VHDL - 기초문법

2015.07.17

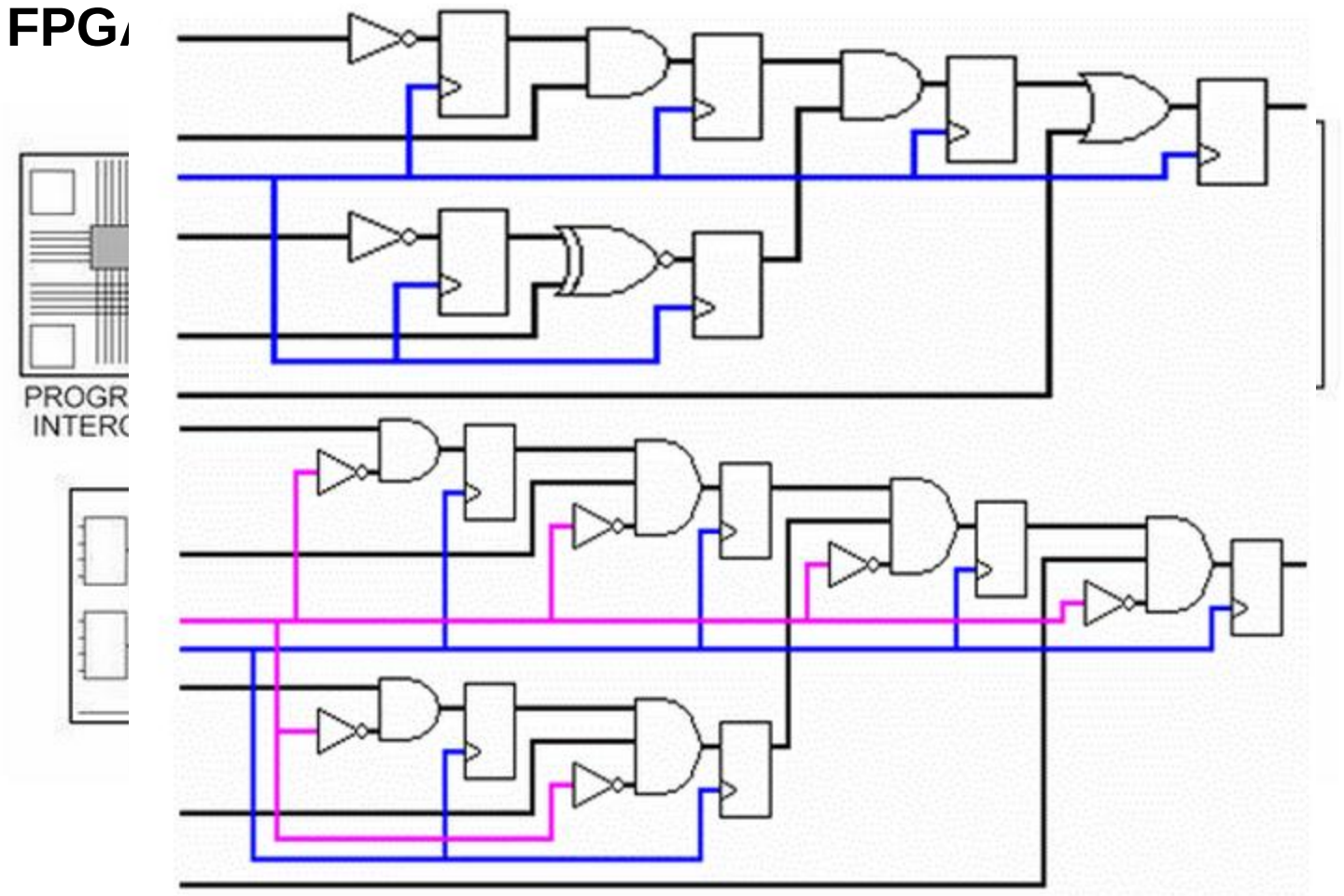
강의자 : **23**기 백두현

BARAM
KW univ
New generation of Robotics

