2장 VHDL의 기본구성

제1절 VHDL의 기초 표현

■ VHDL에서의 주석문(comments)

- 설계의 내용을 설계자가 쉽게 이해할 수 있도록 기술한 것으로 VHDL 컴파일러(compiler)입장에서는 무시하고 넘어가는 부분.
- 두 개의 하이픈(hyphen) 즉, "--"으로 시작하고, 그 끝 줄에서 종료
- 주석문의 활용

[예제 2.1] 주석문의 활용 예

```
1  process(rst, clk)
2   begin
3    -- if문
4    if rst='0' then            -- low active reset
5      cnt <= (others => '0');
6       E <= '0'; RS <= '0'; RW <= '0';
7       DB <= "00000000";
8    elsif clk'event and clk='1' then    -- clock's rising edge
9     if 10000 = '1' then
10      -- if 100 = '1' then
11       cnt <= cnt+1;
12       if cnt = 1 then
13          E <= '1'; RS <= '0'; RW <= '0';
14          DB <= "00111000";          -- function
15       elsif cnt =3 then
16          E <= '1'; RS <= '0'; RW <='0';
17          DB <= "00001110";                -- DB : display on
18       end if;
```

제1절 VHDL의 기초 표현

▪ 식별자(identifiers)
이름을 의미, 즉 VHDL 코드(code)를 구성하는 엔티티(entity), 아키텍처(architecture), 포트(port) 등 여러 가지 기능을 정의하여 식별하기 위한 공백이 없는 문자열

▪ 식별자 활용
[예제 2.2] 식별자의 활용(4bit 비교기)

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  -- entity structure
4  entity compare_4 is
5     port(x, y : in std_logic_vector( 3 downto 0);
6          equal : out std_logic);
7  end compare_4;
8  -- 아키텍처 몸체
9  architecture equal_logic of compare_4 is
10    begin
11      equal <= '1' when (x=y) else '0';
12 end equal_logic;
```

▪ 식별자의 규칙
① 첫 번째 문자는 반듯이 영문자(a ~ z)로 시작, 두 번째 문자부터 영문자, 숫자(0 ~ 9) 및 밑줄 문자 등의 혼합이 가능
② 마지막 문자는 밑줄 문자와 밑줄 문자를 두 개 이상 연속 사용 불가
③ 대문자, 소문자의 구별은 없으며, VHDL구문의 예약어는 식별자로 사용 불가

제1절 VHDL의 기초 표현

▪ 올바른 식별자의 사용
[예제 2.3] 올바른 식별자의 사용

♂ 올바른 식별자
FET
Decoder_4
sig_N
compare_eq
main_clock

♂ 잘못된 식별자
① _ Decoder_4 -- 첫 문자는 영문자로 시작
② 2FET_2 -- 첫문자는 영문자로 시작
③ sig_*R -- 문자, 숫자, 밑줄 문자로만 구성
④ MOS-FET -- 문자, 숫자, 밑줄 문자로만 구성
⑤ Decoder_ -- 밑줄 문자가 마지막에 기술되면 안됨
⑥ Main__clk -- 밑중 문자가 연속 2개 사용 불가

제1절 VHDL의 기초 표현

■ 리터럴(literals)
VHDL 코드로 설계를 할 때, 직접적으로 표현하는 값 또는 문자를 말함.

▪ 리터럴(literal)의 사용

[예제 2.4] 리터럴의 사용

