ALTERA Quartus II Web Edition 8.0sp1의 VHDL

한밭대학교 컴퓨터공학과
이 재 흥 교수

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL의 역사

-VHDL(Very High Speed Integrated Circuit Hardware Description Language)

-첫 글자 'V'는 VHSIC(Very High Speed Integrated Circuit)로서 미국방성의 고속 IC 칩을 만들려는 프로그램 때문에 사용되었다. 그리고 'HDL'은 상위 시스템 레벨에서부터 하위 게이트 레벨까지 하드웨어를 기술하고 설계하도록 하는 하드웨어 설계 언어(Hardware Description Language)를 의미한다.

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 발달과정

-미 국방성은 1970년 초부터 싹튼 VHSIC 계획 동안 그들이 구입하는 전자장비의 효율적인 관리를 위하여 기술이나 설계의 변화에 관계없는 강력한 HDL을 찾았다.

-1983년 7월 Intermetrics, IBM, TI 등의 합동개발팀은 정부로부터 공식적으로 VHDL을 개발할 것을 허락받아 VHDL 버전 7.2를 1987년 2월에 완성하였다.
-IEEE는 1986년 3월에 자체의 VASG(VHDL Analysis and Standardization Group)를 형성 하여 VHDL을 연구 및 발전시키도록 하였다.

-IEEE는 1987년에 IEEE Standard_1076_1987을 만들어 미 국방성의 인정을 받고 각 EDA 회사들의 지지를 받으면서 표준 VHDL을 내놓게 된다. IEEE는 또한 LRM(Language Reference Manual)을 발간하여 이 VHDL의 사용을 적극 장려하고 있다.

-1993년에 IEEE는 다시 새 버전인 IEEE Standard_1076_1993을 내놓았는데 이것은 버전 IEEE Standard_1076_1087을 수정한 버전이다.

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL의 장점과 문제점

VHDL의 장점

-VHDL은 표준화되어 사용됨에 따라 호환 문제가 없다.
표준화된 VHDL을 이용하여 국가 및 회사간의 설계 정보 교환이 가능하므로 사용이 증가하였다. VHDL은 IEEE가 지원하는 표준일 뿐만 아니라 미국 정부에서 인정한 공인 HDL이고, 유럽에서도 유럽연합(EU)이 결성되기 전에 각 나라가 사용하던 하드웨어 표현 및 문서화를 위한 언어의 표준화에 대해서 고민하던 차에 VHDL이 출현하자 적극적으로 사용하기 시작하였고 전세계적으로 확대되는 계기가 되었다.

-VHDL은 특정한 공정기술(process technology)과 설계방법론(design methodology)과 관계 없이 회로 구현이 가능하다.
설계를 위하여 기술된 내용들을 시뮬레이션 하여 구현하는데 있어서 제조공정(바이폴라, CMOS, nMOS, GaAs)에 상관없이 사용할 수 있다. 따라서 VHDL로 기술한 것을 시뮬레이션하고 합성하면 반도체 제조업체의 특정한 라이브러리와 상관없이 회로구현이 가능하다.

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL의 장점과 문제점

VHDL의 장점

-VHDL은 시스템 레벨에서부터 게이트 레벨까지의 하드웨어의 동작적 표현(behavioral representation) 및 구조적 표현(structural representation)으로 설계할 수 있도록 만들어졌기 때문에 다양한 형태의 대규모 설계가 가능하다.

-VHDL은 Top-Down 설계 방식을 사용하기 때문에 설계 기간이 훨씬 단축되고 설계자가 놓치기 쉬운 여러 단계들에서의 다양한 오류들을 철저히 검증할 수 있다. 따라서 시장에 설계 제품을 적기에 내놓을 수 있으므로 제품 설계 비용을 크게 줄일 수 있다.

.

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL의 장점과 문제점

VHDL의 문제점

-VHDL 언어 자체가 복잡하기 때문에 이해하고 원활히 사용하기 위해서는 많은 시간과 노력이 필요하다.

-Bottom-Up 설계에 익숙한 전통적이고 보수적인 엔지니어들에는 Top-Down 설계 방식은 어려울 수 있다.

-하드웨어 설계를 위한 도구로서 VHDL 자체의 완벽성도 중요하지만 그에 못지 않게 이를 지원하는 도구의 완벽성도 중요하다. 그러나 오늘날의 합성 툴은 VHDL의 모든 구문을 지원하지 않고 일부분만 지원하기 때문에 VHDL의 특징을 모두 이용할 수 없을 뿐 아니라 상업용 설계 도구마다 제공하는 합성의 범위가 다르므로 불편하다.

-하드웨어 엔지니어의 소프트웨어적 설계 기법에 익숙지 못함과 아날로그 시스템 표현의 어려움 등이 있다.

-그러나 최근에는 CAD 업체들의 많은 노력으로 이러한 문제들이 부분적으로 극복되어 가고 있다.

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

1. VHDL 표현 시 고려사항

1. 설계 시작 전에 입출력 사양(input/output specifications)이 확실히 정해져야 한다.

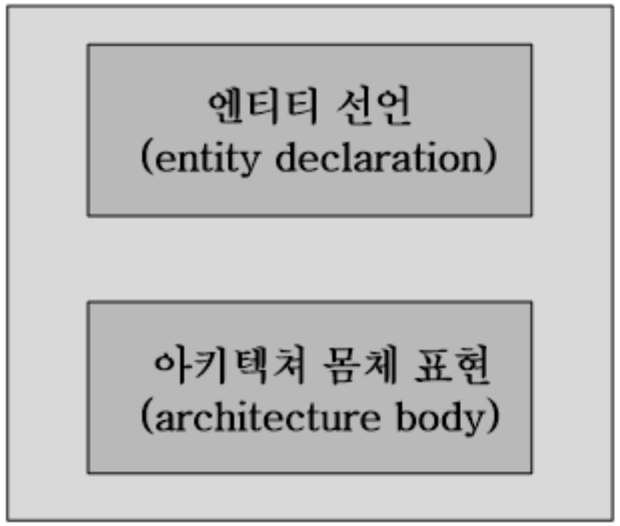
2. 가능한 작은 단위로 프로그래밍 해야 한다.

3. 사용자가 사용하는 VHDL 합성 툴이 제공하는 제한요소들(constraints)을 정확히 알고 사용하여야 한다.

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

2. VHDL 에 의한 설계단계



◆ 1단계 : 엔티티 선언(entity declaration)
- 하드웨어 외부 입출력 인터페이스를 정의
- 하드웨어 블록의 이름과 입출력 포트를 선언

◆ 2단계 : 아키텍쳐 몸체(architecture body) 표현
- 하드웨어 내부를 표현
- 내부회로의 연결, 동작 또는 구조 등을 표현

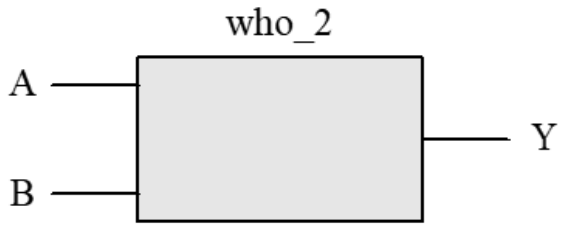
VHDL 설계의 구조

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

2. VHDL 에 의한 설계단계

▪ 2입력 가상 게이트의 엔티티 선언




```
ENTITY who_2 IS
  PORT(A, B : IN bit;
               Y : OUT bit);
END who_2;
```

