Verilog HDL

Assignment 3

By: Magdy Ahmed Abbas Abdelhamid

Question 1

D - Type Flip - Flop

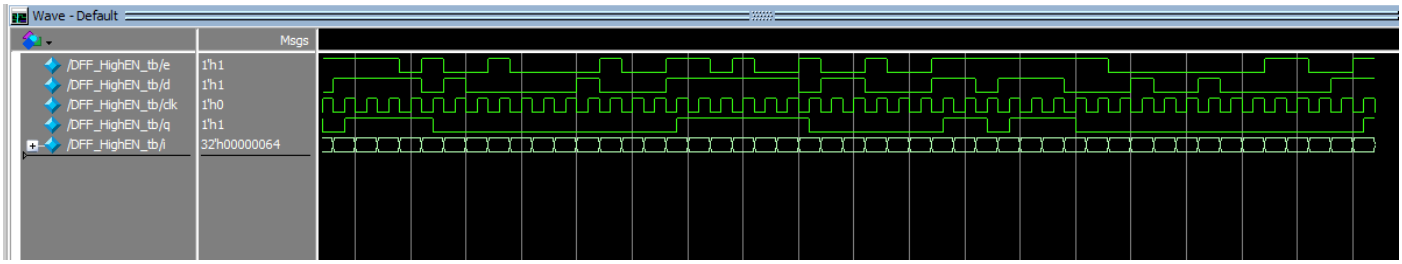
Verilog Code:

```
module DFF_HighEN (D, E, CLK, Q);

input D, E, CLK;
output reg Q;

always @(posedge CLK) begin
   if (E)
      Q  <=  D;
end

endmodule
```

Testbench Code:

```
module DFF_HighEN_tb ();
reg e, d, clk;
wire q;
DFF_HighEN DUT (.E(e),.D(d),.CLK(clk),.Q(q));
initial begin
  clk = 0;
  forever
  #10 clk = ~clk;
End
integer i;
initial begin
  // Test Normal Operation of FF
  @(negedge clk);
  e = 1;
  for (i = 0; i < 100; i = i + 1) begin
    d = $random;
    @(negedge clk);
  End
  // Test Holding Q to prev value
  e = 0;
  for (i = 0; i < 100; i = i + 1) begin
    d = $random;
    @(negedge clk);
  End
  // Randomization
  for (i = 0; i < 100; i = i + 1) begin
    d = $random;
    e = $random;
    @(negedge clk);
  end
  #1 $stop;
End
initial begin
  $monitor ("D = %b, EN = %b, CLK = %b, Q = %b", d, e, clk, q);
end

endmodule
```

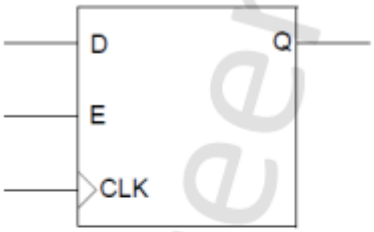
Waveform:




```
# D = x, EN = x, CLK = 0, Q = x
# D = x, EN = x, CLK = 1, Q = x
# D = 0, EN = 1, CLK = 0, Q = x
# D = 0, EN = 1, CLK = 1, Q = 0
# D = 1, EN = 1, CLK = 0, Q = 0
# D = 1, EN = 1, CLK = 1, Q = 1
# D = 1, EN = 1, CLK = 0, Q = 1
# D = 1, EN = 1, CLK = 1, Q = 1
# D = 1, EN = 1, CLK = 0, Q = 1
# D = 1, EN = 1, CLK = 1, Q = 1
# D = 1, EN = 1, CLK = 0, Q = 1
# D = 1, EN = 1, CLK = 1, Q = 1
# D = 1, EN = 1, CLK = 0, Q = 1
# D = 1, EN = 1, CLK = 1, Q = 1
# D = 1, EN = 1, CLK = 0, Q = 1
# D = 1, EN = 1, CLK = 1, Q = 1
# D = 0, EN = 1, CLK = 0, Q = 1
# D = 0, EN = 1, CLK = 1, Q = 0
# D = 1, EN = 1, CLK = 0, Q = 0
# D = 1, EN = 1, CLK = 1, Q = 1
# D = 1, EN = 1, CLK = 0, Q = 1
```

```
# D = 0, EN = 0, CLK = 1, Q = 0
# D = 1, EN = 0, CLK = 0, Q = 0
# D = 1, EN = 0, CLK = 1, Q = 0
# D = 1, EN = 0, CLK = 0, Q = 0
# D = 1, EN = 0, CLK = 1, Q = 0
# D = 0, EN = 0, CLK = 0, Q = 0
# D = 0, EN = 0, CLK = 1, Q = 0
# D = 0, EN = 0, CLK = 0, Q = 0
# D = 0, EN = 0, CLK = 1, Q = 0
# D = 0, EN = 0, CLK = 0, Q = 0
# D = 0, EN = 0, CLK = 1, Q = 0
# D = 1, EN = 0, CLK = 0, Q = 0
# D = 1, EN = 0, CLK = 1, Q = 0
# D = 0, EN = 0, CLK = 0, Q = 0
# D = 0, EN = 0, CLK = 1, Q = 0
# D = 1, EN = 0, CLK = 0, Q = 0
# D = 1, EN = 0, CLK = 1, Q = 0
# D = 0, EN = 0, CLK = 0, Q = 0
# D = 0, EN = 0, CLK = 1, Q = 0
```

```
# D = 0, EN = 1, CLK = 1, Q = 0
# D = 1, EN = 0, CLK = 0, Q = 0
# D = 1, EN = 0, CLK = 1, Q = 0
# D = 1, EN = 1, CLK = 0, Q = 0
# D = 1, EN = 1, CLK = 1, Q = 1
# D = 1, EN = 1, CLK = 0, Q = 1
# D = 1, EN = 1, CLK = 1, Q = 1
# D = 0, EN = 1, CLK = 0, Q = 1
# D = 0, EN = 1, CLK = 1, Q = 0
# D = 1, EN = 1, CLK = 0, Q = 0
# D = 1, EN = 1, CLK = 1, Q = 0
# D = 0, EN = 1, CLK = 0, Q = 0
# D = 0, EN = 1, CLK = 1, Q = 0
```



