

---

# VHDL 및 실습

Signal vs Variable / ROM & RAM

---

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

## 1. 주제 및 배경이론

**i** VGA Display 에 대해 알고 이를 통해 원하는 결과를 얻어낼 수 있는 방법에 대해 살펴보자.

### Signal 과 Variable

VHDL 을 이용한 설계에서 Signal 은 우리가 일반적으로 알아온 컴파일러의 방식으로 동작하지 않는다. Signal 과 Variable 은 둘 다 디지털 신호를 나타내는 하나의 변수로서의 역할을 한다. 하지만 그 동작에서 차이를 가지는데 일반적으로 signal 로 선언된 변수들은 마지막 라인에 동시에 적용이 되므로 한클럭 전의 신호를 통해 연산이 적용되게 된다. 따라서 원하는 결과를 얻기 위해서는 변수의 값이 라인 별로 변경되어야 하는데 이를 가능하게 하는 것이 바로 Variable 변수이다. 이 변수는 라인 별로 값이 변경되어 우리가 익히 사용하던 다른 언어와 동일하게 적용되게 된다. 그렇다면 둘의 차이는 어떻게 나타날까 둘의 특징에 대해 알아보자.

#### Signal

- 순차코드 내부에서 사용된 경우 값의 갱신이 즉각적이지 않음
- 새로운 값의 갱신은 해당 signal 에 값이 할당되는 process, procedure, function 이 종결되는 시점에 이뤄진다.
- Signal 에 값이 다중으로 할당될 경우 컴파일러 경고 혹은 마지막의 값만 적용되어 합성되므로 주의하여야 한다.
- '<=' 기호를 사용하여 할당

#### Variable

- 낱위과 대조적으로 선언된 process, function, procedure 내에서만 사용되며 외부로 전달될 수 없음
- 순차코드 내부에서만 사용되며 값의 갱신이 즉각적으로 이루어지므로 새로운 값이 해당영역의 종결과 관계없이 바로 사용이 가능함
- ':=' 기호를 사용하여 할당

	SIGNAL	VARIABLE
할당	<=	:=
활용	회로의 상호연결 표현	지역적 정보(값)를 표현
유효범위	전역적 (코드 전체에서 사용가능)	지역적 (해당 process, function, procedure 내부에서만 사용 가능)
동작	순차코드 내에서는 갱신이 즉각적으로 이루어지지 않고 해당 process, function, procedure 가 종결될 때 값이 갱신됨	즉각적으로 값이 갱신되어 코드의 바로 다음 행에서 바로 사용이 가능
사용처	Package, entity, architecture 에서 사용되며 entity 의 모든 port 는 기본적으로 signal 임	순차코드 내부에서만 사용

### RAM & ROM

이번 실습에서는 RAM 과 ROM 이 가지는 의미는 중요하지 않지만 가볍게 알아보도록 하자.

#### RAM(Random Access Memory)

- 이름 그대로 메모리 주소를 가진 채 어느 곳이든(random) 접근하여 읽고 쓸 수 있다.

#### ROM(Read Only Memory)

- 초기에 한 번만 쓰여진 채 이후에는 읽어오기만 가능하다.

위의 내용 정도를 이해하고 VHDL 을 작성한다면 이해가 좀 더 쉽게 될 것이다.

## 2. 소스코드 및 코드 설명(3, 4 단계 함께 기술)

**i** 소스코드를 설명과 함께 기술한다.

## 3. 시뮬레이션 결과 및 설명(3, 4 단계 함께 기술)

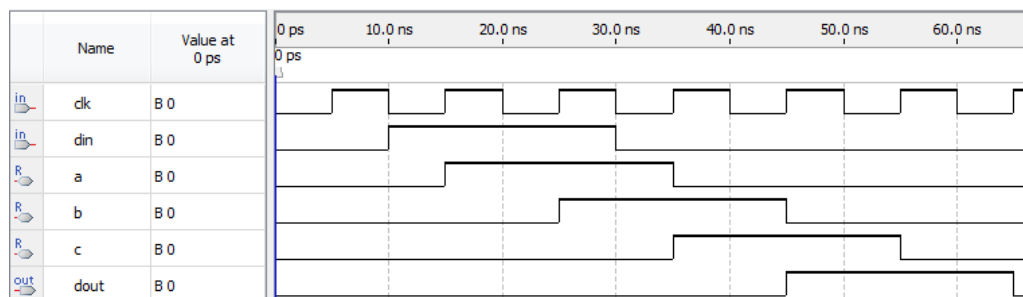
**i** 코드 시뮬레이션과 그 결과를 기술한다.

