

---

# VHDL 및 실습

Flip -Flop & Counter

---

과목	VHDL 및 실습
학과	전자공학과
학번	2011144024
이름	유대성 (오전)
제출일	
담당교수	최종성 교수님

# || FLIP - FLOP , COUNTER

## 1. 주제 및 배경이론

i 실습 주제에 대한 내용을 기술하시오

### 플립플롭(Flip-flop)

: 연산을 하기 위해서는 상태에 대한 값이 존재해야 한다. 예를 들어  $5+6$  을 연산하기 위해서는 5 와 6 이라는 값을 데이터로 가지고 있어야 결과값을 나타낼 수 있는 것이다. 이를 저장하기 위해서 저장매체를 사용하는 데 플립플롭이 이에 해당한다. 플립플롭의 종류는 다음과 같다.

- SR-F/F (Set-Reset Flop-Flop)
  - 기본적인 형태의 플립플롭으로 1 비트를 저장한다. 피드백에 의한 동작을 통해 Q 의 값을 세팅하는 동작을 하며 /SET, /RESET 이 모두 활성화 되는 조건 (0, 0)에서 나오면 안되는 조건인 Unused State 가 발생한다.
- Gated SR-F/F
  - 앞의 SR-F/F 에서 앞에 게이트를 달아 CK(Clock)가 활성화 되었을 때만 동작하도록 설계하였다. 하지만 마찬가지로 활성화 된 상태에서 같은 문제가 발생하게 된다.
- D-F/F
  - 문제가 발생하는 상황을 아예 사용하지 않도록 설계된 플립플롭. D 의 값이 곧 Q 값이 되도록 설계되어 D-F/F 이라는 이름을 갖게되었다.
- Master-Slave D-F/F
  - NOT 과 같은 게이트를 지날 때는 Time delay 가 발생한다는 것을 알 것이다. 이렇게 되면 Clock 이 동기화 되지 않아 원하지 않는 동작을 할 위험이 있다. 이를 막기위해 각 클럭에 동작을 Master, Slave 에 나누어 순차적으로 실행되게 하였다.
- Master-Slave D-F/F with Preset & Clear
  - 위의 Master-Slave D-F/F 에서 Preset 과 Clear 동작을 추가하였다. 해당 동작은 어떤 것보다 우선적으로 행해지게끔 설계 되어 Q 값을 바로 조작할 수 있다.
- Toggled Master-Slave D-F/F
  - 위의 Master-Slave D-F/F 에서 /Q 를 입력으로 다시 피드백하여 클럭 주파수를 절반으로 만드는 방식. 동작이 위의 Master-Slave D-F/F 의 동작이 두 클럭에 걸쳐 이루어지는 점에서 착안한 것으로 생각된다.
- JK-F/F
  - SR-F/F 에서 문제가 되었던 둘다 활성화 되는 조건(1, 1)의 동작을 토글로 만들고 프리셋, 클리어 동작 또한 구현할 수 있도록 한 플립플롭. Falling Edge 에서만 동작하므로 클럭 주파수는 반감되어 동작한다.

### 카운터(Counter)

: 저장된 값을 통해 2 진수의 개념을 확장할 수 있게 해주는 카운터 4, 8 진 카운터부터 클리어 동작을 통한 10 진, 2-13 진 카운터 등을 구현할 수 있다.

- Ripple Counter
  - 각 클럭의 주파수를 반감하는 방식으로  $2^n$ 을 구현할 수 있다.
- Divide-by-N Ripple Counter
  - 위의 Ripple Counter 는  $2^n$ 을 표현하지만 10 진 카운터와 같은 2 의 n 승이 성립하지 않는 카운터를 제작하기 위해서는 나머지 구간을 카운팅을 막아야한다.제외 조건을 찾아 Clear 를 통해 0 을 입력하여 초기값으로 돌릴 수 있다. 하지만 이러한 동작은 게이트를 통과함으로써 구현되므로 Time delay 가 발생되게 된다. 아주 미세하지만 이러한 시간 간격이 치명적으로 작용하는 시스템에서는 다른 방식을 사용해야 할 것이다.
- Synchronous Counter
  - 모든 클럭을 동시에 사용하여 위의 전파 지연을 개선할 수 있다. 이를 구현한 것이 Synchronous(동기) Counter. 동기식은 카르노맵을 사용해 간략화 한 뒤 회로를 구성하게 된다.

- Presetable Counter

- Divide-by-N Ripple Counter 가 Clear 을 통해 초기값을 주었다면 Presetable Counter 는 0 이 아닌 특정 값으로 시작하게끔 동작시키는 역할을 한다. 이를 통해 5-6-7 과 같은 구간을 반복하는 카운터를 개발할 수 있다.

## STD\_LOGIC 논리 체계

: 이전에 한 비트의 발생가능한 경우의 수는 2 가 아닌 8 이라고 배웠던 적이 있다. LOW, HIGH 가 아닌 동작에는 무엇이 있는 지 살펴보도록 하자.

X	Forcing Unknown (합성 가능한 unknown)
0	Forcing Low (합성 가능한 low)
1	Forcing High (합성 가능한 high)
Z	High Impedance (합성 가능한 3 상 버퍼) - <b>입력 또한 받지 않음</b>
W	Weak unknown (합성 불가의 약한 신호)
L	Weak Low (합성 불가의 약한 신호)
H	Weak high (합성 불가의 약한 신호)
-	Don't care

실제로 합성 테이블을 보면 이해가 더 쉬울 것이다.

	H	0	1	Z	W	L	H	-
H	X	X	X	X	X	X	X	X
0	X	0	X	0	0	0	0	X
1	X	X	1	1	1	1	1	X
Z	X	0	1	Z	W	L	H	X
W	X	0	1	W	W	W	W	X
L	X	0	1	L	W	L	W	X
H	X	0	1	H	W	W	H	X
-	X	X	X	X	X	X	X	X

## 2. 소스코드 및 코드 설명 (2, 3 단계 함께 기술)

**i** 소스코드를 설명과 함께 기술하시오.

## 3. 시뮬레이션 결과 및 설명 (2, 3 단계 함께 기술)

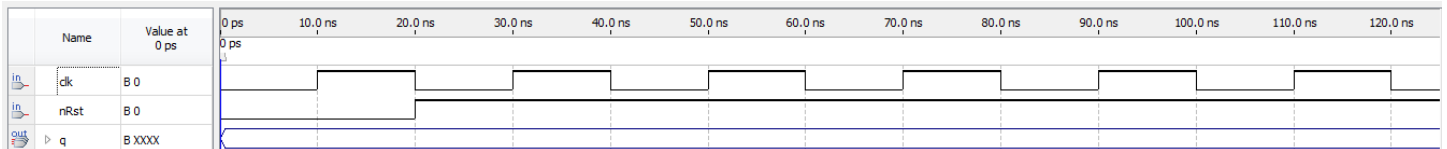
**i** 위 코드를 시뮬레이션 하고 그에 대한 결과를 작성하시오.