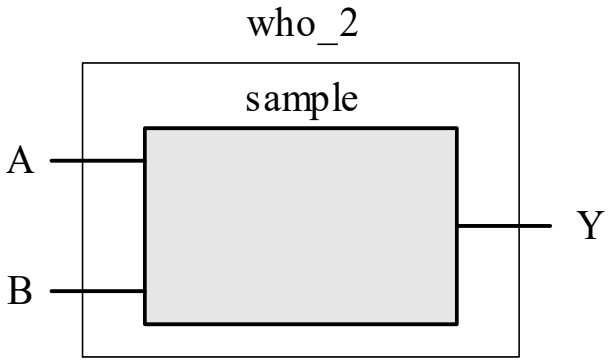
- 엔티티 선언 : ENTITY에서 END로 끝난다.
- 엔티티 이름(entity name)은 설계하고자 하는 회로(또는 컴포넌트)를 잘 표현하는 이름을 사용자가 부여한다.
- 엔티티 내에는 외부와의 인터페이스를 담당하는 PORT를 사용한다.
- 입출력 신호선들을 포트 신호(port signal)라 하고, 이들 신호의 입력 또는 출력 여부는 IN 또는 OUT 모드(mode)에 의해서 결정된다.
- bit 자료형은 신호가 '0' 또는 '1'의 값을 가질 수 있음을 나타낸다.
- 신호와 모드 사이에 콜론(:)을 사용하여 신호의 모드와 자료형을 나타낸다.
- END 다음에 가상 게이트의 엔티티 이름을 다시 한번 써준다.

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

2. VHDL 에 의한 설계단계

2입력 가상 게이트의 아키텍쳐 몸체 표현




```
ARCHITECTURE sample OF who_2 IS
BEGIN
   Y <= A and B;
END sample;
```

❖아키텍쳐 몸체의 시작은 ARCHITECTURE로 시작한다.

❖ARCHITECTURE 다음에 사용자가 정의하는 아키텍쳐 몸체 이름(architecture body name)을 쓰고, 이어서 아키텍쳐 몸체가 소속한 엔티티 이름이 OF 다음에 쓰여진다.

❖실제 동작 표현은 IS 다음의 BEGIN과 END 사이에서 이루어진다.

❖만일 가상 게이트가 AND 동작을 한다면, AND 동작을 위한 연산자 'and'를 이용하여 입력신호 A와 B를 AND 연산하고 그 결과를 출력신호선 Y로 전달하도록 설계한다.

❖END 다음에 가상 게이트의 아키텍쳐 몸체 이름을 다시 한번 써준다.

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

3. VHDL 표현 방법

❖동작적 표현(behavioral representation)
=>가장 높은 상위 레벨의 추상적 표현으로서 시스템의 하드웨어의 구조와는 상관없이 시스템 동작을 프로세스문(process statement)을 사용하여 알고리즘(algorithm) 레벨로 표현한다

❖자료흐름적 표현(dataflow representation)
=>동작적 표현보다는 하드웨어에 가까운 표현 방식으로 주로 부울 함수, 레지스터 전송언어(Resister Transfer Language, RTL), 연산자(operater) 등을 사용하여 표현한다.

❖구조적 표현(structural representation)
=>가장 하드웨어에 가까운 하위 레벨의 표현으로서 컴포넌트(component)들의 상호 연결을 이용하여 표현한다.

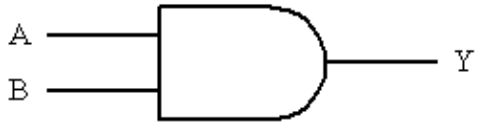
❖혼합적 표현(mixed representation)
=> 앞의 3가지 표현방법들을 혼합하여 표현한다.

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

4. VHDL 표현 예

2입력 AND 게이트의 VHDL 표현



(a) 논리기호

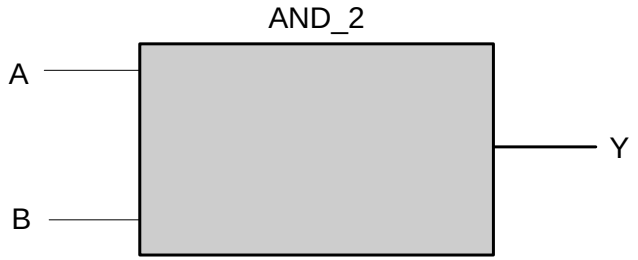
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

(b) 진리표

$Y = AB$

(c) 논리식

2입력 AND 게이트




```
ENTITY AND_2 IS
   PORT(A, B : in bit;
                Y : out bit);
END AND_2;
```

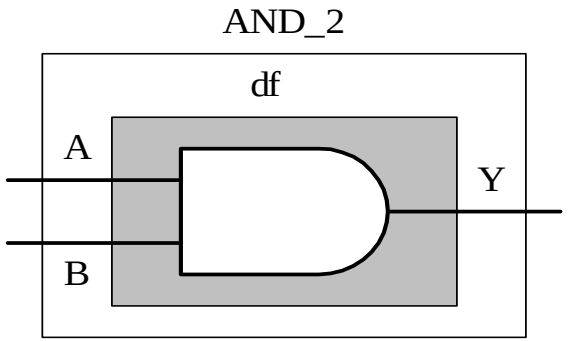
2입력 AND 게이트의 엔티티 선언

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

4. VHDL 표현 예

자료 흐름적 표현의 아키텍처 몸체




```
ARCHITECTURE df OF AND_2
IS
BEGIN
    Y <= A and B;
END df;
```

2입력 AND 게이트

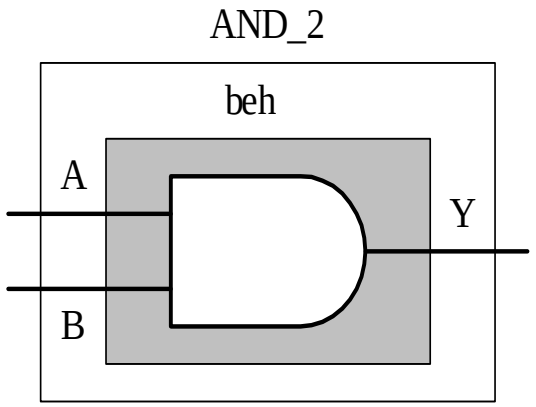
왼쪽 방향의 화살표(<=)는 왼쪽으로 전달되는 값이
자료가 아니라 일정한 지연이 지난 신호이기 때문에 등호 (=)와는 구별하여 사용한다.

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

4. VHDL 표현 예

동작적 표현의 아키텍쳐 몸체




```
ARCHITECTURE beh OF AND_2 IS
BEGIN
  PROCESS(A, B)
  BEGIN
       IF A = '1' and B = '1'  THEN   Y <=
'1';
      ELSE                              Y <= '0';
      END IF;
  END PROCESS;
END beh;
```

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

4. VHDL 표현 예 2입력 AND 게이트의 동작적 표현과 자료흐름적 표현

동작적 표현	자료흐름적 표현

ENTITY AND_2 IS
PORT(A, B : in bit;
Y : out bit);
END AND_2;
ARCHITECTURE beh OF AND_2 IS
BEGIN
PROCESS(A, B)
BEGIN
IF A = '1' and B = '1' THEN Y <= '1';
ELSE Y <= '0';
END IF;
END PROCESS;
END beh;
	ENTITY AND_2 IS
	PORT(A, B : in bit;
	Y : out bit);
	END AND_2;
	ARCHITECTURE df OF AND_2 IS
	BEGIN
	Y <= A and B;
	END df;

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

4. VHDL 표현 예

한편, VHDL 표현에서는 실제적으로 비트 자료형은 잘 사용하지 않고, std_logic 자료형을 사용한다.
왜냐하면 비트 자료형은 '0' 또는 '1' 등만 사용할 수 있으나, std_logic자료형은 '0' 과 '1' 뿐만 아니라,
다른 형태의 값(Z:high impedance, X:unknown)들을 사용할 수 있기 때문이다.

