실험 8 카운터(Counter)

실험 목표

비동기식 리셋(reset) 입력과 병렬 입력(parallel-input)을 가지는 동기식 상향(count-up) 카운터와 상하향(count-up/down) 카운터의 동작 원리를 이해하고 설계한 후, 실험을 통해 그 동작 특성을 확인한다.

실험 부품

FPGA 실험 보드(EP4CE6)
USB Blaster Ⅱ
Quartus Ⅱ

관련 이론

1. 카운터(counter)

카운터는 플립플롭의 트리거 방식에 따라 비동기식(asynchronous)과 동기식(synchronous)으로 나뉜다. 비동기식 카운터는 직렬 카운터 또는 리플(ripple) 카운터라고도 하며, 앞단의 플립플롭의 출력이 뒷단의 플립플롭을 트리거 시킨다. 반면에 동기식 카운터는 모든 플립플롭이 같은 클럭 펄스에 의해서 동시에 트리거 되며, 병렬 카운터라고도 한다.

비동기식 카운터는 동기식 카운터에 비해 회로가 간단한 장점이 있으나 전달지연시간이 길다는 단점이 있다.

동기식 카운터는 비동기식 카운터에 비해서 전송지연이 매우 작으며, 이는 높은 주파수로 동작시킬 수 있다는 의미가 된다. 동기식 카운터는 모든 플립플롭이 동시에 동작하므로 한 상태와 다음 상태 사이에 잠정적인 중간상태가 존재하지 않는다.