# 16진 Counter

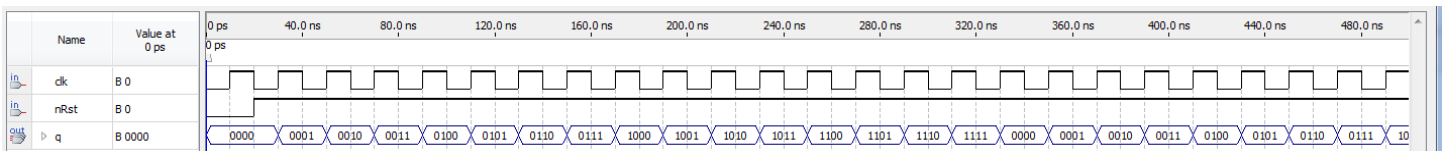
## 1. VHDL 작성

```
1 library ieee;
2
3 use ieee.std_logic_1164.all;
4 use ieee.std_logic_arith.all;
5 use ieee.std_logic_unsigned.all;
6
7 entity counter_16 is
8
9 port(
10     nRst : in std_logic;
11     clk : in std_logic;
12     q : out std_logic_vector(3 downto 0)
13 );
14 end counter_16;
15
16 architecture BEH of counter_16 is
17
18     signal cnt : std_logic_vector(3 downto 0);
19
20 begin
21     process(nRst, clk)
22     begin
23         if(nRst = '0') then
24             cnt <= (others => '0');
25         elsif rising_edge(clk) then
26             cnt <= cnt + 1;
27         end if;
28     end process;
29
30     q <= cnt;
31
32 end BEH;
```

## 2. VMF 파일 생성 및 입력레벨 설정

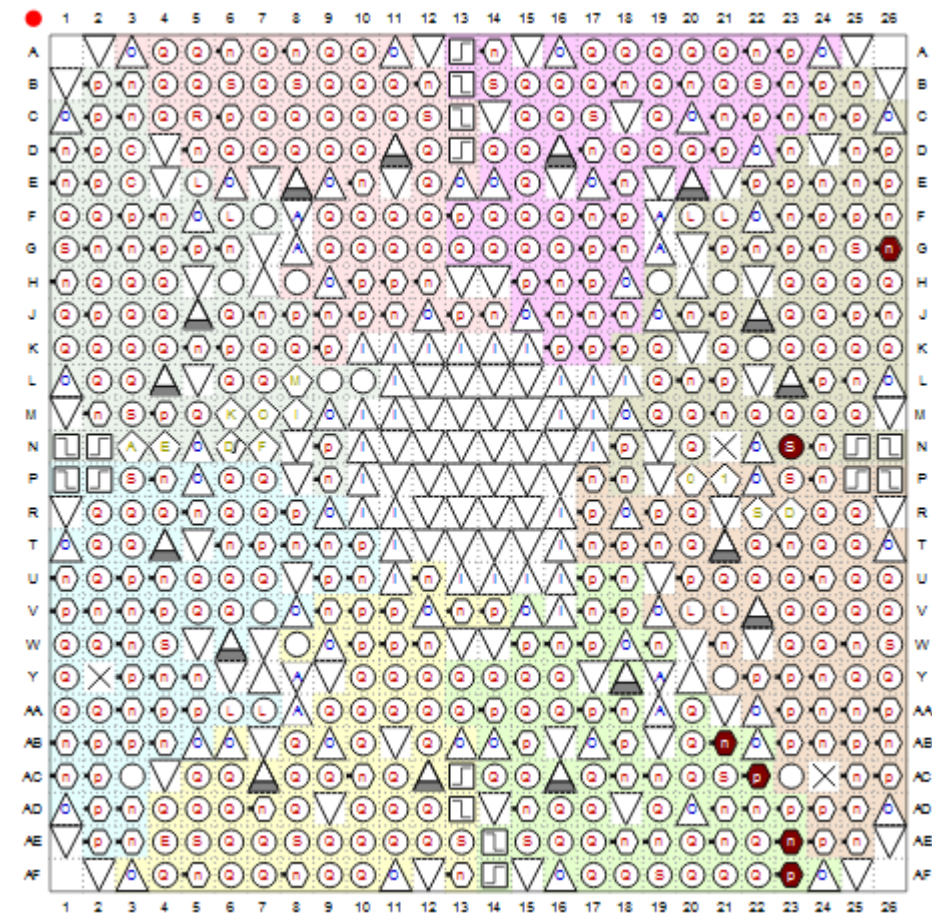


## 3. Function Simulation



1111(=16)인 지점 이후에 다시 0000 이 되는 것을 볼 수 있다.

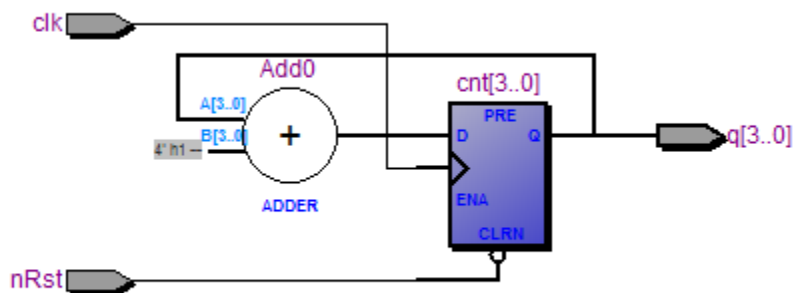
#### 4. Pin Planner



Node Name	Direction	Location
in clk	Input	PIN_G26
in nRst	Input	PIN_N23
out q[3]	Output	PIN_AC22
out q[2]	Output	PIN_AB21
out q[1]	Output	PIN_AF23
out q[0]	Output	PIN_AE23
<<new node>>		

스위치, LED 의 직관적인 이해를 위해 순차적으로 배치하였다.

#### 5. RTL viewer



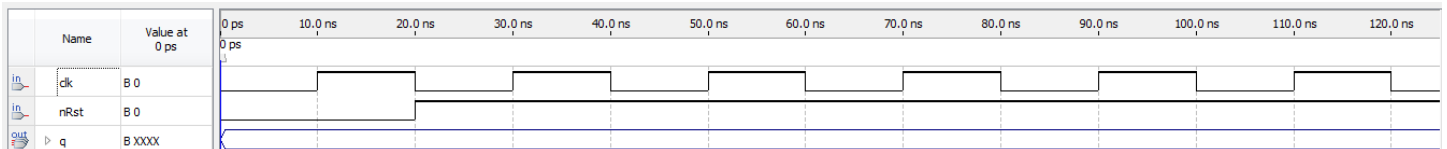
# 10진 counter

## 1. VHDL 작성

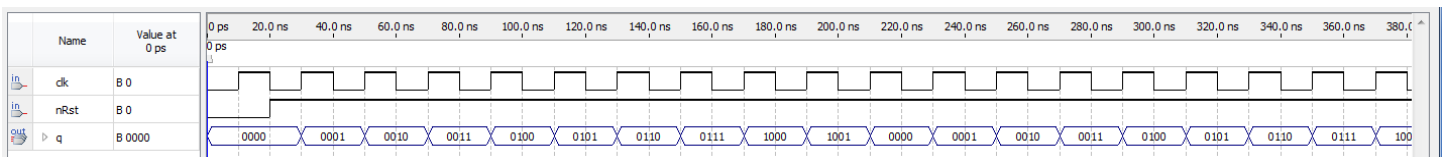
```
1 library ieee;
2
3 use ieee.std_logic_1164.all;
4 use ieee.std_logic_arith.all;
5 use ieee.std_logic_unsigned.all;
6
7 entity counter_10 is
8
9 port(
10     nRst : in std_logic;
11     clk : in std_logic;
12     q : out std_logic_vector(3 downto 0)
13 );
14 end counter_10;
15
16 architecture BEH of counter_10 is
17
18     signal cnt : std_logic_vector(3 downto 0);
19
20 begin
21     process(nRst, clk)
22     begin
23         if(nRst = '0') then
24             cnt <= (others => '0');
25         elsif rising_edge(clk) then
26             if(cnt = 9) then
27                 cnt <= (others => '0');
28             else
29                 cnt <= cnt + 1;
30             end if;
31         end if;
32     end process;
33
34     q <= cnt;
35
36 end BEH;
```

