컴퓨터공학실험 2 13 주차 결과보고서

전공: 컴퓨터공학과 학년: 2 학번: 20191559 이름: 강상원

1. 4-bit Shift Register의 결과 및 Simulation 과정에 대해서 설명하시오. (verliog source, 출력 예시, 과정 상세히 적을 것!)

Shift Register는 단일 데이터가 register를 거쳐 다수의 출력으로 나타내지는 구조이다. 데이터가 입력되면 clock이 변화할 때까지 기다렸다가 rising edge(descending edge)가 관측되면 데이터를 q0으로 출력한다. 두 번째 clock에서 새로운 입력이 q0으로 출력되고 q0에 저장되어 있던 데이터는 q1으로 출력된다. 따라서 clock이 발생하지 않는 동안에는 입력되었던 데이터를 유지시켜 준다.

대표적인 경우의 4-bit Shift register의 진리표는 다음과 같다.

	OUTPUTS			
	P0	P1	P2	P3
0	L	L	L	L
1	H	L	L	L
2	L	H	L	L
3	L	L	H	L
4	L	L	L	H
5	L	L	L	L

4-bit Shift Register는 다음과 같이 표현될 수 있다.

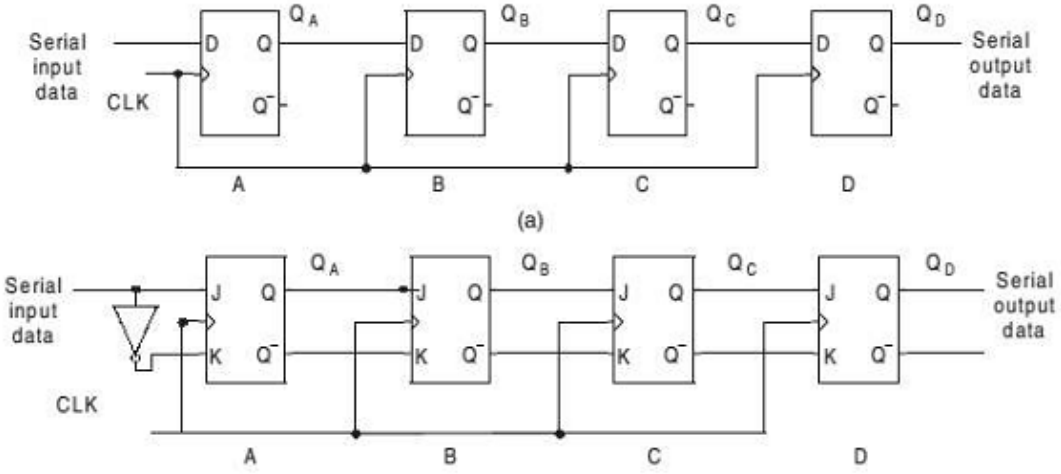


Figure of Shift-right register (a) using D flip-flops, (b) using J-K flip-flops.

이 중 JK Flip-Flop을 이용한 구현을 하고자 한다. 보이는 바와 같은 구조를 코드로 작성한다.

