12주차 결과보고서

컴퓨터공학과
20191621 이민영

1. 2-bit counter의 결과 및 Simulation 과정에 대해서 설명하시오.
(Verilog source, 출력 예시, 과정 상세히 적을 것)

Two_bit_counter.v

```
`timescale 1ns / 1ps
module two_bit_counter(
    input clk, reset,j,k,
    output wire [1:0] q
    );
    wire[1:0] qc;
    wire[1:0] tmpq;
    jk_ff b1(j,k,clk,tmpq[0],qc[0]);
    jk_ff b2(tmpq[0],tmpq[0],clk,tmpq[1],qc[1]);

    and(q[0],~reset,tmpq[0]);
    and(q[1],~reset,tmpq[1]);
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
```

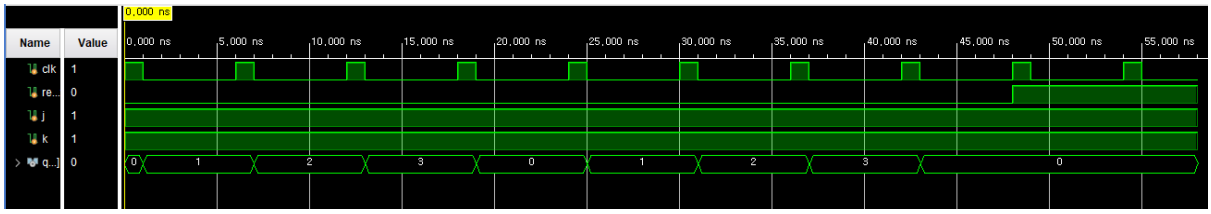
```
endmodule
```

Two_bit_counter_tb.v

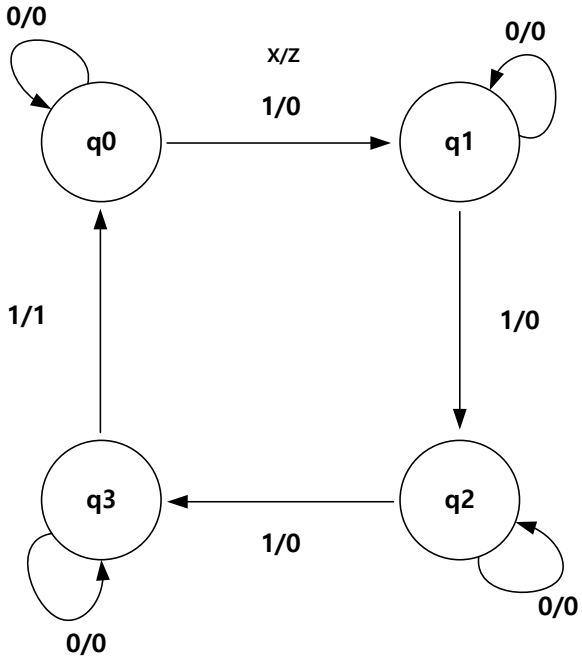
```
`timescale 1ns / 1ps
module two_bit_counter_tb;
reg clk;
reg reset,j,k;
wire [1:0] q;
two_bit_counter func(
.clk(clk),
.j(j),
.k(k),
.reset(reset),
.q(q)
);
initial begin
	clk=1'b1;
	reset=1'b0;
	j=1'b1;
	k=1'b1;
end
always begin
	clk=#1~clk;
	clk=#5~clk;
end
initial begin
#48
reset=~reset;
#10
$finish;
end
endmodule
```

Two_bit_counter_c.xdc

```
set_property IOSTANDARD LVCMOS18 [get_ports clk]
set_property IOSTANDARD LVCMOS18 [get_ports reset]
set_property IOSTANDARD LVCMOS18 [get_ports j]
set_property IOSTANDARD LVCMOS18 [get_ports k]
set_property IOSTANDARD LVCMOS18 [get_ports q[1]]
set_property IOSTANDARD LVCMOS18 [get_ports q[0]]
set_property PACKAGE_PIN J4 [get_ports clk]
set_property PACKAGE_PIN L3 [get_ports reset]
set_property PACKAGE_PIN K3 [get_ports j]
set_property PACKAGE_PIN M2 [get_ports k]
set_property PACKAGE_PIN F15 [get_ports q[1]]
set_property PACKAGE_PIN F13 [get_ports q[0]]


set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets {clk_IBUF}]
```