Colored by Color Scripter CS

## 2. VMF 파일 생성 및 입력레벨 설정

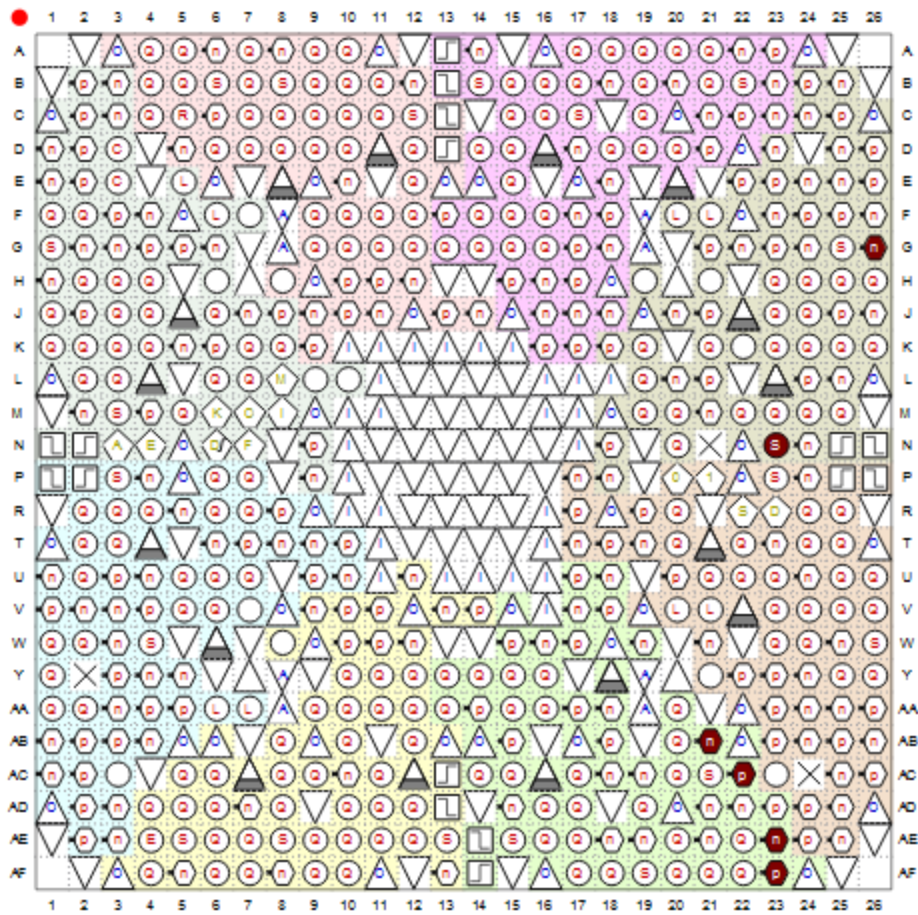


## 3. Function Simulation



1001 (=9)인 지점 이후에 0000(=0)으로 다시 돌아오는 것을 볼 수 있다. 이는 ~9 까지의 카운팅 동작을 제대로 수행하는 것이다.

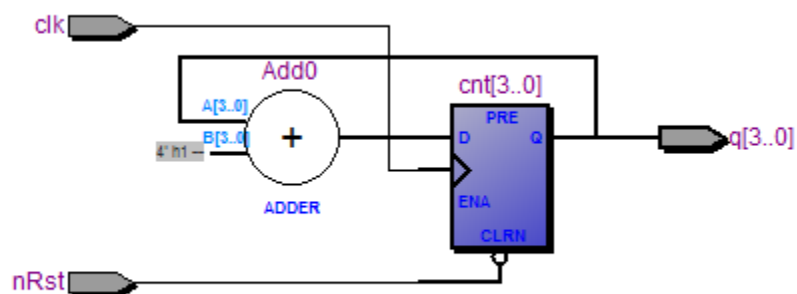
## 4. Pin Planner



Node Name	Direction	Location
in clk	Input	PIN_G26
in nRst	Input	PIN_N23
out q[3]	Output	PIN_AC22
out q[2]	Output	PIN_AB21
out q[1]	Output	PIN_AF23
out q[0]	Output	PIN_AE23
<<new node>>		

스위치, LED 의 직관적인 이해를 위해 순차적으로 배치하였다.

## 5. RTL viewer



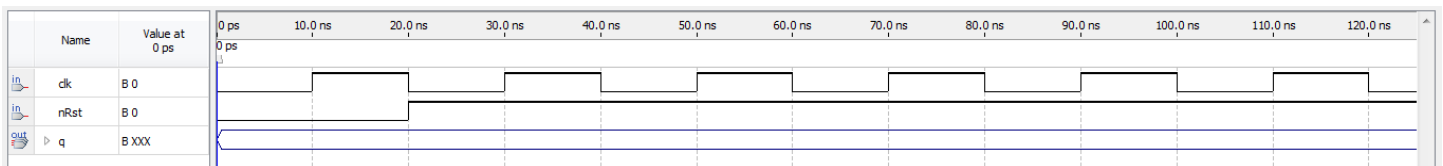


## 5-6-7 반복 3진 Counter

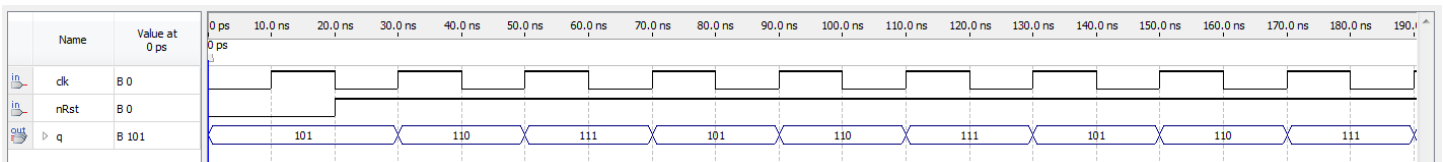
### 1. VHDL 작성

```
1 library ieee;
2
3 use ieee.std_logic_1164.all;
4 use ieee.std_logic_arith.all;
5 use ieee.std_logic_unsigned.all;
6
7 entity counter_567 is
8
9 port(
10     nRst : in std_logic;
11     clk : in std_logic;
12     q : out std_logic_vector(2 downto 0)
13 );
14 end counter_567;
15
16 architecture BEH of counter_567 is
17
18     signal cnt : std_logic_vector(2 downto 0);
19
20 begin
21     process(nRst, clk)
22     begin
23         if(nRst = '0') then
24             cnt <= "101";
25         elsif rising_edge(clk) then
26             if(cnt = 7) then
27                 cnt <= "101";
28             else
29                 cnt <= cnt + 1;
30             end if;
31         end if;
32     end process;
33
34     q <= cnt;
35
36 end BEH;
```