## SIGNAL 을 이용한 시프트레지스터 설계

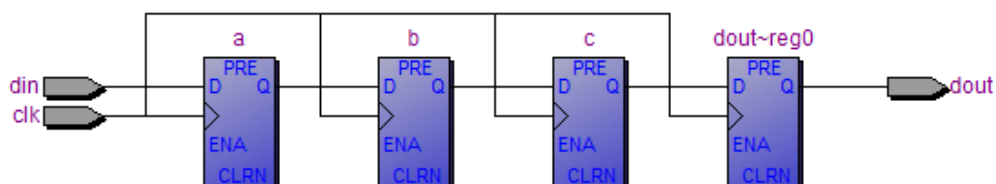
### 1. VHDL 작성

```
1 library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.std_logic_arith.all;
4   use ieee.std_logic_unsigned.all;
5
6 entity shift is
7   port (
8     clk : in std_logic;
9     din : in std_logic;
10    dout : out std_logic
11  );
12 end shift;
13
14 architecture BEH of shift is
15   signal a, b, c : std_logic;
16
17 begin
18   process(clk)
19   begin
20     if rising_edge(clk) then
21       a <= din;
22       b <= a;
23       c <= b;
24       dout <= c;
25     end if;
26   end process;
27 end BEH;
```

### 2. Function Simulation



### 3. RTL Viewer

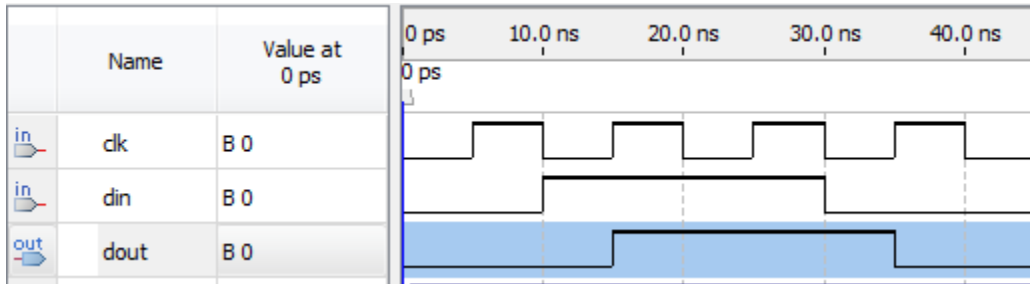


# VARIABLE 을 이용한 시프트레지스터 설계

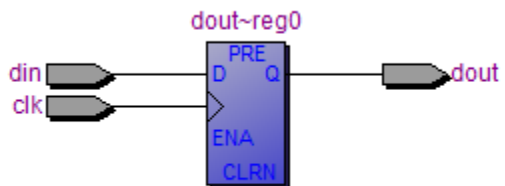
## 1. VHDL 작성

```
1 library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.std_logic_arith.all;
4   use ieee.std_logic_unsigned.all;
5
6 entity shift is
7   port (
8     clk : in std_logic;
9     din : in std_logic;
10    dout : out std_logic
11  );
12 end shift;
13
14 architecture BEH of shift is
15 begin
16   process(clk)
17     variable a, b, c : std_logic;
18   begin
19     if rising_edge(clk) then
20       a := din;
21       b := a;
22       c := b;
23       dout <= c;
24     end if;
25   end process;
26 end BEH;
```

