VHDL 및 실습

샘플링 이론, 클록 & 신호

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

MULTIPLEXER, DECODER

1. 주제 및 배경이론

소스코드를 설명과 함께 기술하시오.

멀티플렉서(Multiplexer)

: 멀티플렉서는 여러 개의 입력을 어떠한 조합으로 선택하여 하나의 출력으로 나타내는 조합논리회로이다.

- 선택 변수 SEL 을 통해 입력선이 선택되어 출력에 나타난다.
- 입력의 조합개수 만큼 선택변수 SEL 이 있어야한다.
- MUX 라는 약어로 표현한다.

디멀티플렉서(Demultiplexer)

: 디멀티플렉서는 하나의 입력을 통해 전달받은 정보를 여러 개의 출력선 중 하나를 통해 나타내는 조합논리회로이다. 출력선은 멀티플렉서와 같이 어떠한 조합에 의해 결정된다.

- 선택 변수 SEL 을 통해 출력선이 선택되어 출력된다.
- 출력의 조합개수 만큼 선택변수 SEL 이 있어야한다.
- DEMUX 라는 약어로 표현한다.
- DEMUX 는 디코더(Decoder)와 유사한 관계가 있다.

디코더(Decoder)

: 디코더란 n 비트 입력을 받아 최대 2^n 개의 서로 다른 정보로 바꾸어주는 조합회로를 말한다.

- 디코더는 코드를 정보화하는 회로이다.
- n 비트의 2 진수는 2^n 개의 정보를 담고 있다.
- $n \times 2^n$ 디코더와 같이 나타낸다.

디멀티플렉서(Demultiplexer)와 디코더(Decoder)의 차이 (해당하는 설명에 색상을 매치하여 확인)

- 디코더는 입력에 따라 출력을 결정한다. 출력 하나가 1 이면 나머지는 0(반전하여 출력하는 경우도 있다)
- 구동입력(E)있는 경우 활성화 될 때만 디코더로써 기능을 발휘한다.
- 이진 입력코드를 출력으로 변환시키는 해독기. (코딩이 암호화라면 디코딩은 해독)
- 입력, 선택(SEL)으로 출력선을 선택하여 1 개의 데이터입력(E)에 들어온 신호를 내보낸다. (반전 출력하는 경우도 있음)
- 입력된 데이터를 선택(SEL)에 따라 출력하는 데이터 분배기

2. 소스코드 및 코드 설명 (2, 3 단계 함께 기술)

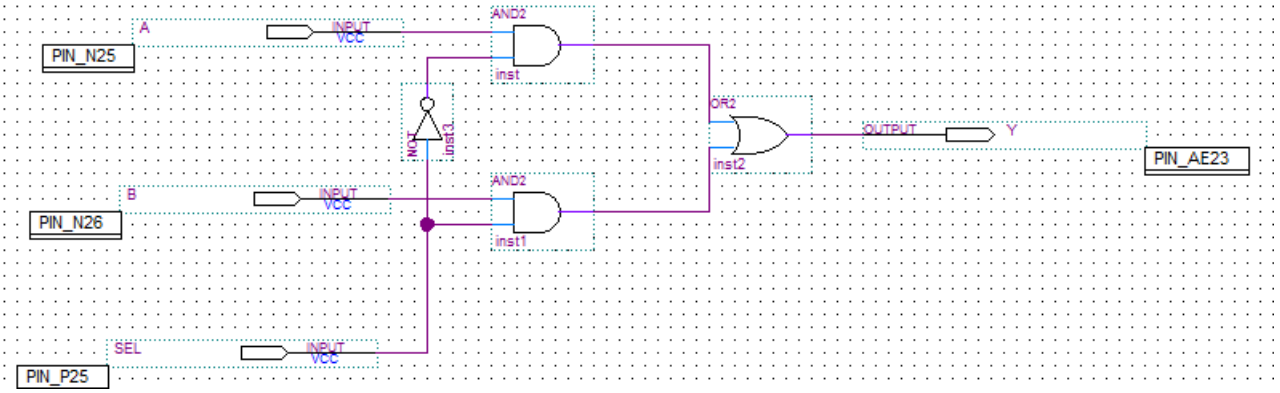
소스코드를 설명과 함께 기술하시오.

3. 시뮬레이션 결과 및 설명 (2, 3 단계 함께 기술)

위 코드를 시뮬레이션 하고 그에 대한 결과를 작성하시오.

2x1 MUX

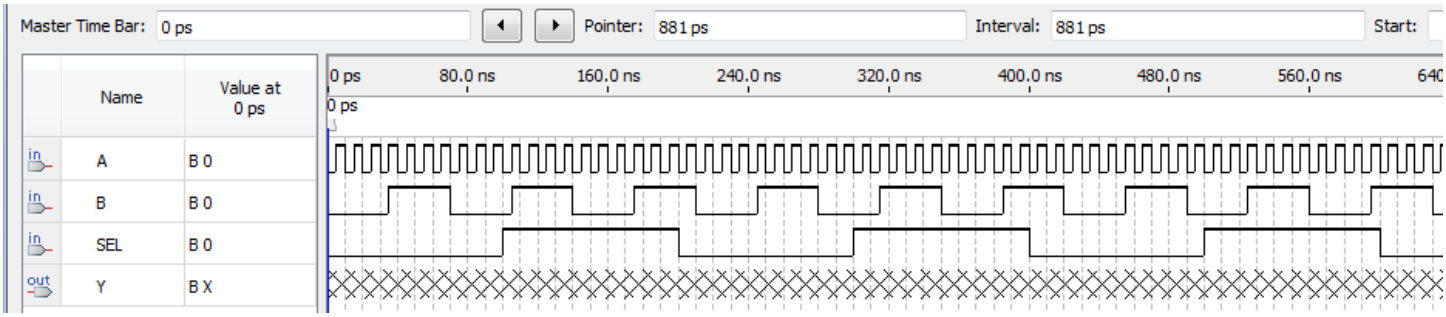
1. Schematic 그리기



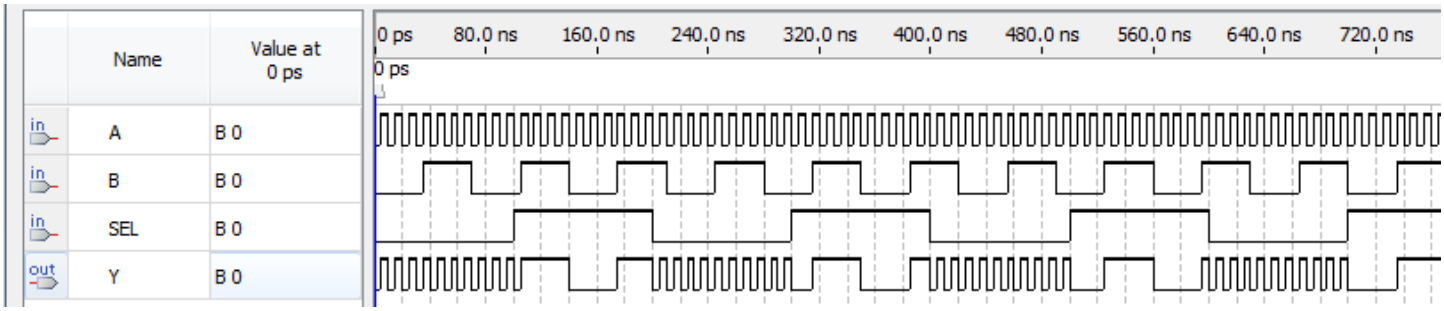
2. VHDL 작성

```vhdl
library ieee;
    use ieee.std_logic_1164.all;
    use ieee.std_logic_arith.all;
    use ieee.std_logic_unsigned.all;

entity MUX_2x1 is
    port(
        A : in std_logic;
        B : in std_logic;
         SEL : in std_logic;
        Y : out std_logic
    );
end MUX_2x1;

architecture BEH of MUX_2x1 is
begin
    Y <= A when SEL = '0' else
            B when SEL = '1' else
            'Z';    -- 예외 출력 조건을 다음과 같이 Z 로 설정, Z 는 대문자 사용
end BEH;
```