Input	Output
D, E, CLK	Q

Truth Table

E	CLK	D	Q_{n+1}
0	X	X	Q_n
1	not Rising	X	Q_n
1	↑	D	D

Question 2

Data Latch

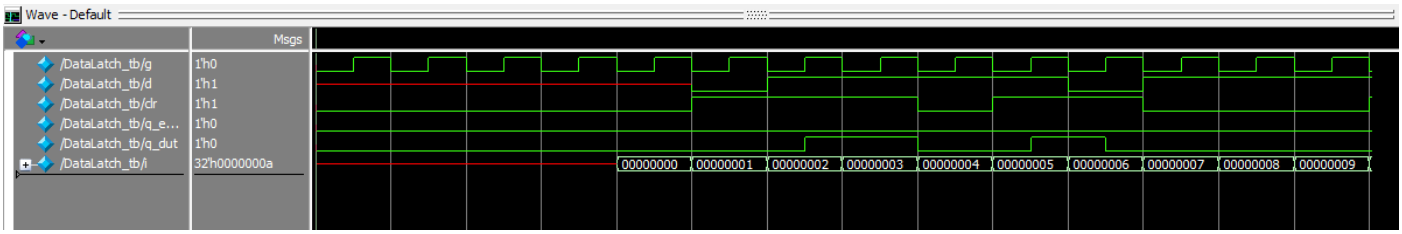
Verilog Code:

```
module DataLatch (CLR, D, G, Q);

input CLR, D, G;
output reg Q;

always @(*) begin
   if (~CLR)
      Q  <=  1'b0;
   else if (G)
      Q  <=  D;
end

endmodule
```

Testbench Code:

```
module DataLatch_tb ();

reg g, d, clr, q_expected;
wire q_dut;

DataLatch DUT (.G(g), .D(d), .CLR(clr), .Q(q_dut));

initial begin
  g = 0;
  forever
  #10 g = ~g;
end

integer i;
initial begin
  clr = 0; // Clear is on for 4 Cycles
  q_expected = 0;
  repeat(4) @(negedge g) begin
    if (q_dut != q_expected) begin
      $display ("Clear Fails");
      $display ("Error @ CLR = %b, G = %b, D = %b, Q = %b", clr, g, d, q_dut);
      $stop;
    end
  end

  for (i = 0 ; i < 10 ; i = i + 1) begin
    @(negedge g);
    d = $random;
    clr = $random;
  end
  #1 $stop;
end

initial
$monitor("@ time = %g \t D = %b \t G = %b \t CLR = %b \t
                                        Q = %b", $time, d, g, clr, q_dut);

endmodule
```

Waveform:

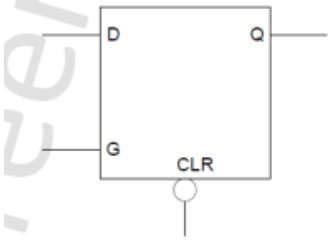



```
# @ time = 0    D = x    G = 0    CLR = 0    Q = 0
# @ time = 10    D = x    G = 1    CLR = 0    Q = 0
# @ time = 20    D = x    G = 0    CLR = 0    Q = 0
# @ time = 30    D = x    G = 1    CLR = 0    Q = 0
# @ time = 40    D = x    G = 0    CLR = 0    Q = 0
# @ time = 50    D = x    G = 1    CLR = 0    Q = 0
# @ time = 60    D = x    G = 0    CLR = 0    Q = 0
# @ time = 70    D = x    G = 1    CLR = 0    Q = 0
# @ time = 80    D = x    G = 0    CLR = 0    Q = 0
# @ time = 90    D = x    G = 1    CLR = 0    Q = 0
# @ time = 100    D = 0    G = 0    CLR = 1    Q = 0
# @ time = 110    D = 0    G = 1    CLR = 1    Q = 0
# @ time = 120    D = 1    G = 0    CLR = 1    Q = 0
# @ time = 130    D = 1    G = 1    CLR = 1    Q = 1
# @ time = 140    D = 1    G = 0    CLR = 1    Q = 1
# @ time = 150    D = 1    G = 1    CLR = 1    Q = 1
# @ time = 160    D = 1    G = 0    CLR = 0    Q = 0
# @ time = 170    D = 1    G = 1    CLR = 0    Q = 0
# @ time = 180    D = 1    G = 0    CLR = 1    Q = 0
# @ time = 190    D = 1    G = 1    CLR = 1    Q = 1
# @ time = 200    D = 0    G = 0    CLR = 1    Q = 1
# @ time = 210    D = 0    G = 1    CLR = 1    Q = 0
# @ time = 220    D = 1    G = 0    CLR = 0    Q = 0
# @ time = 230    D = 1    G = 1    CLR = 0    Q = 0
# @ time = 240    D = 1    G = 0    CLR = 0    Q = 0
# @ time = 250    D = 1    G = 1    CLR = 0    Q = 0
# @ time = 260    D = 1    G = 0    CLR = 0    Q = 0
# @ time = 270    D = 1    G = 1    CLR = 0    Q = 0
# @ time = 280    D = 1    G = 0    CLR = 1    Q = 0
```

Input	Output
CLR, D, G	Q

Truth Table

CLR	G	D	Q
0	X	X	0
1	0	X	Q
1	1	D	D



Question 3

Specific Latch

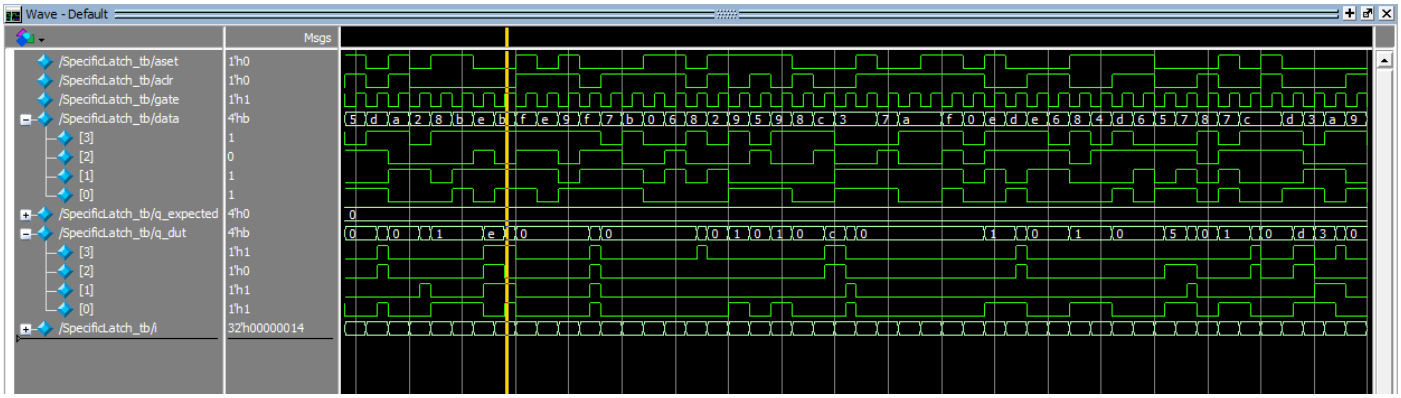
Verilog Code:

```
module SpecificLatch (ASET, DATA, GATE, ACLR, Q);

parameter LAT_WIDTH = 4;
input ASET, GATE, ACLR;
input [LAT_WIDTH - 1 : 0] DATA;
output reg [LAT_WIDTH - 1 : 0] Q;

always @(*) begin
   if (ACLR)
      Q <= 1'b0;
   else if (ASET)
      Q <= 1'b1;
   else if (GATE)
      Q <= DATA;
end

endmodule
```