```
1    elsif cnt =3 then
2      E <= '1'; RS <= '0'; RW <= '0';
3      DB <= "00001100";
```

▪ 리터럴의 표현법

숫자(numbers)의 표현: 숫자는 10진수 표현(decimal literals)과 2진, 8진, 16진과 같이 밑 수(base)에 따라 표시되는 방법(based literals)이 있으며, 숫자에 소수점이 있으면 실수(real literal), 없으면 정수(integer literal) 표현

[예제 2.5] 10진수에 의한 표현

0 1 123_456_78- **9987E6** -- 정수
0.0 0.5 2.718_28 **12.4E-9** -- 실수

[예제 2.6] 밑 수에 의한 표현

16#FE -- 16진수 : FE = 정수 254
2#1111_1110# -- 2진수 : 11111110 = 정수 254
16#D#E1 -- 16진수 : D=13, E1= 161 = 13 x 161 = 실수 208

제1절 VHDL의 기초 표현

▪ 문자(characters literals)의 표현

[예제 2.7] 문자의 표현

'1', '?', 'B', 'b'

▪ 문자열(strings literals)의 표현

① 단일문자들의 집합인 문자열은 이중 인용 부호(큰 따옴표: " ")에 넣어 표시

② 한 줄을 벗어나는 길이의 문자열은 접합연산자 &를 사용하여 구성

[예제 2.8] 문자열의 표현

"B string", "A string in a strong"

▪ 비트 열(bit string)의 표현

① 비트는 단일 인용 부호(작은 따옴표:' ') 사이에 숫자를 넣어 '1' 또는 '0'으로 표시하고, 비트 열은 이중 인용 부호(큰 따옴표:" ") 사이에 비트 열을 표시

② 비트 벡터의 값을 정의하기 위해 사용하며, 기수를 정의하는 문자가 앞에 표시되는 수열을 나타내고, 2진수는 B, 8진수는 O, 16진수는 X로 시작

[예제 2.9] 비트열의 표현

B"1010110" -- 2진수로 bit의 길이가 7
O"127" -- 8진수로 bit의 길이가 9이며, 2진수로 "001_010_111"과 동등 값
X"FF" -- 16진수로 bit길이가 8이며, 2진수로 "1111_1111"과 동등 값

제2절 VHDL의 설계단위 표현

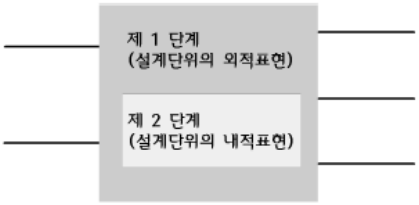
■ 기본 설계단위**(desgin unit)**
VHDL에서 독립적으로 분석할 수 있는 VHDL Source File(또는 VHDL Code)

▪ **VHDL** 설계단위의 분류

- 1단계(Primary unit) : **설계 단위을 외적 관점에서 본 것**
- 2단계(Secondary unit)**: 설계 단위을 내적 관점에서 본 것**

[표 2.1] 설계단위구분

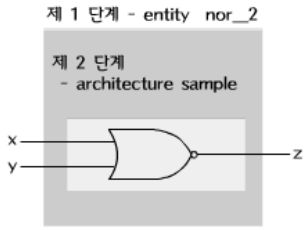
구 분	제1단계(primary unit)	제2단계(secondary unit)
하드웨어 계층	엔티티(entity)	아키텍처 몸체 (architecture body)
	구성(configuration)	
소프트웨어 계층	패키지(package)	패키지 몸체(package body)



[그림 2.1]VHDL의 기본 설계단위

제2절 VHDL의 설계단위 표현

■ 설계단위의 **VHDL** 구문 예



[그림 2.2] **2입력 NOR게이트 설계**

[예제 2.10] 설계단위의 예
<VHDL 구문>

```
1  entity nor_2 is                  -- Primary unit
2     port (x, y : in std_logic;          -- NOR 게이트의 외적 표현
3           z : out std_logic);
4  end nor_2;
5  architecture example of nor_2 is   -- secondary unit
6     begin
7       z <= x nor y;                    -- NOR 게이트의 내적 표현
8  end example;
```

제2절 VHDL의 설계단위 표현

■ VHDL의 기본 구성
- **필수적인 설계단위 - 엔티티(entity)와 아키텍처 몸체(architecture body)**

① 제1단계 : 엔티티(entity)를 선언
- 하드웨어 외부의 입・출력 Interface를 정의한 디지털 시스템의 본체
- 하드웨어 블럭의 이름과 입・출력 port를 선언

② 제2단계 : 아키텍처 몸체(architecture body)를 표현
- 디지털 시스템인 하드웨어 내부를 표현
- 내부회로의 연결, 동작 또는 구조 등을 표현

- VHDL의 기본구조

VHDL의 기본 구조

Entity 선언

```
entity nand_2 is
   port (a,b : in bit; y  : out bit);
end nand_2
```

Architecture