## 2. Function Simulation



## 3. RTL Viewer



## ROM 설계

## 1. VHDL 작성

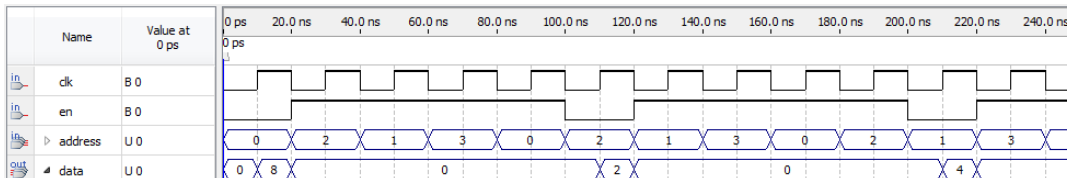
```

1 library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.std_logic_arith.all;
4   use ieee.std_logic_unsigned.all;
5
6 entity ROM_4x8 is
7   port (
8     clk : in std_logic;
9     en : in std_logic;
10    address : in std_logic_vector(1 downto 0);
11    data : out std_logic_vector(7 downto 0)
12  );
13 end ROM_4x8;
14
15 architecture BEH of ROM_4x8 is
16   type mem_array is array(0 to 3) of std_logic_vector(7 downto 0);
17   signal mem : mem_array;
18 begin
19   process(clk, address)
20     variable index : integer range 0 to 3 := 0;
21   begin
22     mem(0) <= x"08";
23     mem(1) <= x"04";
24     mem(2) <= x"02";
25     mem(3) <= x"01";
26     index := conv_integer(address (1 downto 0));
27     if(en = '1') then
28       data <= (others => '0');
29       elsif rising_edge(clk) then
30         data <= mem(index);
31       end if;
32     end process;
33 end BEH;

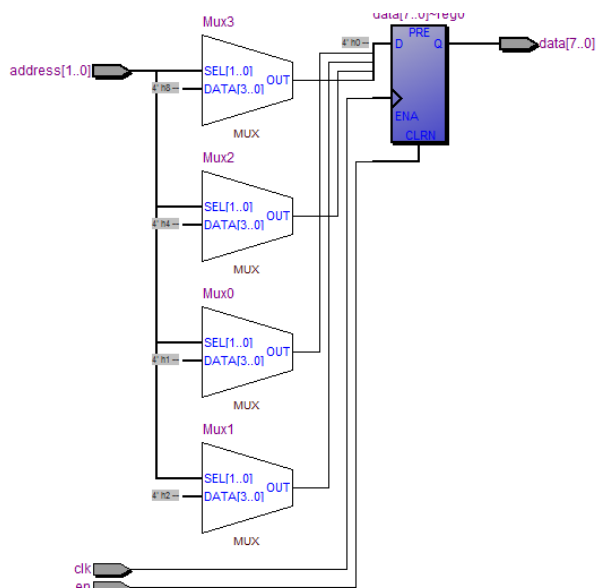
```