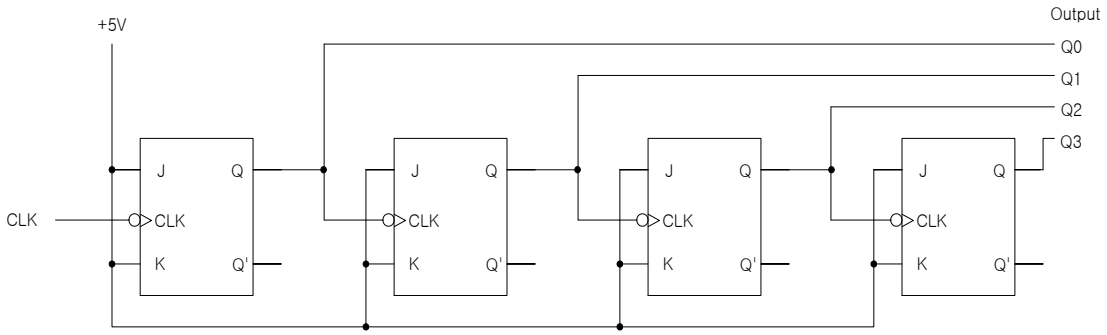

그림 8-1. 비동기식 카운터

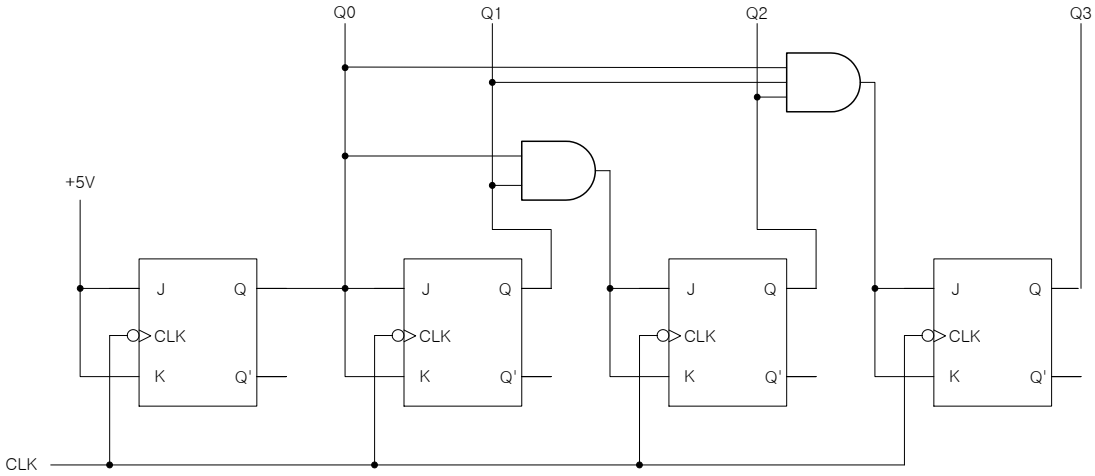

그림 8-2. 동기식 카운터

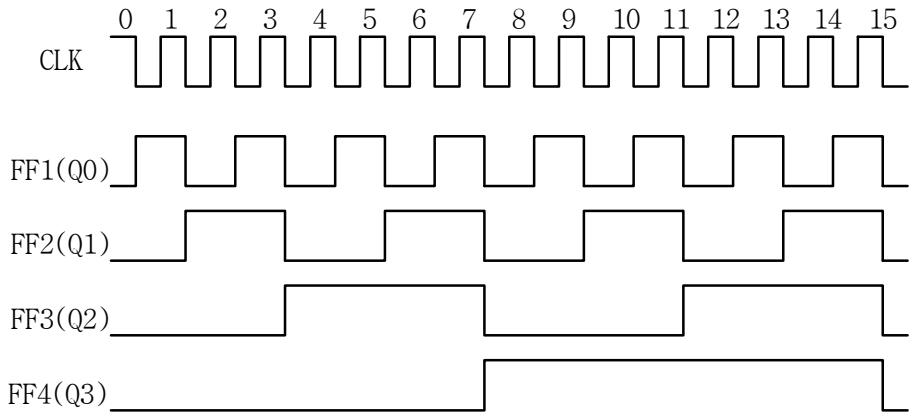

그림 8-3. 카운터 파형

2. BCD 카운터(counter)

BCD 카운터는 2진화 10진수(binary-coded decimal)를 0000에서 1001까지 세고, 다시 0000으로 되돌아간다.

3. 모듈로(modulo)-N 카운터

모듈로(modulo)-N (줄여서 mod-N) 카운터는 N개의 상태를 차례로 되풀이해서 계수하는 카운터이다. 예를 들어 모듈로(modulo)-5 카운터는 0부터 4까지 5가지의 상태를 되풀이하면서 계수한다.

실험 순서

아래의 모든 카운터의 출력을 7-segment LED에 나타내시오.

실험 A 동기식 상향 카운터
(mod-10 counter with asynchronous active low reset)

FPGA/VHDL을 이용하여 rising edge triggered synchronous up counter를 설계 및 구현하고, 그 기능을 시험해 본다. 이 카운터는 0부터 9까지의 10개의 상태를 계수할 수 있는 modulo-10 counter이다. (9까지 계수한 후에 다음 클럭 신호가 입력되면 다시 0으로 돌아가서 계수를 계속한다.) 외부에는 별도의 비동기식 active low `Reset` 입력이 있으며, 이 `Reset` 입력에 논리값 '0'이 입력되면 이 카운터는 0으로 리셋된다.

입력

`CLK` (Clock)
`Reset` (Active low reset)

출력

`a, b, c, d, e, f, g` - common cathode type의 SSD 구동 출력.

