실험 10 Rotary Encoder

실험 목표

모터의 회전각 측정 및 입력 장치로 널리 사용되고 있는 로터리 엔코더의 동작을 이해하고 이를 이용하여 카운터를 설계해 본다.

실험 부품

FPGA Board (EP4CE6)
Rotary Encoder (GA1F1724B, FPGA Board에 부착)
Seven-segment display module (FPGA Board에 부착)
USB Blaster Ⅱ
Quartus Ⅱ
디지털 멀티미터(digital multimeter)

관련 이론

1. 로터리 엔코더(rotary encoder)

그림 10-1에 나타낸 로터리 엔코더는 축의 회전에 따른 디지털 신호를 만들어낸다. 일반적으로 모터의 속도 제어를 위한 피드백에 사용되며, 그 외에도 각종 전자 기기의 입력장치로도 사용된다.

그림 10-1 모터축에 연결되어 있는 로터리 엔코더의 모양

2. 실험에 사용할 로터리 엔코더 스위치 및 응용 회로

이 실험에서는 그림 10-2의 인크리멘탈 로터리 엔코더(incremental rotary encoder)를 응용한 회로를 사용하여 엔코더 축의 회전 방향에 따라 그림 10-3과 같은 출력 파형을 얻게 된다.

그림 10-3에 나타낸 바와 같이 로터리 엔코더의 축이 시계방향으로 회전할 때 A와 B의 출력은 "00 → 10 → 11 → 01 → 00 … "으로 변하는 파형을 반복하여 출력하게 되며, 반시계 방향으로 회전할 때에는 "00 → 01 → 11 → 10 → 00 … "으로 출력한다. 이 때, '00'에서 시작하여 '00'으로 돌아올 때까지가 엔코더 출력 파형의 한 주기이며, 로터리 엔코더의 분해능은 엔코더 축이 한 바퀴 회전할 때 발생하는 펄스의 주기 수로 나타낸다.