현재 상태	다음 상태		Output (Z)		D1 D2	
	X=0	X=1	X=0	X=1	X=0	X=1
00	00	01	0	0	00	01
01	01	10	0	0	01	10
10	10	11	0	0	10	11
11	11	00	0	1	11	00



2bit 2진 Counter는 q0에서 시작해서 X가 0일 때 현재상태로 다시 돌아오며, X가 1일 때는 q1으로 간다. 다른 경우에서도 X가 0일 경우에는 자기 자신으로 돌아오게 되고 Z는 0으로 설정된다. X가 1인 경우에는 00->01->10->11->00 과 같이 다음 상태로 넘어가게 되고 11에서 00으로 변화되는 것은 reset되는 상태이므로 Z가 1로 설정되게 된다. 이를 시뮬레이션을 통해서도 볼 수 있다.

Q의 값이 0,1,2,3,0,1,2,3, 을 반복해서 이어지는 것을 알 수 있다. 이때 reset이 1로 설정되었을 경우에는 q의 값이 변화하지 않고 0으로만 지속되는 것을 알 수 있다.
2bit counter는 JK 플립플롭을 이용해서 구성된다.

Xdc 코드를 살펴보면 clk,reset,j,k가 각각 J4, L3, K3, M2 핀에 연결되어 있으며 q[1],q[0]이 F15,F13에 연결되어서 결과값을 살펴볼 수 있었다.

2. 4-bit decade counter의 결과 및 Simulation 과정에 대해서 설명하시오.
(Verilog source, 출력 예시, 과정 상세히 적을 것)

Four_bit_decade_counter.v

```
`timescale 1ns / 1ps

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

    wire[3:0] qc;
    wire[3:0] tmpq;
    wire[6:0] tmpwire;
    jk_ff b1(1'b1,1'b1,clk,tmpq[0], qc[0]);
    and(tmpwire[0], tmpq[0], qc[3]);
    jk_ff b2(tmpwire[0], tmpwire[0], clk, tmpq[1], qc[1]);
    and(tmpwire[1], tmpq[0], tmpq[1]);
    jk_ff b3(tmpwire[1], tmpwire[1], clk, tmpq[2], qc[2]);
    and(tmpwire[2], tmpwire[1], tmpq[2]);
    and(tmpwire[3], tmpq[0], tmpq[3]);
    or(tmpwire[4], tmpwire[2], tmpwire[3]);
    jk_ff b4(tmpwire[4], tmpwire[4], clk, tmpq[3], qc[3]);

    and(q[0], ~reset, tmpq[0]);
    and(q[1], ~reset, tmpq[1]);
    and(q[2], ~reset, tmpq[2]);
    and(q[3], ~reset, tmpq[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
```

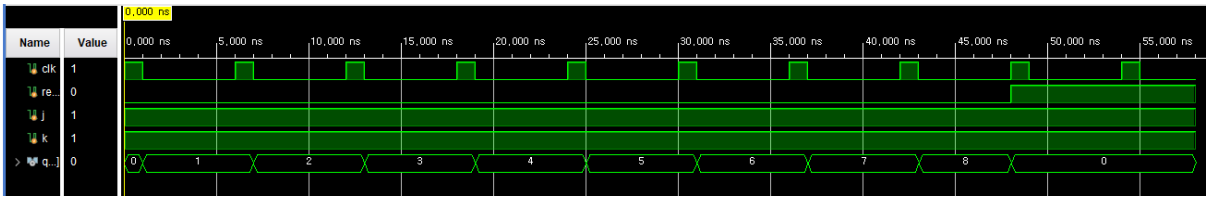
Four_bit_decade_counter_tb.v

```
    `timescale 1ns / 1ps
    module four_bit_decade_counter_tb;
    reg clk;
    reg reset;
    reg j;
    reg k;
    wire [3:0] q;
    four_bit_decade_counter func(
    .clk(clk),
    .reset(reset),
    .q(q)
    );
    initial begin
        clk=1'b1;
        reset=1'b0;
        j=1'b1;
        k=1'b1;
    end
    always begin
        clk=#1~clk;
        clk=#5~clk;
    end
    initial begin
    #48
    reset=~reset;
    #10
    $finish;
    end
endmodule
```