목차

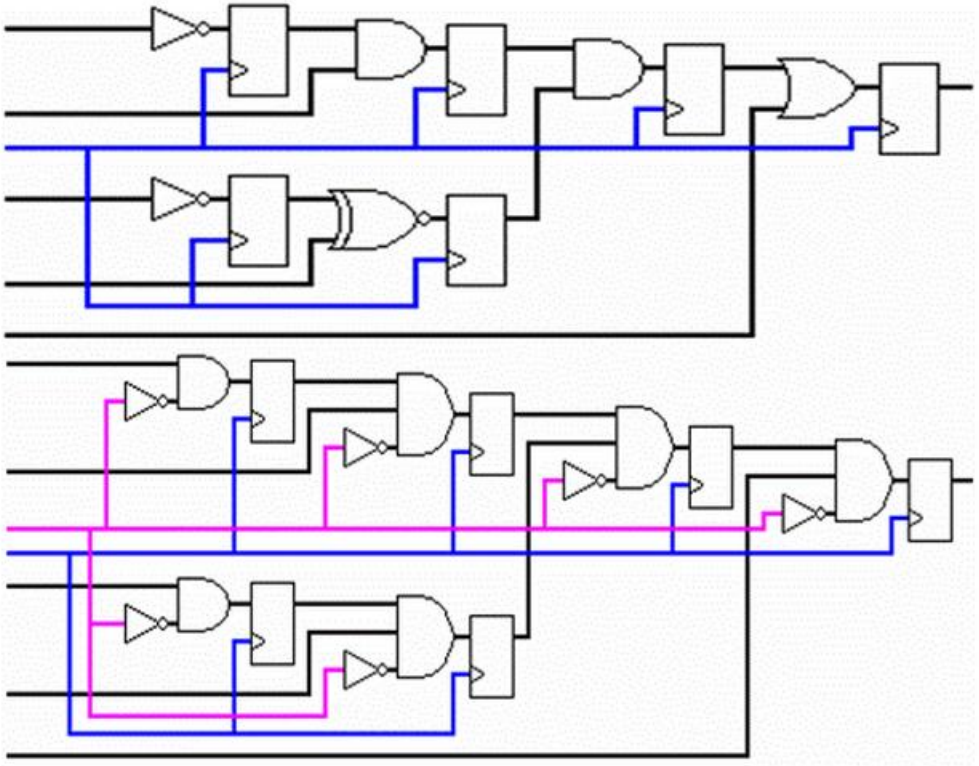
VHDL 에서 컴파일이란?

- FPG



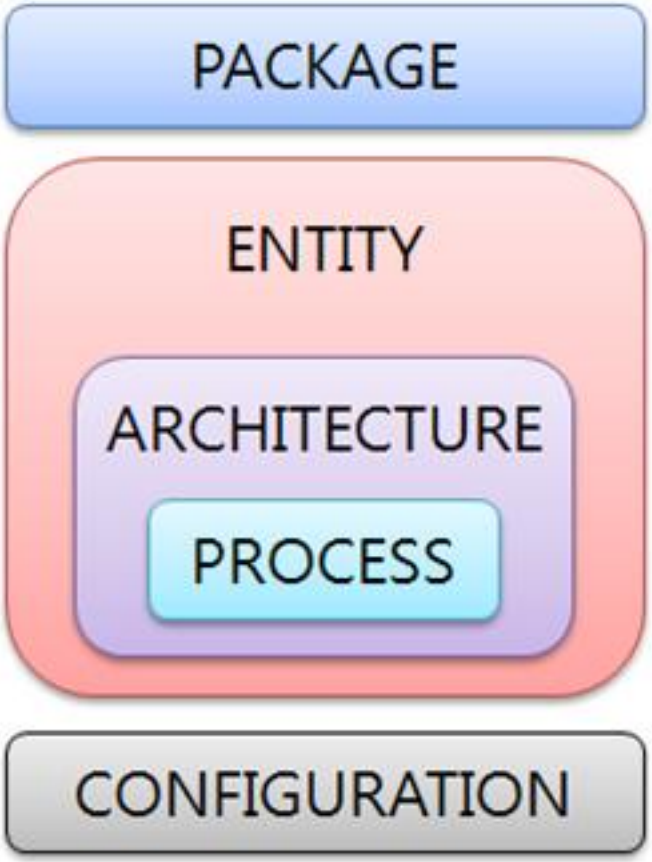
VHDL 에서 컴파일이란?

