컴퓨터공학실험 2 11 주차 결과보고서

전공: 컴퓨터공학과 학년: 2 학번: 20191559 이름: 강상원

1. RS Flip-Flop 의 결과 및 Simulation 과정에 대해서 설명하시오.
(verliog source, 출력 예시, 과정 상세히 적을 것!)

RS(SR) 플립-플롭의 동작은 아래의 상태도와 특성표로 확인할 수 있다.

Table 5.5 *SR* flip flop behavioral tables.

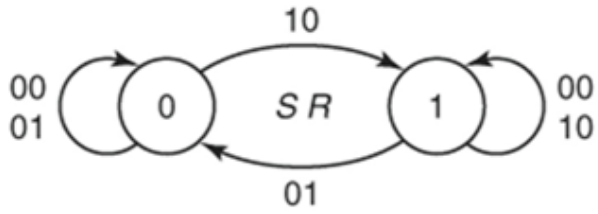
S	*R*	*q*	*q**
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	—
1	1	1	—

not allowed

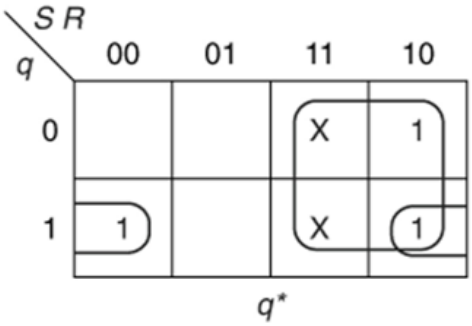
S	*R*	*q**
0	0	*q*
0	1	0
1	0	1
1	1	—

not allowed

Figure 5.15 *SR* flip flop state diagram.



Map 5.1 *SR* flip flop behavioral map.



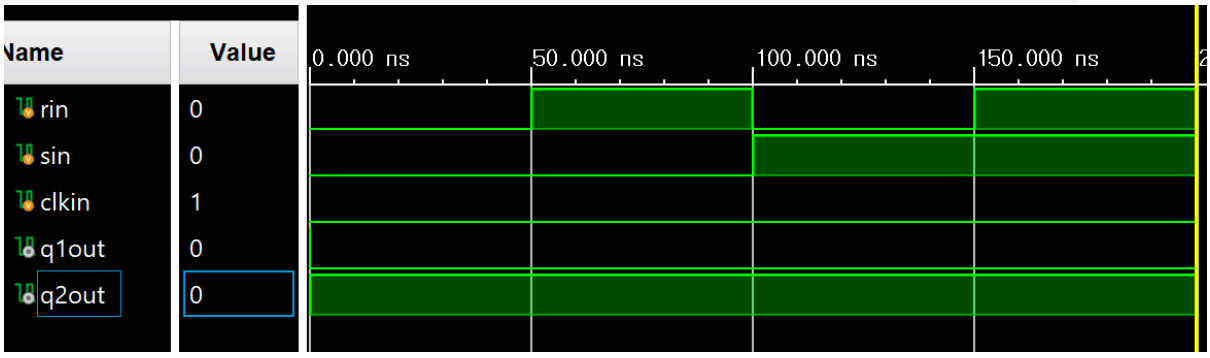
위와 같은 카르노 맵으로 정리하면, q* = S + R'q와 같이 나타낼 수 있다.

다음은 이를 Verilog로 나타낸 것이다. (q* = outq2, q = outq1, S = ins)

```
`timescale 1ns / 1ps

module inv(
    input inr, ins, inclk,
    output outq1, outq2  );

    assign outq1 = ((~ins & inclk) & outq2);
    assign outq2 = ~((~(inr & inclk)) & outq1);

endmodule

```

이를 아래의 시뮬레이션 코드로 시뮬레이션을 실행해 본다.

```
`timescale 1ns / 1ps

module inv_sim;
reg rin, sin, clkin;
wire q1out, q2out;

inv inv_sim(.inr(rin), .ins(sin), .inclk(clkin), .outq1(q1out), .outq2(q2out));

initial rin = 1'b0;
initial sin = 1'b0;
initial clkin = 1'b0;

always rin = #50 ~rin;
always sin = #100 ~sin;
always clkin = #200 ~clkin;

initial begin
    #800
    $finish;
end
endmodule

```



시뮬레이션 결과, 앞서 서술한 진리표와 같은 결과가 나옴을 확인할 수 있다.