### 2. VMF 파일 생성 및 입력레벨 설정








### 3. Function Simulation

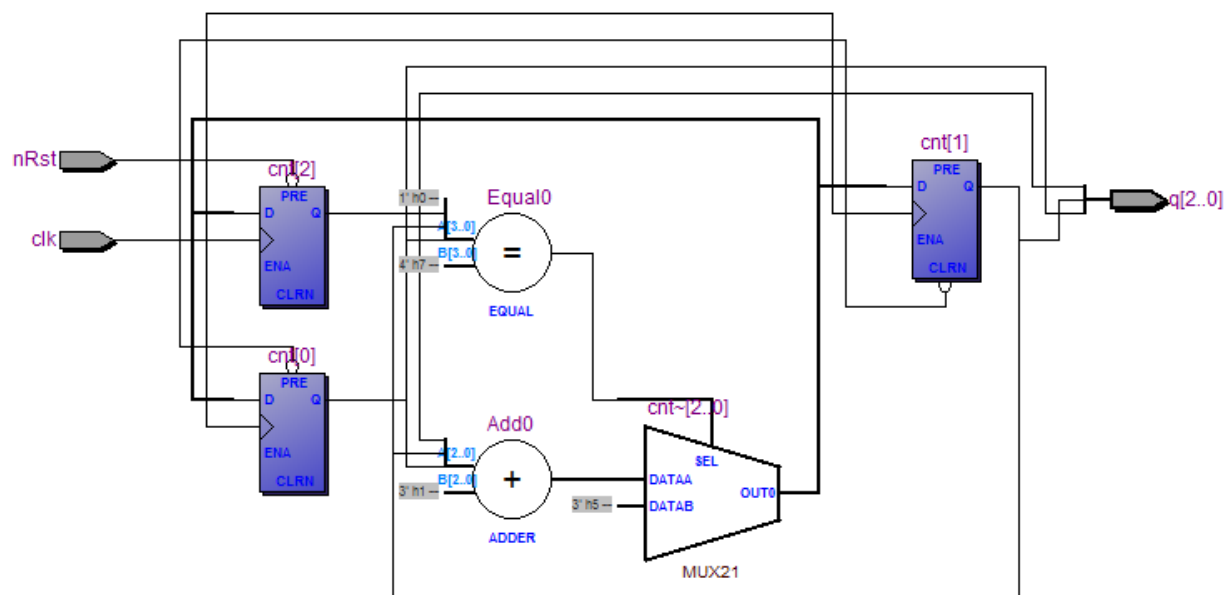


101(=3), 110(=4), 111(=5)를 반복적으로 수행하는 것을 볼 수 있다. 선택지가 3 개이므로  $2^2$ 으로 두 비트를 사용하고 불빛을 3, 4, 5 다른 불을 켤 수 있지 않을까 생각해 보았다.



Node Name	Direction	Location
 clk	Input	PIN_G26
 nRst	Input	PIN_N23
 q[2]	Output	PIN_AB21
 q[1]	Output	PIN_AF23
 q[0]	Output	PIN_AE23

## 5. RTL Viewer

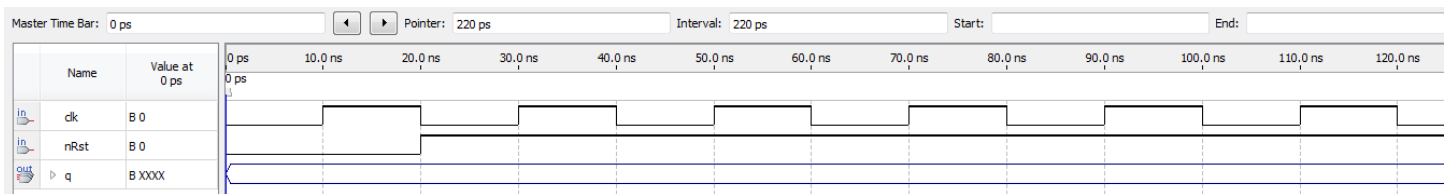


## 2-13 반복 12 진 카운터

### 1. VHDL 작성

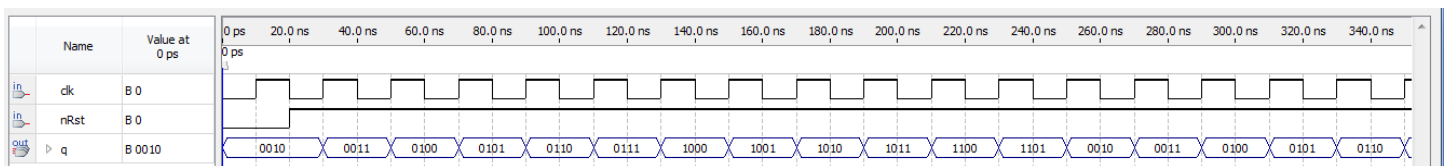
```
1 library ieee;
2
3 use ieee.std_logic_1164.all;
4 use ieee.std_logic_arith.all;
5 use ieee.std_logic_unsigned.all;
6
7 entity counter_213 is
8
9 port(
10     nRst : in std_logic;
11     clk : in std_logic;
12     q : out std_logic_vector(3 downto 0)
13 );
14 end counter_213;
15
16 architecture BEH of counter_213 is
17
18     signal cnt : std_logic_vector(3 downto 0);
19
20 begin
21     process(nRst, clk)
22     begin
23         if(nRst = '0') then
24             cnt <= "0010";
25         elsif rising_edge(clk) then
26             if(cnt = 13) then
27                 cnt <= "0010";
28             else
29                 cnt <= cnt + 1;
30             end if;
31         end if;
32     end process;
33
34     q <= cnt;
35
36 end BEH;
```