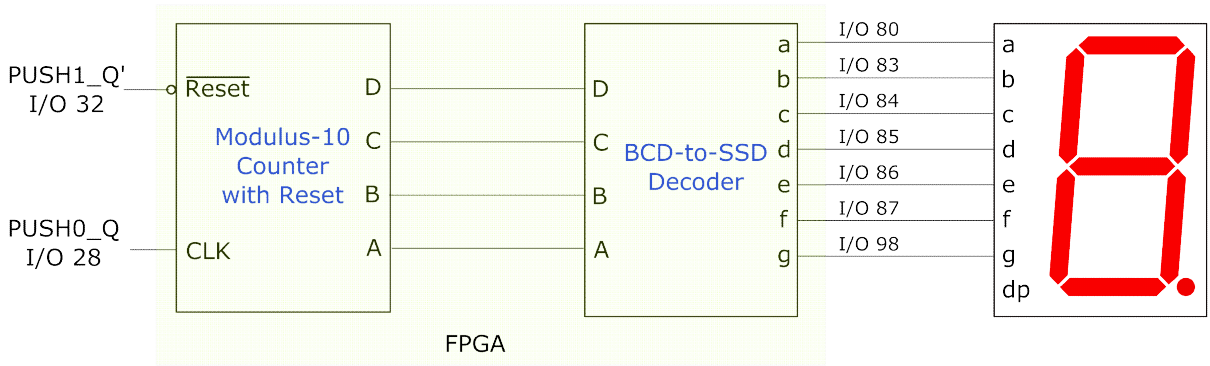

그림 8-4 Mod-10 동기식 상향 카운터

`mod_10_counter` process와 `bcd_to_ssd` process의 두 개의 process를 만든다.

1. `mod_10_up_counter` process
   1) 입력: `CLK`, `Reset`
   2) 출력: 계수값 (BCD 형태)

2. `bcd_to_ssd` process
   1) 입력: `mod_10_up_counter`의 출력 (BCD 형태)
   2) 출력: SSD를 구동하기 위한 7개의 출력 신호(`a, b, c, d, e, f, g`)

실험 B 병렬 데이터 적재가 가능한 동기식 상하향 카운터
(Presettable mod-10 up/down counter with asynchronous active low reset)

Parallel Load, Count Up, Count Down의 세 가지 기능을 가지는 4-bit presettable rising edge triggered synchronous up counter를 FPGA/VHDL을 이용하여 설계 및 구현하고, 그 기능을 시험해 본다.
또한, asynchronous active low reset($\overline{\text{Reset}}$) 입력이 있으며, 이 $\overline{\text{Reset}}$ 입력이 '0'일 경우에는 카운터가 0으로 리셋되고, 이 $\overline{\text{Reset}}$ 입력이 '1'일 경우에는 아래의 표 8-1에 나타낸 것처럼 동작을 결정하는 2개의 제어선 S1과 S0의 값에 따라 동작한다.

병렬 입력 데이터(parallel input data)의 값이 9보다 크면 9로 처리한다.
상향 계수 때, 9까지 계수한 후에는 0으로 돌아가서 계수를 계속한다.
하향 계수 때, 0까지 계수한 후에는 9로 돌아가서 계수를 계속한다.

$\overline{\text{Reset}}$	S1	S0	Functions
0	×	×	Clear
1	0	0	No Change
1	0	1	Parallel Load
1	1	0	Count Up
1	1	1	Count Down