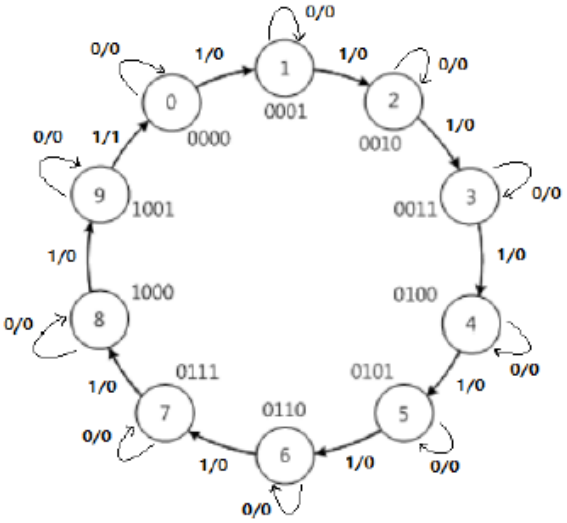
Four_bit_decade_counter_c.xdc

```
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[0]]
set_property IOSTANDARD LVCMOS18 [get_ports q[1]]
set_property IOSTANDARD LVCMOS18 [get_ports q[2]]
set_property IOSTANDARD LVCMOS18 [get_ports q[3]]

set_property PACKAGE_PIN J4 [get_ports j]
set_property PACKAGE_PIN L3 [get_ports k]
set_property PACKAGE_PIN K3 [get_ports clk]
set_property PACKAGE_PIN M2 [get_ports reset]

set_property PACKAGE_PIN F15 [get_ports q[3]]
set_property PACKAGE_PIN F13 [get_ports q[2]]
set_property PACKAGE_PIN F14 [get_ports q[1]]
set_property PACKAGE_PIN F16 [get_ports q[0]]

set_property ALLOW_COMBINATORIAL_LOOPS TRUE [get_nets {q[1]}]
set_property ALLOW_COMBINATORIAL_LOOPS TRUE [get_nets {q[1]_OBUF}]
set_property ALLOW_COMBINATORIAL_LOOPS TRUE [get_nets {q[0]}]
set_property ALLOW_COMBINATORIAL_LOOPS TRUE [get_nets {q[0]_OBUF}]
set_property ALLOW_COMBINATORIAL_LOOPS TRUE [get_nets {q[3]}]
set_property ALLOW_COMBINATORIAL_LOOPS TRUE [get_nets {q[3]_OBUF}]
set_property ALLOW_COMBINATORIAL_LOOPS TRUE [get_nets {q[2]}]
set_property ALLOW_COMBINATORIAL_LOOPS TRUE [get_nets {q[2]_OBUF}]
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets {clk_IBUF}]
set_property SEVERITY {Warning}   [get_drc_checks LUTLP-1]
set_property SEVERITY {Warning} [get_drc_checks NSTD-1]
```



현재상태	다음 상태		Output		D1 D2 D3 D4	
	X=0	X=1	X=0	X=1	X=0	X=1
0000	0000	0001	0	0	0000	0001
0001	0001	0010	0	0	0001	0010
0010	0010	0011	0	0	0010	0011
0011	0011	0100	0	0	0011	0100
0100	0100	0101	0	0	0100	0101
0101	0101	0110	0	0	0101	0110
0110	0110	0111	0	0	0110	0111
0111	0111	1000	0	0	0111	1000
1000	1000	1001	0	0	1000	1001
1001	1001	0000	0	1	1001	0000