3. VMF 파일 생성 및 입력레벨 설정

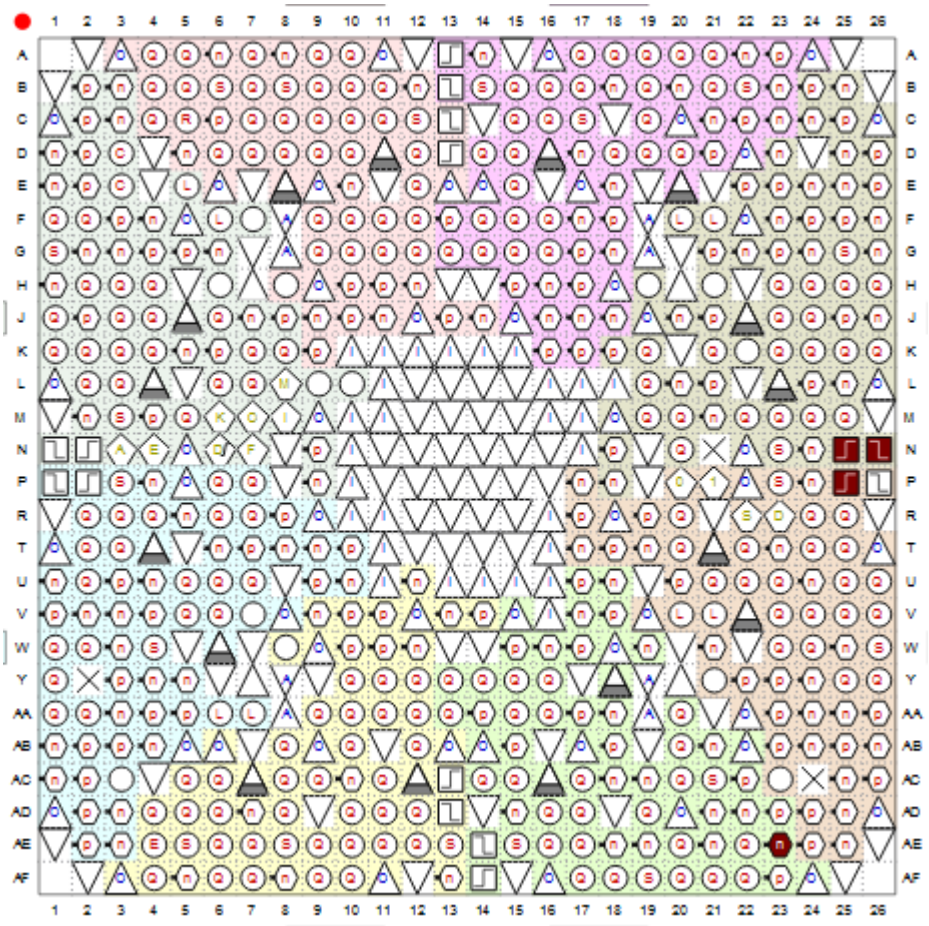


4. Function Simulation



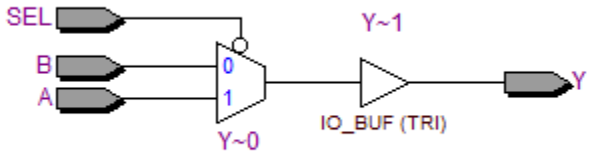
출력 레벨로 보아 SEL 입력이 LOW 일 경우 A 를, HIGH 일 경우 B 를 출력하는 것을 볼 수 있다. SEL 의 역할은 입력을 '선택'하는 것임을 알 수 있다.

5. Pin Planner



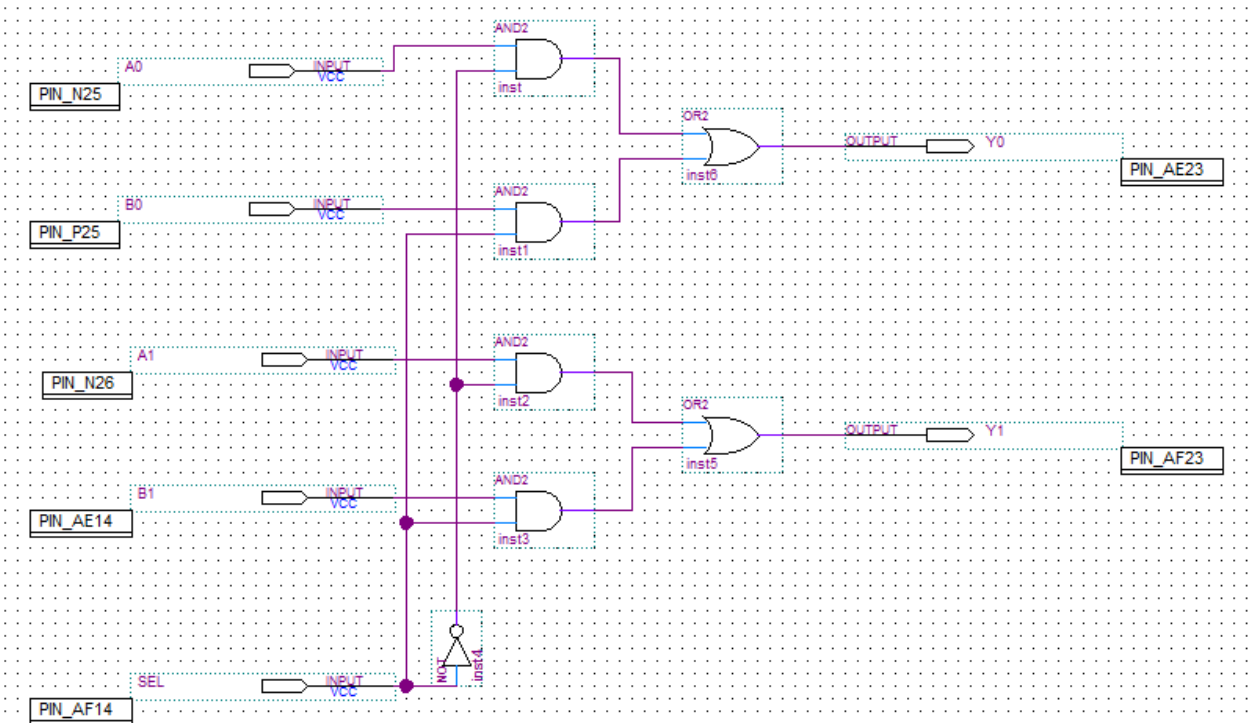
스위치, LED 의 직관적인 이해를 위해 순차적으로 배치하였다.

