4

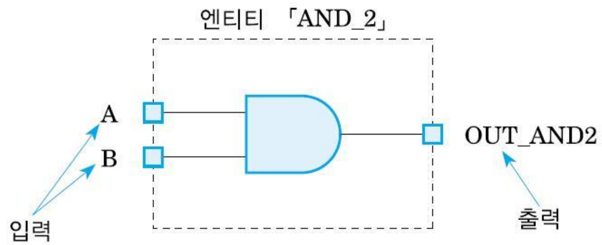
하드웨어 기술 언어

IT CookBook, VHDL을 이용한 디지털 회로 입문

❖ AND 회로

▪ 엔티티 선언

• 설계 대상 회로의 이름(엔티티명)과 입출력을 정의함



[그림 4-1] 2AND 회로의 엔티티 선언

엔티티 선언의 예제

엔티티명	AND_2
입력	A, B
출력	OUT_AND

```
entity 엔티티명 is
    port 문 ——— 구문 정의에서 엔티티 헤더에 포함되는 전형적인 문장이다.
end 엔티티명;
    생략 가능
```

[그림 4-2] entity 문의 구문 정의

```
port ( 포트명: 모드    데이터 타입;
              :
       포트명: 모드    데이터 타입; ) ;
              포트 리스트
```

[그림 4-3] port 문의 구문 정의

리스트 4-2 2AND 회로의 엔티티 선언

행번호 / 엔티티문 / 포트문

```
--       AND_2  설명문

entity AND_2 is
           엔티티명
  port (A, B      : in  std_logic;        -- 입력 선언 설명문

        OUT_AND2 : out std_logic );      -- 출력 선언 설명문

end AND_2;
      엔티티명
```

[표 4-1] 모드의 종류

모드	기능
in	입력
out	출력. 내부 신호로 이용할 수 없다.
inout	쌍방향
buffer	출력. 내부 신호로 이용할 수 있다.
linkage	방향지정 없음

❖ architecture 문

- 설계 대상 회로의 내부 구조를 기술 [begin] 이후의 concurrent 문(동시 처리 신호 대입문)으로 구성

```
architecture 아키텍처명 of 엔티티명 is
        신호 선언  ——— 아키텍처 선언부의 블록(block) 선언 항목인 신호 선언에 해당한다.
begin
        concurrent 문
end   아키텍처명;
       생략 가능
```

[그림 4-4] architecture 문의 구문 정의

리스트 4-4 2AND 회로의 아키텍처 본체

행 번호, 엔티티명, 아키텍처명, 아키텍처 문

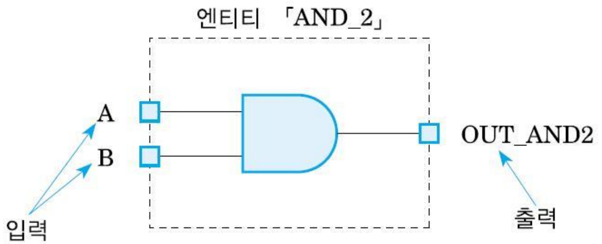
```
architecture AND_2 of AND_2 is
begin
OUT_AND2 <= A and B; ← concurrent문
end AND_2;
```

- VHDL 논리연산자의 종류
  - and, or, nand, nor, xor, xnor, not

❖ 엔티티 선언과 아키텍처 본체의 결합

리스트 4-5 2AND 회로의 VHDL 기술

```
        library IEEE;
        use IEEE.std_logic_1164.all;
        -- AND_2
        entity AND_2 is
엔티티          port ( A, B      : in  std_logic;          -- 입력 선언
선언                   OUT_AND2 : out std_logic );         -- 출력 선언
        end AND_2;
        architecture AND_2 of AND_2 is
아키텍처  begin
본체            OUT_AND2 <= A and B;
        end AND_2;
```



[그림 4-1] 2AND 회로의 엔티티 선언

```
library IEEE;
use IEEE.std_logic_1164.all;
entity 엔티티명 is
        port 문에 의한 포트 리스트
end 엔티티명;
architecture 아키텍처명 of 엔티티명 is
        신호선언
begin
        concurrent 문
end 아키텍처명;
```