상태표와 상태도를 보면 Input이 1이 될 때 값이 1씩 증가한다는 것을 알 수 있다. 또한 상태가 1001에서 다시 0000으로 돌아갈 때 output이 1로 설정된다는 것을 알 수 있다.
X가 0일 때는 다음상태에도 동일한 값을 유지한다. 즉 0000이고 X=0일 때 0000으로 동일한 값이 된다. X가 1일 때는 1씩 증가한 값으로 다음상태가 넘어가게 된다. 즉 0000은 0001로, 0001은 0010으로 등 0000->0001->0010->0011->0100->0101->0110->0111->1000->1001->0000 순으로 변화되게 된다. 이 때 다시 1001에서 0000으로 돌아오게 되면 Output이 1로 설정되게 된다. 즉 output은 X가 0일때는 모두 0으로 설정되며 X가 1일때는 1001이 0000으로 바뀔 때 1, 나머지는 0으로 설정되게 된다.
이를 시뮬레이션과 FPGA를 통해서도 확인해볼 수 있었다. Reset은 0일 때 영향을 미치지 않으며, 1로 셋팅되게 되면 값이 더 이상 증가하지 않고 유지되게 된다.
FPGA를 통해서도 확인할 수 있었다. 값을 증가시키면서 변화를 살펴보면 값이 변화되는 것을 확인할 수 있었다.

Xdc 코드를 살펴보면 j,k,clk,reset이 각각 J4, L3, K3, M2 핀에 연결되어 있으며 q[3],q[2],q[1],q[0]이 F15,F13,F14,F16에 연결되어서 결과값을 살펴볼 수 있었다.

3. 4-bit 2421 decade counter의 결과 및 Simulation 과정에 대해서 설명하시오. (Verilog source, 출력 예시, 과정 상세히 적을 것)

Four_bit_2421.v

```
`timescale 1ns / 1ps
module four_bit_2421(
    input clk, reset,
    output wire [3:0] q
    );
    wire [3:0] tmpq;
    wire check;
    wire tmpwire;
    reg [3:0] B;
    four_bit_decade_counter fourbit(clk, reset, tmpq);
    assign check = (tmpq[3]&~tmpq[2])|(tmpq[2]&tmpq[0])|(tmpq[2]&tmpq[1]&~tmpq[0]);
    always @(check)
        if(check == 1'b1) begin
          B = 4'b0110;
        end
        else begin
          B = 4'b0000;
        end
    adder4bit add0(tmpq, B, 0, q, tmpwire);
endmodule

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

    wire[3:0] qc;
    wire[3:0] tmpq;
    wire[6:0] tmpwire;
    jk_ff b1(1'b1,1'b1,clk,tmpq[0], qc[0]);
    and(tmpwire[0], tmpq[0], qc[3]);
    jk_ff b2(tmpwire[0], tmpwire[0], clk, tmpq[1], qc[1]);
    and(tmpwire[1], tmpq[0], tmpq[1]);
    jk_ff b3(tmpwire[1], tmpwire[1], clk, tmpq[2], qc[2]);
    and(tmpwire[2], tmpwire[1], tmpq[2]);
    and(tmpwire[3], tmpq[0], tmpq[3]);
    or(tmpwire[4], tmpwire[2], tmpwire[3]);
    jk_ff b4(tmpwire[4], tmpwire[4], clk, tmpq[3], qc[3]);

    and(q[0], ~reset, tmpq[0]);
    and(q[1], ~reset, tmpq[1]);
    and(q[2], ~reset, tmpq[2]);
    and(q[3], ~reset, tmpq[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

module adder1bit(A, B, Ci, S, Co);

input A, B, Ci;
output S, Co;
assign S=A^B^Ci;
assign Co=(A&B)|((A^B)&Ci);

endmodule

module adder4bit(A, B, Ci, S, Co);
input [3:0] A, B; input Ci;
output [3:0] S; output Co;

wire [3:0] A, B, S; wire Ci, Co;
wire [2:0] C;

adder1bit add1(A[0], B[0], Ci, S[0], C[0]);
adder1bit add2(A[1], B[1], C[0], S[1], C[1]);
adder1bit add3(A[2], B[2], C[1], S[2], C[2]);
```

```
adder1bit add4(A[3], B[3], C[2], S[3], Co);


endmodule
```