자료형 std_logic을 사용할 경우에는 LIBRARY ieee;와 USE ieee.std_logic_
1164.all;을 반드시 포함하여야 한다.

이후의 모든 VHDL 표현에서는 std_logic 자료형을 사용하기로 한다.

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

4. VHDL 표현 예

2입력 AND 게이트의 동작적 표현과 자료흐름적 표현

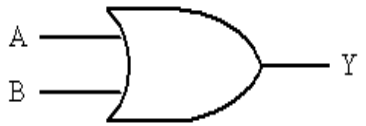
동작적 표현	자료흐름적 표현

LIBRARY ieee; USE
ieee.std_logic_1164.all;
ENTITY AND_2 IS
PORT(A, B : in std_logic;
Y : out std_logic);
END AND_2;
ARCHITECTURE beh OF AND_2 IS
BEGIN
PROCESS(A, B)
BEGIN
IF A = '1' and B = '1' THEN Y <= '1';
ELSE Y <= '0';
END IF;
END PROCESS;
END beh;
	LIBRARY ieee; USE
	ieee.std_logic_1164.all;
	ENTITY AND_2 IS
	PORT(A, B : in std_logic;
	Y : out std_logic);
	END AND_2;
	ARCHITECTURE df OF AND_2 IS
	BEGIN
	Y <= A and B;
	END df;

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

4. VHDL 표현 예 2입력 OR 게이트의 VHDL 표현



(a) 논리기호

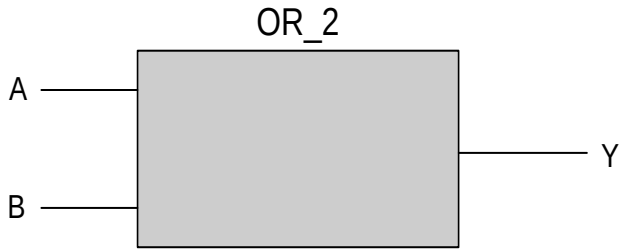
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

(b) 진리표

$Y = A + B$

(c) 논리식

2입력 OR 게이트



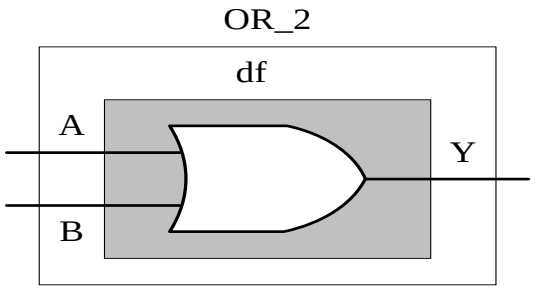

```
LIBRARY ieee; USE ieee.std_logic_1164.all;
ENTITY OR_2 IS
  PORT(A, B : in std_logic;
               Y : out std_logic);
END OR_2;
```

2입력 OR 게이트의 엔티티 선언

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

4. VHDL 표현 예




```
ARCHITECTURE df OF OR_2 IS
BEGIN
   Y <=  A or B;
END df;
```

2입력 OR 게이트의 아키텍쳐 선언

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

4. VHDL 표현 예

동작적 표현	자료흐름적 표현

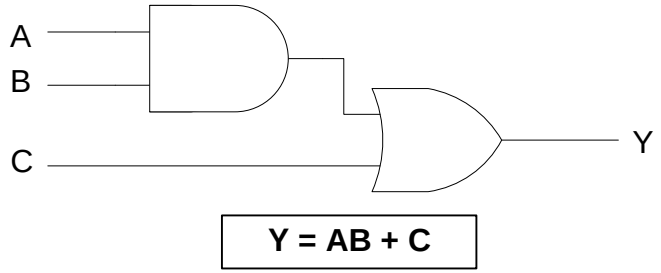
LIBRARY ieee; USE ieee.std_logic_1164.all;
ENTITY OR_2 IS
PORT(A, B : IN std_logic;
Y : OUT std_logic);
END OR_2;
ARCHITECTURE beh OF OR_2 IS
BEGIN
PROCESS(A, B)
BEGIN
IF A = '1' or B = '1' THEN Y <= '1';
ELSE Y <= '0';
END IF;
END PROCESS;
END beh;
	LIBRARY ieee; USE ieee.std_logic_1164.all;
	ENTITY OR_2 IS
	PORT(A, B : IN std_logic;
	Y : OUT std_logic);
	END OR_2;
	ARCHITECTURE df OF OR_2 IS
	BEGIN
	Y <= A or B;
	END df;

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

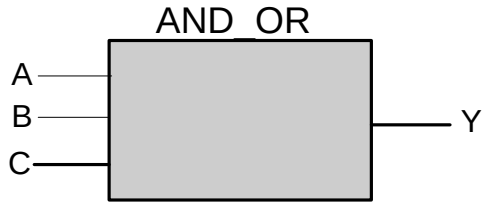
4. VHDL 표현 예

AND_OR 회로의 VHDL 표현



Y = AB + C

논리회로 및 논리식



블록도

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

진리표

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

4. VHDL 표현 예 **동작적 표현과 자료흐름적 표현**

동작적 표현	자료흐름적 표현

LIBRARY ieee; USE ieee.std_logic_1164.all;
ENTITY AND_OR IS
PORT(A, B, C : IN std_logic;
Y : OUT std_logic);
END AND_OR;
ARCHITECTURE beh OF AND_OR IS
BEGIN
PROCESS(A, B, C)
BEGIN
IF C = '1' or (A = '1' and B = '1') THEN Y <= '1';
ELSE
Y <= '0';
END IF;
END PROCESS;
END beh;
	LIBRARY ieee;
	USE ieee.std_logic_1164.all;
	ENTITY AND_OR IS
	PORT(A, B, C : IN std_logic;
	Y : OUT std_logic);
	END AND_OR;
	ARCHITECTURE df OF AND_OR IS
	BEGIN
	Y <= (A and B) or C;
	END df;

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

4. VHDL 표현 예

구조적 표현

먼저 AND_2와 OR_2를 각각 VHDL로 표현하여 저장하여야 한다.

ARCHITECTURE와 BEGIN 사이에는 신호(signal)와 컴포넌트(component) 선언이 이루어진다. 신호 선언(signal declaration)에서 선언된 tmp는 자료형(data type)을 std_logic 형태를 가지며 AND_2와 OR_2 컴포넌트간에 연결되는 내부신호이다.

컴포넌트 선언(component declaration)에서는 시스템 내부에서 사용되는 회로들을 가상적으로 선언하여 인터페이스를 제공한다.

컴포넌트 사례화(component instantiation)문에서는 선언한 컴포넌트들의 지역포트 신호에 실제포트 신호가 연결되도록 해준다.

컴포넌트 사례화문은 레이블(lable)을 반드시 쓰고, 콜론(:) 다음에 컴포넌트 이름을 쓴다.

port map 문에서는 port를 통해서 지역포트(local port) 신호의 값을 실제포트(actual port) 신호에 전달하는 역할을 한다.

AND_2의 지역포트 신호들은 A, B, Y를 말하며 실제포트 신호들은 A, B, tmp가 된다. OR_2의 지역포트 신호들은 A, B, Y를 말하며 실제포트 신호들은 tmp, C, Y가 된다.

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

4. VHDL 표현 예

▪지역포트 신호와 실제포트 신호들을 연결하는 결합방법은 다음과 같이 위치결합(positional association)과 이름결합(named association) 등의 2가지 방법이 있다.