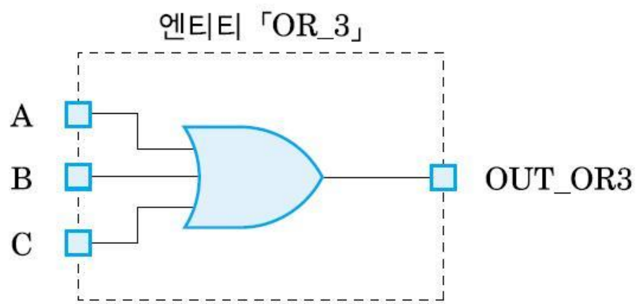
[그림 4-5] 정리해 놓은 VHDL 구문 정의

❖ VHDL의 구문 정의 정리

- 설명문은 [--]부터 그 줄의 끝까지
- 식별자(Identifier)를 붙이는 방법은
  - 최초 문자는 영문자
  - 사용 가능한 문자는 영문자와 숫자 및 [_]
  - _은 계속해 사용할 수 없다. (예, AND__2는 잘못된 표현)
  - _을 마지막에 사용할 수 없다. (예, AND_2_는 잘못된 표현)
  - 대문자와 소문자는 구별되지 않는다. 즉, AND_2와 and_2는 동일한 식별자

- entity 문은 [entity – end] 쌍과 port 문을 기술
- port 문으로 입출력을 선언
- architecture 문은 [architecture – end] 쌍과 신호 선언, [begin] 뒤의 concurrent 문을 기술
- 엔티티 선언과 아키텍처 본체를 결합
- 제일 앞 두줄에는 [library IEEE;]와 [use IEEE.std_logic_1164. all;]을 기술
- 이 책에서는 예약어(reserved word)는 소문자, 그 이외는 대문자로 기술

❖ OR회로



[그림 4-6] 3OR 회로의 엔티티 정의

리스트 4-6 OR 회로의 VHDL 기술

```
library IEEE;
use IEEE.std_logic_1164.all;

--      OR_3
entity OR_3 is
        port ( A, B, C : in  std_logic;
               OUT_OR3 : out std_logic );
end OR_3;

architecture OR_3 of OR_3 is
begin
        OUT_OR3 <= A or B or C;
end OR_3;
```

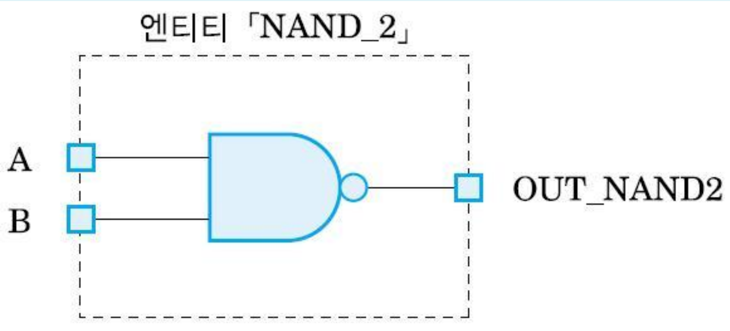
❖ NAND회로

리스트 4-7 2NAND 회로의 VHDL 기술(nand 연산자)

```
library IEEE;
use IEEE.std_logic_1164.all;

--      NAND_2
entity NAND_2 is
        port ( A, B       : in  std_logic;
               OUT_NAND2 : out std_logic );
end NAND_2;

architecture NAND_2 of NAND_2 is
begin
        OUT_NAND2 <= A nand B;
end NAND_2;
```



[그림 4-7] 2NAND 회로의 엔티티 정의

❖ NAND회로

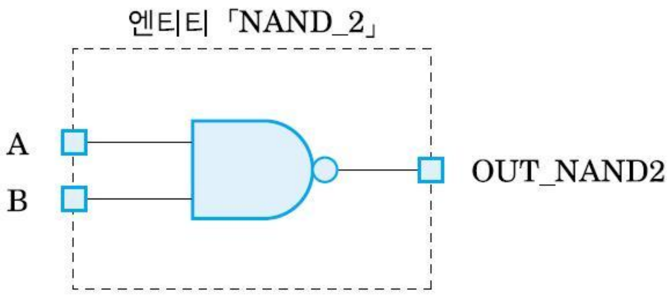
리스트 4-8 2NAND 회로의 VHDL 기술(not, and 연산자)

```
library IEEE;
use IEEE.std_logic_1164.all;

--      NAND_2
entity NAND_2 is

        port ( A, B       : in  std_logic;
               OUT_NAND2 : out std_logic );
end NAND_2;

architecture NAND_2 of NAND_2 is
begin
        OUT_NAND2 <= not ( A and B );
end NAND_2;
```