- FPGA 내부 구조



*컴파일 : 논리 합성 가능!

VHDL의 기본구성




```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity test is
5   port();
6   end test;
7
8   architecture arc of test is
9   begin
10      process()
11      begin
12      end process;
13  end arc;
```

PACKAGE

ENTITY

ARCHITECTURE

PROCESS

VHDL의 기본구성_PACKAGE

- **PACKAGE**
  - 단일 목적을 위한 선언(declaration)들의 모임.
  - 공통적으로 사용되는 선언이나 자원을 공유하기 위한 용도.
  - C언어의 라이브러리 및 헤더파일과 같은 기능을 제공.

- **VHDL 표준 라이브러리에서 제공하는 PACKAGE**
  - Standard
  - textio
  - Env
  - std_logic_1164
  - Std_logic_arith
  - Std_logic_signed
  - Std_logic_unsigned
  - Std_logic_textio
  - Std_logic_misc

VHDL의 기본구성_ENTITY

ENTITY

```
entity Entity_이름 is
    port(
        -- 입출력 선언부
    );
end Entity_이름;
```

- 설계 회로의 인터페이스에 관한 정보를 기술한 본체 부분으로서 설계된 블록의 이름과 입출력 포트를 선언.

- PORT의 종류
  - IN : 입력, 신호가 해당 ENTITY로 입력되는 경우에 사용.
  - OUT : 출력, 해당 ENTITY에서 신호가 출력되는 경우에 사용.
  - INOUT : 입출력, 신호가 해당 ENTITY에서 양방향으로 사용.
  - BUFFER : 출력 기능 신호에 ENTITY 내에서 다시 읽는 기능을 추가한 것.
  - LINKAGE : 동작에 영향을 주지 않으며, 단지 포트로서 연결된 경우에 사용.

VHDL의 기본구성_ARCHITECTURE

ARCHITECTURE

```
architecture Architecture_이름 of Entity_이름 is
    -- 변수 선언부
begin
    -- 동작 구현부
end Architecture_이름;
```

- Hardware의 내부동작 및 연결상태 표시

ARCHITECTURE_병행구문과 순차구문

병행구문

```
architecture arc of test is
begin
    W <= A and B;
    X <= A OR  B;
    Z <= A NOR B;
end arc;
```

- VHDL은 기본적으로 병행구문을 지원한다.
- 병행구문이란 동시처리 되는 것을 말한다.

순차구문

```
architecture arc of test is
begin
    process(감지리스트)
    begin
        W <= A and B;
        X <= A OR  B;
        Z <= A NOR B;
    end process;
end arc;
```

- 순차구문이란 C와 같이 위에서부터 순차적으로 처리되는 구문을 말한다.
- VHDL에서는 PROCESS 문을 통해 순차구문을 지원한다.

ARCHITECTURE_PROCESS

PROCESS문(순차 처리문)

- 기본적으로 병행처리인 VHDL에서 순차적으로 처리를 할 수 있도록 해주는 문법.

```
architecture arc of test is
begin
    process(감지리스트)
        -- 변수 선언부
    begin
        -- 동작 선언부
    end process;
end arc;
```

- PROCESS문은 감지리스트에 들어가는 식별자가 변화가 있을 때 마다 실행 된다.
- 변수는 PROCESS선언과 BEGIN 사이에 선언한다.
- 감지리스트에 들어가는 신호는 PROCESS내에서 선언할 수 없다.