Testbench Code:

```
module SpecificLatch_tb ();

parameter N = 4;
reg aset, aclr, gate;
reg [N - 1: 0] data, q_expected;
wire [N - 1: 0] q_dut;

SpecificLatch #(.LAT_WIDTH(N)) DUT (.ACLR(aclr), .ASET(aset), .DATA(data), .GATE(gate), .Q(q_dut));

initial begin
  gate = 0;
  forever #10 gate = ~gate;
end

integer i;
initial begin
  aclr = 0;
  aset = 1;
  q_expected = 1;
  repeat(2) @(negedge gate);
  if (q_dut != q_expected) begin
    $display ("The Latch Design is Wrong! ");
    $stop;
  end
  aclr = 1;
  aset = 1;
  q_expected = 0;
  repeat(2) @(negedge gate);
  if (q_dut != q_expected) begin
    $display ("The Latch Design is Wrong! ");
    $stop;
  end
  for (i = 0 ; i < 100 ; i = i + 1) begin
    aclr = $random;
    aset = $random;
    data = $random;
    @(negedge gate);
  end
  #2 $stop;
end

initial
$monitor("DATA = %b \t ACLR = %b \t ASET = %b \t GATE = %b \t Q = %b", data, aclr, aset, gate, q_dut);

endmodule
```

Waveform:




```
# DATA = 0011   ACLR = 0   ASET = 1   GATE = 1   Q = 0001
# DATA = 1110   ACLR = 0   ASET = 0   GATE = 0   Q = 0001
# DATA = 1110   ACLR = 0   ASET = 0   GATE = 1   Q = 1110
# DATA = 0001   ACLR = 0   ASET = 1   GATE = 0   Q = 0001
# DATA = 0001   ACLR = 0   ASET = 1   GATE = 1   Q = 0001
# DATA = 1111   ACLR = 0   ASET = 0   GATE = 0   Q = 0001
# DATA = 1111   ACLR = 0   ASET = 0   GATE = 1   Q = 1111
# DATA = 1101   ACLR = 0   ASET = 0   GATE = 0   Q = 1111
# DATA = 1101   ACLR = 0   ASET = 0   GATE = 1   Q = 1101
# DATA = 1001   ACLR = 0   ASET = 0   GATE = 0   Q = 1101
# DATA = 1001   ACLR = 0   ASET = 0   GATE = 1   Q = 1001
# DATA = 0011   ACLR = 0   ASET = 0   GATE = 0   Q = 1001
# DATA = 0011   ACLR = 0   ASET = 0   GATE = 1   Q = 0011
# DATA = 0110   ACLR = 1   ASET = 1   GATE = 0   Q = 0000
# DATA = 0110   ACLR = 1   ASET = 1   GATE = 1   Q = 0000
# DATA = 1001   ACLR = 0   ASET = 0   GATE = 0   Q = 0000
# DATA = 1001   ACLR = 0   ASET = 0   GATE = 1   Q = 1001
# DATA = 1111   ACLR = 0   ASET = 0   GATE = 0   Q = 1001
# DATA = 1111   ACLR = 0   ASET = 0   GATE = 1   Q = 1111
# DATA = 0010   ACLR = 0   ASET = 0   GATE = 0   Q = 1111
# DATA = 0010   ACLR = 0   ASET = 0   GATE = 1   Q = 0010
# DATA = 0100   ACLR = 0   ASET = 1   GATE = 0   Q = 0001
# DATA = 0100   ACLR = 0   ASET = 1   GATE = 1   Q = 0001
# DATA = 1001   ACLR = 0   ASET = 0   GATE = 0   Q = 0001
# DATA = 1001   ACLR = 0   ASET = 0   GATE = 1   Q = 1001
# DATA = 1011   ACLR = 1   ASET = 0   GATE = 0   Q = 0000
# DATA = 1011   ACLR = 1   ASET = 0   GATE = 1   Q = 0000
# DATA = 1001   ACLR = 1   ASET = 1   GATE = 0   Q = 0000
```

Name	Type	Description
aset	Input	Asynchronous set input. Sets q[] output to 1.
data[]		Data Input to the D-type latch with width LAT_WIDTH
gate		Latch enable input
aclr		Asynchronous clear input. Sets q[] output to 0.
q[]	Output	Data output from the latch with with LAT_WIDTH

If both aset and aclr are both asserted, aclr is dominant.

Question 4

(A) T - type Flip - flop

Verilog Code

```
module T_FlipFlop (t, rstn, clk, q, qbar);

input t, rstn, clk;
output reg q; // output reg q, qbar;
output qbar;

always @(posedge clk or negedge rstn) begin
  if (~rstn)
    q <= 1'b0; // qbar <= 1'b1;
  else if (t)
    q <= ~q;  // qbar <= ~qbar;
end

assign qbar = ~q;

endmodule
```

(B) D Flip - flop

Verilog Code:

```
module D_FlipFlop (d, rstn, clk, q, qbar);

input d, rstn, clk;
output reg q; // output reg q, qbar;
output qbar;

always @(posedge clk or negedge rstn) begin
  if (~rstn)
    q <= 1'b0; // qbar <= 1'b1;
  else
    q <= d;  // qbar <= ~d;
end

assign qbar = ~q;

endmodule
```

(C) Parameterized Flip - flop

Verilog Code:

