 0; #5;
33          D = 1; #13;
34          D = 0; #15;
35          D = 1; #8;
36          D = 0; #3;
37          D = 1; #18;
38
39          $finish;
40      end
41
42  endmodule
```



Q3- Counter that counts from 7 to 0 (e.g., 7, 6, 5, 4, 3, 2, 1, 0)

```
Assignment_3 > q3 >  ≡ a3.v
1  module downcounter (input clk, output reg [2:0] q);
2      initial q = 3'b000;
3      always @(posedge clk)
4          begin
5              q <= q-1;
6          end
7  endmodule
```

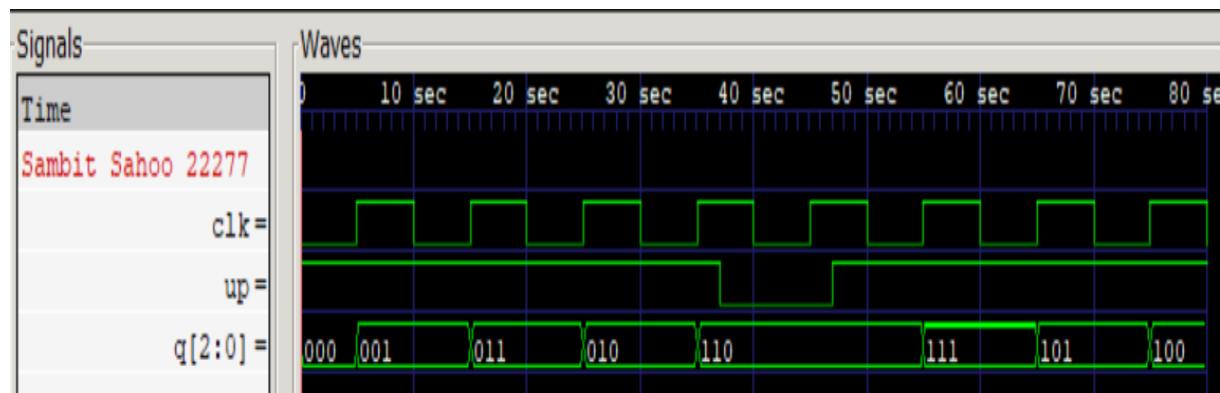
```
Assignment_3 > q3 > ≡ a3.v
 9   module a3_tb();
10     reg clk;
11     wire [2:0] q;
12
13     downcounter DC (.clk(clk), .q(q));
14
15     initial clk = 0;
16     always begin
17       #5;
18       clk = ~clk;
19     end
20
21     initial begin
22       $dumpfile("a3_tb.vcd");
23       $dumpvars(0,a3_tb);
24
25       #90;
26       $finish;
27     end
28
endmodule
```



Q4 - UP/DOWN modulo 8 Gray Code Counter

```
Assignment_3 > q4 >  ≡ a4.v
1  module grayupcounter (input up, input clk, output reg [2:0] q);
2      initial q = 3'b000;
3      always @ (posedge clk)
4          begin
5              if (up)
6                  begin
7                      q[2] <= q[0] ? q[2] : q[1];
8                      q[1] <= q[0] ? ~q[2] : q[1];
9                      q[0] <= ~(q[2] ^ q[1]);
10                 end
11             end
12         endmodule
```