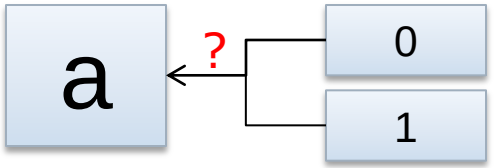
ARCHITECTUR_PROCESS

다중 PROCESS

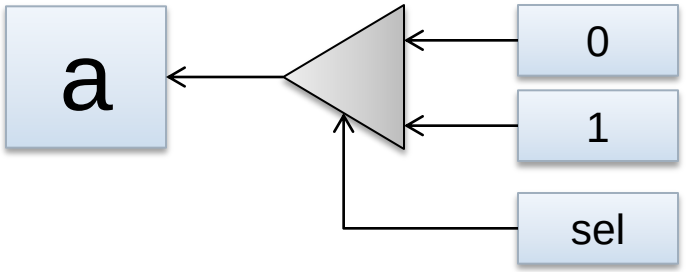
```
architecture arc of test is
begin
    process(감지리스트)
    begin
        a <= '1';
    end process;

    process(감지리스트)
    begin
        a <= '0';
    end process;
end arc;
```

주의사항

- 하나의 변수를 다른 PROCESS문에서 값을 대입할 수 없다!!!



- A 변수가 '0'을 받을지 '1'을 받을지 판단 불가!
- 해결책



기초문법

- **beign ~ end**
  - Entity, architecture, process, if, case등을 사용시 블록을 지정할 때 사용한다.
  - C언어에서 { … }와 같은 개념이다.
- **주석**
  - -- : 한 줄을 주석 처리할 때 사용한다.
  - 여러 줄을 주석 처리?? => 불가
    - 에디트 프로그램을 통해서 사용가능.
- **수 표현**
  - 단일 bit : ‘0’,’1’
  - 다중 bit : “0000”,”1111”
    - (others => ‘1’) == “1111” ex) a <= (others =>’1’) ;

std_logic

- **std_logic과 std_logic_vector형**
  - 회로를 합성하는 목적으로 VHDL을 사용할 때, 가장 권장하는 자료형으로서 로직의 다양한 상태를 표현할 수 있음.
  - 3상태, 연결 논리 등을 표현 할 수 있을 뿐 아니라 같은 신호에서의 다구동을 모델링 할 수 있다.
  - std_lgogic
    - 1 bit의 데이터 register.
  - std_logic_vector
    - 2 bit이상의 데이터 register.
    - downto 와 to
      - std_logic_vector(7 downto 0);

7	6	5	4	3	2	1	0

      - std_logic_vetor(0 to 7);

0	1	2	3	4	5	6	7

변수

- **SIGNAL**
  - VHDL 합성 시에 선(WIRE)으로 구현.
  - 각 부품(COMPONENT)의의 연결에 사용되는 외적 변수.
  - 대입기호 (‘<=’)
    - 즉시 대입되는 것이 아닌 VHDL문에서 필요한 어떤 시점에서 대입.
    - EX) PROCESS문 내에서는 ‘end process;’를 만나야 대입 완료.

```
signal signal_이름 : 자료형 [:= 초기값]; -- 초기값은 선택적

ex)    signal a     : std_logic;
       signal b,c   : std_logic;
       signal d,e   : std_logic_vector(7 downto 0);

       a  <= '1';
       d  <= "11111111";
       e  <= (others => '1');
```