[그림 4-7] 2NAND 회로의 엔티티 정의

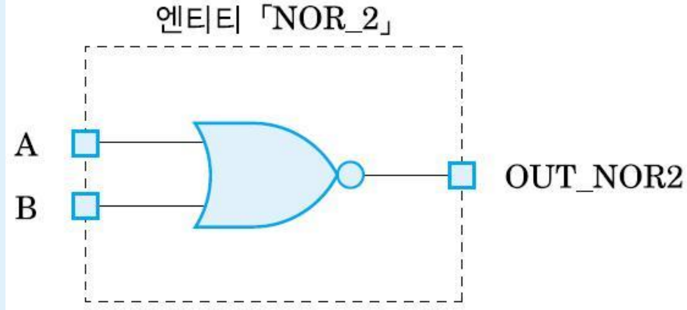
❖ NOR회로

리스트 4-9 2NOR 회로의 VHDL 기술

```
library IEEE;
use IEEE.std_logic_1164.all;

--      NOR_2
entity NOR_2 is
        port ( A, B       : in  std_logic;
               OUT_NOR2 : out std_logic );
end NOR_2;

architecture NOR_2 of NOR_2 is
begin
        OUT_NOR2 <= A nor B;
end NOR_2;
```



[그림 4-8] 2NOR 회로의 엔티티 정의

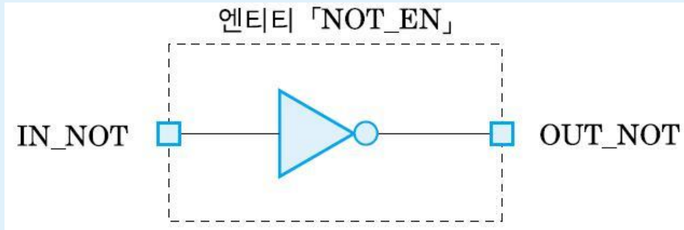
❖ NOT회로

리스트 4-10 NOT 회로의 VHDL 기술

```
library IEEE;
use IEEE.std_logic_1164.all;

--      NOT
entity NOT_EN is
       port ( IN_NOT  : in  std_logic;
              OUT_NOT : out std_logic );
end NOT_EN;

architecture NOT_AR of NOT_EN is
begin
       OUT_NOT <= not IN_NOT;
end NOT_AR;
```



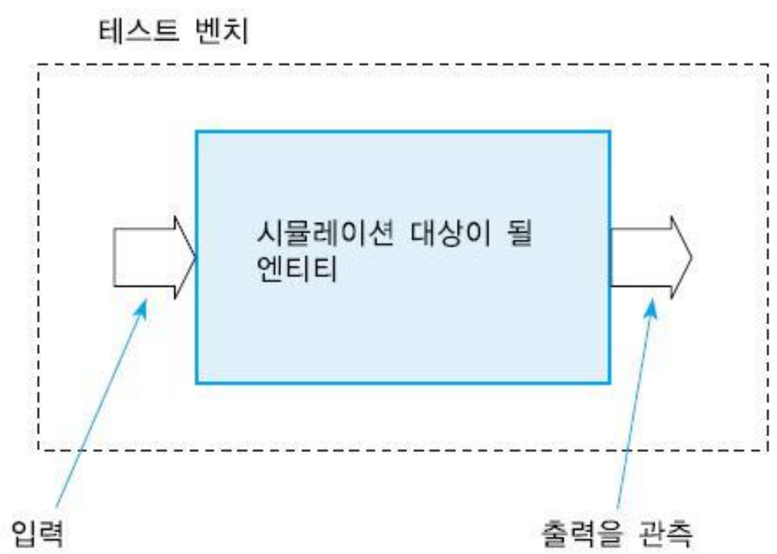
[그림 4-9] NOT 회로의 엔티티 정의

❖ 시뮬레이션(simulation)

- [모의 실험]이라는 의미로, 여기에서는 회로 동작에 필요한 입력을 사용자가 주고 이에 따른 출력의 변화를 관찰하는 것
- VHDL로 설계한 엔티티에 입력을 주기 위해 필요한 기술