```
Assignment_3 > q4 >  ≡ a4.v
14    module a4_tb();
15        reg up;
16        reg clk;
17        wire [2:0] q;
18
19        grayupcounter GUC (.clk(clk), .up(up), .q(q));
20
21        initial clk = 0;
22        always begin
23            #5;
24            clk = ~clk;
25        end
26
27        initial begin
28            $dumpfile("a4_tb.vcd");
29            $dumpvars(0,a4_tb);
30
31            up = 1; #37;
32            up = 0; #10;
33            up = 1; #33;
34
35            $finish;
36        end
37    endmodule
```

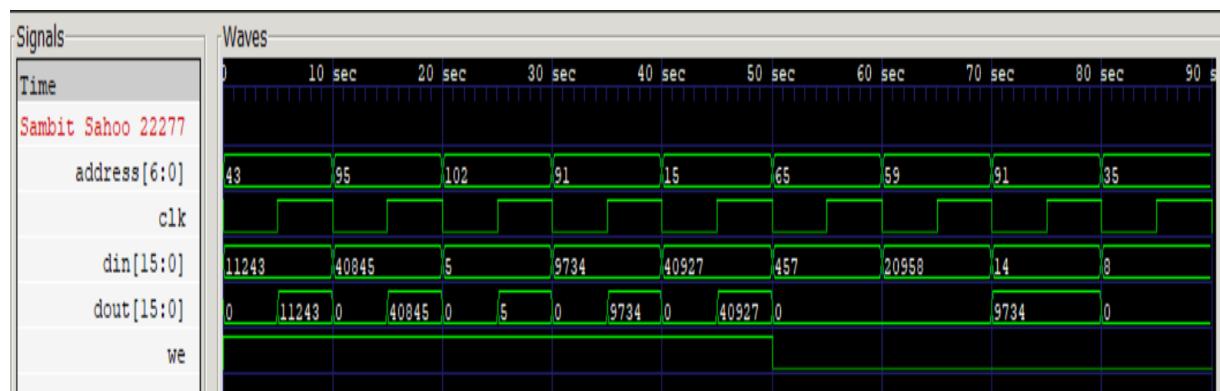


Q5- N-bit bidirectional shift register

```
Assignment_3 > q5 > Ξ a5.v
1  module nbitbidshifter #(parameter N = 4) (input clk, input s, input n0, output reg [N-1:0] a);
2    initial a = 3'b101;
3    always @ (posedge clk)
4      begin
5        if (s)
6          begin
7            a <= a >> 1;
8            a[N-1] <= n0;
9          end
10       else
11         begin
12           a <= a << 1;
13           a[0] <= n0;
14         end
15       end
16     endmodule
17 
```

```
Assignment_3 > q5 > Ξ a5.v
19  module a5_tb();
20    reg s;
21    reg clk;
22    reg n0;
23    wire [3:0] a;
24
25    nbitbidshifter NBBS (.s(s), .clk(clk), .n0(n0), .a(a));
26
27    initial clk = 0;
28    always begin
29      #5;
30      clk = ~clk;
31    end
32
33    initial begin
34      $dumpfile("a5_tb.vcd");
35      $dumpvars(0, a5_tb);
36
37      s = 0; n0 = 0; #11
38      s = 0; n0 = 1; #11
39      s = 1; n0 = 0; #11
40      s = 1; n0 = 1; #11
41      $finish;
42    end
43  endmodule

```



Q6- A single port memory of address space 128 with 16-bit addressability

```
Assignment_3 > q6 >  a6.v
1  module memory (input clk, input we, input [6:0] address, input [15:0] din, output [15:0] dout);
2    reg [15:0] mem [127:0];
3    integer i;
4    initial begin
5      for(i = 0; i <= 127; i++)
6        begin
7          mem[i] = 0;
8        end
9    end
10   always @(posedge clk)
11     if (we) mem[address] <= din;
12
13   assign dout = mem[address];
14 endmodule
```

```
Assignment_3 > q6 >  a6.v
15
16  module a6_tb();
17    reg we;
18    reg clk;
19    reg [6:0] address;
20    reg [15:0] din;
21    wire [15:0] dout;
22
23    memory M (.we(we), .clk(clk), .address(address), .din(din), .dout(dout));
24
25    initial clk = 0;
26    always begin
27      #5;
28      clk = ~clk;
29    end
30
31    initial begin
32      $dumpfile("a6_tb.vcd");
33      $dumpvars(0, a6_tb);
34
35      we = 1; address = 7'b0101011; din = 16'd11243; #10
36      we = 1; address = 7'b1011111; din = 16'd40845; #10
37      we = 1; address = 7'b1100110; din = 16'd5; #10
38      we = 1; address = 7'b1011011; din = 16'd9734; #10
39      we = 1; address = 7'b0001111; din = 16'd40927; #10
40      we = 0; address = 7'b1000001; din = 16'd457; #10
41      we = 0; address = 7'b0111011; din = 16'd20958; #10
42      we = 0; address = 7'b1011011; din = 16'd14; #10
43      we = 0; address = 7'b0100011; din = 16'd8; #10
44
45      $finish;
46    end
47  endmodule
```