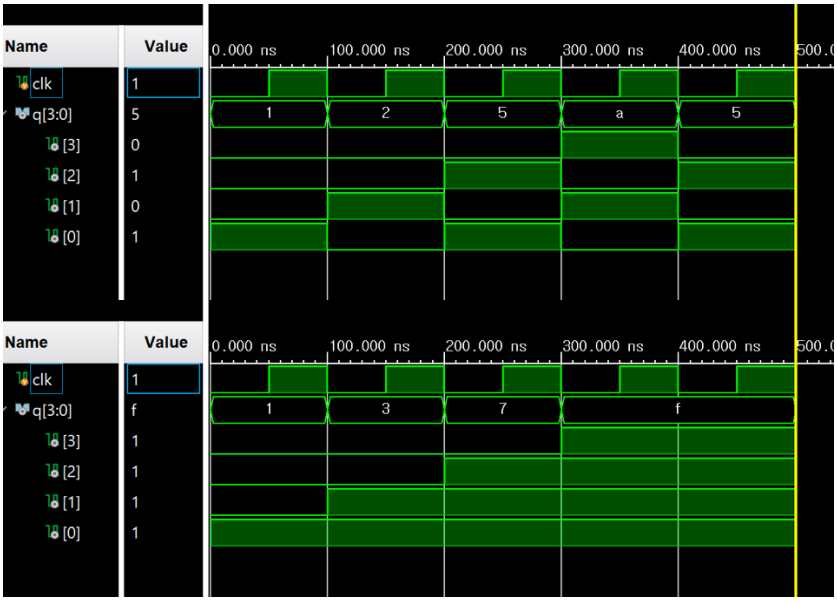
Design Source는 다음과 같다.

```
`timescale 1ns / 1ps

module four_shr(
    input clk,
    output wire [3:0] q
    );

    wire[3:0] qc;
    wire[3:0] tmpq;
    wire[6:0] tmpwire;
    jk_ff b1(1'b1,1'b0,clk,q[0], qc[0]);
    jk_ff b2(q[0], qc[0], clk, q[1], qc[1]);
    jk_ff b3(q[1], qc[1], clk, q[2], qc[2]);
    jk_ff b4(q[2], qc[2], clk, q[3], qc[3]);
  endmodule

module jk_ff(
    input j, k, clk,
    output reg q, qc
);
    initial begin
        q=0;
        qc=1;
    end
    always@(negedge clk)begin
        if(j==0&&k==0) begin
            q<=q;
            qc<=qc;
        end
        if(j==0&&k==1) begin
            q=0;
            qc=1;
        end
        if(j==1&&k==0) begin
            q=1;
            qc=0;
        end
        if(j==1&&k==1)begin
            q<=qc;
            qc<=q;
        end
    end
endmodule
```

JK Flip-Flop 4개가 연속으로 연결된 모양으로서 clk(clock) 값이 변화할 때 shift가 일어나게 된다. 이에 따른 testbench code는 다음과 같다.

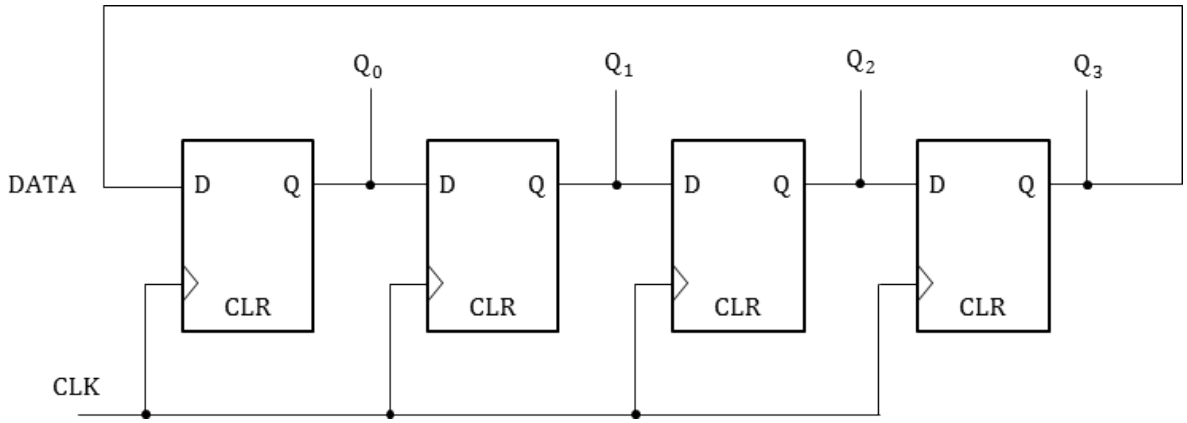
```
1  `timescale 1ns / 1ps
2
3  module four_shr_tb;
4  reg clk; wire [3:0] q;
5
6  //four_shr connect(clk, q);
7  four_shr u_four_shr(
8      .clk(clk),
9      .q(q)
10 );
11
12
13 initial clk = 1'b0;
14
15 always clk = #50 ~clk;
16
17 initial begin
18     #500
19     $finish;
20 end
21
22 endmodule
```

Design source, testbench code에 따른 시뮬레이션 결과는 다음과 같다.