Constraints (.xdc) 파일을 이용해 각 변수를 할당하고 FPGA에 업로드할 수 있었다. 다음은 RS Flip-Flop의 Constraints 파일이다.

```
set_property IOSTANDARD LVCMOS18 [get_ports {inr}]
set_property IOSTANDARD LVCMOS18 [get_ports {ins}]
set_property IOSTANDARD LVCMOS18 [get_ports {inclk}]
set_property IOSTANDARD LVCMOS18 [get_ports {outq1}]
set_property IOSTANDARD LVCMOS18 [get_ports {outq2}]
set_property PACKAGE_PIN W10 [get_ports {inr}]
set_property PACKAGE_PIN Y11 [get_ports {ins}]
set_property PACKAGE_PIN Y12 [get_ports {inclk}]
set_property PACKAGE_PIN W11 [get_ports {outq1}]
set_property PACKAGE_PIN W12 [get_ports {outq2}]
set_property ALLOW_COMBINATORIAL_LOOPS TRUE [get_nets {outq1}]
set_property ALLOW_COMBINATORIAL_LOOPS TRUE [get_nets {outq1_OBUF}]
set_property ALLOW_COMBINATORIAL_LOOPS TRUE [get_nets {outq2}]
set_property ALLOW_COMBINATORIAL_LOOPS TRUE [get_nets {outq2_OBUF}]
```

2. **D Flip-Flop 의 결과 및 Simulation 과정에 대해서 설명하시오.**
   (verliog source, 출력 예시, 과정 상세히 적을 것!)

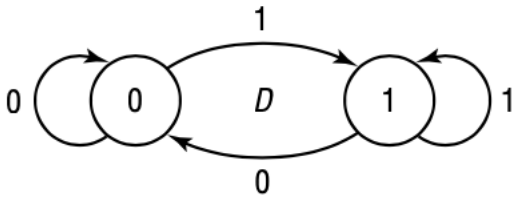
D 플립-플롭의 동작은 아래의 상태도와 특성표로 확인할 수 있다.

Table 6.3 The *D* flip flop behavioral tables.

D	q	$q^\star$
0	0	0
0	1	0
1	0	1
1	1	1

D	$q^\star$
0	0
1	1

Figure 6.9 *D* flip flop state diagram.



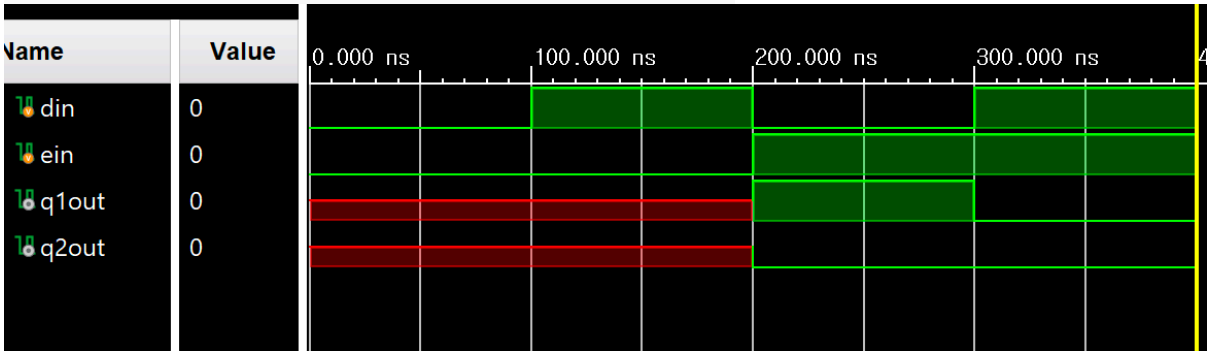
정리하면, q* = D 와 같이 나타낼 수 있다.