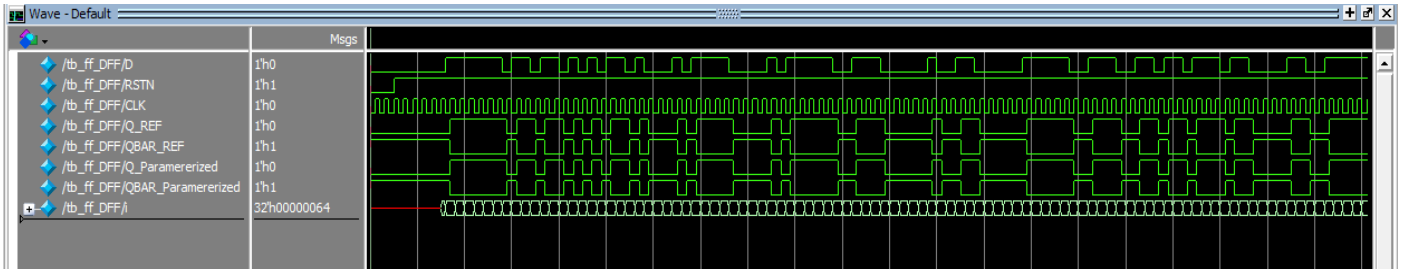
```
module Param_FF (d, rstn, clk, q, qbar);

parameter FF_TYPE = "DFF";
// d to be used as D input for DFF and Toggle input for TFF
input d, rstn, clk;
output reg q, qbar;

always @(posedge clk or negedge rstn) begin
  if (~rstn) begin
    q <= 1'b0;
    qbar <= 1'b1;
  end
  else begin
    if (FF_TYPE == "DFF") begin
      q <= d;
      qbar <= ~d;
    end
    else if (FF_TYPE == "TFF") begin
      if (d) begin
        q <= ~q;
        qbar <= ~qbar;
      end
    end
  end
end

endmodule

/*
module Param_FF (d, clk, rstn, q, qbar);
parameter FF_TYPE = "DFF";
input d, clk, rstn;
output q, qbar;
generate
  case(FF_TYPE)
    "DFF" : D_FlipFlop FF1(.d(d),.clk(clk),.rstn(rstn),.q(q),.qbar(qbar));
    "TFF" : T_FlipFlop FF2(.t(d),.clk(clk),.rstn(rstn),.q(q),.qbar(qbar));
  endcase
endgenerate
endmodule */
```

(D) Test for (C)

Testbench 1 Code For DFF:

```
module tb_ff_DFF ();

reg D, RSTN, CLK;
wire Q_REF, QBAR_REF, Q_Paramererized, QBAR_Paramererized;

D_FlipFlop DUT_REF (.d(D), .rstn(RSTN), .clk(CLK), .q(Q_REF), .qbar(QBAR_REF));
Param_FF #("DFF")
DUT_Parameterized (.d(D), .rstn(RSTN), .clk(CLK), .q(Q_Paramererized), .qbar(QBAR_Paramererized));

// Clock Generation
initial begin
  CLK = 0;
  forever
    #10 CLK = ~CLK;
end

integer i;

// Reset and Initial Values for Inputs
initial begin
  RSTN = 0;
  D = 0;
  #50; // repeat(5) @(negedge CLK);
  RSTN = 1;
  #100;
  for (i = 0 ; i < 100 ; i = i + 1) begin
    D = $random;
    @(negedge CLK);
  end
  #2 $stop;
end

// Checking
always @(posedge CLK) begin
  if ((Q_REF != Q_Paramererized) || (QBAR_REF != QBAR_Paramererized)) begin
    $display ("The DFF Design is Wrong! ");
    $stop;
  end
end

// Test Monitor and Results
initial begin
  $monitor ("time = %0t, D = %b, RSTN = %b, Q_DFF = %b, QBAR_DFF = %b",
                                                    $time, D, RSTN, Q_REF, QBAR_REF);
End
endmodule
```

Waveform tb 1:

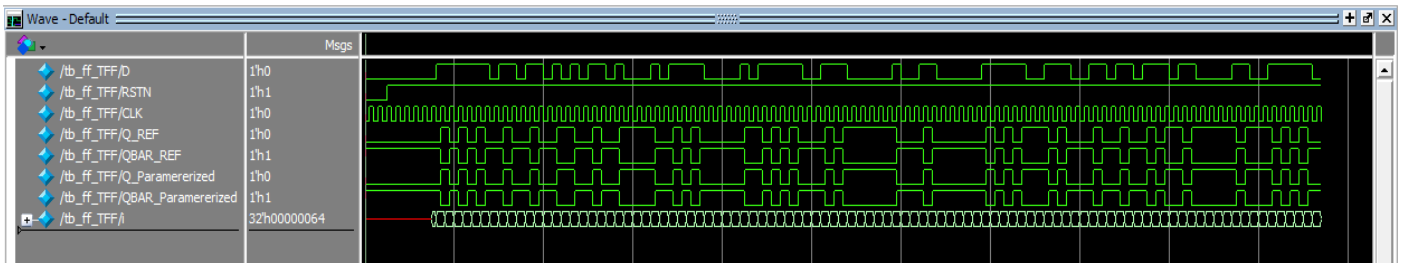



```
# time = 1200, D = 0, RSTN = 1, Q_DFF = 1, QBAR_DFF = 0
# time = 1210, D = 0, RSTN = 1, Q_DFF = 0, QBAR_DFF = 1
# time = 1240, D = 1, RSTN = 1, Q_DFF = 0, QBAR_DFF = 1
# time = 1250, D = 1, RSTN = 1, Q_DFF = 1, QBAR_DFF = 0
# time = 1280, D = 0, RSTN = 1, Q_DFF = 1, QBAR_DFF = 0
# time = 1290, D = 0, RSTN = 1, Q_DFF = 0, QBAR_DFF = 1
# time = 1380, D = 1, RSTN = 1, Q_DFF = 0, QBAR_DFF = 1
# time = 1390, D = 1, RSTN = 1, Q_DFF = 1, QBAR_DFF = 0
# time = 1480, D = 0, RSTN = 1, Q_DFF = 1, QBAR_DFF = 0
# time = 1490, D = 0, RSTN = 1, Q_DFF = 0, QBAR_DFF = 1
# time = 1520, D = 1, RSTN = 1, Q_DFF = 0, QBAR_DFF = 1
# time = 1530, D = 1, RSTN = 1, Q_DFF = 1, QBAR_DFF = 0
# time = 1580, D = 0, RSTN = 1, Q_DFF = 1, QBAR_DFF = 0
# time = 1590, D = 0, RSTN = 1, Q_DFF = 0, QBAR_DFF = 1
# time = 1620, D = 1, RSTN = 1, Q_DFF = 0, QBAR_DFF = 1
# time = 1630, D = 1, RSTN = 1, Q_DFF = 1, QBAR_DFF = 0
# time = 1660, D = 0, RSTN = 1, Q_DFF = 1, QBAR_DFF = 0
# time = 1670, D = 0, RSTN = 1, Q_DFF = 0, QBAR_DFF = 1
# time = 1680, D = 1, RSTN = 1, Q_DFF = 0, QBAR_DFF = 1
# time = 1690, D = 1, RSTN = 1, Q_DFF = 1, QBAR_DFF = 0
# time = 1720, D = 0, RSTN = 1, Q_DFF = 1, QBAR_DFF = 0
# time = 1730, D = 0, RSTN = 1, Q_DFF = 0, QBAR_DFF = 1
# time = 1740, D = 1, RSTN = 1, Q_DFF = 0, QBAR_DFF = 1
# time = 1750, D = 1, RSTN = 1, Q_DFF = 1, QBAR_DFF = 0
# time = 1800, D = 0, RSTN = 1, Q_DFF = 1, QBAR_DFF = 0
# time = 1810, D = 0, RSTN = 1, Q_DFF = 0, QBAR_DFF = 1
# time = 1820, D = 1, RSTN = 1, Q_DFF = 0, QBAR_DFF = 1
# time = 1830, D = 1, RSTN = 1, Q_DFF = 1, QBAR_DFF = 0
# time = 1860, D = 0, RSTN = 1, Q_DFF = 1, QBAR_DFF = 0
# time = 1870, D = 0, RSTN = 1, Q_DFF = 0, QBAR_DFF = 1
# time = 1940, D = 1, RSTN = 1, Q_DFF = 0, QBAR_DFF = 1
# time = 1950, D = 1, RSTN = 1, Q_DFF = 1, QBAR_DFF = 0
# time = 1980, D = 0, RSTN = 1, Q_DFF = 1, QBAR_DFF = 0
# time = 1990, D = 0, RSTN = 1, Q_DFF = 0, QBAR_DFF = 1
# time = 2020, D = 1, RSTN = 1, Q_DFF = 0, QBAR_DFF = 1
# time = 2030, D = 1, RSTN = 1, Q_DFF = 1, QBAR_DFF = 0
```