2. 4-bit Ring Register의 결과 및 Simulation 과정에 대해서 설명하시오.
(verliog source, 출력 예시, 과정 상세히 적을 것!)

Ring Counter는 입력되었던 데이터가 전체적으로 회전하는 구조의 shift register이다. 따라서 다음 그림과 같이 마지막 Flip-Flop의 출력이 다시 맨 처음 Flip-Flop의 입력으로 연결되어 있다.



4-bit Ring Counter의 대표적인 형태의 진리표는 다음과 같다.

	OUTPUTS			
	P0	P1	P2	P3
0	H	L	L	L
1	L	H	L	L
2	L	L	H	L
3	L	L	L	H
4	H	L	L	L
5	L	H	L	L
6	L	L	H	L
7	L	L	L	H

Design Source는 다음과 같다. (다음 쪽)

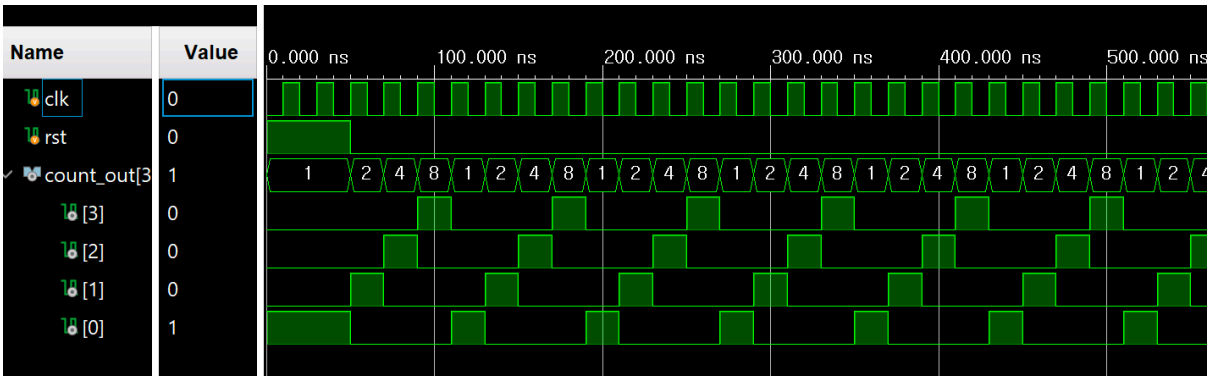
```
`timescale 1ns / 1ps

module ring_counter(
    clk,
    rst,
    count_out
    );
    input clk;
    input rst;
    output [3:0] count_out;
    reg [3:0] count_temp;

    always @(posedge(clk),rst)
    begin
        if(rst == 1'b1)   begin
            count_temp = 4'b0001;
            end
        else if(clk == 1'b1)  begin
            count_temp = {count_temp[2:0],count_temp[3]};
            end
    end
    assign count_out = count_temp;
endmodule
```

이에 따른 test bench code는 다음과 같다.

```
`timescale 1ns / 1ps

module tb_ring;
    reg clk, rst;
    wire [3:0] count_out;
    ring_counter uut (
        .clk(clk),
        .rst(rst),
        .count_out(count_out)
    );
    initial clk = 0;
    always #10 clk = ~clk;
    initial begin
        rst = 1;
        #50;
        rst = 0;
    end
endmodule

```

clk(clock)을 10마다 전환하고, rst(reset)은 처음부터 50까지 신호를 주어 보았다.