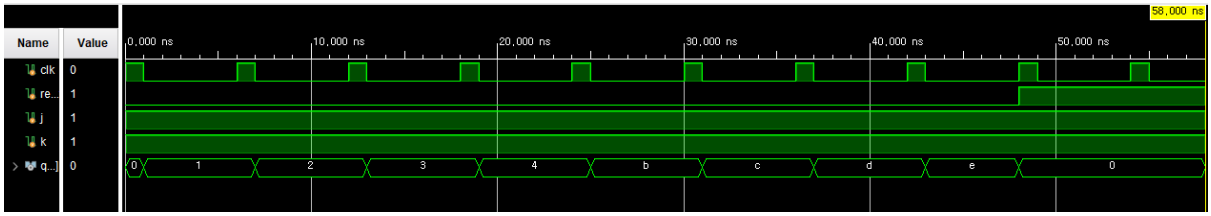
Four_bit_2421_tb.v

```
`timescale 1ns / 1ps
module four_bit_2421_tb;
reg clk;
reg reset;
reg j;
reg k;
wire [3:0] q;
four_bit_2421 func(
.clk(clk),
.reset(reset),
.q(q)
);
initial begin
	clk=1'b1;
	reset=1'b0;
	j=1'b1;
	k=1'b1;
end
always begin
	clk=#1~clk;
	clk=#5~clk;
end
initial begin
#48
reset=~reset;
#10
$finish;
end
endmodule
```

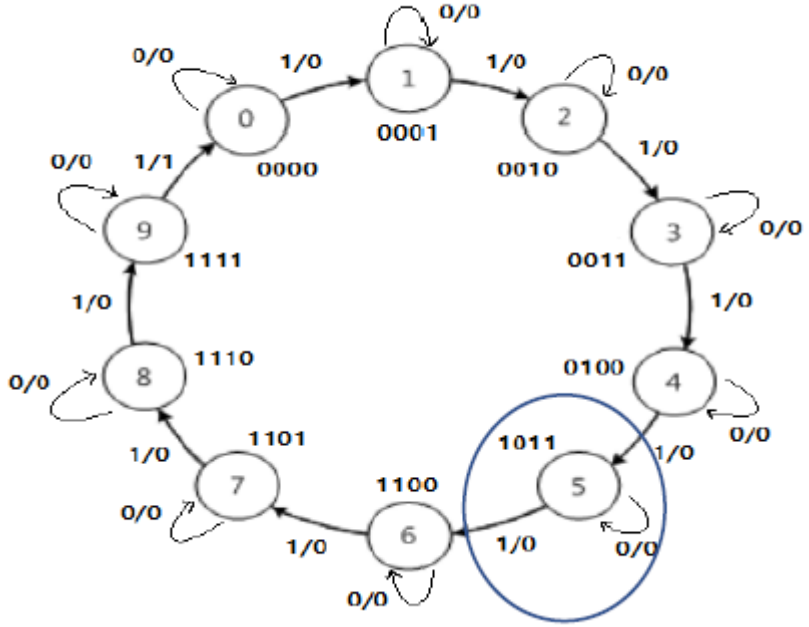
Four_bit_2421_c.xdc

```
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[0]]
set_property IOSTANDARD LVCMOS18 [get_ports q[1]]
set_property IOSTANDARD LVCMOS18 [get_ports q[2]]
set_property IOSTANDARD LVCMOS18 [get_ports q[3]]

set_property PACKAGE_PIN J4 [get_ports j]
set_property PACKAGE_PIN L3 [get_ports k]
```

```
set_property PACKAGE_PIN K3 [get_ports clk]
set_property PACKAGE_PIN M2 [get_ports reset]

set_property PACKAGE_PIN F15 [get_ports q[3]]
set_property PACKAGE_PIN F13 [get_ports q[2]]
set_property PACKAGE_PIN F14 [get_ports q[1]]
set_property PACKAGE_PIN F16 [get_ports q[0]]

set_property ALLOW_COMBINATORIAL_LOOPS TRUE [get_nets {q[1]}]
set_property ALLOW_COMBINATORIAL_LOOPS TRUE [get_nets {q[1]_OBUF}]
set_property ALLOW_COMBINATORIAL_LOOPS TRUE [get_nets {q[0]}]
set_property ALLOW_COMBINATORIAL_LOOPS TRUE [get_nets {q[0]_OBUF}]
set_property ALLOW_COMBINATORIAL_LOOPS TRUE [get_nets {q[3]}]
set_property ALLOW_COMBINATORIAL_LOOPS TRUE [get_nets {q[3]_OBUF}]
set_property ALLOW_COMBINATORIAL_LOOPS TRUE [get_nets {q[2]}]
set_property ALLOW_COMBINATORIAL_LOOPS TRUE [get_nets {q[2]_OBUF}]
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets {clk_IBUF}]
set_property SEVERITY {Warning}   [get_drc_checks LUTLP-1]
set_property SEVERITY {Warning} [get_drc_checks NSTD-1]
```