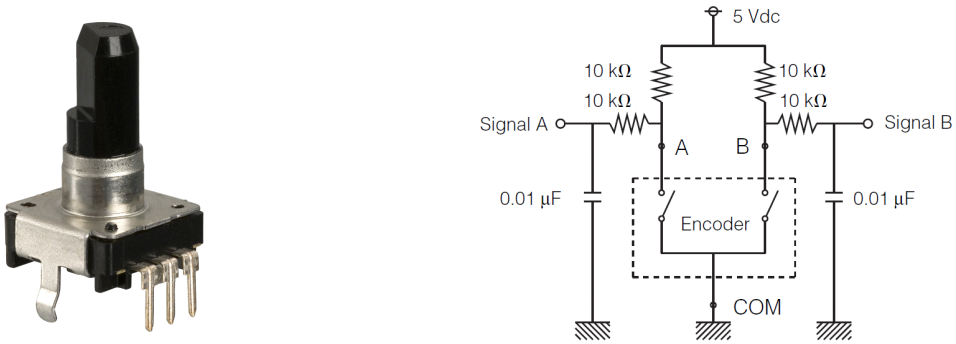

그림 10-2 실험에 사용할 로터리 엔코더 스위치의 모양과 이를 이용한 펄스 발생 회로

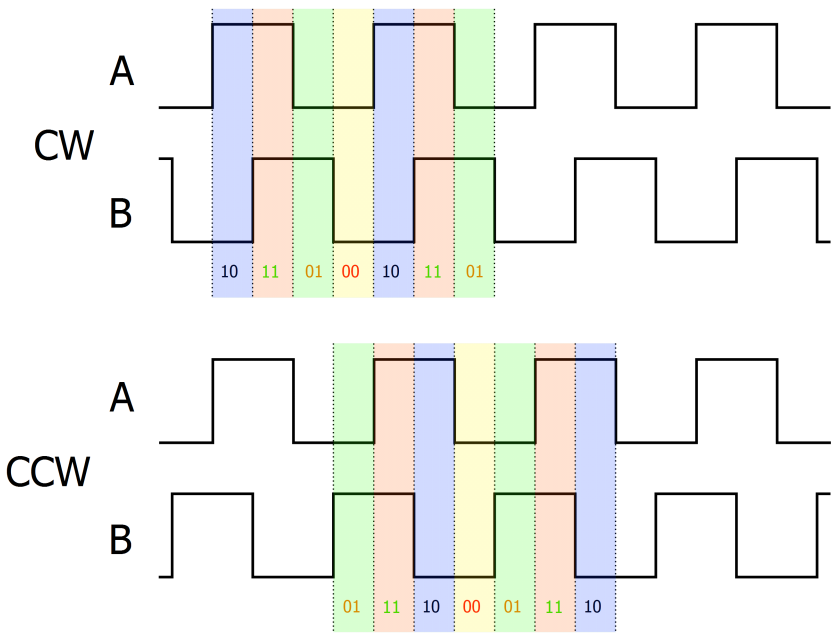

그림 10-3 회전 방향에 따른 Rotary Encoder 출력 파형

실험 순서

실험 A Rotary encoder 출력에 대한 state graph

그림 10-3을 참고하여 Rotary encoder의 출력 A와 B가 가지는 값에 따라 네 가지의 상태를 정의하고 이들 사이의 천이를 state graph로 나타내시오.

S0: A=__, B=__	S2: A=__, B=__	S3: A=__, B=__	S1: A=__, B=__

그림 10-4 Rotary Encoder의 State Graph

실험 B 로터리 엔코더를 이용한 mod-16 up/down counter 설계

로터리 엔코더 GA1F1724B의 datasheet와 그림 10-2, 그림 10-3 및 그림 10-4를 참고하여 A와 B의 값에 따라 상향 또는 하향 계수를 하는 16진 상하향 카운터(mod-16 up/down counter)를 FPGA/VHDL로 설계 및 구현한다. 이 카운터의 Block Diagram을 그림 10-5에 나타내었다.

