1번.함수표현

```
library IEEE; -- 사용할 라이브러리 지정
use IEEE.std_logic_1164.all; --iEEE 라이브러리에서 std_logic_1164패키지 가져온다

entity ORNAND_GATE is -- entity 선언

        port( A, B: in std_logic; -- 입출력선언
                X, Y : out std_logic);
end ORNAND_GATE;

architecture func of ORNAND_GATE is -- 아키텍쳐 선언
        function OR_GATE ( A, B: std_logic) -- 함수 선언 및 입력과 리턴값 선언
                return std_logic is
begin
        return A or B; -- 리턴값
end OR_GATE;

function NAND_GATE ( B, C: std_logic) --함수 선언 및 입력과 리턴값 선언
        return std_logic is
begin
        return B nand C; --리턴값
end NAND_GATE;
begin
        X <= OR_GATE(A, B); --OR함수의 리턴값을 X 에
        Y <= NAND_GATE(A, B); -- NAND함수 리턴값을 Y에
end func;
```

2번.프로시저 표현

```
library IEEE; -- 사용할 라이브러리 지정
use IEEE.std_logic_1164.all; --iEEE 라이브러리에서 std_logic_1164패키지 가져온다

entity ORNAND_GATE is          -- entity선언
        port( A, B: in std_logic;          --입출력 선언
                X, Y : out std_logic);
end ORNAND_GATE;

architecture proc of ORNAND_GATE is -- 아키텍쳐 선언
procedure OR_GATE    --프로시저 선언
        (signal A, B: in std_logic;        -- 입출력 선언
        signal O : out std_logic) is
begin
        O <= A or B;    -- A or B 출력
end OR_GATE;

procedure NAND_GATE --프로시저 선언
        (signal A, B: in std_logic; -- 입출력 선언
        signal O : out std_logic) is
begin
        O <= A nand B; -- A nand B 출력
end NAND_GATE;
begin
        OR_GATE(A, B, X); --X에 출력값
        NAND_GATE(B, C, Y); --Y에 출력값
end proc;
```