### 2. VMF 파일 생성 및 입력레벨 설정



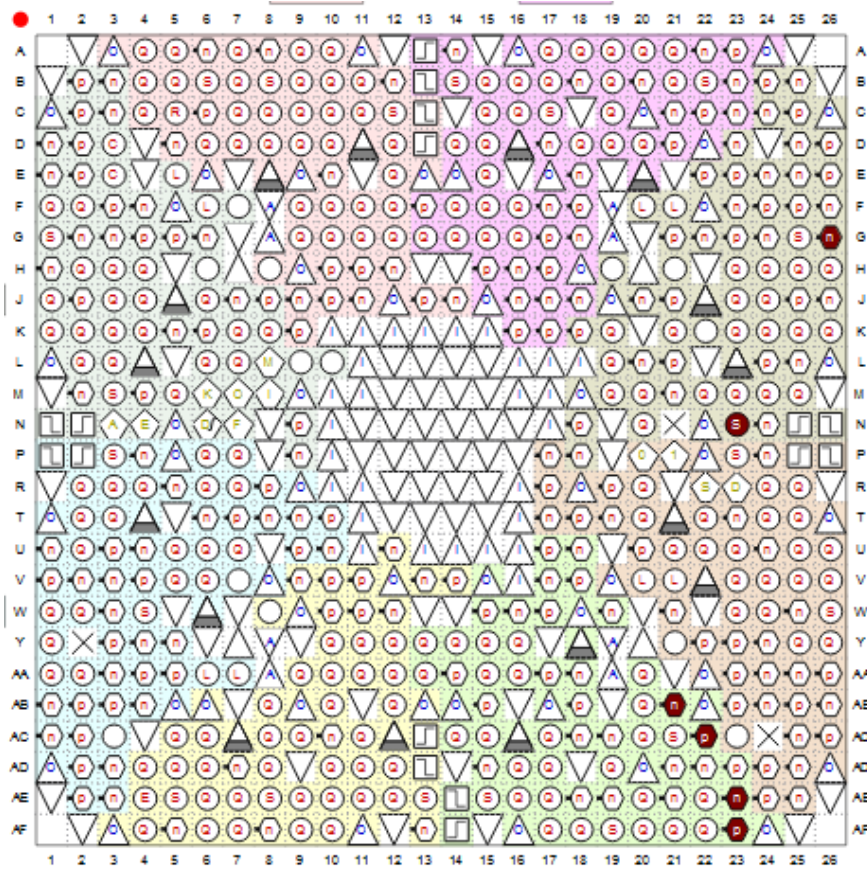
입력 펄스를 조절하여 모든 경우의 수를 쉽게 나타낼 수 있었다.

### 3. Function Simulation



0010(=2)에서 시작하여 13(=1101)까지의 카운팅이 반복되는 것을 볼 수 있다.

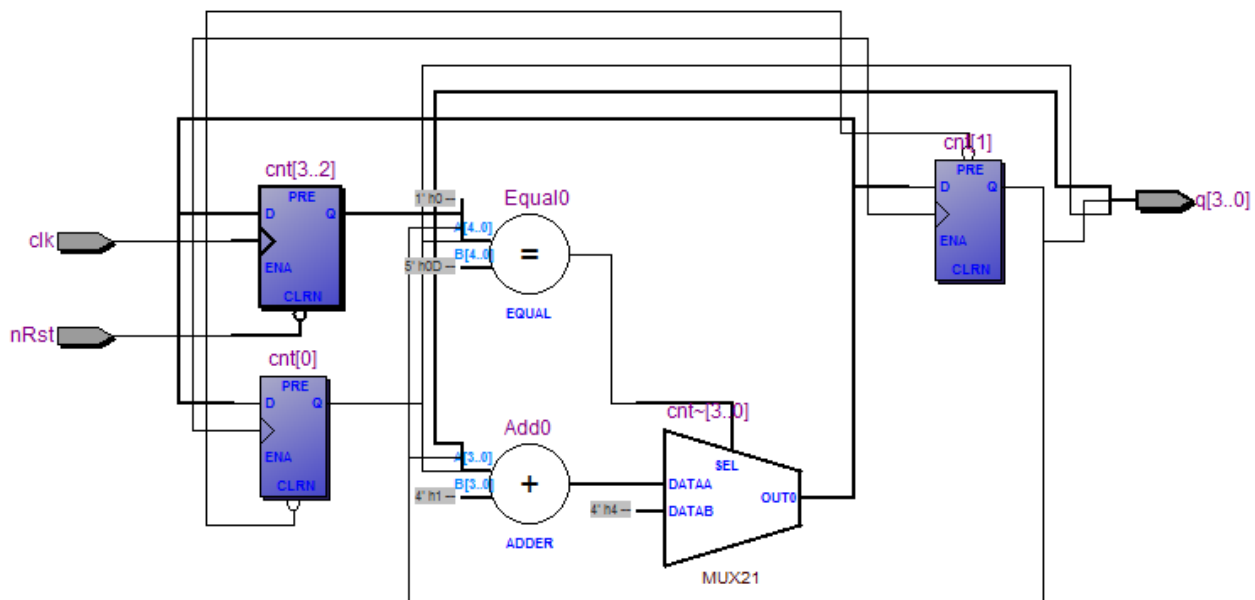
#### 4. Pin Planner



Node Name	Direction	Location
in clk	Input	PIN_G26
in nRst	Input	PIN_N23
out q[3]	Output	PIN_AC22
out q[2]	Output	PIN_AB21
out q[1]	Output	PIN_AF23
out q[0]	Output	PIN_AE23


스위치, LED 의 직관적인 이해를 위해 순차적으로 배치하였다.