6. RTL viewer



2bit Multiplexer

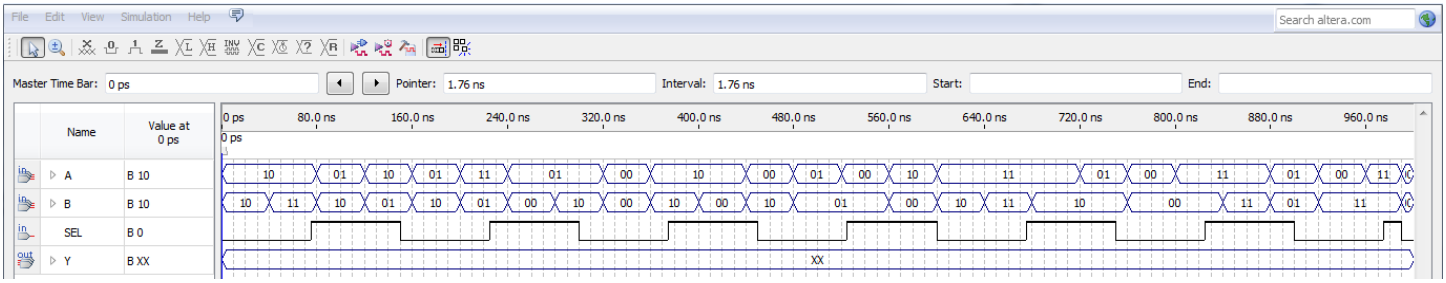
1. Schematic 그리기



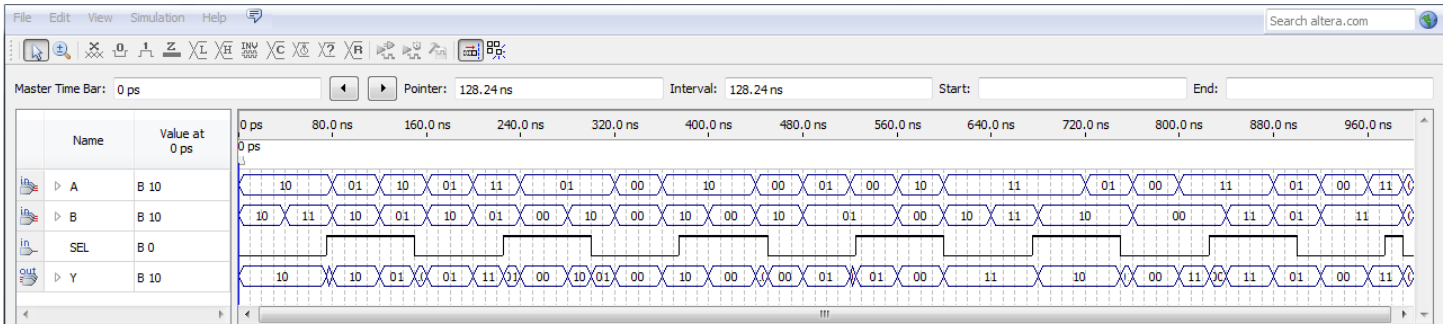
2. VHDL 작성

```vhdl
library ieee;
   use ieee.std_logic_1164.all;
   use ieee.std_logic_arith.all;
   use ieee.std_logic_unsigned.all;

entity MUX_2x1 is
   port(
      A : in std_logic_vector(1 downto 0); -- 2 비트 입력을 다음과 같이 간소화 하여 나타낼 수 있다.
      B : in std_logic_vector(1 downto 0);
       SEL : in std_logic;
      Y : out std_logic_vector(1 downto 0)
   );
end MUX_2x1;

architecture BEH of MUX_2x1 is
begin
   Y <= A when SEL = '0' else
         B when SEL = '1' else
         (others => 'Z'); -- z 는 잘못된 데이터를 의미한다. Z 가 대문자임에 유의하자.
end BEH;
```

3. VMF 파일 생성 및 입력레벨 설정

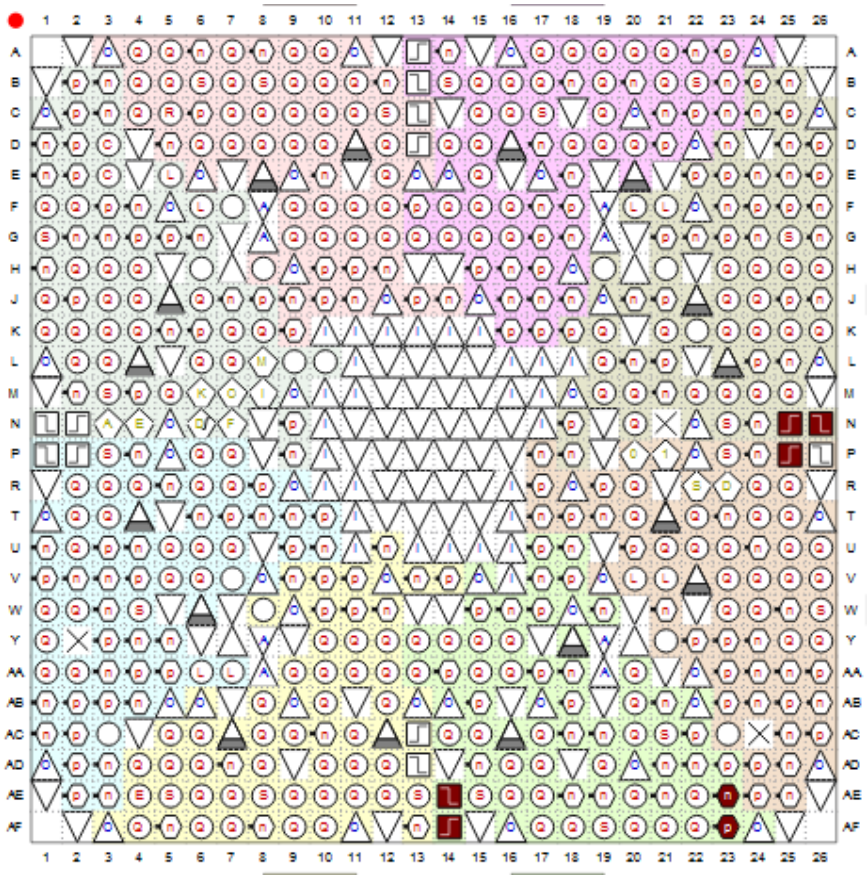


4. Function Simulation



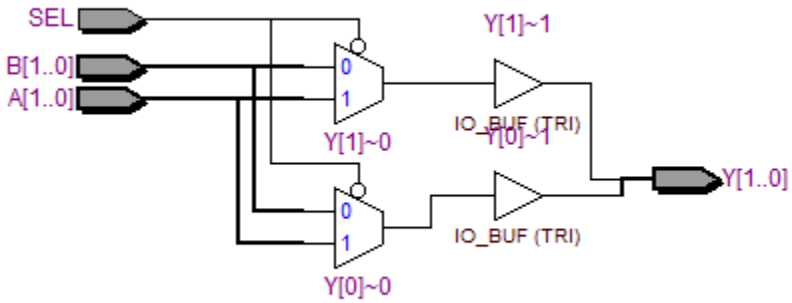
출력 레벨로 보아 SEL 입력이 LOW 일 경우 A 를, HIGH 일 경우 B 를 출력하는 것을 볼 수 있다. 단순히 입력 비트수가 늘어났을 뿐 SELECT 조건은 2 가지 경우로 동일한 것을 알 수 있었다.