변수

VARIABLE

- PROCESS나 부 프로그램(FUNCTION,PROCEDURE)에서만 사용.
- VARIABLE의 값은 PROCESS나 부 프로그램 내에서만 유효한 내적 변수.
- VARIABLE은 SIGNAL과 같이 VHDL 합성 시에 선으로 구현되는 것이 아니며, 중간 연산 단계에 주로 사용.
- 대입기호(:=) : 즉시대입

```
variable variable_이름 : 자료형;

ex)    process(감지리스트)
           variable temp1 : std_logic;
           variable temp2 : std_logic;
       begin
           temp1 := '1';
           temp2 := a or b;   -- a,b,는 signal
           k  <= temp1;       -- k는 signal
       end process;
```

변수

- **CONSTANT**
  - CONSTANT는 초기에 선언한 상수의 값을 유지하는데 사용.
  - 대입기호(:=)
    - 초기값은 즉시 대입되며, 한 번 대입된 값은 바꿀 수 없다.

```
constant constant_이름 : 자료형 := 초기값;

ex) constant data_size : std_logic_vector(7 downto 0) := (others=>'0');
```

연산자

관계연산자

=	같다(equal to)
/=	같지 않다.(not equal to)
<	(왼쪽이) 작다 (less than)
<=	(왼쪽이) 작거나 같다 (less than or equal to)
>	(왼쪽이) 크다 (greater than)
>=	(왼쪽이) 크거나 같다 (greater than or equal to)

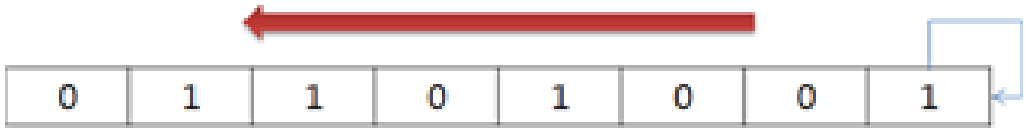
연산자

- **순환연산자**
  - 쉬프트
    - sll : 논리적인 왼 방향 쉬프트

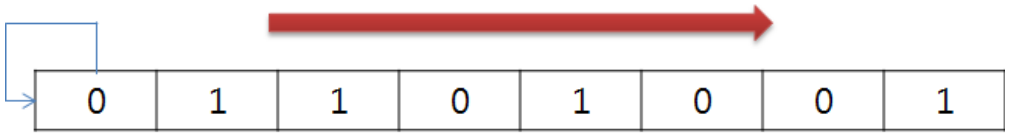
    - srl : 논리적인 오른 방향 쉬프트

연산자

- sla : 산술적인 왼 방향 쉬프트



- sra : 산술적인 오른 방향 쉬프트



연산자

□ 연산자 우선 순위

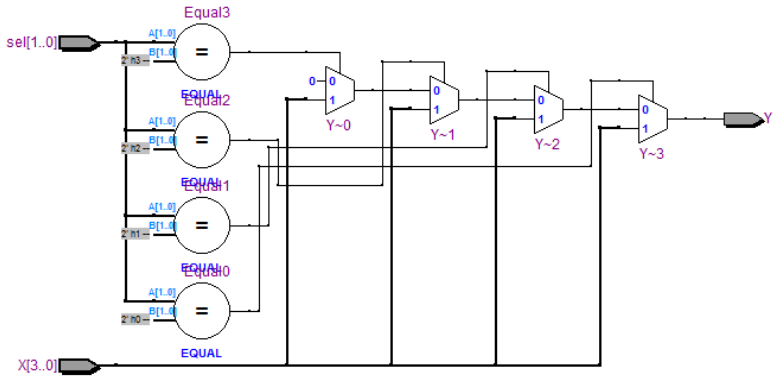
상위	기타연산자		**	abs	not		
	곱셈연산자		*	/	mod	rem	
	부호연산자			+	-		
	덧셈연산자		+	-	&		
	시프트연산자	sll	srl	sla	sra	rol	ror
	관계연산자	=	/=	<	<=	>	>=
하위	논리연산자	and	or	nand	nor	xor	xnor