#### 5. RTL Viewer



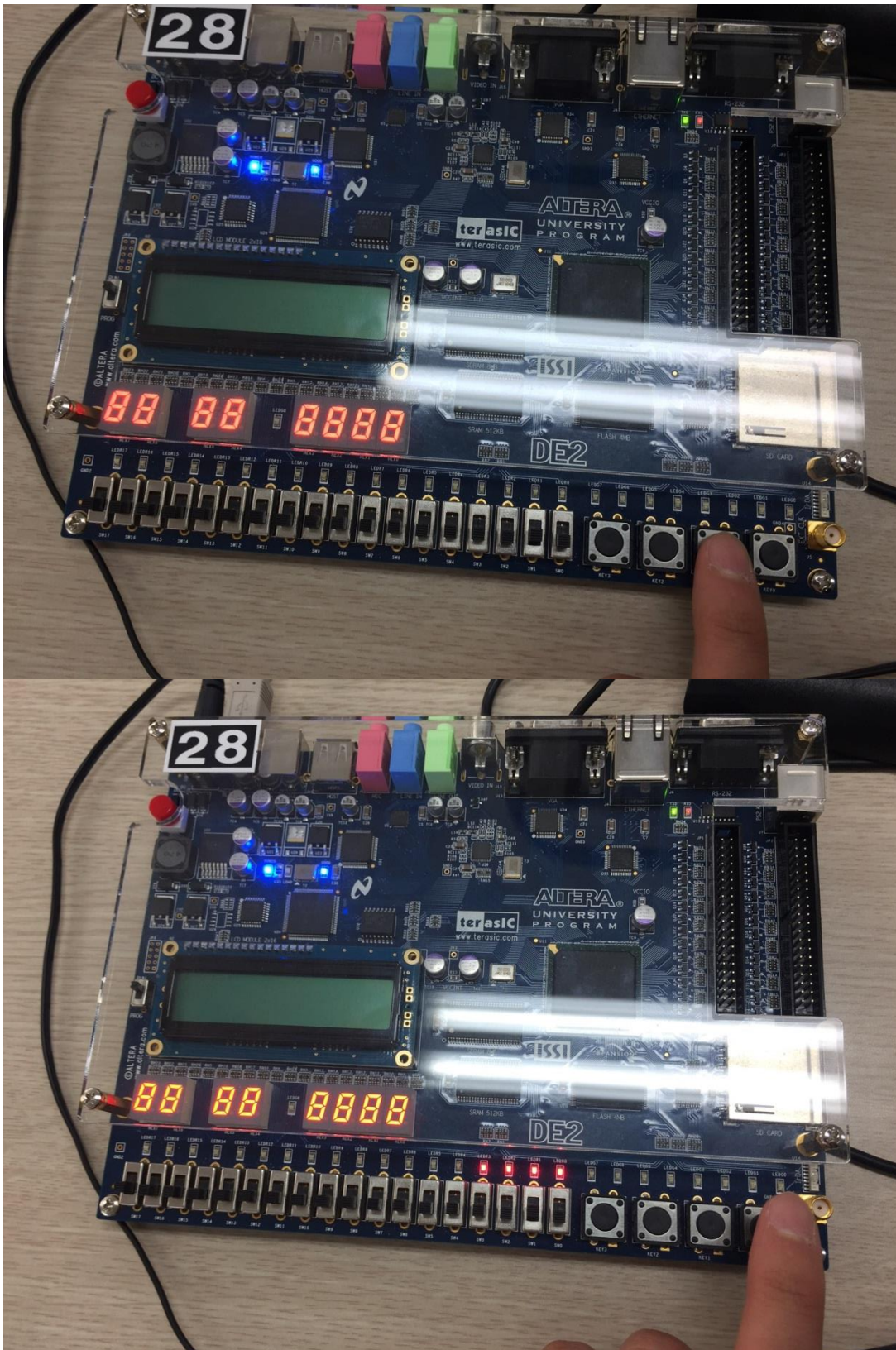


#### 4. 실습보드 적용 결과

 해당 소스를 보드에 다운로드하여 예상한 실행 결과에 맞게 동작하는 지 테스트하시오.

### 16 진 Counter

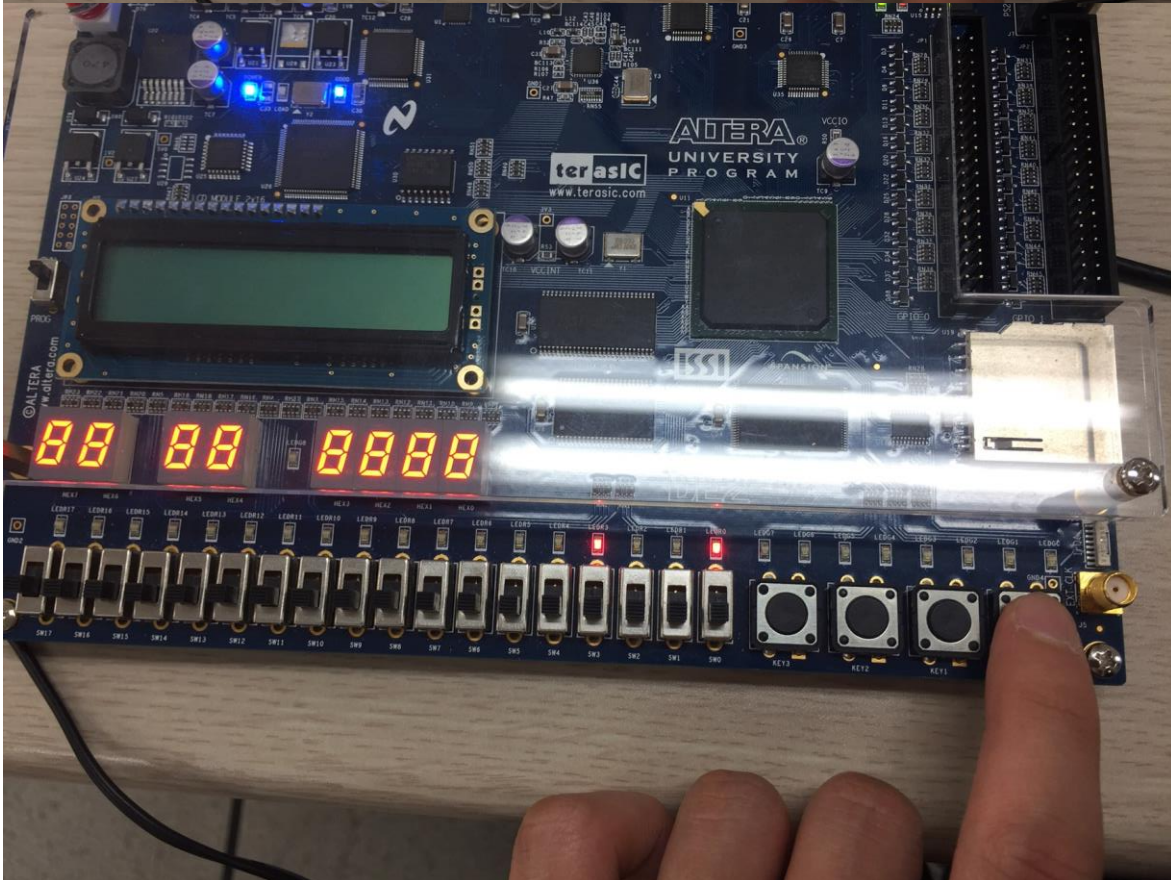
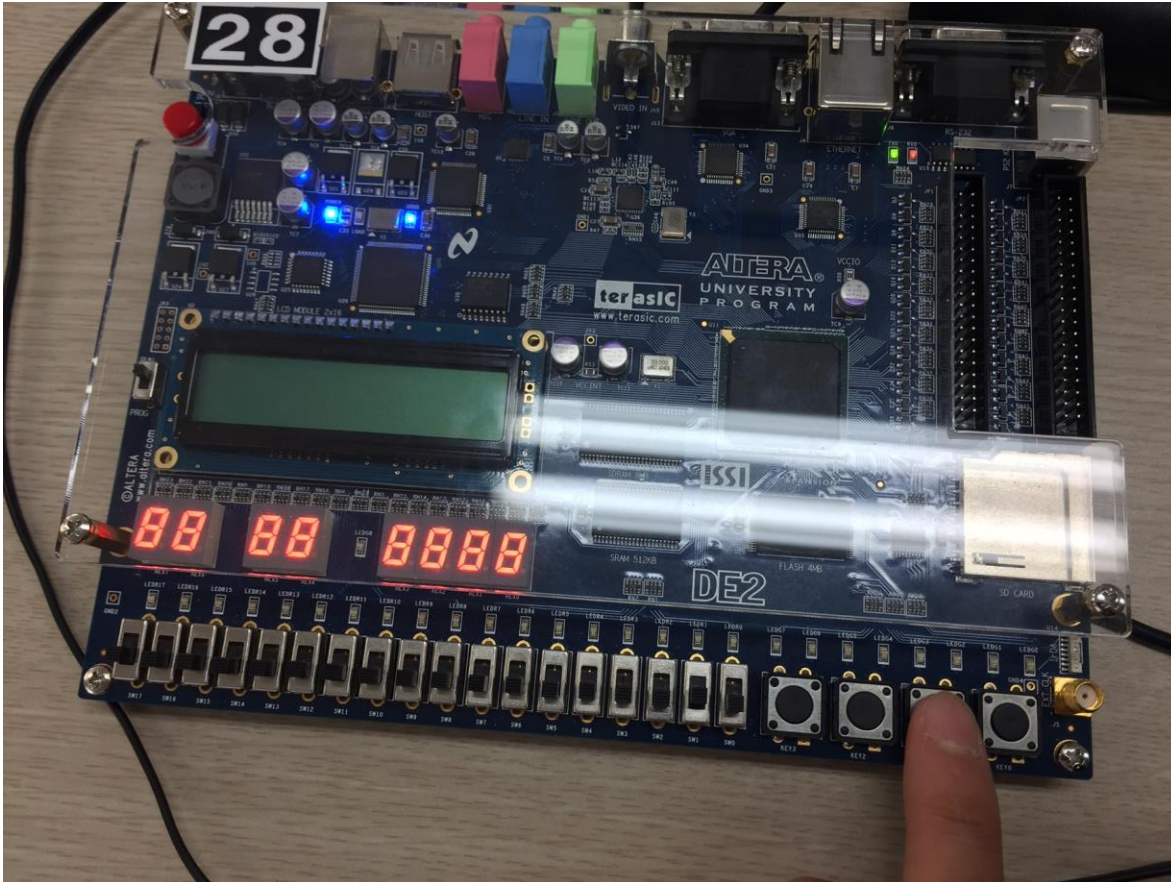
16 진 카운팅 동작과 RESET 버튼을 시뮬레이팅해 보았습니다.