5. Pin Planner



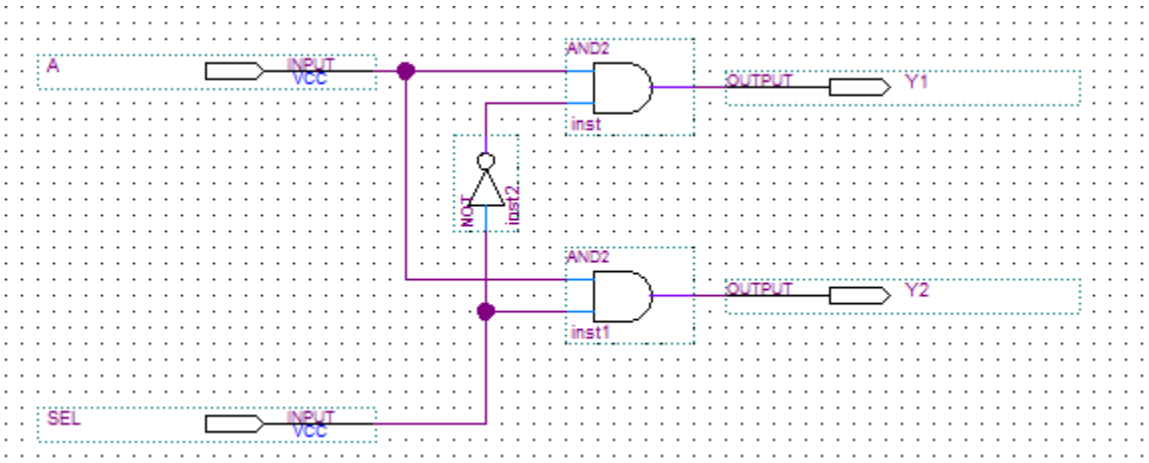
스위치, LED 의 직관적인 이해를 위해 순차적으로 배치하였다.

6. RTL viewer



DeMultiplexer

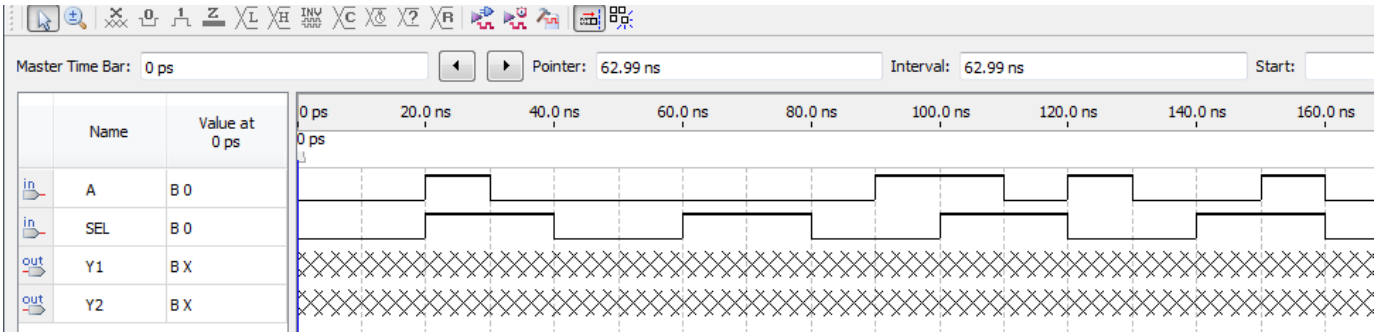
1. Schematic 그리기



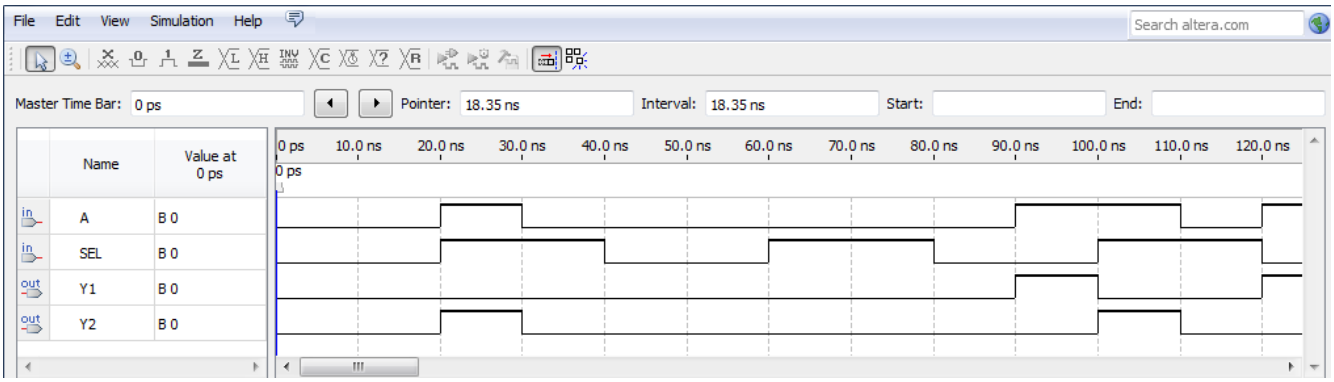
2. VHDL 작성

```vhdl
library ieee;

    use ieee.std_logic_1164.all;
    use ieee.std_logic_arith.all;
    use ieee.std_logic_unsigned.all;

entity DEMUX_1x2 is

    port(
        A : in std_logic;
        SEL : in std_logic;
        Y1 : out std_logic;
          Y2 : out std_logic

    );
end DEMUX_1x2;

architecture BEH of DEMUX_1x2 is

begin
    Y1 <= A when SEL = '0' else '0'; -- 각 비트가 가질 수 있는 경우의 수는 8 가지
    Y2 <= A when SEL = '1' else '0'; -- Z 는 잘못된 데이터, 여기서 0 은 사용자가 처리하고자 하는 데이터
end BEH;
```

3. VMF 파일 생성 및 입력레벨 설정

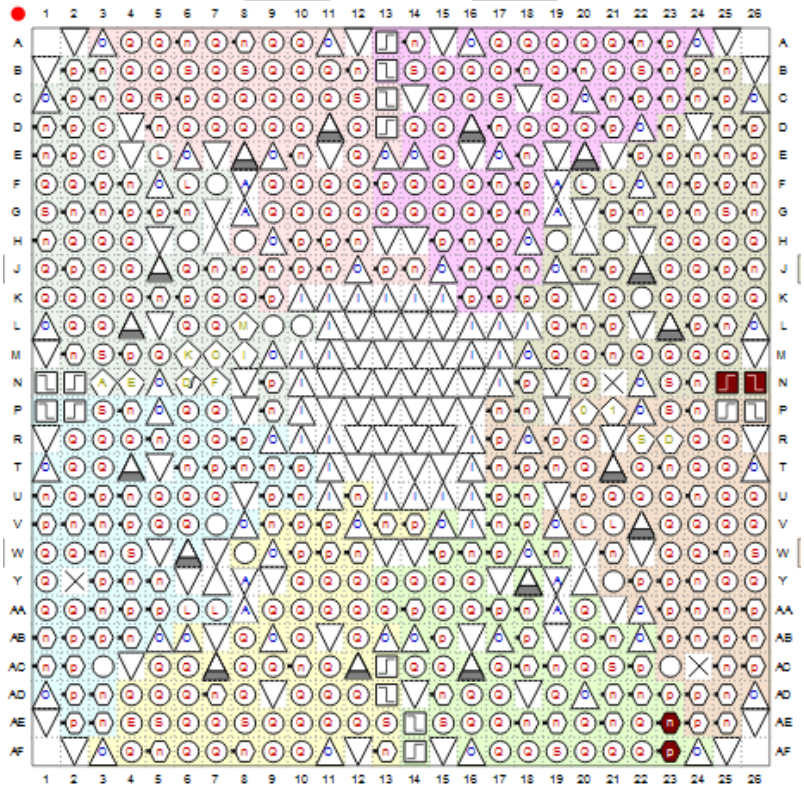


4. Function Simulation



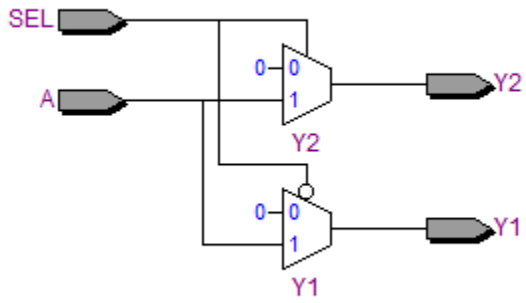
입력레벨을 난수로 설정하고 SEL 에 따라 어떤식으로 나타나는 지 확인하였다. 그 결과 SEL 의 값에 따라 입력값이 표시되는 출력위치가 달라지는 것을 확인 할 수 있었다..