```
architecture example of nand_2 is
   begin
        y <= a nand b ;
end example
```

[그림 2.3] VHDL의 기본 구조

제3절 엔티티(entity)의 구조

■ 기본 형식
- 엔티티(entity) 선언
  설계할 영역의 이름과 외부 환경과의 입・출력 정보를 나타낸 포트(port) 부분을 합한 것

- [형식 2.1] 일반적 형식

```
entity 엔티티_이름 is
  port( 포트_이름 : [모드] 자료형;
        포트_이름 : [모드] 자료형;
        포트_이름 : [모드] 자료형
      );
end 엔티티_이름;
```

제3절 엔티티(entity)의 구조

[예제 2.11] 엔티티 선언(NAND게이트)

♪ 엔티티(entity)의 영역

[그림 2.5] entity nand_2 영역

♪ NAND게이트의 엔티티 선언

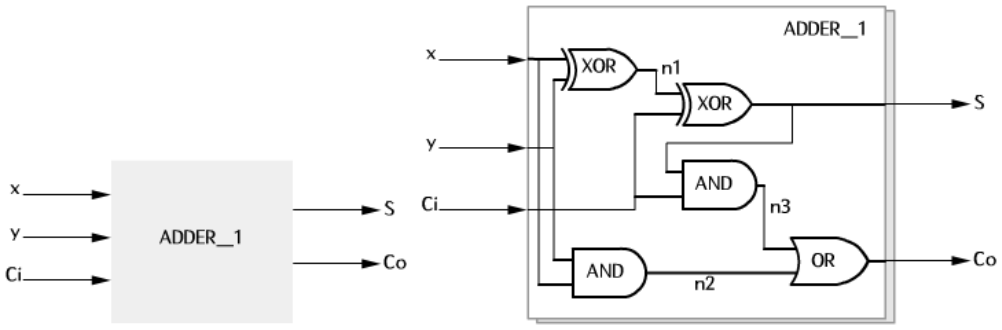
<VHDL 구문>

```
1  entity nand_2 is          -- 하드웨어 블록 이름이 nand_2
2       port ( x, y : in std_logic;   -- 단자(port)신호: x, y는 입력, bit형 데이터
3              z : out std_logic );   -- z는 출력, bit형 데이터
4  end nand_2 ;              -- nand_2의 마감
```


제3절 엔티티(entity)의 구조

[예제 2.12] 엔티티 선언(1bit full_adder)

♪ 엔티티(entity) 영역과 회로도



(a) 엔티티 영역 (b) 회로도

[그림 2.6] 1비트(bit) 전가산기 회로

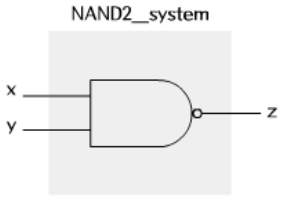
♪ VHDL의 엔티티 선언

```
1  entity adder_1 is
2       port ( X, Y, Ci : in std_logic;
3              S : inout std_logic;
4              Co : out std_logic);
5  end adder_1;
```

제3절 엔티티(entity)의 구조

■ 엔티티의 기본활용

[예제 2.13] 엔티티 선언(2_입력 NAND게이트)



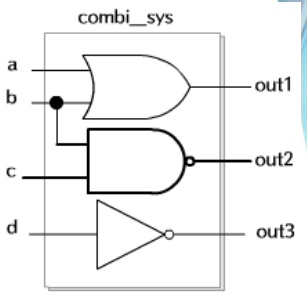
[그림 2.8] 2_입력 NAND게이트

<VHDL 구문>

```
1  entity NAND2_system is
2        port ( x, y : in std_logic;
3                z : out std_logic );
4   end NAND2_system ;
5  architecture example of NAND2_system is
6         begin
7            z <= x nand y ;
8   end example ;
```


제3절 엔티티(entity)의 구조

[예제 2.14] 엔티티의 구조(간단한 조합논리회로)



[그림 2.9] 간단한 조합논리회로

<VHDL구문>