표 8-1 제어선 S1과 S0의 값에 따른 동작

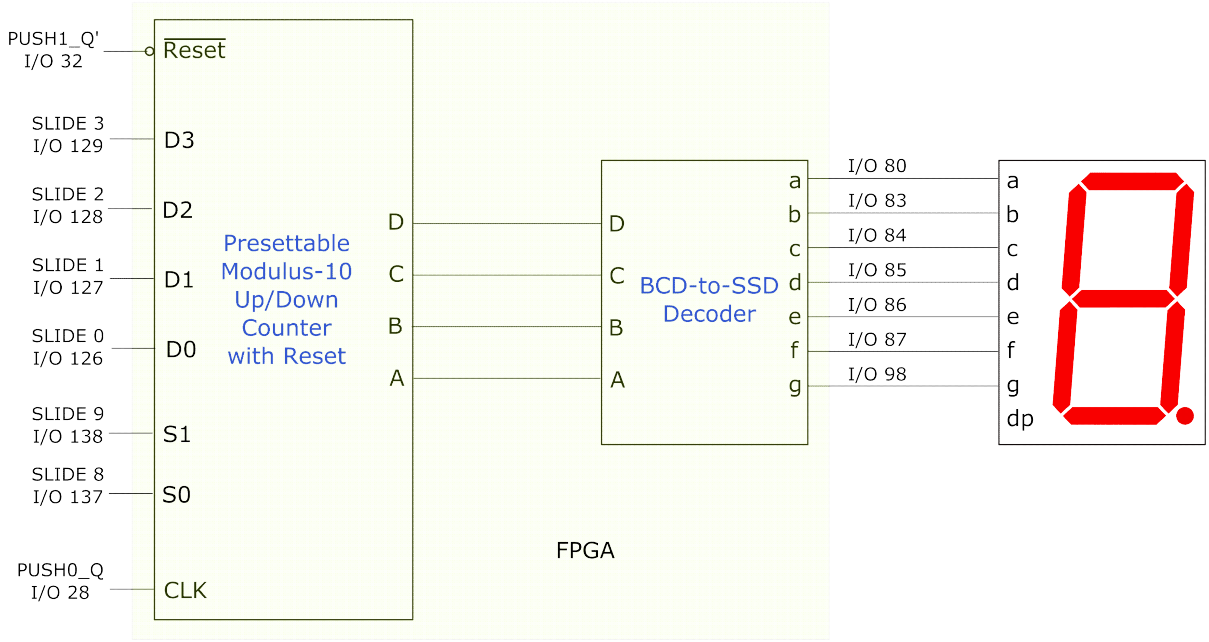

그림 8-5 병렬 데이터 적재가 가능한 동기식 상하향 카운터

입력

`CLK` (clock)

$\overline{\texttt{Reset}}$ (active low reset)

`S1`, `S0` - function control signals

`D3`, `D2`, `D1`, `D0` - parallel input data

출력

`a, b, c, d, e, f, g` - common cathode type의 SSD 구동 출력.

`mod_10_up_down_counter` process와 `bcd_to_ssd` process의 두 개의 process를 만든다.

1. `mod_10_up_down_counter` process
   1) 입력: `CLK`, $\overline{\texttt{Reset}}$, `S1`, `S0`, `D3`, `D2`, `D1`, `D0`
   2) 출력: 계수값 (BCD 형태)

2. `bcd_to_ssd` process
   1) 입력: `mod_10_up_down_counter`의 출력 (BCD 형태)
   2) 출력: SSD를 구동하기 위한 7개의 출력 신호(`a, b, c, d, e, f, g`)

실험 C 클럭 분주(clock frequency division)

FPGA 실험 보드(EP4CE6)에는 50MHz의 crystal oscillator가 부착되어 있으며, 여기에서 발생한 50MHz의 신호는 EP4CE6의 I/O 23 핀으로 공급되어 소자 전체의 클럭 신호로 사용된다. 이 클럭 신호를 분주하여 실험 B의 동기식 상하향 카운터가 1초에 1번씩 카운트 되도록 구현하시오.

입력

`CLK` (clock, I/O 23번 핀으로 공급되는 50MHz)
$\overline{\texttt{Reset}}$ (active low reset)
`S1`, `S0` - function control signals
`D3`, `D2`, `D1`, `D0` - parallel input data

출력

`a, b, c, d, e, f, g` - common cathode type의 SSD 구동 출력.