## 10 진 Counter

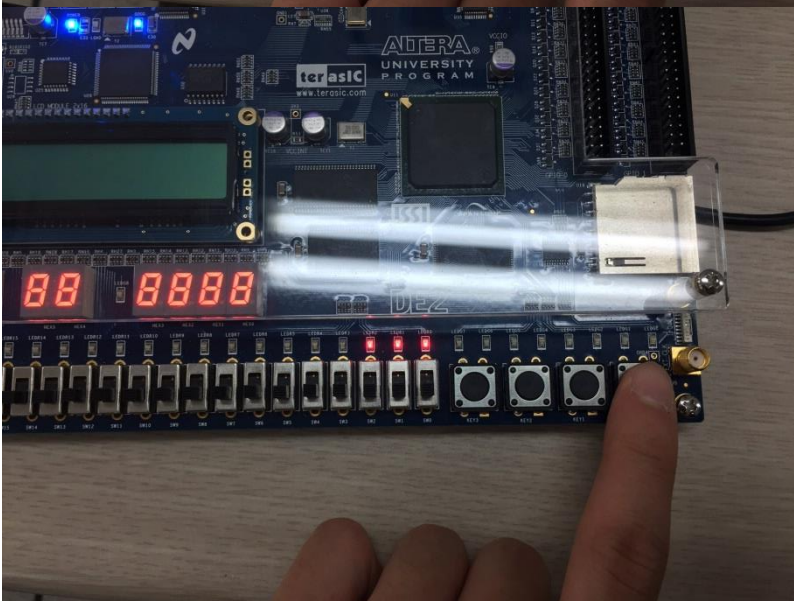
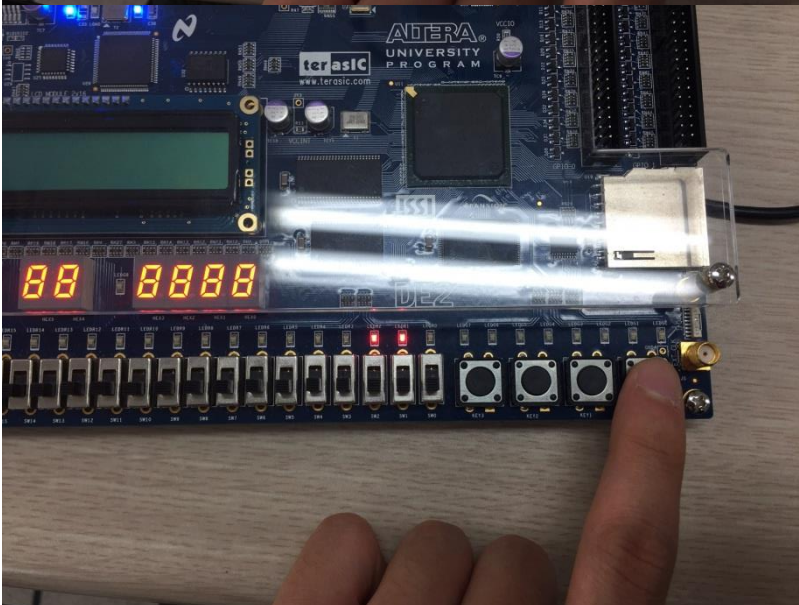
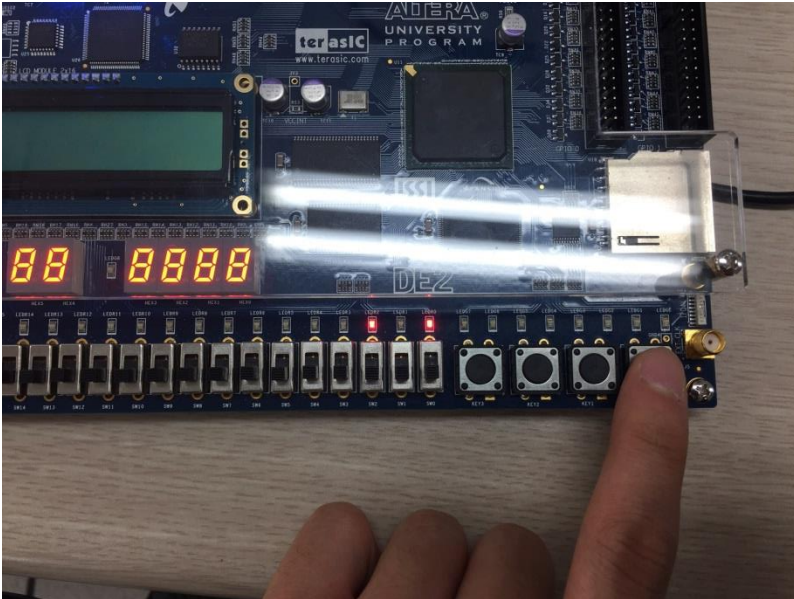
10 진 카운팅 동작과 RESET 버튼을 시뮬레이팅해 보았습니다.