```
1  entity combi_sys is
2        port ( a, b, c, d : in std_logic ;
3                  out1, out2, out3 : out std_logic);
4   end combi_sys ;
5  architecture example of combi_sys is
6        begin
7            out1 <= a or b ;
8            out2 <= b nand c ;
9            out3 <= not(d);
10  end example ;
```

제3절 엔티티(entity)의 구조

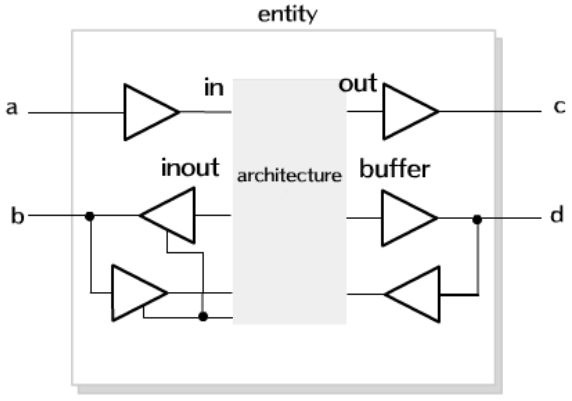
■ 포트(port) 기능의 이해
하드웨어 부품(component)상의 단자들을 표현하며 포트의 이름, 신호의 흐름 및 자료형으로 나타내는 부분으로 엔티티 내 외부 신호선의 연결상태를 기술

[형식 2.2] 포트(port)의 일반적 형식
port(포트_이름, 포트_이름 : [모드] 자료형;
포트_이름,포트_이름 : [모드] 자료형);

[예제 2. 15] 포트(port)의 활용
port(X, Y, Ci : **in** std_logic;
S : **inout** std_logic;
Co : **out** std_logic);

제3절 엔티티(entity)의 구조

■ 모드(mode)의 종류
- **in(안으로)** : 입력(input)으로 신호가 해당 엔티티로 들어가는 경우에 사용
- **out(밖으로)** : 출력(output)으로 해당 엔티티에서 신호가 출력되는 경우에 사용
- **inout(안 밖으로)** : 입/출력(input/output)으로 해당 엔티티에서 신호가 양방향으로 사용
- **buffer(밖으로, 되 읽음)** : 출력기능과 같으나, 단지 자신의 신호를 되 읽는 경우에 사용



[그림 2.10] 포트(port)에서 모드(mode)의 종류

제3절 엔티티(entity)의 구조

▪ 포트 신호(port signal)의 자료형태
  ▪ 자료형에 대한 bit와 bit_vector
  ♂ bit : signal의 개수가 1개인 경우에 사용
  ♂ bit_vector: signal의 개수가 여러 개인 경우에 사용
  ♂ 오름차순: bit_vector(0 to 7), 내림차순: bit_vector(7 downto 0)

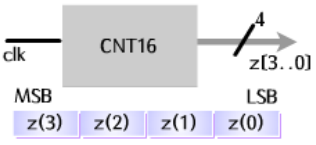
[그림 2.11] 포트(port)에서 자료의 형태

```
1   entity block_port is
2         port( x, c :  in        std_logic;
3               y :  in        std_logic_vector (7 downto 0);
4               z :  out       std_logic_vector (3 downto 0);
5               d :  out       std_logic);
6   end block_port;
                  ▲          ▲
              모드(mode)   형태(type)
```


제3절 엔티티(entity)의 구조

▪ bus signal의 내림차순과 오름차순
♂ **downto** : 내림차순 표현

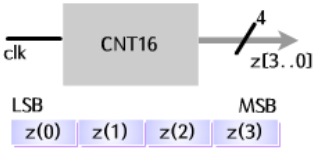
```
port ( clk : in std_logic ;
       z  : buffer std_logic_vector (3 downto 0)) ;
```




```
z <="0011" ;    -- z(3)=0, z(2)=0, z(1)=1, z(0)=1
```

♂ **to** : 오름차순표현

```
port ( clk : in std_logic ;
       z  : buffer std_logic_vector (0 to 3)) ;
```



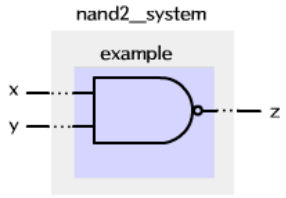

```
z <="0011" ;    -- z(0)=0, z(1)=0, z(2)=1, z(3)=1
```

[그림 2.12] bus signal의 내림차순과 오름차순

제4절 아키텍처(architecture)의 구조

■ 기본 형식
- 아키텍처(architecture)
  - 설계할 회로의 실질적인 내부 동작 또는 각 부품들 사이의 연결구조를 기술하는 부분.
  - 하나의 엔티티 선언에는 여러 개의 아키텍처 몸체(architecture body)가 연결 가능.
  - 아키텍처 기술방법은 동작적・자료흐름적 기법과 구조적 기법.

[예제 2.16] 아키텍처 활용(NAND게이트)



[그림 2.13] 아키텍처 example의 영역

<VHDL구문> 아키텍처 몸체 선언