◦ 위치 결합(positional association)

→ 컴포넌트의 지역포트 신호와 실제포트 신호들을 위치 순서대로 연결

```
ANDG : AND_2 port map(A, B, tmp);
```

◦ 이름 결합(named association)

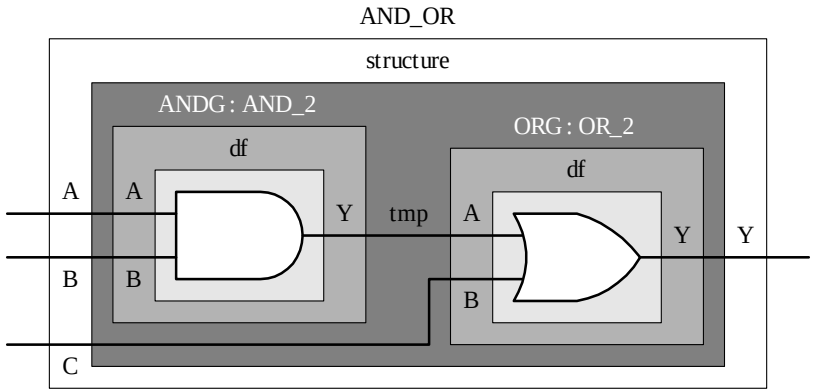
→ 컴포넌트의 지역포트 신호와 실제포트 신호들을 위치와 상관없이 직접 이름으로 연결

```
ANDG : AND_2 port map(A => A, B => B, Y => tmp);
```

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

4. VHDL 표현 예



구조적 표현

```
LIBRARY ieee; USE ieee.std_logic_1164.all;          -- 라이브러리
ENTITY AND_OR IS                                    -- 엔티티 선언
  PORT(A, B, C : IN std_logic;
               Y : OUT std_logic);
END AND_OR;
ARCHITECTURE structure OF AND_OR IS                 -- 아키텍처 몸체
  SIGNAL tmp : std_logic;                             -- 신호선언
  COMPONENT AND_2                                   -- 컴포넌트 선언
     PORT(A, B : in std_logic;  Y : out std_logic);
  END COMPONENT;
  COMPONENT OR_2                                     -- 컴포넌트 선언
     PORT(A, B : in std_logic;  Y : out std_logic);
  END COMPONENT;
BEGIN
  ANDG : AND_2 port map(A, B, tmp);                   -- 컴포넌트 사례화문
   ORG : OR_2   port map(tmp, C, Y);                  -- 컴포넌트 사례화문
END structure;
```

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

4. VHDL 표현 예

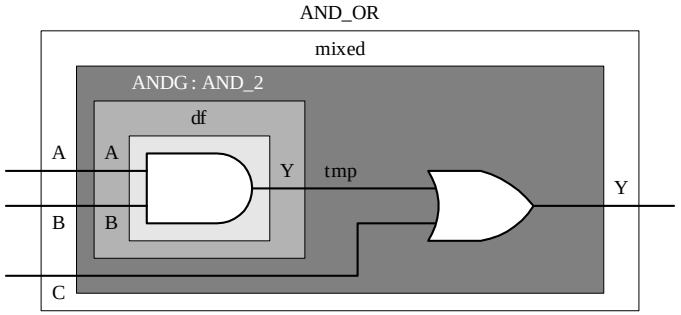
혼합적 표현

- AND_OR 회로를 구조적 표현과 자료흐름적 표현을 사용하여 혼합적으로 표현한다.
- 먼저 AND_2 회로를 VHDL로 표현하고 저장한다.
- 다음에 AND_2 회로를 구조적으로 표현하고 OR_2 회로를 자료흐름적으로 표현한다.
- ARCHITECTURE와 BEGIN 사이에는 신호(signal)와 컴포넌트(component) 선언이 이루어진다.
- 설계하는 AND_OR 회로는 1개의 신호 tmp가 있고 1개의 컴포넌트(AND_2)를 가지고 있다.
- 컴포넌트 AND_2를 사례화하기 위해 1개의 컴포넌트 사례화문이 필요하고 컴포넌트의 출력 Y는 내부신호 tmp에 연결되고, 이 내부신호는 C 신호와 'or' 연산하도록 설계하였다.
- 이 회로는 OR 연산 부분을 자료흐름적 표현으로 설계하였다.

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

4. VHDL 표현 예



혼합적 표현

```
LIBRARY ieee; USE ieee.std_logic_1164.all;
ENTITY AND_OR IS
   PORT(A, B, C : IN std_logic;
                  Y : OUT std_logic);
END AND_OR;
ARCHITECTURE mixed OF AND_OR IS
    SIGNAL tmp : std_logic;
    COMPONENT AND_2                                    -- 컴포넌트 선언
        PORT(A, B : IN std_logic; Y : OUT std_logic);
    END COMPONENT;
BEGIN
    ANDG : AND_2 port map(A, B, tmp);                  -- 컴포넌트 사례화문
    Y <= tmp or C;
END mixed;
```

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

5. VHDL 어휘 요소와 기본 구성

어휘 요소

주석(comment)

주석은 프로그램에 대한 설명을 위해 사용하며 VHDL의 수행에는 영향을 미치지 않는다.
주석은 설명문이라고도 한다.

◎ 주석 표시 : 연자부호(-)를 연속적으로 2개로 구성 "--"

◦ 사용 예

```
ENTITY AND_OR IS                              -- Entity declaration
   PORT(A, B, C : IN std_logic;
                Y : OUT std_logic);
END AND_OR;
ARCHITECTURE Structure OF AND_OR IS       -- Architecture body
   signal tmp : std_logic;                        -- declaration of local signal
```

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

5. VHDL 어휘 요소와 기본 구성

②식별어(identifier)

식별어는 설계자가 정의하는 VHDL에서의 공백을 가지지 않는 문자열(string)을 말한다.
식별어는 대문자와 소문자 구별이 없다.
식별어 만드는 법

◎ 기본법칙 ◎ 기본법칙
- 식별어는 영문자('a' ~ 'z', 'A' ~ 'Z'), 숫자('0' ~ '9') 그리고 밑줄('_')로 구성된다.
- 식별어는 반드시 영문자로 시작해야 한다.
- 식별어는 밑줄(_)로 시작 또는 끝이 나면 안 된다.
- 식별어는 연속적으로 밑줄을 포함해서는 안 된다.

◦ 사용 예

```
A  X0    counter     Next_value      generate_read_cycle
```

◦ 잘못된 사용 예

```
last@value          -- 사용할 수 없는 문자(@)사용
5bit_counter        -- 숫자로 시작
_A0                 -- 밑줄로 시작
A0_                 -- 밑줄로 끝남
clock__pulse         -- 연속적인 밑줄 사용
```

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

5. VHDL 어휘 요소와 기본 구성

식별어 표기 예

```
ENTITY comp IS
  PORT(A, B : IN std_logic;
           E : OUT std_logic);
END comp;
ARCHITECTURE dataflow OF comp IS
BEGIN
  E <= (A AND B) OR (NOT(A) and NOT(B));
END dataflow;
```

엔티티 이름인 comp와 포트문의 신호를 나타내는 A, B, E 등이 식별어이다. 또한 아키텍쳐 몸체 이름인 dataflow도 식별어가 된다.