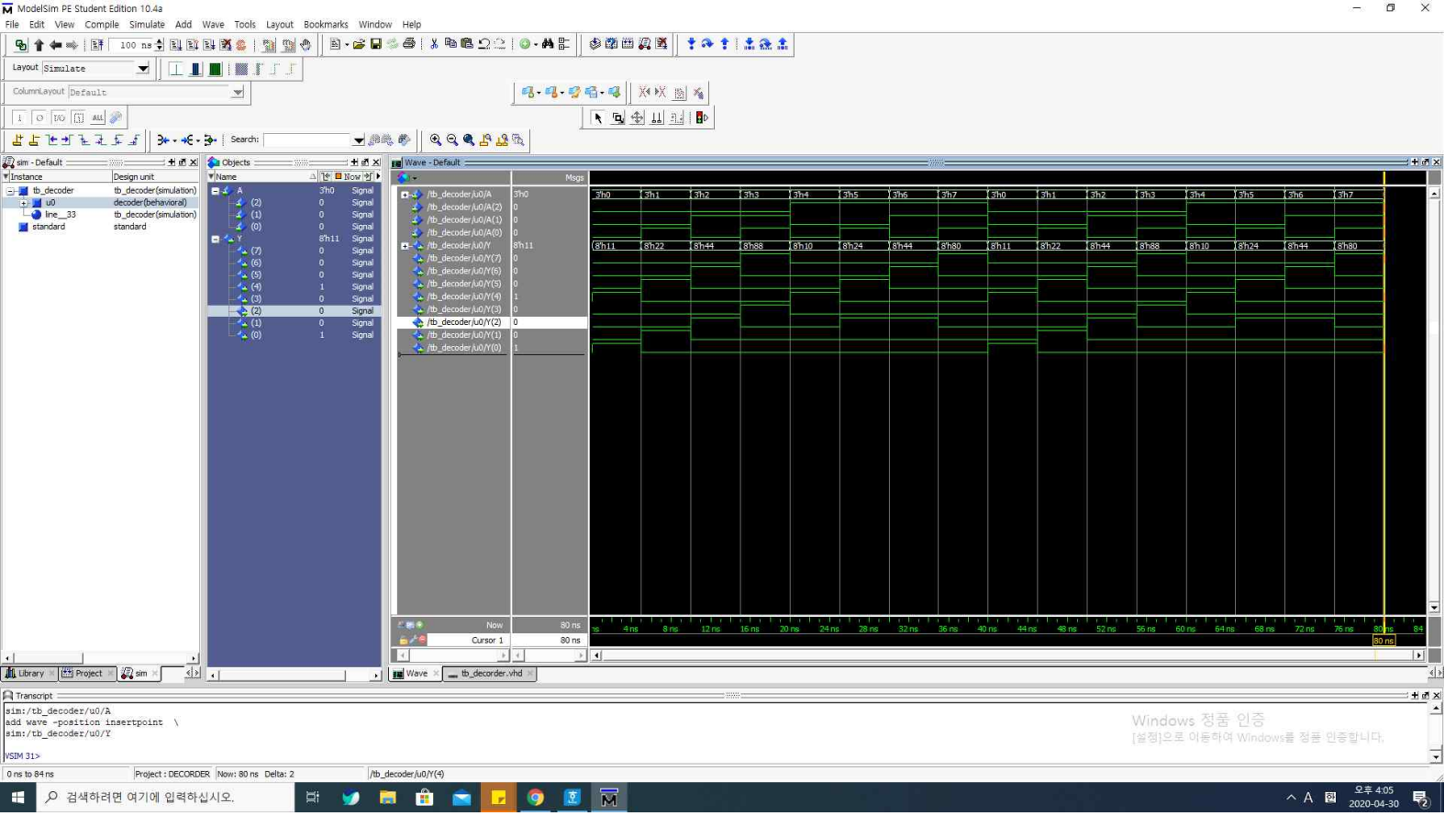
3번.디코더

```
entity decoder is-- entity 선언
	port (A : in bit_vector(2 downto 0); Y : out bit_vector(7 downto 0));
end decoder; -- 입력 3비트, 출력8비트

architecture behavioral of decoder is -- 아키텍쳐 선언
begin
	process(A) -- 프로세스문 A가 변할 때 마다
	begin
		case A is -- A에 맞는 Y값 넣는다.
			when "000" => Y <= "00010001";
			when "001" => Y <= "00100010";
			when "010" => Y <= "01000100";
			when "011" => Y <= "10001000";
			when "100" => Y <= "00010000";
			when "101" => Y <= "00100100";
			when "110" => Y <= "01000100";
			when "111" => Y <= "10000000";
			when others=> Y <= "00000001";
		end case; --case문 종료
	end process; -- 프로세스문 종료
end behavioral; --아키텍쳐 종료

entity tb_decoder is--테스트벤치 선언
end tb_decoder ;

architecture simulation of tb_decoder is --테스트벤치 아키텍쳐
	component decoder--컴포넌트 선언
		port ( A : in bit_vector(2 downto 0); --입력 3비트
			Y : out bit_vector(7 downto 0)); -- 출력 8비트
	end component; --컴포넌트문 종료
	signal signal_a : bit_vector(2 downto 0); -- 3비트 siganl
begin
	signal_a <= "000", "001" after 5 ns, "010" after 10 ns, "011" after 15 ns,
		"100" after 20 ns, "101" after 25 ns, "110" after 30 ns, "111" after 35 ns,
		"000" after 40 ns, "001" after 45 ns, "010" after 50 ns, "011" after 55 ns,
		"100" after 60 ns, "101" after 65 ns, "110" after 70 ns, "111" after 75 ns,
		"000" after 80 ns; --파형생성문으로 입력신호생성
	u0: decoder port map (A=>signal_a, Y=>open); -- 컴포넌트 실체화문
end ;
```



run 80ns 실행 화면

5ns마다 입력이 바뀌며 출력값이 아래와 같이 출력된다.