다음은 이를 Verilog로 나타낸 것이다.

```
`timescale 1ns / 1ps

module inv(
    input ind, ine,
    output outq1, outq2  );

    assign outq1 = ((ine & (~ind)) | outq2);
    assign outq2 = ~((ine & ind) | outq1);

endmodule
```

이를 아래의 시뮬레이션 코드로 시뮬레이션을 실행해 본다.

```
1  `timescale 1ns / 1ps
2
3  module inv_sim;
4  reg din, ein;
5  wire q1out, q2out;
6
7  inv inv_sim(.ind(din), .ine(ein), .outq1(q1out), .outq2(q2out));
8
9  initial
10 begin
11 din = 1'b0;
12 ein = 1'b0;
13
14 end
15 always@(din or ein)begin
16 din<=#100 ~din;
17 ein<=#200 ~ein;
18 end
19 endmodule
20
```



시뮬레이션 결과, 앞서 서술한 진리표와 같은 결과가 나옴을 확인할 수 있다.

(처음 값은 이전 값이 없으므로 미정의 값이 나온다.)

Constraints (.xdc) 파일을 이용해 각 변수를 할당하고 FPGA에 업로드할 수 있었다. 다음은 D Flip-Flop의 Constraints 파일이다.

```
1  set_property IOSTANDARD LVCMOS18 [get_ports {ind}]
2  set_property IOSTANDARD LVCMOS18 [get_ports {ine}]
3  set_property IOSTANDARD LVCMOS18 [get_ports {outq1}]
4  set_property IOSTANDARD LVCMOS18 [get_ports {outq2}]
5  set_property PACKAGE_PIN J4 [get_ports {ind}]
6  set_property PACKAGE_PIN L3 [get_ports {ine}]
7  set_property PACKAGE_PIN F15 [get_ports {outq1}]
8  set_property PACKAGE_PIN F13 [get_ports {outq2}]
9  set_property ALLOW_COMBINATORIAL_LOOPS TRUE [get_nets {outq1}]
10 set_property ALLOW_COMBINATORIAL_LOOPS TRUE [get_nets {outq1_OBUF}]
11 set_property ALLOW_COMBINATORIAL_LOOPS TRUE [get_nets {outq2}]
12 set_property ALLOW_COMBINATORIAL_LOOPS TRUE [get_nets {outq2_OBUF}]
```

3. **JK Flip-Flop 의 결과 및 Simulation 과정에 대해서 설명하시오. (verliog source, 출력 예시, 과정 상세히 적을 것!)**

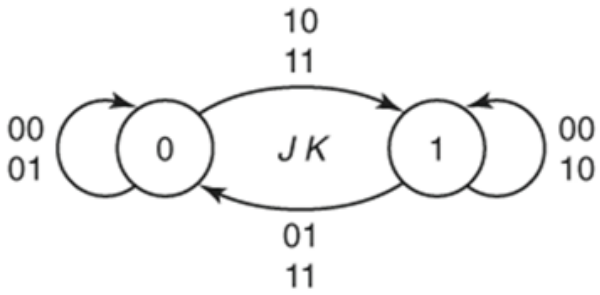
JK 플립-플롭의 동작은 아래의 상태도와 특성표로 확인할 수 있다.

Table 5.7 *JK* flip flop behavioral tables.