5. Pin Planner



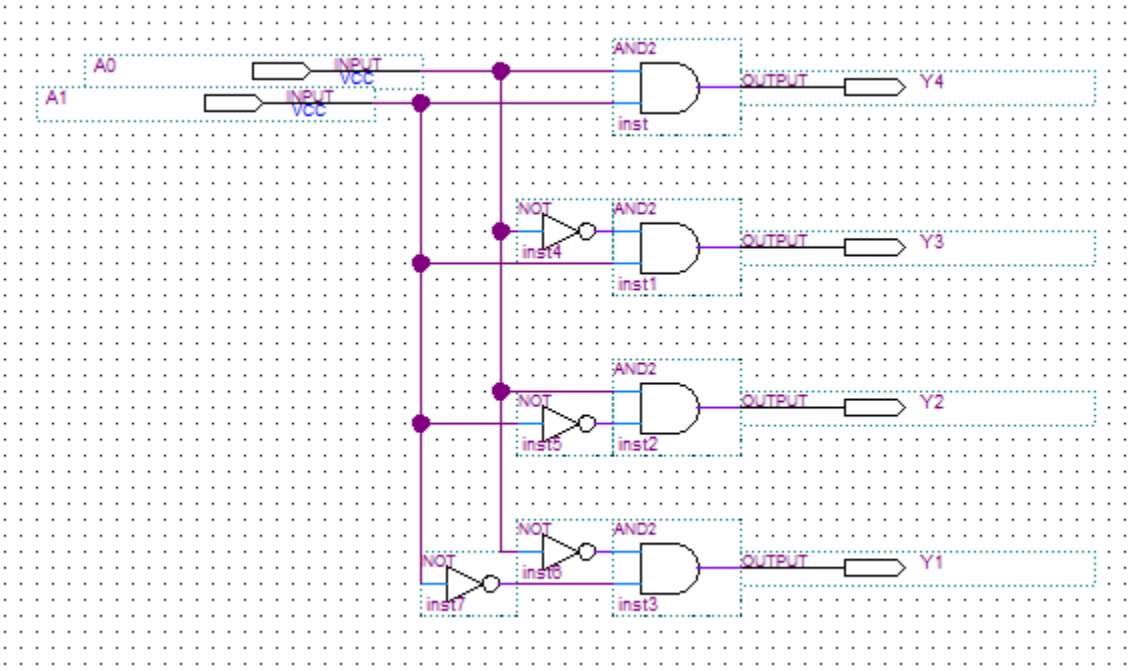
스위치, LED 의 직관적인 이해를 위해 순차적으로 배치하였다.

6. RTL Viewer



Decoder 2x4

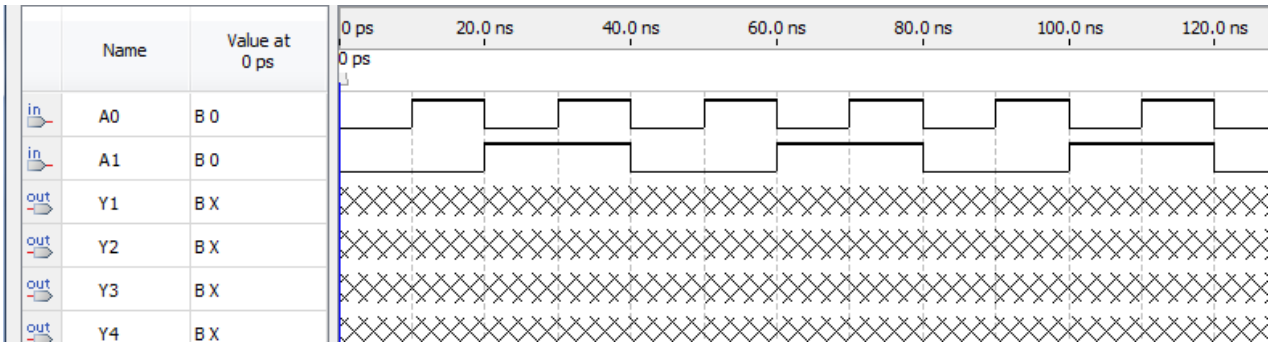
1. Schematic 그리기



2. VHDL 작성

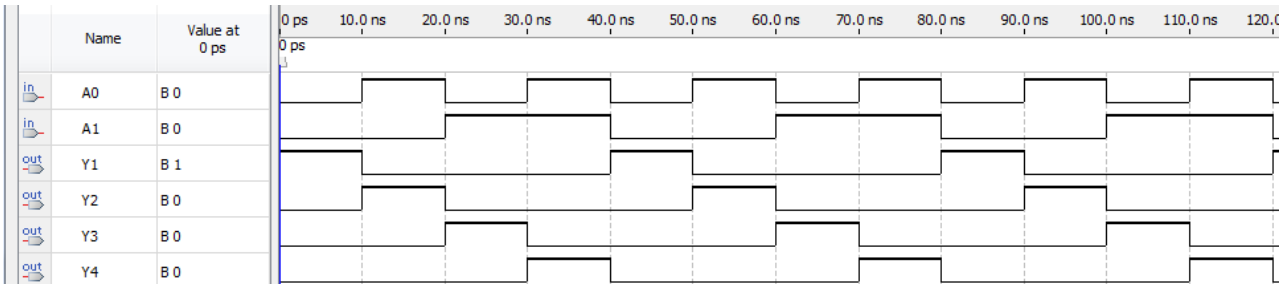
```vhdl
library ieee;
    use ieee.std_logic_1164.all;
    use ieee.std_logic_arith.all;
    use ieee.std_logic_unsigned.all;

entity decoder_2x4 is
    port(
        A0 : in std_logic;
        A1 : in std_logic;
        Y1 : out std_logic;
        Y2 : out std_logic;
        Y3 : out std_logic;
        Y4 : out std_logic
    );
end decoder_2x4;

architecture BEH of decoder_2x4 is
begin
    Y1 <= '1' when A0 = '0' and A1 = '0' else '0'; -- 각 비트가 가질 수 있는 경우의 수는 8 가지 임
    Y2 <= '1' when A0 = '1' and A1 = '0' else '0'; -- 예외는 모두 0 으로 처리
     Y3 <= '1' when A0 = '0' and A1 = '1' else '0';
    Y4 <= '1' when A0 = '1' and A1 = '1' else '0';
end BEH;
```