현재상태	다음 상태		Output		D1 D2 D3 D4	
	X=0	X=1	X=0	X=1	X=0	X=1
0000	0000	0001	0	0	0000	0001
0001	0001	0010	0	0	0001	0010
0010	0010	0011	0	0	0010	0011
0011	0011	0100	0	0	0011	0100
0100	0100	1011	0	0	0100	1011
0101	1011	1100	0	0	1011	1100
0110	1100	1101	0	0	1100	1101
0111	1101	1110	0	0	1101	1110
1000	1110	1111	0	0	1110	1111
1001	1111	0000	0	1	1111	0000



4bit 2421 decade counter는 일반적인 decade와 동일하게 구현되며, 5일 때 0101을 1011로 바꾸어서 1을 더해서 실행될 수 있도록 하였다.
상태표와 상태도를 보면 2421 decade counter는 0000->0001->0010->0011->0100->1011->1100->1101->1110->1111 순서대로 이동하면서 값을 변화시키게 된다. X가 0일 때는 원래 상태 값을 그대로 유지시킨다. 0000일때는 그대로 0000, 0010일때도 원래 값인 0010 그대로 값을 유지하게 된다. X가 1일 때는 위의 순서대로 1씩 증가시킨 값을 다음 상태로 가지게 된다. 이 때 2421이기 때문에 5인 값을 가질 때 0101이 아닌 1011인 값으로 변화되어 증가되게 된다.
Output의 경우, X가 0일 때는 모두 0으로 설정되며, X가 1인 경우는 1111이 0000으로 변화될 때 1로 설정되고 나머지는 0으로 설정되게 된다.

이를 시뮬레이션과 FPGA를 통해서도 확인해볼 수 있었다. Reset은 0일 때 영향을 미치지 않으며, 1로 셋팅되게 되면 값이 더 이상 증가하지 않고 유지되게 된다.
FPGA를 통해서도 확인할 수 있었다. 값을 증가시키면서 변화를 살펴보면 값이 변화되는 것을 확인할 수 있었다.

Xdc 코드를 살펴보면 j,k,clk,reset이 각각 J4, L3, K3, M2 핀에 연결되어 있으며 q[3],q[2],q[1],q[0]이 F15,F13,F14,F16에 연결되어서 결과값을 살펴볼 수 있었다.