```
library ieee ;
use ieee.std_logic_1164.all ; --사용하는 라이브러리

entity pattern is ㅡ엔티티 정의
	port (
		bitstream : in std_logic;
		detect_flag : out std_logic;
		reset_disp_n : in std_logic;
		dispclk : in std_logic); --입출력 신호 정의
end pattern;

architecture behavioral of pattern is ㅡ아키텍쳐 정의

	TYPE state_type IS (idle, tap1, tap2, tap3, tap4); -- 타입 선언
	SIGNAL current_state, next_state: state_type; --신호 선언

begin

	next_state_logic : process(current_state, bitstream) --프로세스문
	begin
	case current_state is --case문
		when idle =>
			detect_flag <= '0';
			if bitstream = '1' then --1이면 idle로
				next_state <= idle;
			else  --0이면 tap1로
				next_state <= tap1;
			end if;
		when tap1 =>
			detect_flag <= '0';
			if bitstream = '0' then --0이면 tap2로
				next_state <= tap2;
			else --1이면 idle로
				next_state <= idle;
			end if;
		when tap2 =>
			detect_flag <= '0';
			if bitstream = '1' then --1이면 tap3으로
				next_state <= tap3;
			else --0이면 tap2로
				next_state <= tap2;
			end if;
		when tap3 =>
			detect_flag <= '0';
			if bitstream = '1' then --1이면 tap4로
				next_state <= tap4;
			else --0이면 tap1로
				next_state <= tap1;
			end if;
		when tap4 =>
			detect_flag <= '1';
			if bitstream = '1' then --1이면 idle로
```

```
                        next_state <= idle;
                else --0이면 tap1로
                        next_state <= tap1;
                end if;
        when others =>
                detect_flag <= '0';
                next_state <= idle;
    end case; --case문 종료
end process; --프로세스문 종료

state_register: process(dispclk, reset_disp_n) 프로세스문
begin
    if (reset_disp_n='0') then --reset신호가 0이면
            current_state <= idle;
    elsif (dispclk='1' and dispclk'event) then --클럭 라이징엣지일때
            current_state <= next_state;
    end if;
end process; --프로세스문 종료

end ;


library ieee;
use ieee.std_logic_1164.all; --사용하는 라이브러리

entity tb_pattern is --테스트벤치 엔티티 정의
end tb_pattern;

architecture sim of tb_pattern is --테스트벤치 아키텍쳐 정의
component pattern --컴포넌트문
    port (
            bitstream : in std_logic;
            detect_flag : out std_logic;
            reset_disp_n : in std_logic;
            dispclk : in std_logic); --입출력 신호 정의
end component; --컴포넌트문 종료

constant HALF_PERIOD_100M : time := 5 ns ; --클럭신호 주기 설정

signal clk, rst_n : std_logic;
signal bitstream : std_logic; --신호 선언

begin

rst_n <= '0','1'after 20 ns ; --리셋신호 20ns부터 1

clk_gen : process --프로세스문
begin
    while (true) loop
            clk <= '0' ; wait for HALF_PERIOD_100M ;
            clk <= '1' ; wait for HALF_PERIOD_100M ; --클럭 신호주기
    end loop;
```

```
end process;

bitstream <= '1', '1' after 10 ns,
'1' after 20 ns, '1' after 30 ns,
'0' after 40 ns, '1' after 50 ns,
'1' after 60 ns, '0' after 70 ns,
'1' after 80 ns, '0' after 90 ns,
'0' after 100 ns, '1' after 110 ns,
'1' after 120 ns, '0' after 130 ns,
'0' after 140 ns, '1' after 150 ns,
'1' after 160 ns, '0' after 170 ns,
'1' after 180 ns, '1' after 190 ns,
'0' after 200 ns, '0' after 210 ns,
'1' after 220 ns, '1' after 230 ns,
'1' after 240 ns, '1' after 250 ns,
'1' after 260 ns, '0' after 270 ns,
'0' after 280 ns, '0' after 290 ns,
'1' after 300 ns, '1' after 310 ns,
'1' after 320 ns, '1' after 330 ns; -- 파형생성문

u0 : pattern port map(bitstream => bitstream, detect_flag => open, reset_disp_n => rst_n, dispclk=>clk);
end sim ; -- 컴포넌트 실체화문
```