❖ 테스트 벤치 설정

- 시뮬레이션을 위해 대상 회로에 입력을 주거나 대상 회로의 출력을 관측하기 위한 테스트 벤치(test bench)(또는 테스트 벡터(test vector))를 설정



[그림 4-12] 테스트 벤치의 정의

리스트 4-12 2AND 회로의 테스트 벤치

```
library IEEE;
use IEEE.std_logic_1164.all;

entity AND2_TEST is
end AND2_TEST;

architecture AND2_TEST of AND2_TEST is
        signal IN_A,IN_B : std_logic;
        signal OUT_AND_2 : std_logic;
        component AND_2
                port ( A, B     : in  std_logic;
                       OUT_AND2 : out std_logic );
        end component;
begin
        U0:AND_2 port map ( A => IN_A,
                            B => IN_B,
                            OUT_AND2 => OUT_AND_2 );
        process
        begin
                IN_A <= '0'; IN_B <= '0';
                wait for 100 ns;         IN_A <= '1';
                wait for 100 ns;         IN_A <= '0'; IN_B <= '1';
                wait for 100 ns;         IN_A <= '1';
                wait;
        end process;
end AND2_TEST;
```

❖ 신호 선언

- **signal 문**
  - 아키텍처 내부의 신호를 선언하는 것이 signal 문
  - port 문의 구문 정의에 있는 [모드]는 필요 없음
  - signal 신호명: 데이터 타입 ;

```
signal 신호명: 데이터 타입;
```

[그림 4-13] signal 문의 구문 정의

❖ 컴포넌트(부품) 선언

```
component 컴포넌트명
    port 문
end component;
```

[그림 4-14] component 문의 구문 정의

리스트 4-12 2AND 회로의 테스트 벤치

```
library IEEE;
use IEEE.std_logic_1164.all;

entity AND2_TEST is
end AND2_TEST;

architecture AND2_TEST of AND2_TEST is
        signal IN_A,IN_B : std_logic;
        signal OUT_AND_2 : std_logic;
        component AND_2
                port ( A, B      : in  std_logic;
                       OUT_AND2 : out std_logic );
        end component;
begin
        U0:AND_2 port map ( A => IN_A,
                            B => IN_B,
                            OUT_AND2 => OUT_AND_2 );
        process
        begin
                IN_A <= '0'; IN_B <= '0';
                wait for 100 ns;        IN_A <= '1';
                wait for 100 ns;        IN_A <= '0'; IN_B <= '1';
                wait for 100 ns;        IN_A <= '1';
                wait;
        end process;
end AND2_TEST;
```

❖ 하위층 엔티티의 호출

- 행 번호 8~11에서 컴포넌트로 선언한 시뮬레이션의 대상 엔티티 [AND_2]를 호출한다. 이것을 컴포넌트 인스턴스(component instance)

인스턴스 라벨명 : U0

컴포넌트명 : AND_2

```
인스턴스 라벨: 컴포넌트명 port map  (포트 관련 리스트);
```

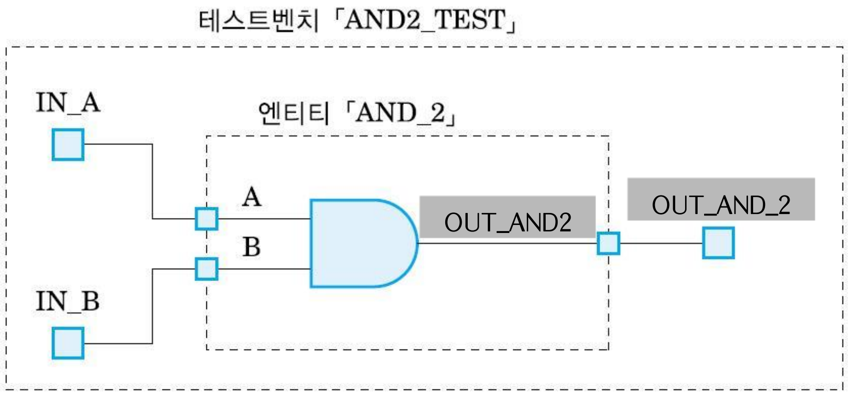
[그림 4-15] 컴포넌트 인스턴스문의 구문 정의

- 포트관련 리스트를 기술하는 방법

① [이름에 의한 관련성] : 포트명 => 신호명 으로 기술

② [위치에 의한 관련성]