Testbench 2 Code For TFF:

```
module tb_ff_TFF ();

reg D, RSTN, CLK;
wire Q_REF, QBAR_REF, Q_Paramererized, QBAR_Paramererized;

T_FlipFlop DUT_REF (.t(D),.rstn(RSTN),.clk(CLK),.q(Q_REF),.qbar(QBAR_REF));
Param_FF #("TFF")
DUT_Parameterized (.d(D),.rstn(RSTN),.clk(CLK),.q(Q_Paramererized),.qbar(QBAR_Paramererized));

// Clock Generation
initial begin
  CLK = 0;
  forever
    #10 CLK = ~CLK;
end

integer i;

// Reset and Initial Values for Inputs
initial begin
  RSTN = 0;
  D = 0;
  #50; // repeat(5) @(negedge CLK);
  RSTN = 1;
  #100;
  for (i = 0 ; i < 100 ; i = i + 1) begin
    D = $random;
    @(negedge CLK);
  end
  $stop;
end

// Checking
always @(posedge CLK) begin
  if ((Q_REF != Q_Paramererized) || (QBAR_REF != QBAR_Paramererized)) begin
    $display ("The DFF Design is Wrong! ");
    #2 $stop;
  end
end

// Test Monitor and Results
initial begin
  $monitor ("time = %0t, D = %b, RSTN = %b, Q_TFF = %b, QBAR_TFF = %b",
                                                    $time, D, RSTN, Q_REF, QBAR_REF);
end

endmodule
```

Waveform tb 2:




```
# time = 1200, D = 0, RSTN = 1, Q_TFF = 0, QBAR_TFF = 1
# time = 1240, D = 1, RSTN = 1, Q_TFF = 0, QBAR_TFF = 1
# time = 1250, D = 1, RSTN = 1, Q_TFF = 1, QBAR_TFF = 0
# time = 1270, D = 1, RSTN = 1, Q_TFF = 0, QBAR_TFF = 1
# time = 1280, D = 0, RSTN = 1, Q_TFF = 0, QBAR_TFF = 1
# time = 1380, D = 1, RSTN = 1, Q_TFF = 0, QBAR_TFF = 1
# time = 1390, D = 1, RSTN = 1, Q_TFF = 1, QBAR_TFF = 0
# time = 1410, D = 1, RSTN = 1, Q_TFF = 0, QBAR_TFF = 1
# time = 1430, D = 1, RSTN = 1, Q_TFF = 1, QBAR_TFF = 0
# time = 1450, D = 1, RSTN = 1, Q_TFF = 0, QBAR_TFF = 1
# time = 1470, D = 1, RSTN = 1, Q_TFF = 1, QBAR_TFF = 0
# time = 1480, D = 0, RSTN = 1, Q_TFF = 1, QBAR_TFF = 0
# time = 1520, D = 1, RSTN = 1, Q_TFF = 1, QBAR_TFF = 0
# time = 1530, D = 1, RSTN = 1, Q_TFF = 0, QBAR_TFF = 1
# time = 1550, D = 1, RSTN = 1, Q_TFF = 1, QBAR_TFF = 0
# time = 1570, D = 1, RSTN = 1, Q_TFF = 0, QBAR_TFF = 1
# time = 1580, D = 0, RSTN = 1, Q_TFF = 0, QBAR_TFF = 1
# time = 1620, D = 1, RSTN = 1, Q_TFF = 0, QBAR_TFF = 1
# time = 1630, D = 1, RSTN = 1, Q_TFF = 1, QBAR_TFF = 0
# time = 1650, D = 1, RSTN = 1, Q_TFF = 0, QBAR_TFF = 1
# time = 1660, D = 0, RSTN = 1, Q_TFF = 0, QBAR_TFF = 1
# time = 1680, D = 1, RSTN = 1, Q_TFF = 0, QBAR_TFF = 1
# time = 1690, D = 1, RSTN = 1, Q_TFF = 1, QBAR_TFF = 0
# time = 1710, D = 1, RSTN = 1, Q_TFF = 0, QBAR_TFF = 1
# time = 1720, D = 0, RSTN = 1, Q_TFF = 0, QBAR_TFF = 1
# time = 1740, D = 1, RSTN = 1, Q_TFF = 0, QBAR_TFF = 1
# time = 1750, D = 1, RSTN = 1, Q_TFF = 1, QBAR_TFF = 0
# time = 1770, D = 1, RSTN = 1, Q_TFF = 0, QBAR_TFF = 1
# time = 1790, D = 1, RSTN = 1, Q_TFF = 1, QBAR_TFF = 0
# time = 1800, D = 0, RSTN = 1, Q_TFF = 1, QBAR_TFF = 0
# time = 1820, D = 1, RSTN = 1, Q_TFF = 1, QBAR_TFF = 0
# time = 1830, D = 1, RSTN = 1, Q_TFF = 0, QBAR_TFF = 1
# time = 1850, D = 1, RSTN = 1, Q_TFF = 1, QBAR_TFF = 0
# time = 1860, D = 0, RSTN = 1, Q_TFF = 1, QBAR_TFF = 0
# time = 1940, D = 1, RSTN = 1, Q_TFF = 1, QBAR_TFF = 0
# time = 1950, D = 1, RSTN = 1, Q_TFF = 0, QBAR_TFF = 1
# time = 1970, D = 1, RSTN = 1, Q_TFF = 1, QBAR_TFF = 0
# time = 1980, D = 0, RSTN = 1, Q_TFF = 1, QBAR_TFF = 0
# time = 2020, D = 1, RSTN = 1, Q_TFF = 1, QBAR_TFF = 0
# time = 2030, D = 1, RSTN = 1, Q_TFF = 0, QBAR_TFF = 1
```