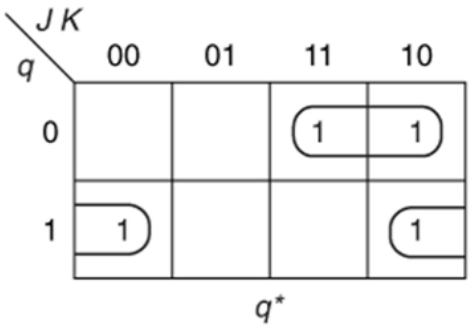
J	*K*	*q*	*q**
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

J	*K*	*q**
0	0	*q*
0	1	0
1	0	1
1	1	*q′*

Figure 5.19 *JK* flip flop state diagram.



Map 5.2 *JK* flip flop behavioral map.



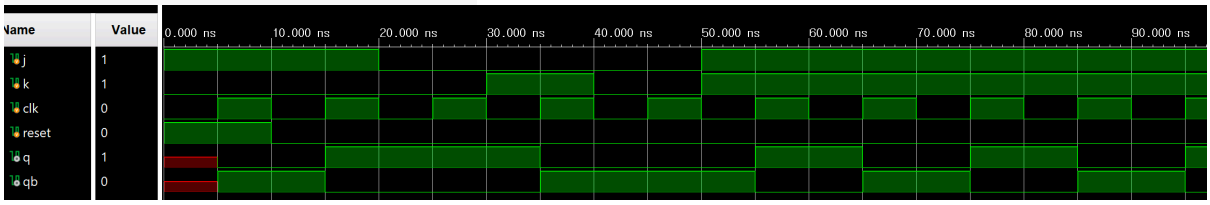
위와 같은 카르노 맵으로 정리하면, q* = Jq' + K'q 와 같이 나타낼 수 있다.

다음은 이를 Verilog로 나타낸 것이다. (q* = outq2, q = outq1, clock = clk)

```
`timescale 1ns / 1ps

module jkffD(q, qb, j, k, clk, reset);
output q, qb;
input j, k, clk, reset;
reg q;
assign qb = ~q;
always @(posedge clk) begin
    if (reset)
        q <= 0;
    else if (k==0 && j==0)
        q <= q;
    else if (k==0 && j==1)
        q <= 1;
    else if (k==1 && j==0)
        q <= 0;
    else
        q <= ~q;
    end
endmodule
```

이를 아래의 시뮬레이션 코드로 시뮬레이션을 실행해 본다.

```
`timescale 1ns / 1ps
module jkfffD_tb();
reg j, k, clk, reset;
wire q, qb;
jkffD con(q, qb, j, k, clk, reset);
initial begin
clk = 0; reset = 1; j = 1; k = 0;
reset = #10 0;
j = #10 0;
k = #10 1;
k = #10 0;
j = #10 1; k = 1;
end;
always clk = #5 ~clk;
endmodule
```



시뮬레이션 결과, 앞서 서술한 진리표와 같은 결과가 나옴을 확인할 수 있다.

(처음 값은 이전 값이 없으므로 미정의 값이 나온다.)

Constraints (.xdc) 파일을 이용해 각 변수를 할당하고 FPGA에 업로드할 수 있었다. 다음은 JK Flip-Flop의 Constraints 파일이다.

```
set_property IOSTANDARD LVCMOS18 [get_ports {j}]
set_property IOSTANDARD LVCMOS18 [get_ports {k}]
set_property IOSTANDARD LVCMOS18 [get_ports {clk}]
set_property IOSTANDARD LVCMOS18 [get_ports {reset}]
set_property IOSTANDARD LVCMOS18 [get_ports {q}]
set_property IOSTANDARD LVCMOS18 [get_ports {qb}]
set_property PACKAGE_PIN J4 [get_ports {inj}]
set_property PACKAGE_PIN L3 [get_ports {ink}]
set_property PACKAGE_PIN K3 [get_ports {clk}]
set_property PACKAGE_PIN M2 [get_ports {reset}]
set_property PACKAGE_PIN F15 [get_ports {q}]
set_property PACKAGE_PIN F13 [get_ports {qb}]
```