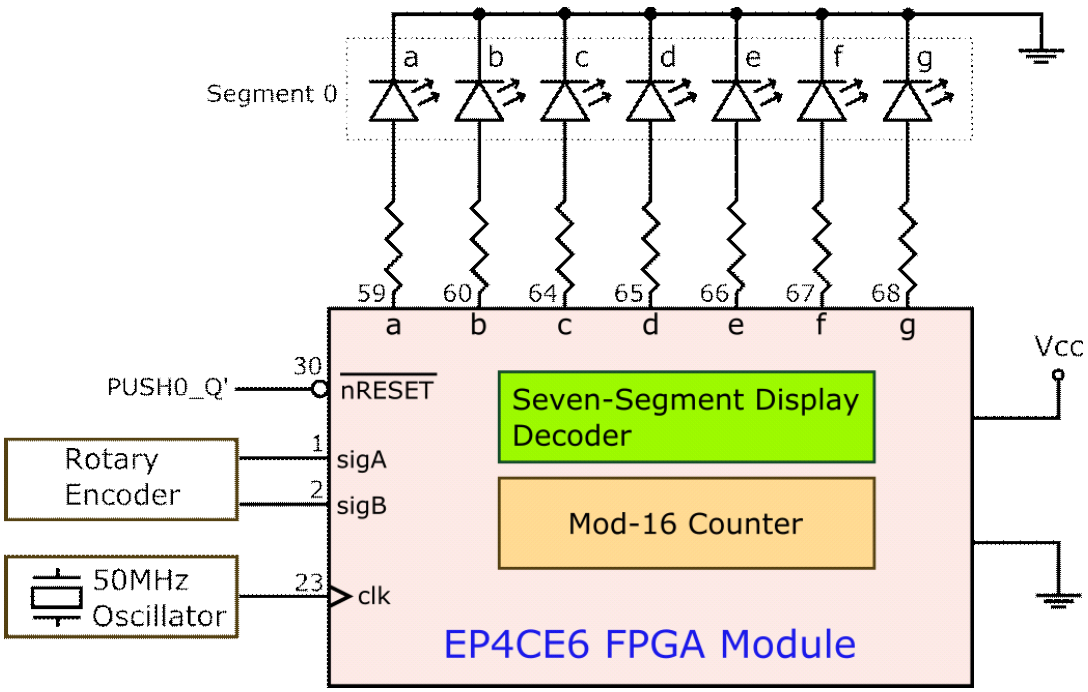

그림 10-5 Rotary Encoder를 이용한 Mod-16 Up/Down Counter

1. 주어진 카운터를 다음과 같이 두 개의 process로 나누어 설계 및 구현한다.

1) mod_16_up_down_counter process

(1) 그림 10-3 및 그림 10-4에 나타낸 바와 같이 rotary encoder로부터의 두 출력 sigA와 sigB의 값에 따라 네 개의 상태를 정의한다.

(2) 클럭 신호가 입력되면 네 개의 '현재 상태' (pr_state) 각각에 대해 sigA 및 sigB 값에 따라 카운터의 계수 방향(상향 혹은 하향)을 결정하고, '다음 상태' (nx_state)를 결정한다.

(3) 계수 방향에 따라 카운터의 값을 갱신하고 '다음 상태' (nx_state)를 '현재 상태' (pr_state)로 갱신한다.

2) count_to_ssd_decoder process

참고: rotary encoder가 한쪽 방향으로만 회전한다면 엔코더 펄스의 한 주기(하나의 detent로부터 다음 detent까지)에 대해 sigA와 sigB 신호의 변화는 네 번 일어난다.

(1) 위 mod_16_up_down_counter process의 3)에서 갱신된 카운터의 값을 4분주한 결과를 seven-segment display에 출력한다.