조건문

IF문

- IF문은 순차문으로 PROCESS문 내에서 사용됨.
  - IF문은 비교기로 생성 된다.

```
process(감지리스트)
begin
    if(조건1) then
        순차문1;
    elsif(조건2) then
        순차문2;
    else(조건3)
        예외처리문;
    end if;
end process;
```



< if문을 이용한 4 to 1 MUX >

조건문 - edge

- event

```
process(sysclk,reset)
begin
   if(reset = 0)
   else if(sysclk'event and sysclk = 1)
   else if(sysclk'event and sysclk = 0)
   end if;
end process;
```

- rising_edge(falling_edge)

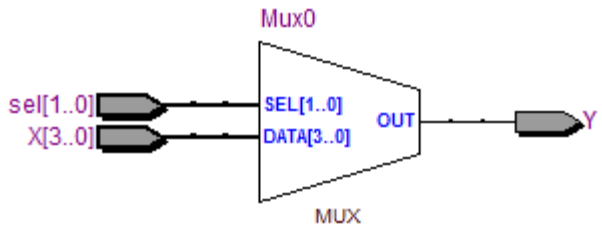
```
process(sysclk,reset)
begin
   if(reset = 0)
   else if(rising_edge(sysclk))
   else if(falling_edge(sysclk))
end if;
```

조건문

CASE문

- CASE문은 순차문으로 PROCESS문 내에서 사용됨
  - CASE문은 MUX로 생성 된다.

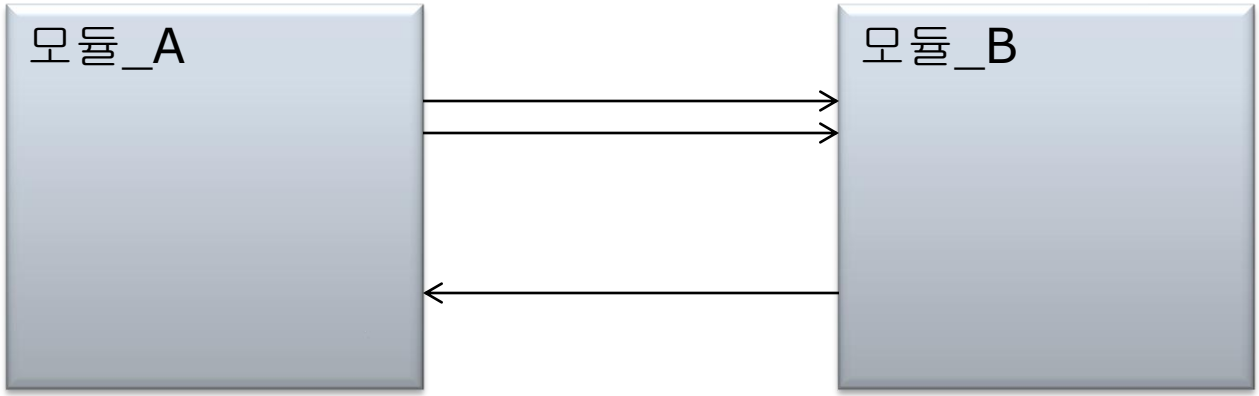
```
process(감지리스트)
begin
    case 조건 is
        when 조건 값1 => 순차문1;
        when 조건 값2 => 순차문2;
        when 조건 값3 => 순차문3;
        when others   => 예외처리문;
    end case;
end process;
```



< CASE문을 이용한 4 to 1 MUX >

Portmap

- **Portmap**
  - 모듈간 데이터를 연결 할 때 사용.



Portmap

```
architecture arc of test is

component 서브모듈
port(
    -- 서브모듈 PORT
    );
end component;

begin

모듈 명: 서브모듈
   port map(
          서브모듈의 포트 이름   => 현재 포트의 포트 이름,
          서브모듈의 포트 이름   => 현재 포트의 포트 이름,
          );

end arc;
```