3. VMF 파일 생성 및 입력레벨 설정



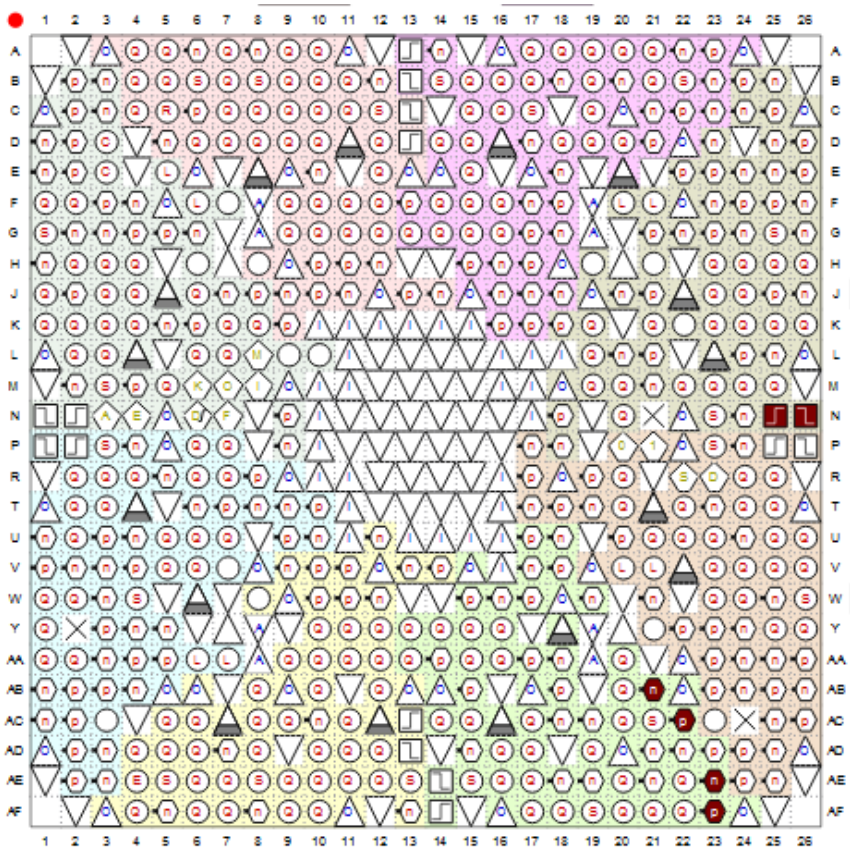
입력 펄스를 조절하여 모든 경우의 수를 쉽게 나타낼 수 있었다.

4. Function Simulation



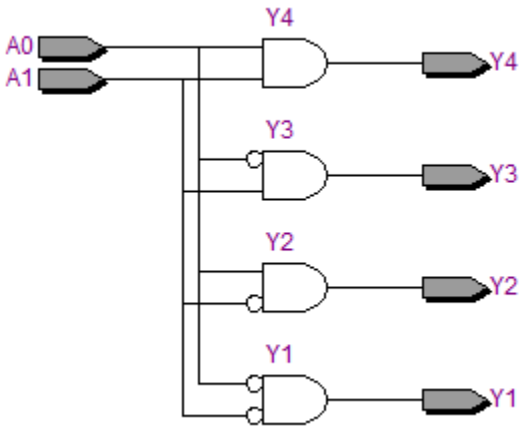
각 입력에 조건에 대해 일대일대응을 하는 것을 볼 수 있었다. 디코더의 의미에 대해 생각해 보면 이 그림이 쉽게 이해될 것이다.

5. Pin Planner



스위치, LED 의 직관적인 이해를 위해 순차적으로 배치하였다.

6. RTL Viewer

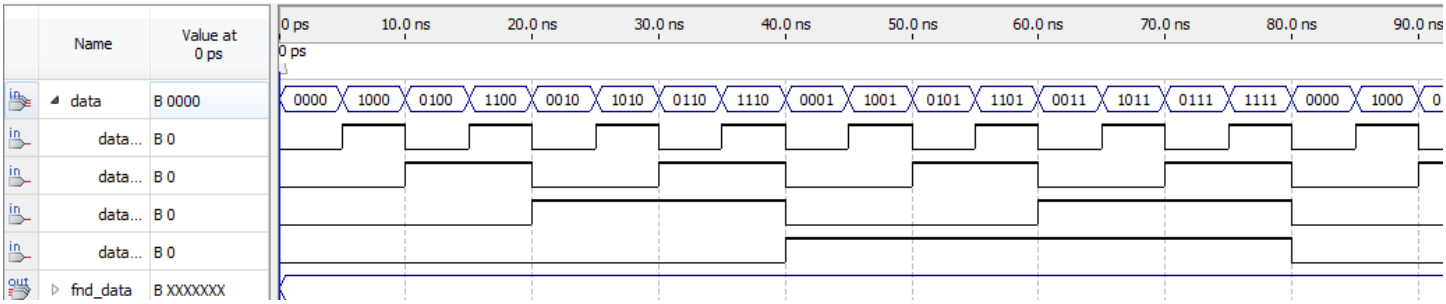


FND Decoder

1. VHDL 작성

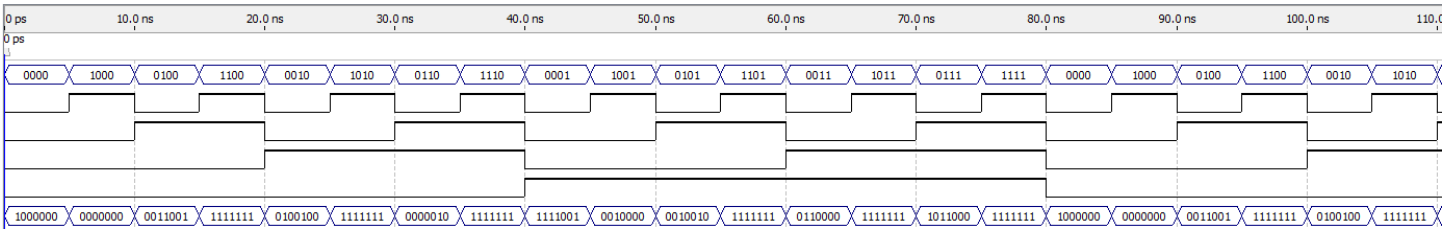
```vhdl
library ieee;

    use ieee.std_logic_1164.all;
    use ieee.std_logic_arith.all;
    use ieee.std_logic_unsigned.all;

entity FND_Decorder is
    port(
        data : in std_logic_vector (3 downto 0);
        fnd_data : out std_logic_vector (6 downto 0)
    );
end FND_Decorder;

architecture BEH of FND_Decorder is

begin
    fnd_data <= "1000000" when data = x"0" else
                "1111001" when data = x"1" else
                "0100100" when data = x"2" else
                "0110000" when data = x"3" else
                "0011001" when data = x"4" else
                "0010010" when data = x"5" else
                "0000010" when data = x"6" else
                "1011000" when data = x"7" else
                "0000000" when data = x"8" else
                "0010000" when data = x"9" else
                "1111111"; -- active row
end BEH;
```