시뮬레이션 결과는 아래와 같다. (다음 쪽)



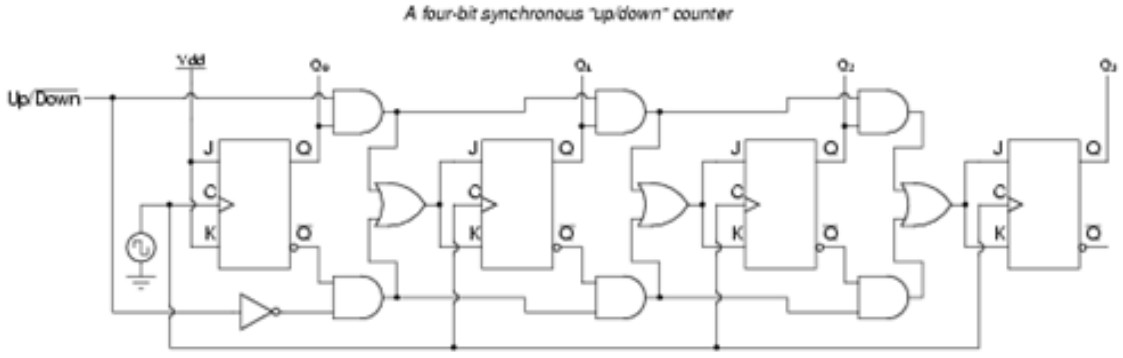
작성하였던 진리표와 동일하게 나옴을 확인할 수 있다.

3. 4-bit Up/Down counter의 결과 및 Simulation 과정에 대해서 설명하시오.
(verliog source, 출력 예시, 과정 상세히 적을 것!)

Up/Down counter는 입력값에 따라 up, down을 수행한다. 일례로 1이 입력되었을 때 up, 0이 입력되었을 때 down이 수행된다. 진리표 예시는 다음과 같다.

BEFORE				Up/DOWN	AFTER			
P0	P1	P2	P3		P0’	P1’	P2’	P3’
0	0	0	0	1	0	0	0	1
0	0	0	1	1	0	0	1	0
0	0	1	0	1	0	0	1	1
0	0	1	1	1	0	1	0	0
…								
1	1	1	1	0	0	1	1	1
0	1	1	1	0	0	1	1	0
0	1	1	0	0	0	1	0	1
…								

4-bit up/Down counter는 아래와 같은 구조로 나타낼 수 있다.



4-bit up/Down counter의 Design source는 다음과 같다.

```
`timescale 1ns / 1ps

module upordown_counter(
    clk,
    rst,
    UpOrDown,
    cnt
    );

    input clk,rst,UpOrDown;
    output [3 : 0] cnt;
    reg [3 : 0] cnt = 0;
     always @(posedge(clk) or posedge(rst))
     begin
        if(rst == 1)
            cnt <= 0;
        else
            if(UpOrDown == 1)
                if(cnt == 15)
                    cnt <= 0;
                else
                    cnt <= cnt + 1;
            else
                if(cnt == 0)
                    cnt <= 15;
                else
                    cnt <= cnt - 1;
     end
endmodule
```

이에 대응하는 Test Bench 코드는 아래와 같다. (다음 쪽)

```
`timescale 1ns / 1ps

module tb_counter;
    reg clk, rst, UpOrDown;
    wire [3:0] cnt;

    upordown_counter u_upordown_counter (
        .clk(clk),
        .rst(rst),
        .UpOrDown(UpOrDown),
        .cnt(cnt)
    );

    initial clk = 0;
    always #5 clk = ~clk;

    initial begin
        rst = 0;
        UpOrDown = 0;
        #300;
        UpOrDown = 1;
      #300;
        rst = 1;
        UpOrDown = 0;
        #100;
          rst = 0;
     end
endmodule
```

300 전까지는 up/down 신호로 0(up)을 주고, 그 이후에는 1(down)을 준다.

따라서 1111에서 시작해 1110, 1101,… 감소로 반복하다 300 이후부터는 0001, 0010, 0011…증가(Up) counter로 전환될 것이다.

위 코드들로 시뮬레이션을 시행하면 다음과 같은 결과를 얻는다.