when "001" => Y <= "00100010";

when "010" => Y <= "01000100";

when "011" => Y <= "10001000";

when "100" => Y <= "00010000";

when "101" => Y <= "00100100";

when "110" => Y <= "01000100";

when "111" => Y <= "10000000";

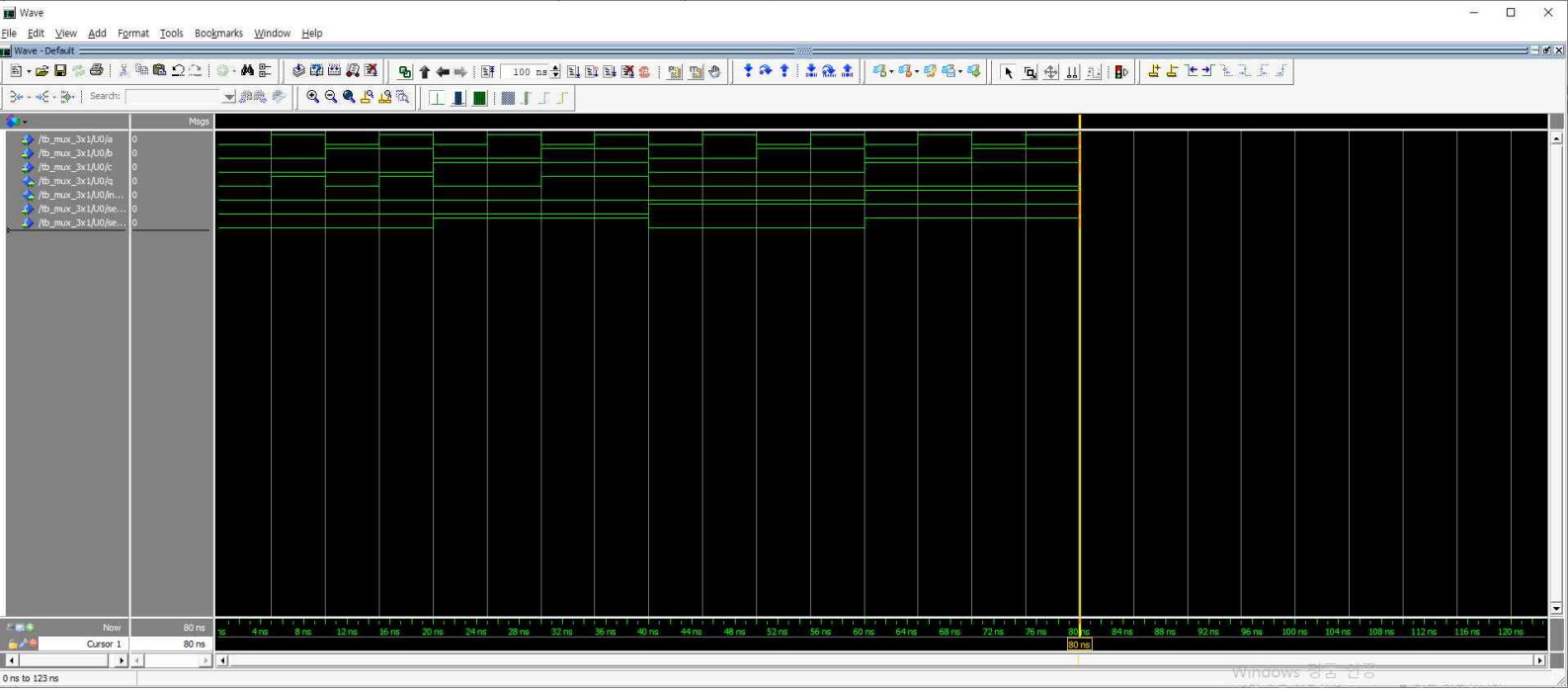
when others=> Y <= "00000001";