: 엔티티 [AND_2]에서는, [A→B→OUT_AND2]의 순서로 기술되어 있기 때문에 행 번호 13~15는, U0: AND_2 port map (IN_A, IN_B, OUT_AND_2); 로 기술



[그림 4-17] 테스트 벤치 「AND2_TEST」와 엔티티 「AND_2」의 관계

❖ 하위층 엔티티로의 입력 파형의 정의

- (1) process 문
- (2) wait 문

리스트 4-12 2AND 회로의 테스트 벤치

행 번호

```
library IEEE;
use IEEE.std_logic_1164.all;

entity AND2_TEST is
end AND2_TEST;

architecture AND2_TEST of AND2_TEST is
       signal IN_A,IN_B : std_logic;
       signal OUT_AND_2 : std_logic;
       component AND_2
              port ( A, B     : in  std_logic;
                     OUT_AND2 : out std_logic );
       end component;
begin
       U0:AND_2 port map ( A => IN_A,
                           B => IN_B,
                           OUT_AND2 => OUT_AND_2 );
       process
       begin
              IN_A <= '0'; IN_B <= '0';
              wait for 100 ns;          IN_A <= '1';
              wait for 100 ns;          IN_A <= '0'; IN_B <= '1';
              wait for 100 ns;          IN_A <= '1';
              wait;
       end process;
end AND2_TEST;
```

[표 4-2] 시간의 물리적인 타입

표기	명칭	크기
fs	펨토초	10^{-15}초
ps	피코초	10^{-12}초
ns	나노초	10^{-9}초
us	마이크로초	10^{-6}초
ms	밀리초	10^{-3}초
sec	초	초
min	분	분
hr	시	시

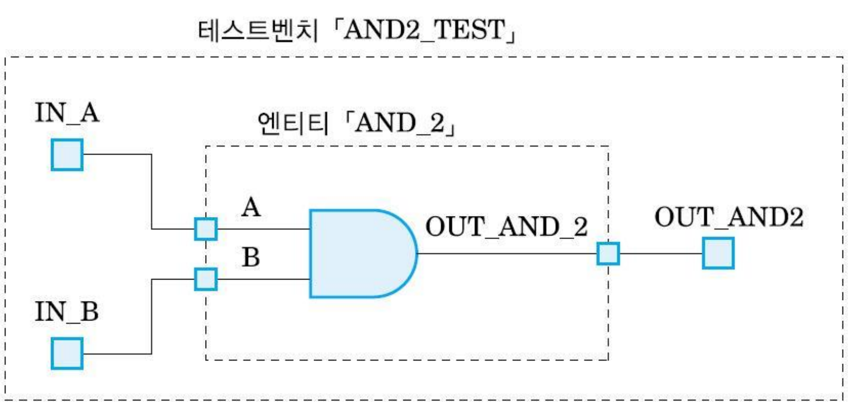
리스트 4-14 2AND 회로의 시뮬레이션을 위한 VHDL 기술(엔티티 「AND_2」)

```
본문(4장) 내 참조 번호
2.1(6)      library IEEE;
2.1(6)      use IEEE.std_logic_1164.all;

            --    AND_2
2.1(2)      entity AND_2 is
2.1(3)           port ( A, B     : in  std_logic;
                      OUT_AND2 : out std_logic );

            end AND_2;

2.1(5)      architecture AND_2 of AND_2 is
            begin
                 OUT_AND2 <= A and B;
            end AND_2;
```



[그림 4-17] 테스트 벤치 「AND2_TEST」와 엔티티 「AND_2」의 관계

리스트 4-13 2AND 회로의 시뮬레이션을 위한 테스트 벤치(엔티티 「AND2_TEST」)