```
1  architecture example of nand2_system is -- nand2_system의
                                           -- 회로의 이름이 example
2       begin                              --내용의 시작
3            z <= x nand y;                -- z에 x nand y를 대입
4  end example;                            -- example의 끝
```


제4절 아키텍처(architecture)의 구조

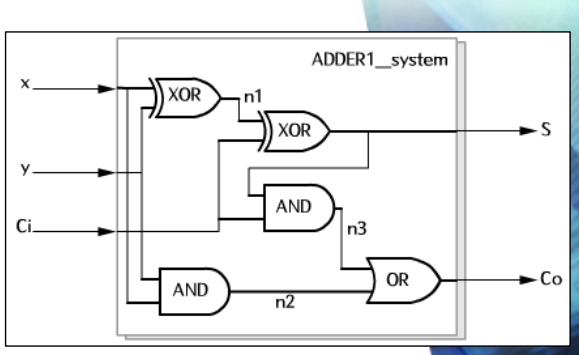
- 아키텍처(architecture)의 일반적 형식

[형식 2.3] 일반적인 형식

```
architecture 아키텍처_이름 of 엔티티_이름 is
  {선언문}
 begin
  {내부적 동작표현}
end 아키텍처_이름;
```

[예제 2.17] 아키텍처의 활용(1bit full_adder)

아키텍처의 영역과 회로도



[그림 2.14] 1비트(bit) 전가산기 회로

VHDL의 아키텍처 구문(선언문이 없는 경우)

```
architecture combi_logic of ADDER1_system is
  begin
   S <= (X xor Y) xor Ci;  -- 덧셈 결과
   Co <= (X and Y) or (S and Ci);
end combi_logic;
```

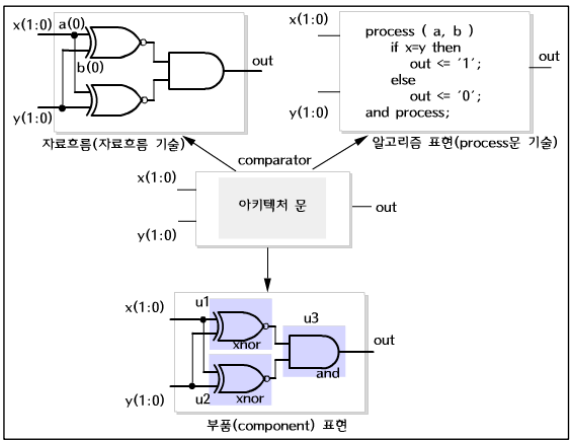
VHDL의 아키텍처 구문(선언문이 있는 경우)

```
1  architecture combi_logic of ADDER1_system is
2    signal n1, n2, n3 : std_logic;   -- n1, n2, n3을 신호로 선언
3    begin
4      n1 <= X xor Y;          -- n1 : 내부 신호 연결
5      S <= n1 xor Ci;         -- 덧셈 결과
6       n2 <= X and Y;         -- n2 : 내부 신호 연결
7       n3 <= S and Ci;        -- n3 : 내부 신호 연결
8      Co <= n2 or n3;         -- 자리올림수 결과
9   end combi_logic;
```

제4절 아키텍처(architecture)의 구조

- 아키텍처의 기술방법
  - 동작적 표현(behavioral description)
  ① 자료 흐름 표현(data flow description) : 부울대수, 논리연산자를 이용하여 자료흐름적으로 기술하는 방법
  ② 동작적 표현 : 디지털 시스템의 동작을 알고리즘으로 기술하는 방법
  - 구조적 표현(structural description)
  ① 디지털 시스템인 하드웨어에서 컴포넌트(component)의 상호연결로 표현
  ② 하드웨어 표현에 매우 가까운 표현 방식으로 계층구조의 하드웨어 설계에 유리

[예제 2.18] 아키텍처 표현방법



[그림 2.15] architecture body의 표현방식