4.3x1멀티플렉서

```
entity mux_3x1 is -- entity선언
        port( a, b, c: in bit; sel : in bit_vector(1 downto 0); -- 입출력선언
        q,invalid : out bit);
end;
architecture with_select of mux_3x1 is -- 아키텍쳐 선언
begin

        with sel select --sel의 변화에 따른 q신호 입력
                q <= a when "00",
                        b when "01",
                        c when "10",
                        '0' when "11",
                        a when others;
        with sel select --sel의 변화에 따른 invalid 신호 입력
                invalid <= '0'when "00",
                        '0' when "01",
                        '0' when "10",
                        '1' when "11",
                        '0' when others;

end;
entity tb_mux_3x1 is --testbench 선언
end tb_mux_3x1 ;
architecture simulation of tb_mux_3x1 is --testbench architecture 선언
        component mux_3x1 -- component 선언
                port ( a, b, c: in bit; sel : in bit_vector(1 downto 0); -- 입출력 선언
                        q,invalid: out bit);
        end component; --component 종료
        signal a,b,c : bit; signal sel : bit_vector(1 downto 0); --signal 선언
begin
        a <= '0', '1' after 5 ns, '0' after 10 ns, '1' after 15 ns, --파형 생성문을 통한 입력신호 생성
        '0' after 20 ns, '1' after 25 ns, '0' after 30 ns, '1' after 35 ns,
        '0' after 40 ns, '1' after 45 ns, '0' after 50 ns, '1' after 55 ns,
        '0' after 60 ns, '1' after 65 ns, '0' after 70 ns, '1' after 75 ns, '0' after 80 ns;
        b <= '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;
        c <= '0', '1' after 20 ns, '0' after 40 ns, '1' after 60 ns, '0' after 80 ns;
        SEL <= "00", "01" after 20 ns, "10" after 40 ns, "11" after 60 ns, "00" after 80 ns;
        U0 : mux_3x1 port map (A=>A,B=>B,C=>C, SEL=> SEL, Q=>open,invalid=>open); --component 실체화문
end ;
```



sel의 신호가 00일 때는 q<=a, invalid<=0

sel의 신호가 01일 때는 q<=b, invalid<=0

sel의 신호가 10일 때는 q<=c, invalid<=0

sel의 신호가 11일 때는 q<=0, invalid<=1 이 된다.