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

5. VHDL 어휘 요소와 기본 구성

③ 지정어(reserved word)

VHDL 구문을 위해서 미리 지정된 문자열을 지정어라고 한다.
지정어는 대문자와 소문자 구별이 없으며, 사용자는 지정어를
식별어로 사용해서는 안 된다.

```
ABS    ACCESS     AFTER     ALIAS    ALL      AND
ARCHITECTURE       ARRAY       ASSERT         ATTRIBUTE
BEGIN      BLOCK      BODY       BUFFER       BUS
CASE       COMPONENT     CONFIGURATION      CONSTANT
DISCONNECT   DOWNTO
ELSE       ELSIF     END      ENTITY      EXIT
FILE      FOR       FUNCTION        GENERATE    GENERIC    GUARD
ED
IF       IN       INOUT      IS
LABEL        LIBRARY      LINKAGE      LOOP      MAP      MOD
NAND       NEW        NEXT        NOR      NOT      NULL
OF         ON        OPEN        OR        OTHERS     OUT
PACKAGE     PORT       PROCEDURE       PROCESS
RANGE       RECORD       REGISTER       REM        REPORT     RETU
RN
SELECT       SEVERITY       SIGNAL      SUBTYPE
THEN        TO         TRANSPORT             TYPE
UNITS       UNTIL       USE       VARIABLE
WAIT        WHEN       WHILE       WITH       XNOR      XOR
```

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

5. VHDL 어휘 요소와 기본 구성

지정어 표기 예

```
LIBRARY IEEE; USE IEEE.std_logic_1164.all;
ENTITY comp IS
   PORT(A, B : IN std_logic;
                E : OUT std_logic);
END comp;
ARCHITECTURE dataflow OF comp IS
BEGIN
    E <= (A AND B) OR (NOT(A) AND NOT(B));
END dataflow;
```

LIBRARY, USE, ENTITY, IS, PORT, IN, OUT, END, ARCHITECTURE, OF, BEGIN AND, OR 등이 지정어이다.

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

5. VHDL 어휘 요소와 기본 구성

④ 리터럴(literal)

VHDL에서 리터럴은 정수 리터럴(integer literal)과
실수 리터럴(real literal)로 나눌 수 있다.

◎ 정수 리터럴 : 소수점이 없는 10진수

◦ 사용 예

23 0 146

◎ 실수 리터럴 : 소수점을 갖는 10진수

◦ 사용 예

23.1 0.0 3.14159

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

5. VHDL 어휘 요소와 기본 구성

진수 표기(base notation) 가능 : 2진수, 8진수, 16진수

◎ 표시 방법
- 수(number)는 # 문자로 앞과 뒤를 둘러싼다.
- 진수(base)는 맨 앞에 쓴다.

◦ 사용 예
2진수 : 2#1101110#
8진수 : 8#0375#
16진수: 16#FD#

밑줄(underline) 사용 표시

◎ 사용 이유
- 긴 숫자를 읽기 쉽도록 한다.

◦ 사용 예
123_456
3.141_592_6
2#1111_0101_1100_0000#

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

5. VHDL 어휘 요소와 기본 구성

⑤ 문자(character)

문자는 각 문자를 별개로 다루는 것으로 A, &, + , 0 등으로 하나의 문자를 의미한다.

◎ 문자 표시

- 단일 따옴표(single quotation) ‘ ’

◦ 사용 예

```
‘A‘                          -- 대문자
‘z‘                          -- 소문자
‘1‘                          -- 숫자 문자
‘,‘                          -- 콤마(comma) 문자
‘’‘                          -- 따옴표 문자
‘-‘                          -- 연자부호(hyphen) 문자
‘ ‘                          -- 공(space) 문자
```

◦ 문자 표기 예

```
IF S ='0'    THEN
  IF A = '1‘   THEN       Y <= '1';
  ELSE                    Y <= '0';
  END IF;
ELSE                      Y <= 'Z';
END IF;
```

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

5. VHDL 어휘 요소와 기본 구성

⑥ 문자열(string)

문자열은 문자들이 2개 이상으로 구성된 문자를 말한다.

◎ 문자열 표시

- **이중 따옴표**(double quotation) " "

◦ 사용 예

```
"A string"
"000110011"
"XXXXXXXX"
"---"
"  "            -- 공(space) 문자 2개
```

◦ 문자열 표기 예

```
CASE D IS
      WHEN "0001" =>     Y <= "00";
      WHEN "0010" =>     Y <= "01";
      WHEN "0100" =>     Y <= "10";
      WHEN "1000" =>     Y <= "11";
      WHEN OTHERS => Y <= "--";
END CASE;
```

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

5. VHDL 어휘 요소와 기본 구성

한 줄에 문자열 표시가 어려울 때 : 결합 연산자(&)를 사용

◦ 사용 예

```
"If a string will not fit on one line"
&"then we can break it into parts on a separate lines."
```

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

5. VHDL 어휘 요소와 기본 구성

⑦비트 문자열(bit string)

비트 문자열은 비트 문자열 부분과 비트의 진수 부분으로 나눠진다.
비트 문자열은 이중 따옴표(double quotation, " ")로 표시하고 진수 표시는 맨 앞에 하며 대문자와 소문자 구별이 없다.

◎ **진수 표시 방법**

- 2 **진수** : B **또는** b
- 8 **진수** : O **또는** o
- 16 **진수** : X **또는** x

◦ **사용 예**

B"01001110“	b"1111_0101_0000"
O"376"	o"00"
X"FA“	x"0d"

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

5. VHDL 어휘 요소와 기본 구성

⑧특수기호(special symbol)

특수기호는 연산자(operator) 표기, 언어 구문의 부분의 한계를 정하거나 구두점으로서 사용한다. 특수기호는 1개의 문자 또는 2개의 문자로 구성한다. 각 특수기호들은 연산자로서 사용된다.

◎ 한 문자

& ' ( ) * + , - . / : ; < = > |

◦ 사용 예

a & b (a + b) a > b a = b

◎ 두 문자

=> ** := /= >= <= <>

◦ 사용 예

a <= b "00" => F <= in0

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

5. VHDL 어휘 요소와 기본 구성

2) 엔티티 선언과 아키텍쳐 몸체

엔티티 선언

엔티티 선언을 통해 설계하는 논리회로 이름과 입출력 인터페이스를 정의한다.
여기서 엔티티_이름은 VHDL 코드 이름이 된다.
따라서 엔티티_이름은 설계하는 논리회로의 특징을 잘 나타낼 수 있는
형태의 이름을 부여하는 것이 좋다.

엔티티에서는 하드웨어 동작에 필요한 설계 매개변수(design parameter) 전달과
여러 아키텍쳐 몸체에서 공통으로 사용할 것을 선언한다.

엔티티 선언 형식

```
ENTITY 엔티티_이름 IS
   포트문;
END 엔티티_이름;
```

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

5. VHDL 어휘 요소와 기본 구성

②포트문(port statement)

포트문은 하드웨어 컴포넌트 상의 핀 또는 연결된 핀 그룹을 표현하며 엔티티 및 컴포넌트 내에서 정의한다.

형식

```
PORT(포트신호_이름, ... : [모드] 자료형[:=표현];   ... );
```

포트신호_이름(port_signal_name)은 외부 입출력 신호선인 신호(signal) 를 선언한다. 따라서 외부 입출력의 특성을 잘 표현하는 이름을 정하는 것 이 좋다.

모드(mode)는 신호의 입출력 방향과 성격을 결정한다.

자료형(data type)은 신호의 자료 형태를 선언한다.

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

5. VHDL 어휘 요소와 기본 구성

모드

모드	입출력	기 능
in	입력	신호가 해당 엔티티로 입력되는 경우에 사용한다.
out	출력	해당 엔티티로 신호가 출력되는 경우에 사용한다.
inout	입출력	신호가 해당 엔티티로 양방향으로 입력되거나 출력되는 경우에 사용한다.
buffer	버퍼	출력 기능에 자신의 신호를 엔티티 내에서 다시 읽는 경우에 사용한다.