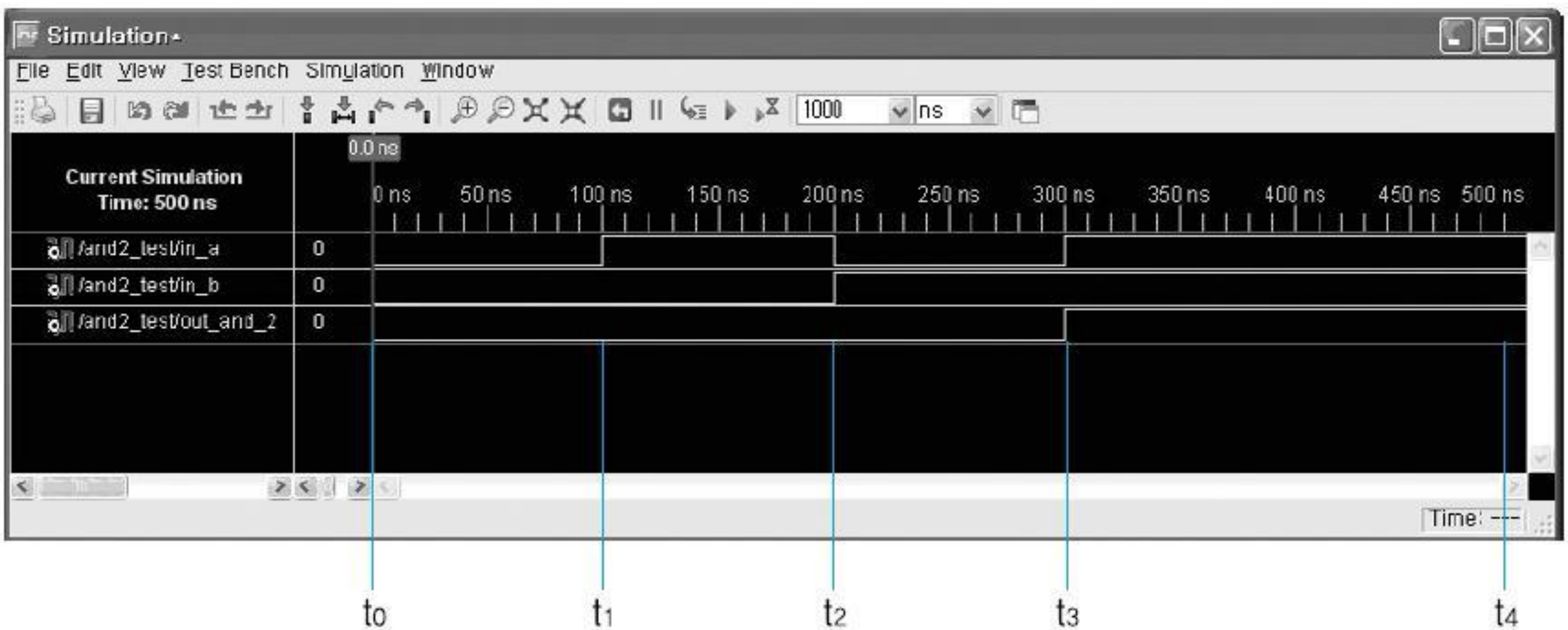
```
본문(4장) 내 참조번호
2.1(6)    library IEEE;
2.1(6)    use IEEE.std_logic_1164.all;

3.1       entity AND2_TEST is
          end AND2_TEST;

2.1(5)    architecture AND2_TEST of AND2_TEST is
3.2              signal IN_A,IN_B : std_logic;
                 signal OUT_AND_2 : std_logic;
3.3 ──────────── component AND_2
2.1(3) ───────────────── port ( A, B     : in  std_logic;
                                OUT_AND2 : out std_logic);
                 end component;
          begin
3.4 ──────────── U0:AND_2 port map ( A => IN_A,
                                     B => IN_B,
                                     OUT_AND2 => OUT_AND_2 );
3.5(1) ───────── process
                 begin
3.5(2) ───────────────── IN_A <= '0'; IN_B <= '0';
                        wait for 100 ns;        IN_A <= '1';
                        wait for 100 ns;        IN_A <= '0'; IN_B <= '1';
                        wait for 100 ns;        IN_A <= '1';
                        wait;
                 end process;
          end AND2_TEST;
```

❖ 2AND 회로의 시뮬레이션 결과

- 입력의 변화
  - 테스트 벤치파일에 작성한 입력의 변화 시간을 관찰
- 출력의 변화
  - 입력에 따라 변화하는 출력을 관찰
  - 설계한 [2AND] 회로의 올바른 동작을 확인



[그림 4-20] 2AND 회로의 시뮬레이션 결과