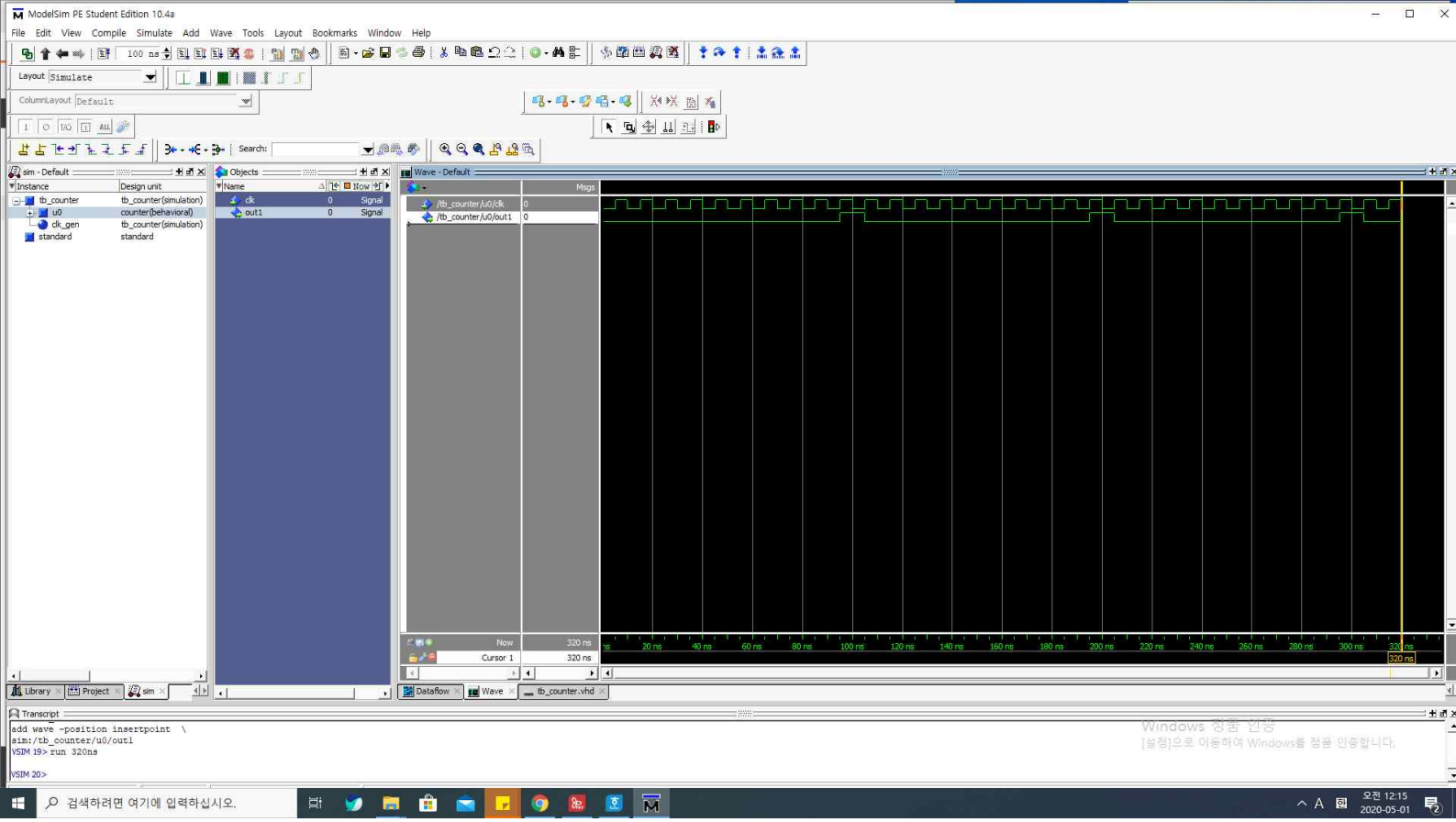
5.카운터

```
entity counter is -- entity 선언
    port(clk: in bit; --입출력 선언
          out1 : out bit);
end counter;

architecture behavioral of counter is --아키텍쳐 선언
begin
    process(clk) --clk에 대한 프로세스문
    variable temp1 : integer range 0 to 10; --0에서 10까지변하는 변수  temp1
    begin
       if(clk'event and clk='1') then -- clk이 라이징 할 때 마다
                out1<='0'; -- out에 0신호
                temp1 := temp1 + 1; --temp+1
            if(temp1 = 10) then -- temp가 10이면
                 temp1 := 0; -- temp=0으로
                out1<='1'; --out에 1신호
            end if;
        end if;
    end process; -- 프로세스문 종료
end behavioral; --아키텍쳐 종료

entity tb_counter is  --테스트벤치 선언
end tb_counter ;

architecture simulation of tb_counter is --testbench architecture 선언
        component counter -- component 선언
                port(clk: in bit;
           out1 : out bit);
        end component; --component 종료
constant HALF_PERIOD_100M : time := 5 ns ; --클럭주기 5ns
        signal out1 : bit; --signal선언
        signal clk: bit;
begin
        clk_gen : process --프로세스문
begin
                while (true) loop --클럭 루프
                        clk <= '0' ; wait for HALF_PERIOD_100M ;
                        clk <= '1' ; wait for HALF_PERIOD_100M ;
                end loop;
        end process; --프로세스문 종료
        u0: counter port map (out1=>out1, clk=>clk); --컴포넌트 실체화
end simulation;
```



10번째 클럭마다 한클럭의 폭을 갖는 펄스가 생성된다.