자료형(data type)

- 모든 신호(signal)는 자료형을 가져야 한다.
- 신호의 개수가 1개일 때는 std_logic으로 표현한다.
- 신호의 개수가 2개 이상일 때는 std_logic_vector로 표현한다.

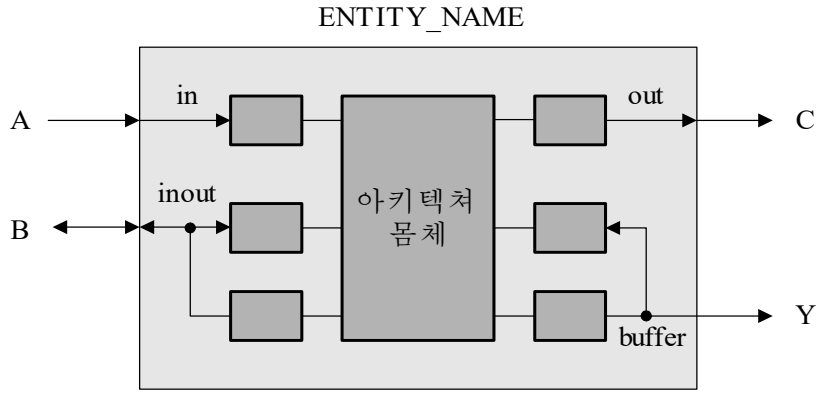
ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

5. VHDL 어휘 요소와 기본 구성

포트문 사용 예

- 포트신호 A는 in 모드로 정의되므로 입력핀이 된다.
- 포트신호 B는 inout 모드로 정의되므로 입,출력핀이 된다.
- 포트신호 C는 out 모드이므로 출력핀이 된다.
- 포트신호 Y는 buffer 모드 출력핀이 된다.




```
ENTITY ENTITY_NANE  IS
  PORT(A : IN             std_logic;
       B : INOUT          std_logic;
       C : OUT            std_logic;
       Y : BUFFER     std_logic);
END ENTITY_NAME;
```

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

5. VHDL 어휘 요소와 기본 구성

포트문 신호의 초기화

신호의 자료형 선언 후 할당연산자 ':='을 사용하여 초기값을 부여한다.

신호 A와 B는 초기값이 '1' 그리고 C는 디폴트 값('0')으로 주어진다.

```
ENTITY ENTITY_NANE  IS
  PORT(A, B  : IN  std_logic := '1';
               C  : OUT std_logic);
END ENTITY_NAME;
```

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

5. VHDL 어휘 요소와 기본 구성

③최소 엔티티 선언

최소 엔티티 선언은 엔티티에서 모든 요소가 생략된 경우를 말하며 입력과 출력이 없는 하드웨어 구조를 표현하기 위해서 사용한다.
형식

```
ENTITY TEST IS
END TEST;
```

TEST라는 엔티티_이름으로 입력과 출력이 없는 하드웨어를 나타낸다.
활용

테스트 벤치(Test Bench)에서 활용한다. 테스트 벤치는 외부와 입출력을 하지 않고 자체 내에서 입력 파형을
주기 때문에 엔티티 선언에서 포트문을 가지지 않는다.
테스트 벤치는 외부와 입출력을 하지 않고 VHDL 자체 내에서 입력파형을 주기 때 문에 entity declaration에서 port문을 갖지 않는다.
다음은 전가산기(full adder)에 대한 테스트 벤치 파일의 예이다.
전가산기 테스트를 위한 입력 signal은 X, Y, Cin을 선언하며, 출력 signal은 Sout, Cout을 선언한다.
full_adder를 컴포넌트로 선언하고 컴포넌트 사례화문과 X, Y, Cin 등의 입력파형 이 주어진다.
테스트를 위해 X, Y, Cin 입력에 "000"부터 "111"까지 차례로 10ns 간격으로 입력시킨다.
X <= '0' after 10ns 문장은 10ns 뒤에 X의 값이 '0'이 된다는 의미이다.

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

5. VHDL 어휘 요소와 기본 구성

◆ 전가산기에 대한 진리표

x	y	Cin	Cout	Sout
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

5. VHDL 어휘 요소와 기본 구성

◆ 전가산기에 대한 VHDL 코드

```
LIBRARY IEEE;  USE IEEE.std_logic_1164.all

-- Entity declaration
entity full_adder is
   port(X, Y, Cin      : in std_logic;
          Sout, Cout   : out std_logic);
end full_adder;

-- Architecture body
architecture Behavioral of full_adder is
begin
   process(X, Y, Cin)
      variable i : integer;
   begin
      i := 0;
      if  X = '1'     then  i := i +  1;  end if;
      if  Y = '1'     then  i := i +  1;  end if;
      if  Cin = '1'  then  i := i +  1;  end if;
      if  i = 0  or  i = 2  then  Sout <= '0'  after 2 ns;
      else  Sout <= '1'    after 2 ns;
      end if;
      if  i = 0  or  i = 1  then Cout <= '0'  after 2 ns;
      else  Cout <= '1'  after 2 ns;
      end if;
   end process;
end Behavioral;
```

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

5. VHDL 어휘 요소와 기본 구성

◆ 전가산기에 대한 테스트 벤치 VHDL 코드

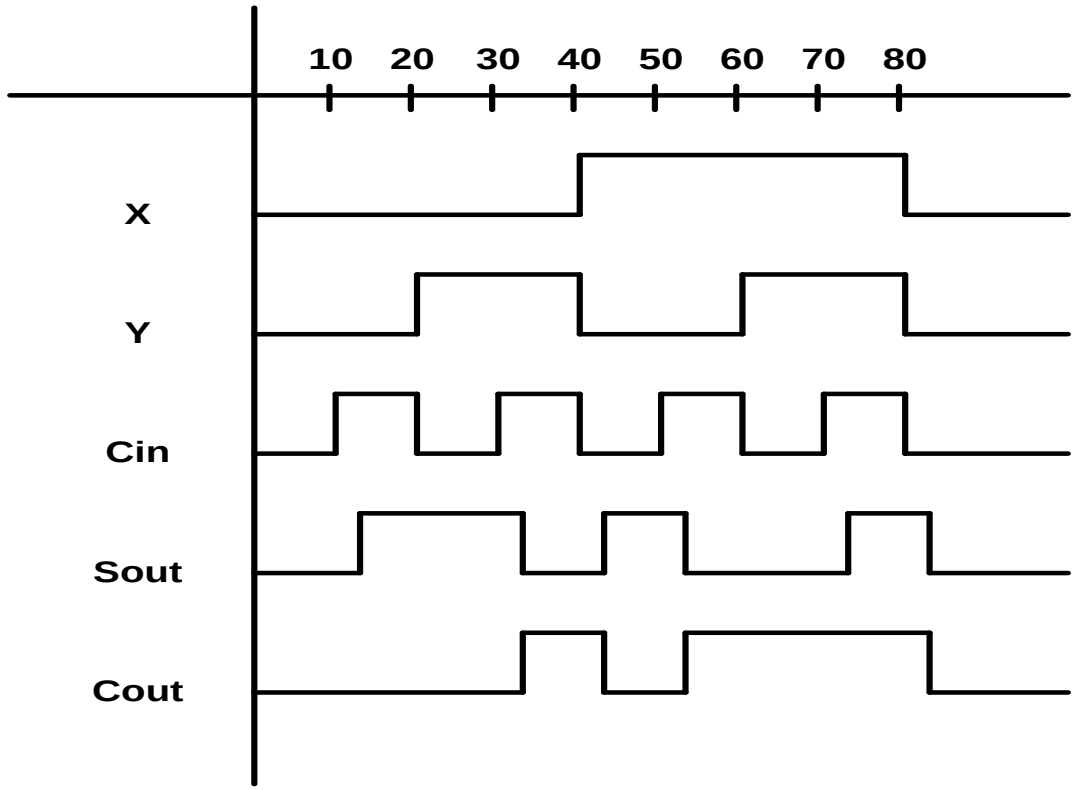
```
LIBRARY IEEE;  USE IEEE.std_logic_1164.all

entity tb_full_adder is
end tb_full_adder;

architecture Test_bench of tb_full_adder is
  signal X, Y : std_logic;
  signal Cin  : std_logic;
  signal Sout, Cout : std_logic;
  component full_adder
    port(X, Y, Cin     : in std_logic;
         Sout, Cout : out std_logic);
  end component;
begin
       X <= '0',
            '1'  after 40 ns,    '0' after 80 ns;
       Y <=  '0',
            '1'  after 20 ns,    '0' after 40 ns,    '1' after 60 ns,   '0' after 80 ns;
       Cin <=  '0',
            '1' after 10 ns,     '0' after 20 ns,     '1' after 30 ns,   '0' after 40 ns,
            '1' after 50 ns,     '0' after 60 ns,     '1' after 70 ns,   '0' after 80 ns;

       FA : full_adder  port map(X, Y, Cin, Sout, Cout);
end Test_bench;
```