2. VMF 파일 생성 및 입력레벨 설정



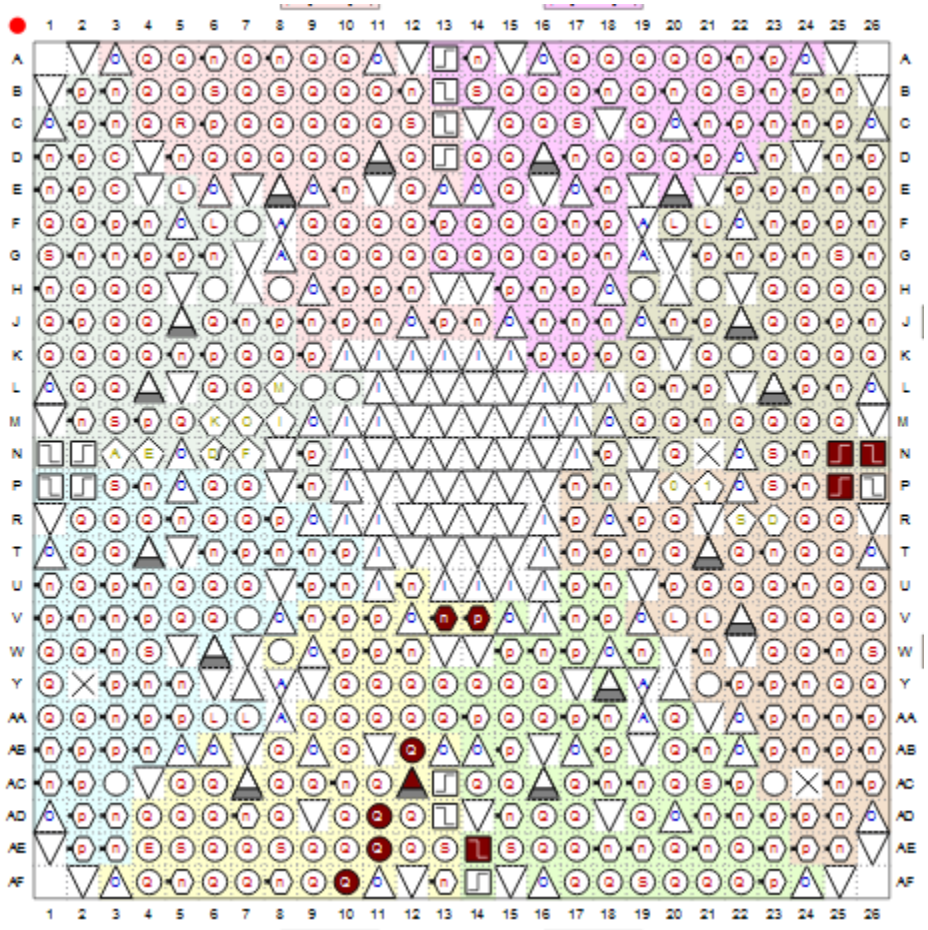
입력 펄스를 조절하여 모든 경우의 수를 쉽게 나타낼 수 있었다.

4. Function Simulation



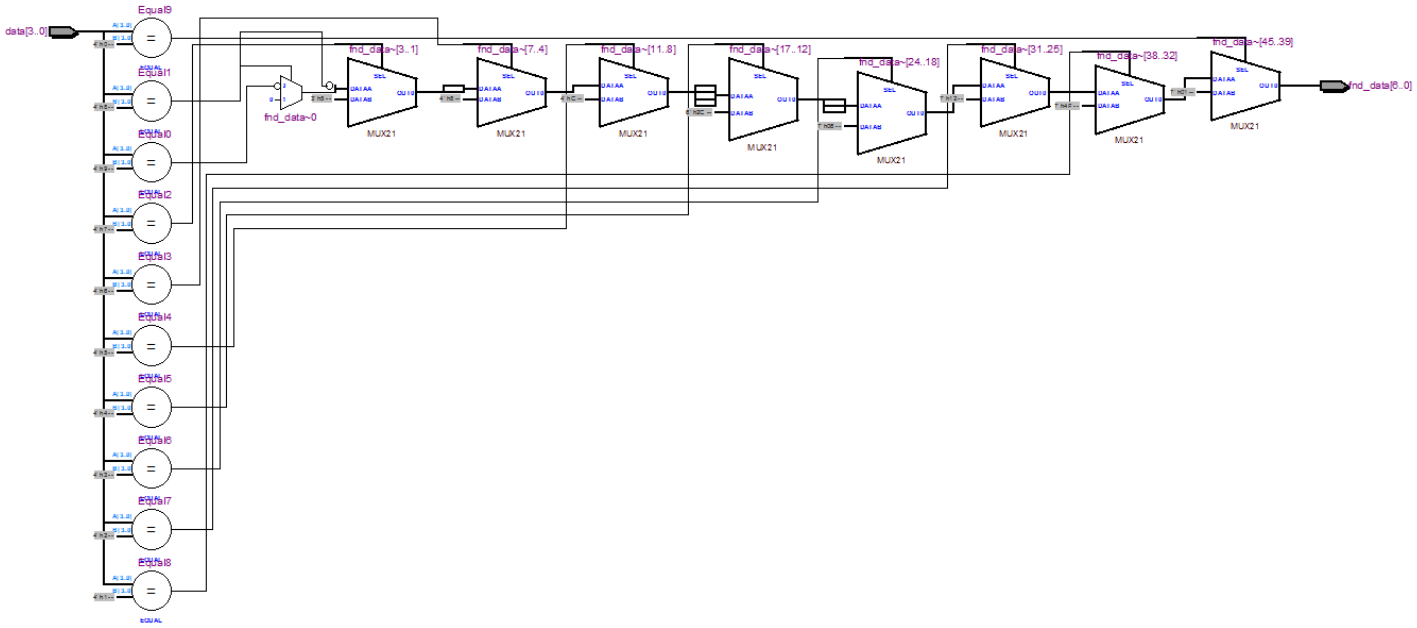
각 입력에 조건에 대해 일대일대응을 하는 것을 볼 수 있었다 FND_DATA 테이블을 보며 숫자가 나타내는 바를 쉽게 이해할 수 있었다.

5. Pin Planner



테이블을 보며 해당 기능에 연결하였다.

6. RTL Viewer



이전과 달리 복잡한 형태의 회로가 나타났다. 어떤식으로 이해해야 하는 지 감이 잡히지 않았다.