2. Simulation을 통해 제대로 동작하는지 확인한 후, FPGA에 download하여 실제의 동작을 확인한다.

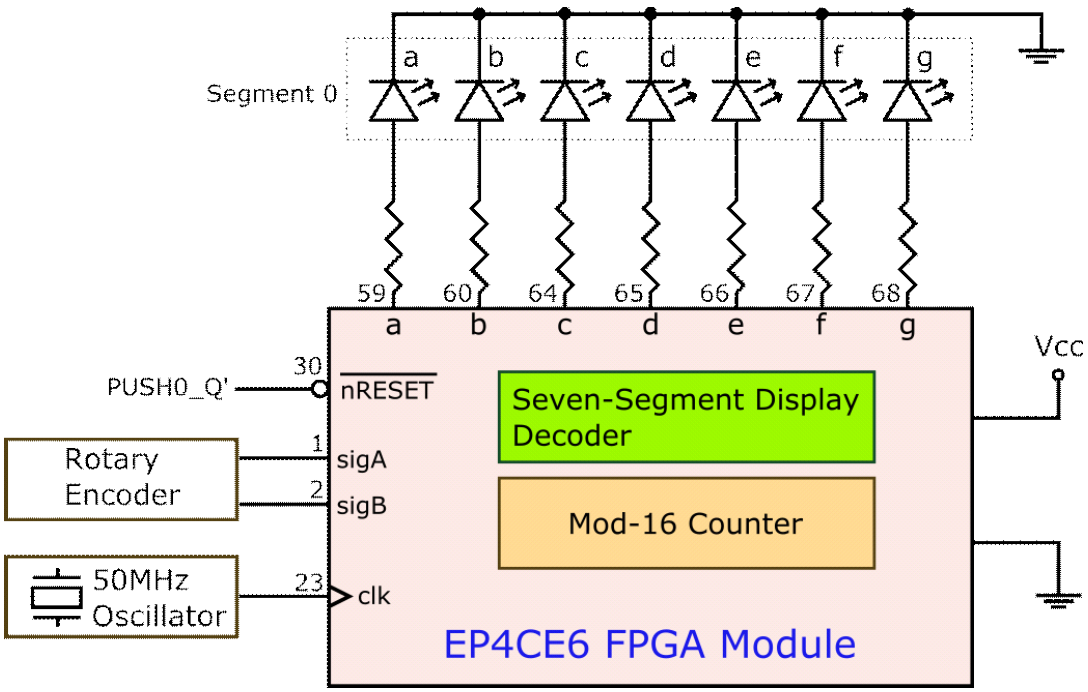

그림 10-5 Rotary Encoder를 이용한 Mod-16 Up/Down Counter

```
library ieee;
use ieee.std_logic_1164.all;

entity rotary_encoder_counter is
    port (clk, nReset, sigA, sigB: in  std_logic;
                          segment7: out std_logic_vector(6 downto 0));
end entity;

architecture my_counter of rotary_encoder_counter is
    signal mod_64_counter: integer range 0 to 63;
    signal pr_state: integer range 0 to 3;
    signal sigAB: std_logic_vector (1 downto 0);
begin
    process (clk, nReset, sigA, sigB)
        variable dir: integer range -1 to 1;
        variable nx_state: integer range 0 to 3;
    begin
        if nReset = '0' then
            mod_64_counter <= 0;
            pr_state <= 0;
        elsif (clk'event and clk = '1') then
            sigAB <= sigA & sigB;
            case pr_state is
                when 0 =>
                    if sigAB="10" then dir _______; nx_state _______;
                    elsif sigAB="01" then dir _______; nx_state _______;
                    else dir _______;  nx_state _______;
                    end if;
                when 1 =>
                    if sigAB="00" then dir _______; nx_state _______;
                    elsif sigAB="11" then dir _______; nx_state _______;
                    else dir _______;  nx_state _______;
                    end if;
                when 2 =>
                    if sigAB="11" then dir _______; nx_state _______;
                    elsif sigAB="00" then dir _______; nx_state _______;
                    else dir _______; nx_state _______;
                    end if;
                when 3 =>
                    if sigdeAB="01" then dir _______; nx_state _______;
                    elsif sigAB="10" then dir _______; nx_state _______;
```

```
                else dir _______;  nx_state _______;
                end if;
            end case;

            mod_64_counter <= _________________;      -- update counter
            pr_state <= _____________;                -- update pr_state
        end if;
    end process;

    process (mod_64_counter)
        variable count: integer range 0 to 15;
    begin
        count := ___________________;          -- 4분주하여 ssd에 표시
        case count is
            when 0 => segment7 <=_____________________;  -- '0'
            when 1 => segment7 <=_____________________;  -- '1'
            when 2 => segment7 <=_____________________;  -- '2'
            when 3 => segment7 <=_____________________;  -- '3'
            when 4 => segment7 <=_____________________;  -- '4'
            when 5 => segment7 <=_____________________;  -- '5'
            when 6 => segment7 <=_____________________;  -- '6'
            when 7 => segment7 <=_____________________;  -- '7'
            when 8 => segment7 <=_____________________;  -- '8'
            when 9 => segment7 <=_____________________;  -- '9'
            when 10 => segment7 <=_____________________;  -- 'A'
            when 11 => segment7 <=_____________________;  -- 'b'
            when 12 => segment7 <=_____________________;  -- 'C'
            when 13 => segment7 <=_____________________;  -- 'd'
            when 14 => segment7 <=_____________________;  -- 'E'
            when 15 => segment7 <=_____________________;  -- 'F'
            when others => segment7 <= null;
        end case;
    end process;
end architecture;
```

결과 및 결론

숙제

1. 위 실험 B에서 A, B의 입력 파형에 Bouncing Noise가 섞여 있을 때의 동작을 "Waveform_with_noise.vwf" 입력 파형을 이용하여 Simulation을 통해 분석하시오.

2. VHDL에서 사용되는 signal과 variable의 차이점에 대해 설명하시오.

3. Rotary encoder의 이용 분야에 대해 조사하여 보고서에 쓰시오.