Colored by Color Scripter

## 2. Function Simulation



### 3. RTL Viewer



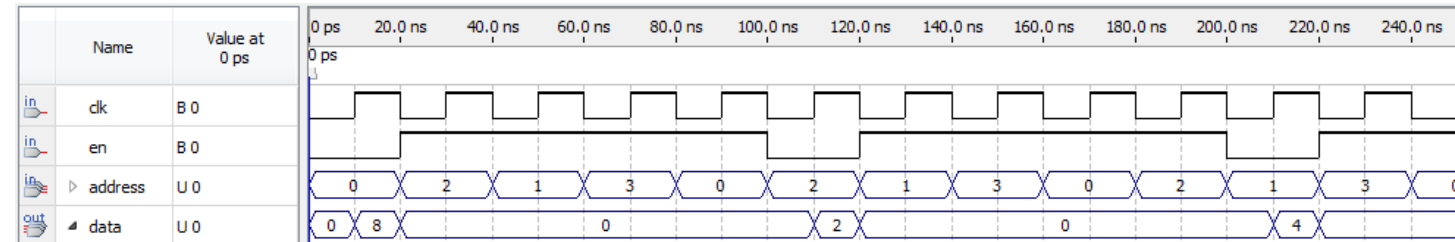
# RAM 설계

## 1. VHDL 작성

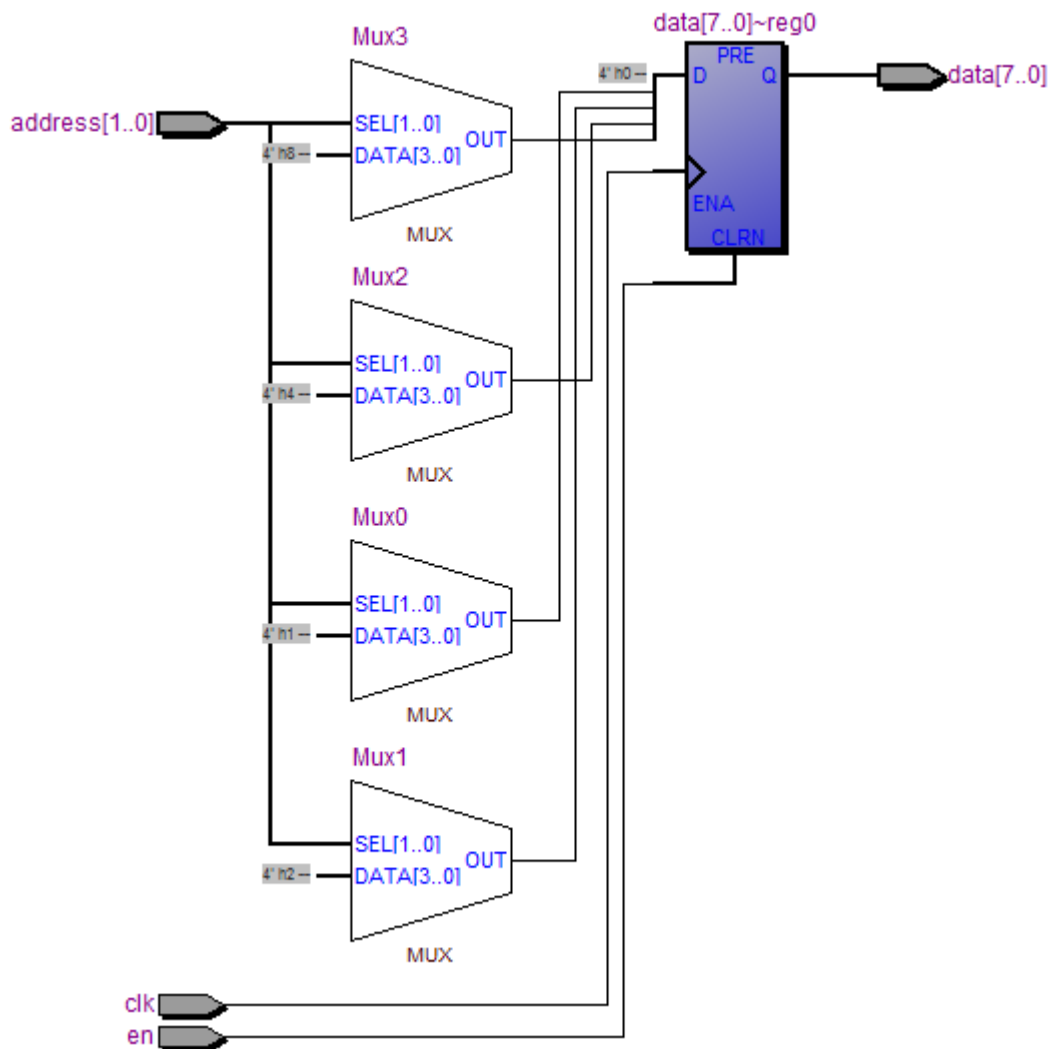
```
1 library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.std_logic_arith.all;
4   use ieee.std_logic_unsigned.all;
5
6 entity DPRAM_8x8 is
7   port (
8     clk_l : in std_logic;
9     wrn : in std_logic;
10    waddr : in std_logic_vector(2 downto 0);
11    data_in : in std_logic_vector(7 downto 0);
12    clk_r : in std_logic;
13    rdn : in std_logic;
14    raddr : in std_logic_vector(2 downto 0);
15    data_out : out std_logic_vector(7 downto 0)
16  );
17 end DPRAM_8x8;
18
19 architecture BEH of DPRAM_8x8 is
20   type mem_tbl is array(0 to 7) of std_logic_vector(7 downto 0);
21   signal mem : mem_tbl;
22 begin
23   write : process(clk_l)
24     variable I_A : integer range 0 to 7 := 0;
25   begin
26     I_A := conv_integer(waddr);
27     if rising_edge(clk_l) then
28       if (wrn = '0') then
29         mem(I_A) <= data_in;
30       end if;
31     end if;
32   end process;
33
34   read : process(clk_r)
35     variable I_A : integer range 0 to 7 := 0;
36   begin
37     I_A := conv_integer(raddr);
38     if falling_edge(clk_r) then
39       if (rdn = '0') then
40         data_out <= mem(I_A);
41       else
42         data_out <= (others => '0');
43       end if;
44     end if;
45   end process;
46 end BEH;
```

Colored by Color ScripterCS

## 2. Function Simulation



### 3. RTL Viewer



### 4. 실습보드 적용 결과

**i** 소스를 보드에 다운로드하여 예상한 결과에 맞게 동작하는 지 테스트하십시오.


금주 과제에 대한 실습은 진행하지 못했습니다. 하지만 코드 시뮬레이션을 통해 signal 과 variable 가 각각 어떤 역할을 하는 지 이해할 수 있었습니다.

### 5. 실습소감

**i** 실습을 통해 경험한 것을 자유롭게 서술하십시오.

금주 수업을 통해 이전에 원하는 방식으로 동작하지 않던 코드에 대한 원인을 알 수 있었고 이제는 원하는 결과에 맞는 신호 변수를 선언하여 사용할 수 있을 것 같다. 따라서 이후 수업에서는 signal 과 variable 변수를 잘 구분하여 원하는 동작을 할 수 있도록 잘 판단하여 사용해야겠다.

## 6. 문의 사항

 실습하다 겪은 어려웠던 점을 기술하시오.

DPR0M\_8x8 실습 시, natural 키워드 사용 시 원하는 동작을 하지 않는 것을 확인하였습니다.

```
1 variable I_A : natural;cs
```

때문에 아래와 같이 수정하여 사용하였습니다.

```
1 variable I_A : integer range 0 to 7 := 0;cs
```

Natural 을 그저 자연수를 지칭하는 말로 이해하고 수업을 마쳤으나 이러한 문제점은 예상하지 못했습니다. 어떠한 이유 때문에 동작하지 않았는 지 궁금합니다.