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

5. VHDL 어휘 요소와 기본 구성

◆ 전가산기에 대한 시뮬레이션 파형 그림



ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

5. VHDL 어휘 요소와 기본 구성

- 입력 X, Y, Cin 등이 모두 '0'이면 출력 Cout과 Sout이 모두 '0'이 된다.
- 10ns에 Cin이 '1'이 되면 Sout이 '1'이 되고 Cout은 '0'이 된다.
- 20ns에 Y가 '1'이 되고 Cin이 '0'이 되면 Sout은 '1'이 되고 Cout은 '0'이 된다.
- 30ns에 Cin이 '1'이 되면 2개의 '1'이 있으므로 Cout은 '1'이 되고 Sout은 '0'이 된다.
- 전가산기에서 출력 Sout과 Cout의 결과가 나오는 지연시간을 2ns로 주었기 때문에 시뮬레이션 결과에서 출력 파형의 지연이 나타나는 것을 알 수 있다.

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

5. VHDL 어휘 요소와 기본 구성

④아키텍쳐 몸체

아키텍쳐 몸체는 설계 회로의 내부 동작 또는 연결 구조를 기술한다.
하나의 엔티티 선언에는 여러 개의 아키텍쳐 몸체가 존재할 수 있다.

아키텍쳐 몸체의 형식

```
ARCHITECTURE 아키텍쳐_이름 OF 엔티티_이름 IS
    선언부;
BEGIN
    병행문;
END 아키텍쳐_이름;
```

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

5. VHDL 어휘 요소와 기본 구성

ARCHITECTURE와 BEGIN 사이는 선언부로서 선언문(declaration statement)들을 기술한다.
선언문은 BEGIN과 END 사이에서 사용할 신호(signal), 변수(variable), 상수(constant),
자료형(data type), 컴포넌트 등을 선언한다.
하드웨어 내부적인 동작 또는 구조를 다른 문장과 병행적으로 수행되는
병행문(concurrentstatement)을 BEGIN과 END 사이에서 사용한다.
병행문의 종류는 다음과 같다

- **컴포넌트 사례화문**(component instantiation statement)
- **병행할당문**(concurrent assignment statement)
- **생성문**(generate statement)
- **프로세스문**(process statement)

이들 병행문들을 디지털 논리회로 하드웨어를 구현할 때 다음과 같은 표현 방법들을 사용한다.

- **동작적 표현**(behavioral representation)
- **자료흐름적 표현**(dataflow representation)
- **구조적 표현**(structural representation)
- **혼합적 표현**(mixed representation)

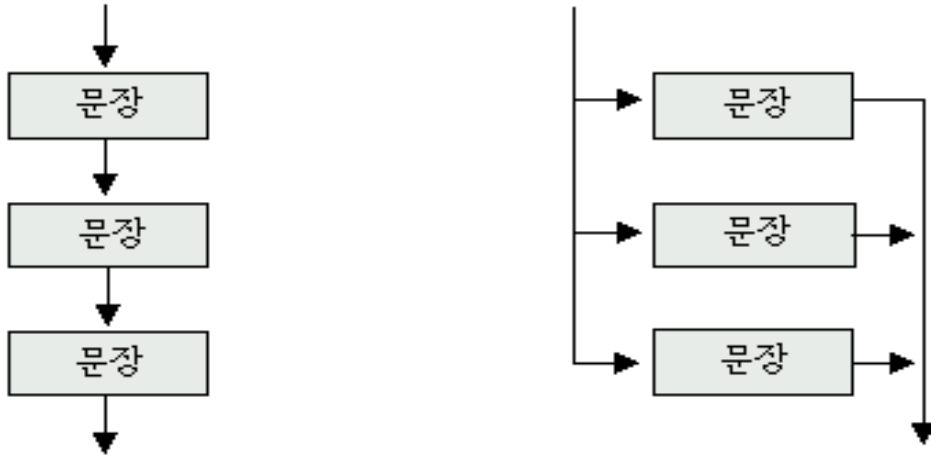
ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

5. VHDL 어휘 요소와 기본 구성

⑤VHDL 문장의 순차문과 병행문

VHDL 문장(statement)은 수행하는 방법에 따라 크게 순차문(sequential statement)과 병행문(concurrent statement)으로 나누어진다.
아키텍처의 BEGIN과 END 사이에 있는 문장들이, 순차적으로 수행되는 문을 순차문 그리고 병행적으로 수행되는 문을 병행문이라 한다.
VHDL에서는 실제로 디지털 논리회로 내의 모든 요소들은 항상 같이 동작하도록 표현해야 하므로 병행문을 사용하여 설계한다.
보통의 프로그램 언어에서처럼 문장들을 순서적으로 동작하는 방법을 이용할 수 있도록 VHDL은 순차문을 제공한다.



순차문 병행문

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

5. VHDL 어휘 요소와 기본 구성

◦ **순차문**(sequential statement)
- 문장들이 하나씩 차례로 수행되는 문
- 프로세스문의 BEGIN ~ END 내부에서 순차적으로 한 문장씩 수행
  → 동작적 표현으로 구성

◦ **병행문**(concurrent statement)
- 문장들이 동시에 병행적으로 수행되는 문
- 문장들이 동시에 병행적으로 수행되므로 문장들의 순서는 중요하지 않음
- 프로세스문 자체는 병행문
- 여러 개의 프로세스문들이 있으면 동시에 병행적으로 수행
  → 자료흐름적 표현을 위해서는 병행 신호 할당문을 사용
  → 구조적 표현을 위해서는 컴포넌트 사례화문을 주로 사용

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

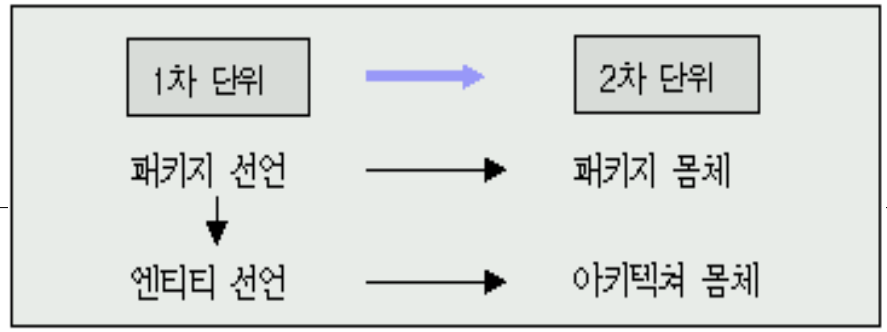
5. VHDL 어휘 요소와 기본 구성

3)설계 라이브러리 (design library)

라이브러리(library) – VHDL을 사용하여 이미 설계한 것들을 저장하여 필요시에 이용할 수 있도록 함으로서 반복된 설계를 피하고 설계한 것을 공유하도록 한 저장공간이다.