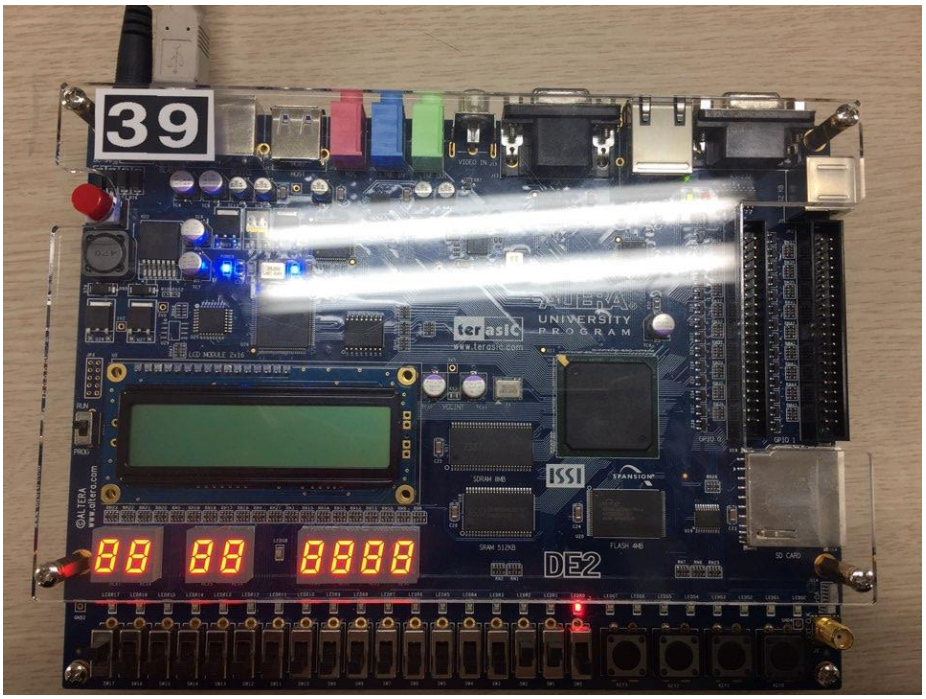
4. 실습보드 적용 결과

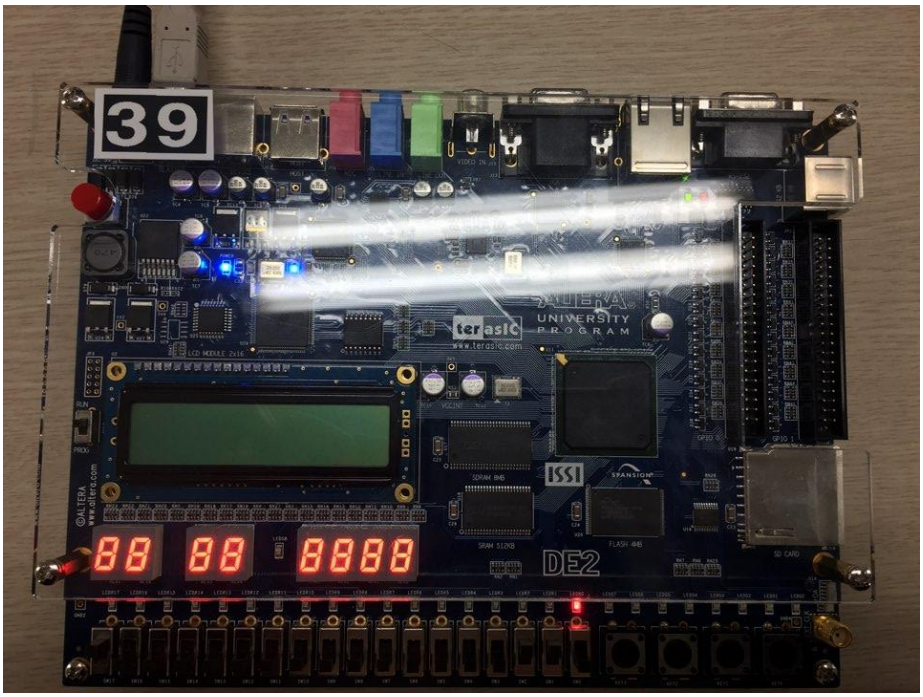
해당 소스를 보드에 다운로드하여 예상한 실행 결과에 맞게 동작하는 지 테스트하시오.

실습 기간 중 모든 sof 파일에 대해 테스트하지 못했습니다. 툴에 좀 더 익숙해져서 모든 실험에 대한 사진을 첨부할 수 있도록 하겠습니다.

Multiplexer / Demultiplexer / Decoder

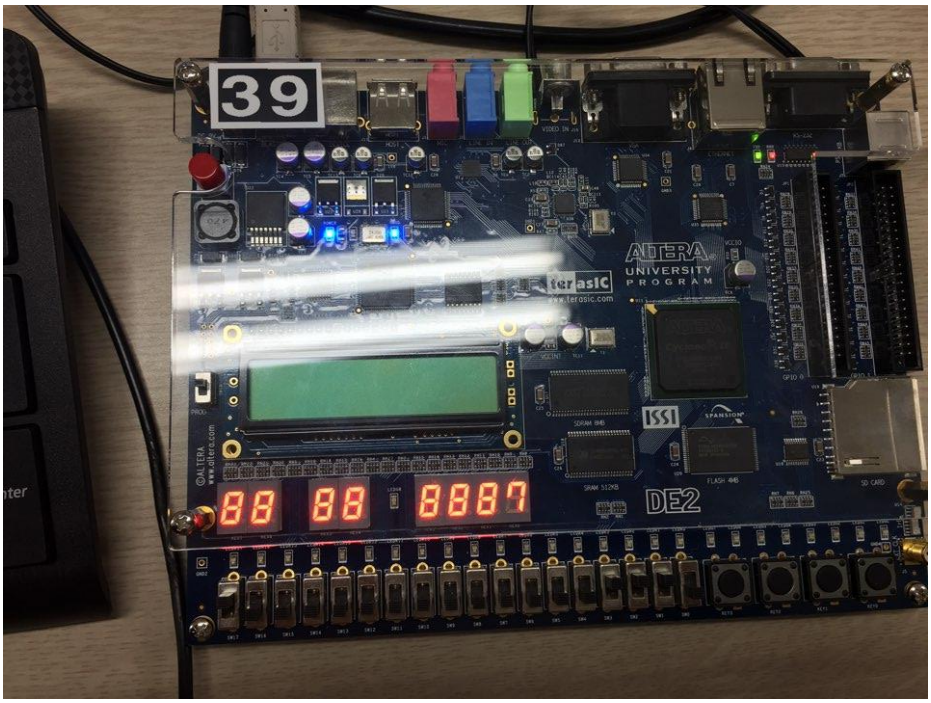
스위치를 동작해보며 각각의 경우에 대해 LED 점등으로 설계한 대로 움직이는 지 확인할 수 있었습니다.

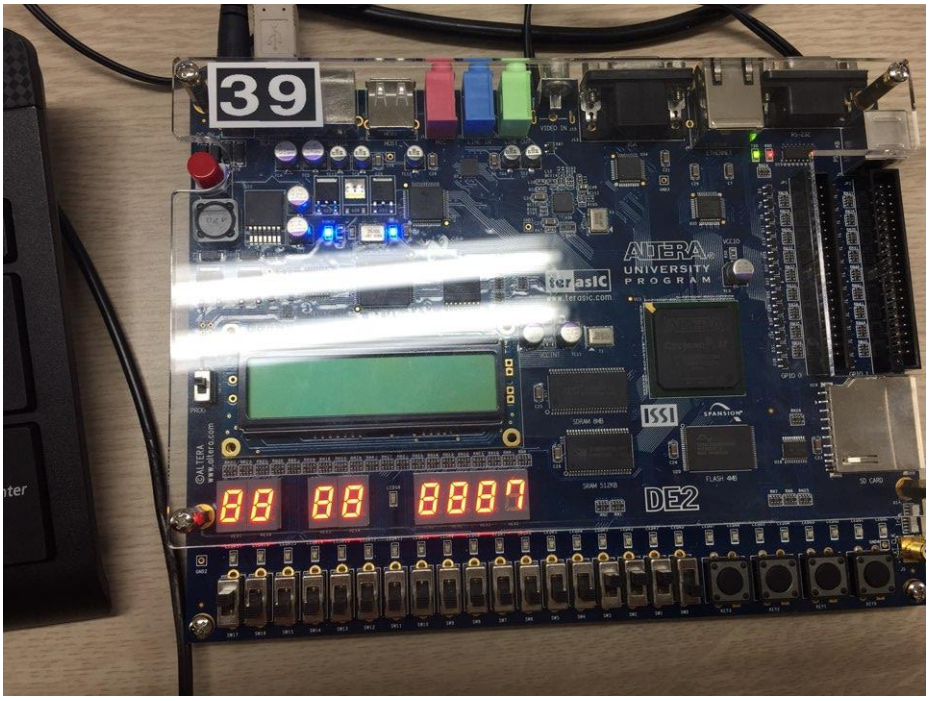




FND Decoder

스위치를 동작해보며 각 경우에 숫자가 올바르게 표시되는 지, 범위를 벗어난 입력에 대해서 ‘1111111’이 올바르게 동작하는 지 테스트 하였습니다.





5. 실습소감

실습을 통해 경험한 것을 자유롭게 서술하시오.

이전 디지털 공학 및 실습 강의를 들으면서 MUX, DEMUX 및 Decoder 에 대한 내용을 공부했었는데 이번 실습을 하며 눈에 보이는 결과물로 회로에서 어떤 역할을 하는 지, 어떤 결과를 보이는 지 이해할 수 있었다. 하지만 아직도 Schematic 도면은 이해하기 어려움이 있다. 오히려 처음 배우며 익숙하지 않은 VHDL 을 사용해 머리속으로 그려지지도 않은 도면을 얻어낼 수 있었다. 이렇게 입력과 선택조건을 통해 결과물을 예측하고 이를 기술해 도면을 얻어냈고 이렇게 작성하는 것이 더 쉽고 편하다라는 점을 느꼈다. 이를 통해 VHDL 의 탄생배경에 대해 다시금 이해할 수 있게 되었다.