Question 5

4 - Bit Ripple Counter

Verilog Code:

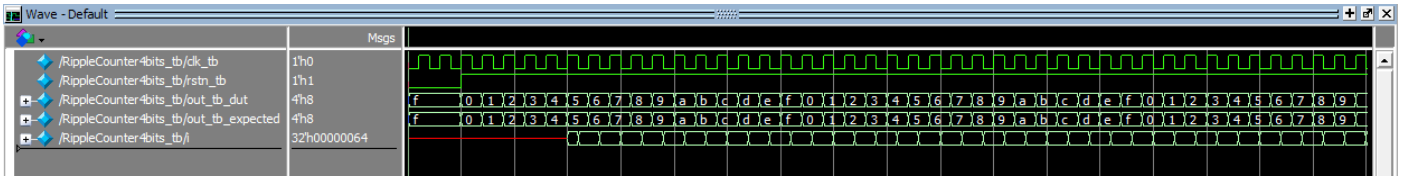
```
module RippleCounter4bits (CLK, RSTN, OUT);

parameter N = 4;
input CLK, RSTN;
output [N - 1: 0] OUT;

wire Q0, Q1, Q2, Q3;
wire Qn0, Qn1, Qn2, Qn3;

D_FlipFlop ff1 (.d(Qn0), .rstn(RSTN), .clk(CLK), .q(Q0), .qbar(Qn0));
D_FlipFlop ff2 (.d(Qn1), .rstn(RSTN), .clk(Q0), .q(Q1), .qbar(Qn1));
D_FlipFlop ff3 (.d(Qn2), .rstn(RSTN), .clk(Q1), .q(Q2), .qbar(Qn2));
D_FlipFlop ff4 (.d(Qn3), .rstn(RSTN), .clk(Q2), .q(Q3), .qbar(Qn3));

assign OUT = {Qn3, Qn2, Qn1, Qn0};

endmodule
```

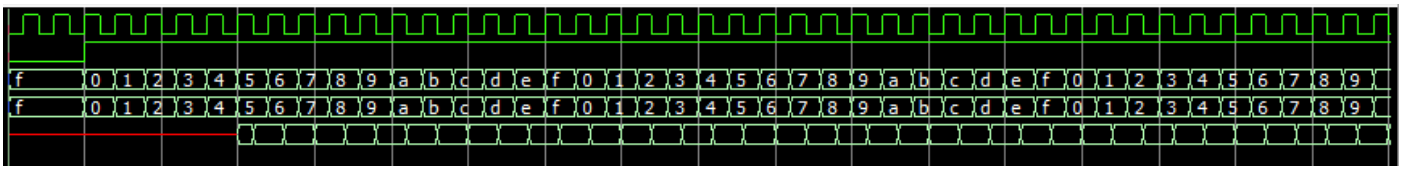
```
module RippleCounter4bits_Gen (CLK, RSTN, OUT);

parameter N = 4;
input CLK, RSTN;
output [N - 1: 0] OUT;

wire [N - 1: 0] Q;
wire [N - 1: 0] Qn;
generate
  genvar i;
    D_FlipFlop ff1 (.d(Qn[0]), .rstn(RSTN), .clk(CLK), .q(Q[0]), .qbar(Qn[0]));
  for (i = 1; i < 4 ; i = i + 1) begin
    D_FlipFlop ffi (.d(Qn[i]), .rstn(RSTN), .clk(Q[i - 1]), .q(Q[i]), .qbar(Qn[i]));
  end
endgenerate

assign OUT = Qn;

endmodule
```

Testbench Code:

```
module RippleCounter4bits_tb ();

parameter n = 4;
reg clk_tb, rstn_tb;
wire [n − 1: 0] out_tb_dut, out_tb_expected;

RippleCounter4bits #(.N(n)) DUT_Ref (.CLK(clk_tb), .RSTN(rstn_tb), .OUT(out_tb_dut));
RippleCounter4bits_Gen #(.N(n)) DUT (.CLK(clk_tb), .RSTN(rstn_tb), .OUT(out_tb_expected));

initial begin
  clk_tb = 0;
  forever
    #10 clk_tb = ~clk_tb;
end

integer i;

initial begin
  rstn_tb = 0;
  // out_tb_expected = 4'b1111;
  #50; // repeat(4) @(negedge clk_tb); = #80
  if (out_tb_dut != out_tb_expected) begin
    $display ("The 4 − Bit Ripple Counter Design is Wrong! ");
    $stop;
  end

  rstn_tb = 1;
  #100; // repeat(8) @(negedge clk_tb);

  for (i = 0 ; i < 100 ; i = i + 1) begin
    @(negedge clk_tb);
    if (out_tb_dut != out_tb_expected) begin
      $display ("The 4 − Bit Ripple Counter Design is Wrong! ");
      $stop;
    end
  end
  #2 $stop;
end

initial
  $monitor ("time = %0t, Reset = %b, CLK = %b, Out = %b", $time, rstn_tb, clk_tb, out_tb_dut);

endmodule
```