## 5-6-7 Counter

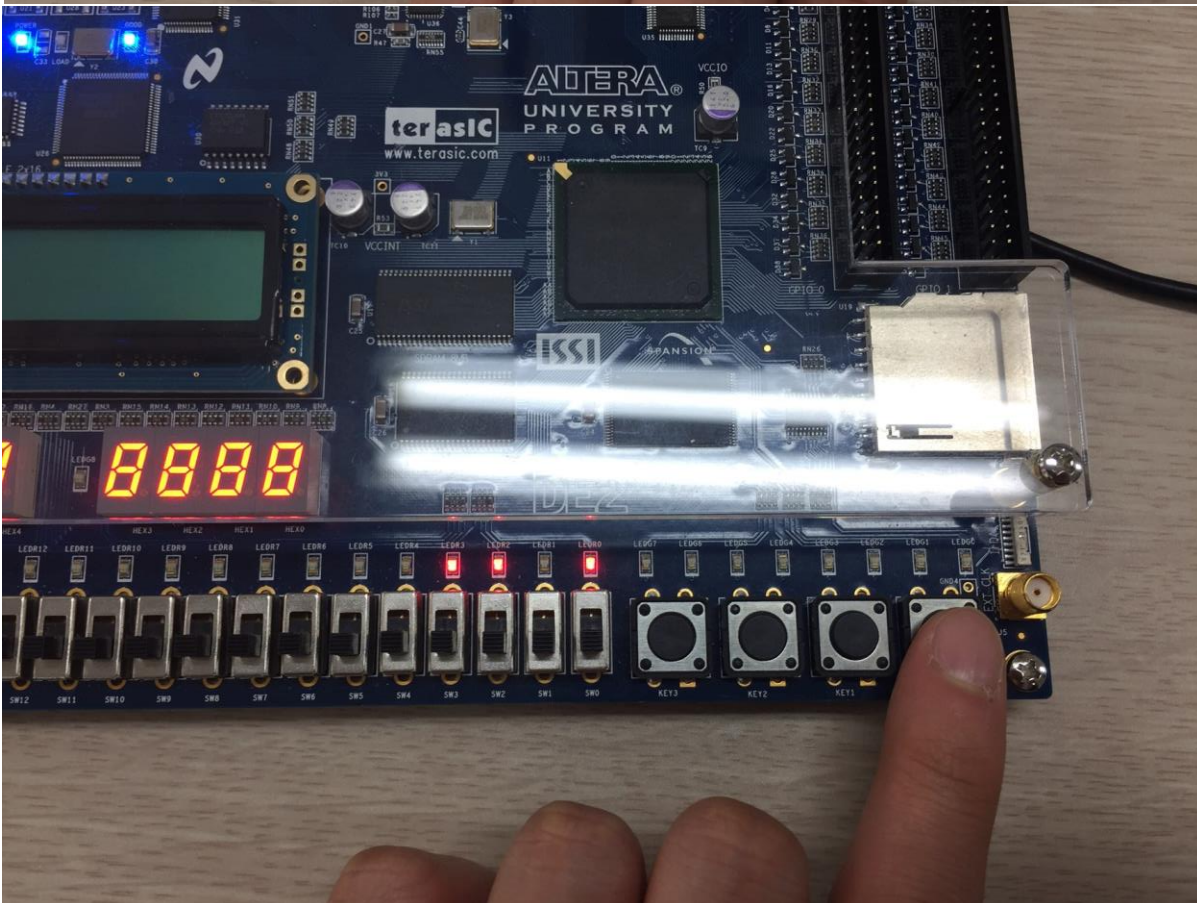
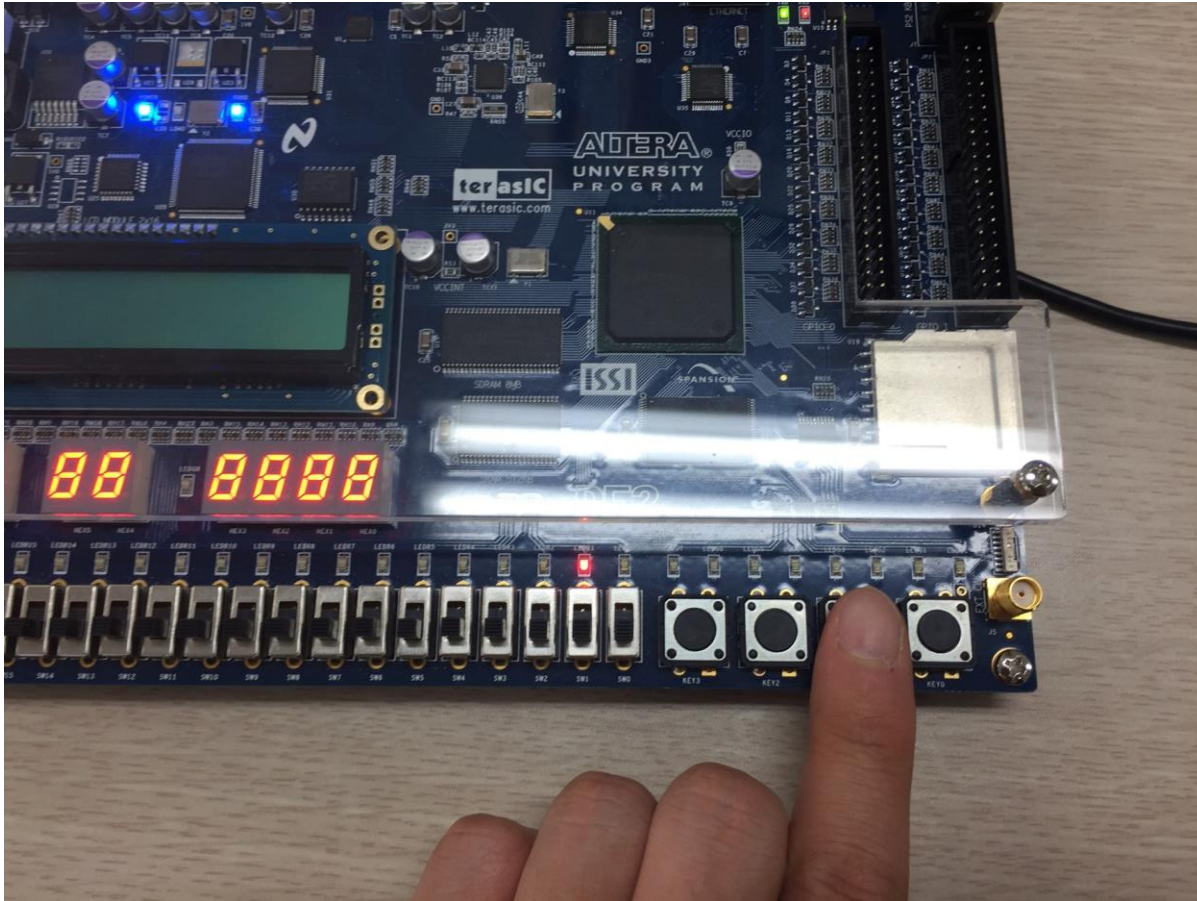
5-6-7 을 순회하는 카운팅 동작과 RESET 버튼을 시뮬레이팅해 보았습니다.





## 5-6-7 Counter

5-6-7 을 순회하는 카운팅 동작과 RESET 버튼을 시뮬레이팅해 보았습니다.





## 5. 실습소감



실습을 통해 경험한 것을 자유롭게 서술하시오.

실습을 진행하며 VMF 파일에서 에러가 발생했는데 사용하는 R led 가 4 개인데 q[4] 핀이 할당되어 시뮬레이팅에서 에러가 발생한 것이였습니다. 항상 순서대로 작업하다 보니 시뮬레이팅 다음 단계인 pin 배치에서 에러가 발생할 거라고 생각하지 못했는데 이후 단계의 설정에 문제가 있으면 시뮬레이팅 과정에 에러가 생길 수 있다는 것을 알았습니다. 에러메시지를 읽고 잘 대처할 수 있도록 하였습니다. 또 이번 실험에서는 16, “1011”과 같이 10 진수 정수형으로 비교, 대입하는 과정과 “1011”으로 2 진수 형태로 비교하는 과정이 혼용되었는데 0b1011 과 같이 사용되지 않는 것을 확인 할 수 있었습니다. 단순히 “”안에 표기하는 것으로 직관적으로 이해할 수 있어 더 보기 편한 것 같습니다. 강의시간에만 보드를 통한 실습을 할 수 있다는 것이 아쉽긴 하지만 눈으로 보면서 이해할 수 있는 수업이라 색다른 경험이 되는 것 같습니다. 다음 강의도 너무 기대됩니다.