엔티티 선언, 아키텍쳐 몸체, 패키지 선언, 패키지 몸체와 같은 설계 단위(design unit)들을 분석하여 이상이 없으면 라이브러리에 저장한다.

1차 단위에서 먼저 분석이 끝나면 2차 단위에서 분석을 하는데, 예를 들어 패키지 선언을 먼저 분석하고 패키지 몸체를 분석하며 이어서 엔티티 선언을 분석하고 대응하는 아키텍쳐 몸체를 분석하게 된다.



- **1차 단위**(primary unit) : **패키지 선언, 엔티티 선언**
- **2차 단위**(secondary unit) : **패키지 몸체, 아키텍쳐 몸체**

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

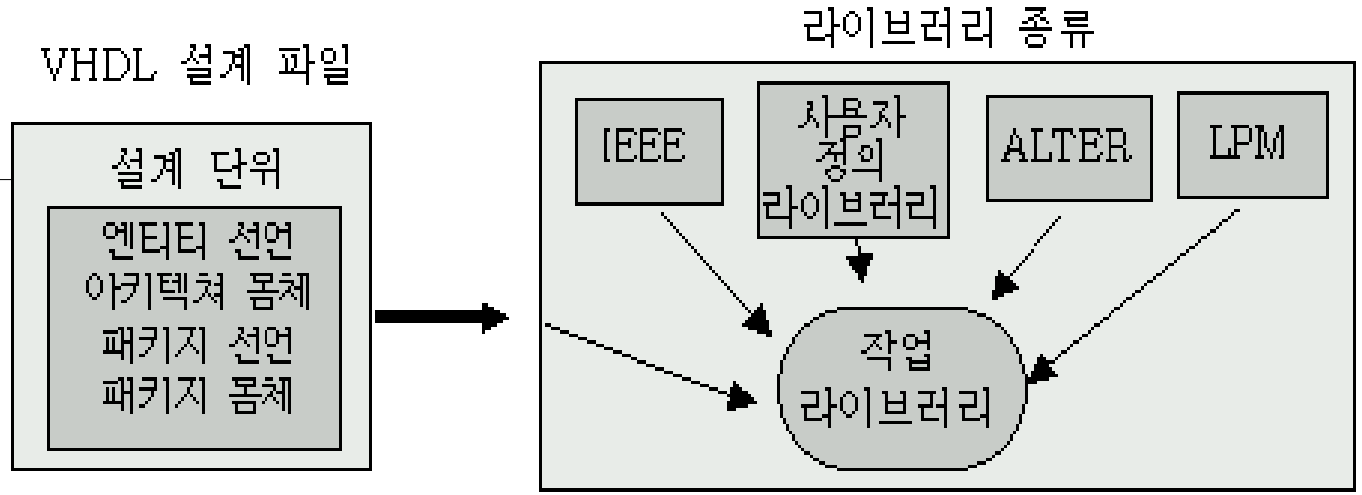
5. VHDL 어휘 요소와 기본 구성

VHDL 설계 파일은 VHDL 해석기를 통하여 합성되고 합성된 문장은 지정된 라이브러리에 저장된다. 만일 특별히 라이브러리를 지정하지 않으면 WORK 라이브러리에 저장된다.

모든 라이브러리는 각각 논리적 이름(logic name)을 가진다. 설계 유니트를 분석한 뒤에 이를 저장할 라이브러리의 이름을 부여할 수 있다. 그러나 이름이 이미 주어진 2가지 라이브러리가 있으며 이는 WORK 와 STD 이다.

WORK 라이브러리는 분석한 설계 유니트를 저장하는 곳이며 여러 사람이 공유한다.

STD 라이브러리는 STANDARD 라는 패키지와 TEXTIO 라는 패키지를 가지고 있다.



ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

5. VHDL 어휘 요소와 기본 구성

라이브러리는 VHDL 및 ASIC 공급자를 통해서 제공되기도 하지만, 사용자가 직접 사용자의 목적에 따라서 만들 수 있다.

라이브러리에는 하나 이상의 패키지들이 있는데, 각 패키지들은 특정 용도에 맞추어 구분되어 사용된다.

VHDL에서 많이 사용하는 라이브러리는 IEEE 라이브러리이다. IEEE 라이브러리는 std_logic_1164, std_logic_unsigned, std_logic_arith등과 같은 패키지들이 저장되어 있는 곳이다.

라이브러리는 다음과 같은 각각 논리적 이름(logical name)을 가진다.이들은 반드시 영어로 쓰여져야 한다.

IEEE, STD, WORK, USER, ALTERA, ASIC vender library

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

5. VHDL 어휘 요소와 기본 구성

이들 라이브러리를 사용하기 위해서는 가시성(visibility)을 부여해야 한다. 가시성은 라이브러리 내에서 선언 및 정의된 것을 사용할 수 있도록 경로(path)를 설정하는 것이다.

라이브러리는 자동으로 가시화되지 않으므로 다음과 같이 라이브러리 구문을 사용하여 가시화 한다.

◎ **형식**

LIBRARY **라이브러리_이름**; USE **라이브러리_이름.패키지_이름.내용**;

◦ **사용 예**

Library IEEE; USE IEEE.STD_LOGIC_1164.ALL;

사용 예에서 Library IEEE는 IEEE라는 라이브러리를 가시화한 것이다. 가시화된 라이브러리내의 패키지를 사용하기 위해서는 USE 문으로 패키지 std_logic_1164를 가시화 한다. 그리고 IEEE내의 std_logic_1164 패키지 내의 모든 내용을 사용하기 위해서는 USE IEEE.std_logic_1164.all로 가시화하면 된다.

라이브러리 WORK와 STD는 항상 가시적이므로 별도의 library 및 use 문을 사용할 필요가 없다.(library work, std; use std.standard.all;)

ALTERA Quartus II Web Edition 8.0sp1의 VHDL

VHDL 표현

5. VHDL 어휘 요소와 기본 구성

ALTERA사가 제공하는 Quartus Ⅱ 라이브러리

라이브러리	패키지	파일	내용
IEEE	std_logic_1164	std1164.vhd std1164b.vhd	VHDL 표현을 위한 상호연결 자료형에 대한 표준 자료형 - STD_LOGIC 자료형 - STD_LOGIC_VECTOR 자료형
	std_logic_arith	arth.vhd arthb.vhd	signed 자료형, unsigned 자료형 정의 산술 연산 함수(arithmetic function) 비교 함수(comparision function) 자료형 변환 함수(type conversion function) - conv_integer, conv_signed, conv_unsigned, conv_std_logic_vector 등
	std_logic_signed	signed.vhd signedb.vhd	STD_LOGIC, STD_LOGIC_VECTOR 등에서 부호있는 산술연산을 할 경우에 사용
	std_logic_unsigned	unsigned.vhd unsignedb.vhd	STD_LOGIC, STD_LOGIC_VECTOR 등에서 부호없는 산술연산을 할 경우 경우에 사용