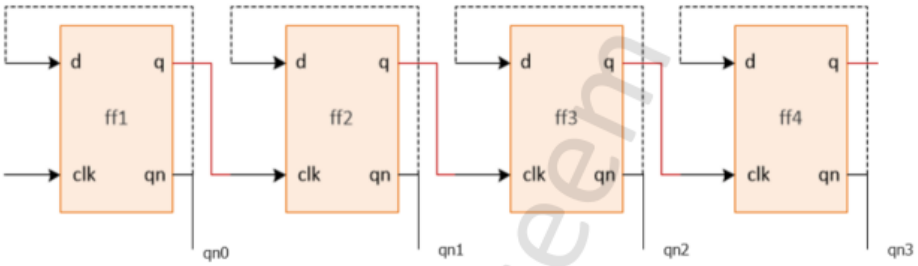
Waveform:





Do File for 4 - Bit Ripple Counter

```
vlib work
vlog RippleCounter4bits. v RippleCounter4bits_tb. v RippleCounter4bits_Gen. v
vsim  −  voptargs  =  +acc work. RippleCounter4bits_tb
add wave  ∗
run  −  all
```




```
# time = 0, Reset = 0, CLK = 0, Out = 1111
# time = 10, Reset = 0, CLK = 1, Out = 1111
# time = 20, Reset = 0, CLK = 0, Out = 1111
# time = 30, Reset = 0, CLK = 1, Out = 1111
# time = 40, Reset = 0, CLK = 0, Out = 1111
# time = 50, Reset = 1, CLK = 1, Out = 0000
# time = 60, Reset = 1, CLK = 0, Out = 0000
# time = 70, Reset = 1, CLK = 1, Out = 0001
# time = 80, Reset = 1, CLK = 0, Out = 0001
# time = 90, Reset = 1, CLK = 1, Out = 0010
# time = 100, Reset = 1, CLK = 0, Out = 0010
# time = 110, Reset = 1, CLK = 1, Out = 0011
# time = 120, Reset = 1, CLK = 0, Out = 0011
# time = 130, Reset = 1, CLK = 1, Out = 0100
# time = 140, Reset = 1, CLK = 0, Out = 0100
# time = 150, Reset = 1, CLK = 1, Out = 0101
# time = 160, Reset = 1, CLK = 0, Out = 0101
# time = 170, Reset = 1, CLK = 1, Out = 0110
# time = 180, Reset = 1, CLK = 0, Out = 0110
# time = 190, Reset = 1, CLK = 1, Out = 0111
# time = 200, Reset = 1, CLK = 0, Out = 0111
# time = 210, Reset = 1, CLK = 1, Out = 1000
# time = 220, Reset = 1, CLK = 0, Out = 1000
# time = 230, Reset = 1, CLK = 1, Out = 1001
# time = 240, Reset = 1, CLK = 0, Out = 1001
# time = 250, Reset = 1, CLK = 1, Out = 1010
# time = 260, Reset = 1, CLK = 0, Out = 1010
# time = 270, Reset = 1, CLK = 1, Out = 1011
# time = 280, Reset = 1, CLK = 0, Out = 1011
# time = 290, Reset = 1, CLK = 1, Out = 1100
# time = 300, Reset = 1, CLK = 0, Out = 1100
# time = 310, Reset = 1, CLK = 1, Out = 1101
# time = 320, Reset = 1, CLK = 0, Out = 1101
# time = 330, Reset = 1, CLK = 1, Out = 1110
# time = 340, Reset = 1, CLK = 0, Out = 1110
# time = 350, Reset = 1, CLK = 1, Out = 1111
# time = 360, Reset = 1, CLK = 0, Out = 1111
```

Question 6

Sequential Logic Element (SLE)

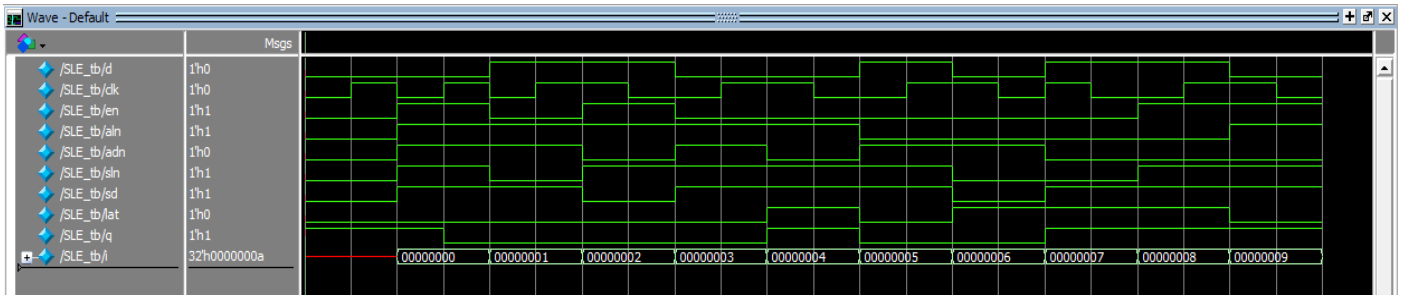
Verilog Code:

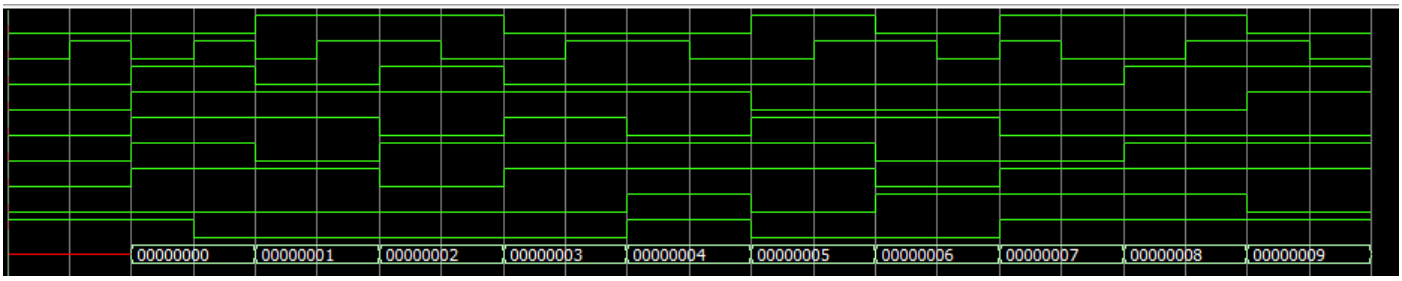
```
module SLE (D, CLK, EN, ALn, ADn, SLn, SD, LAT, Q);

input D, CLK, EN, ALn, ADn, SLn, SD, LAT;
output Q;

reg Q_FF, Q_Latch;

always @(posedge CLK or negedge ALn) begin
   if (~ALn)
      Q_FF  <=  ~ADn;
   else
   if (EN) begin
      if (! SLn)
         Q_FF  <=  SD;
      else
         Q_FF  <=  D;
   end
end

always @(*) begin
   if (~ALn)
      Q_Latch  <=  ~ADn;
   else if (CLK) begin
      if (EN) begin
         if (! SLn)
            Q_Latch  <=  SD;
         else
            Q_Latch  <=  D;
      end
   end
end

assign Q  =  (LAT)?  Q_Latch :  Q_FF;

endmodule
```

Testbench Code:

```
module SLE_tb ();

reg d, clk, en, aln, adn, sln, sd, lat;
wire q;

integer i;

SLE Dut (d, clk, en, aln, adn, sln, sd, lat, q);

initial begin
  clk = 0;
  forever
  #1 clk = ~clk;
end

initial
  begin
    d = 0;
    clk = 0;
    en = 0;
    aln = 0;
    adn = 0;
    sln = 0;
    sd = 0;
    lat = 0;
    #2;
    for (i = 0 ; i < 10 ; i = i + 1) begin
      d = $random;
      clk = $random;
      en = $random;
      aln = $random;
      adn = $random;
      sln = $random;
      sd = $random;
      lat = $random;
      #2;
    end
  $stop;
end

initial
$monitor ("d = %b, clk = %b, en = %b, aln = %b, adn = %b, sln = %b, sd = %b,
                                                    lat = %b, q = %b", d, clk, en, aln, adn, sln, sd, lat, q);

endmodule
```

Waveform:

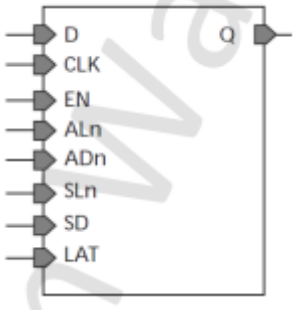





```
# d = 0, clk = 0, en = 0, aln = 0, adn = 0, sln = 0, sd = 0, lat = 0, q = 1
# d = 0, clk = 1, en = 0, aln = 0, adn = 0, sln = 0, sd = 0, lat = 0, q = 1
# d = 0, clk = 0, en = 1, aln = 1, adn = 1, sln = 1, sd = 1, lat = 0, q = 1
# d = 0, clk = 1, en = 1, aln = 1, adn = 1, sln = 1, sd = 1, lat = 0, q = 0
# d = 1, clk = 0, en = 0, aln = 1, adn = 1, sln = 0, sd = 1, lat = 0, q = 0
# d = 1, clk = 1, en = 0, aln = 1, adn = 1, sln = 0, sd = 1, lat = 0, q = 0
# d = 1, clk = 1, en = 1, aln = 1, adn = 0, sln = 1, sd = 0, lat = 0, q = 0
# d = 1, clk = 0, en = 1, aln = 1, adn = 0, sln = 1, sd = 0, lat = 0, q = 0
# d = 0, clk = 0, en = 0, aln = 1, adn = 1, sln = 1, sd = 1, lat = 0, q = 0
# d = 0, clk = 1, en = 0, aln = 1, adn = 1, sln = 1, sd = 1, lat = 0, q = 0
# d = 0, clk = 1, en = 0, aln = 0, adn = 1, sln = 1, sd = 1, lat = 1, q = 1
# d = 0, clk = 0, en = 0, aln = 1, adn = 0, sln = 1, sd = 1, lat = 1, q = 1
# d = 1, clk = 0, en = 0, aln = 0, adn = 1, sln = 1, sd = 1, lat = 0, q = 0
# d = 1, clk = 1, en = 0, aln = 0, adn = 1, sln = 1, sd = 1, lat = 0, q = 0
# d = 0, clk = 1, en = 0, aln = 0, adn = 1, sln = 0, sd = 0, lat = 1, q = 0
# d = 0, clk = 0, en = 0, aln = 0, adn = 1, sln = 0, sd = 0, lat = 1, q = 0
# d = 1, clk = 1, en = 0, aln = 0, adn = 0, sln = 0, sd = 1, lat = 1, q = 1
# d = 1, clk = 0, en = 0, aln = 0, adn = 0, sln = 0, sd = 1, lat = 1, q = 1
# d = 1, clk = 0, en = 1, aln = 0, adn = 0, sln = 1, sd = 1, lat = 1, q = 1
# d = 1, clk = 1, en = 1, aln = 0, adn = 0, sln = 1, sd = 1, lat = 1, q = 1
# d = 0, clk = 1, en = 1, aln = 1, adn = 0, sln = 1, sd = 1, lat = 0, q = 1
# d = 0, clk = 0, en = 1, aln = 1, adn = 0, sln = 1, sd = 1, lat = 0, q = 1
```

Input		Output
Name	**Function**	
D	Data	
CLK	Clock	
EN	Enable	
ALn	Asynchronous Load (Active Low)	Q
ADn*	Asynchronous Data (Active Low)	
SLn	Synchronous Load (Active Low)	
SD*	Synchronous Data	
LAT*	Latch Enable	

***Note**: ADn, SD and LAT are static signals defined at design time and need to be tied to 0 or 1.



Truth Table

ALn	ADn	LAT	CLK	EN	SLn	SD	D	Q_{n+1}
0	ADn	X	X	X	X	X	X	!ADn
1	X	0	Not rising	X	X	X	X	Qn
1	X	0	↑	0	X	X	X	Qn
1	X	0	↑	1	0	SD	X	SD
1	X	0	↑	1	1	X	D	D
1	X	1	0	X	X	X	X	Qn
1	X	1	1	0	X	X	X	Qn
1	X	1	1	1	0	SD	X	SD
1	X	1	1	1	1	X	D	D