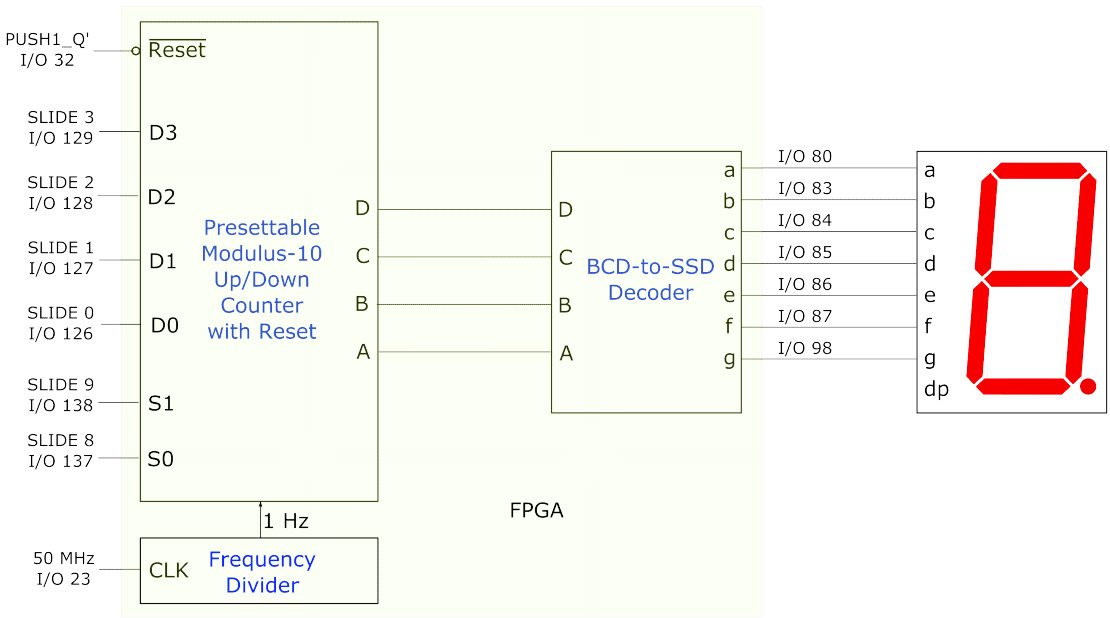

그림 8-5 내부 클럭을 분주한 카운터

`freq_divider` process와 `mod_10_up_down_counter` process 및 `bcd_to_ssd` process의 세 개의 process를 만든다.

1. `freq_divider` process
   1) 입력: `CLK` (clock, I/O 23번 핀으로 공급되는 50MHz)
   2) 출력: 1 Hz 펄스 신호

2. `mod_10_up_down_counter` process statement
   1) 입력: 1 Hz 펄스 신호, $\overline{\texttt{Reset}}$, `S1`, `S0`, `D3`, `D2`, `D1`, `D0`

2) 출력: 계수값 (BCD 형태)

3. `bcd_to_ssd` process statement
   1) 입력: `mod_10_up_down_counter`의 출력 (BCD 형태)
   2) 출력: SSD를 구동하기 위한 7개의 출력 신호(`a, b, c, d, e, f, g`)

결과 및 결론

실험 A 동기식 상향 카운터
(Rising edge triggered synchronous up counter with asynchronous active low reset)

FPGA/VHDL을 이용하여 rising edge triggered synchronous up counter를 설계 및 구현한 후, VHDL code와 simulation 결과를 보고서에 첨부하시오. 필요하다면 별도의 출력 신호를 할당하여 관찰할 것.

실험 B 병렬 데이터 적재가 가능한 동기식 상하향 카운터
(Presettable rising edge triggered synchronous up/down counter with asynchronous active low reset)

Parallel Load, Count Up, Count Down의 세 가지 기능을 가지는 4-bit presettable rising edge triggered synchronous up counter를 FPGA/VHDL을 이용하여 설계 및 구현한 후, VHDL code와 simulation 결과를 보고서에 첨부하시오. 필요하다면 별도의 출력 신호를 할당하여 관찰할 것.

실험 C 클럭 분주(clock frequency division)

FPGA 실험 보드(EP4CE6)에는 50MHz의 oscillator가 부착되어 있으며, 여기에서 발생한 50MHz의 신호는 EP4CE6의 I/O 23번 핀으로 공급되어 소자 전체의 클럭 신호로 사용된다. 이 클럭 신호를 분주하여 실험 B의 동기식 상하향 카운트가 1초에 1번씩 카운트 되도록 구현하고, 주변 하드웨어를 연결한 후 그 기능을 시험하여 그 결과를 VHDL code와 함께 보고서에 첨부하시오. 필요하다면 별도의 출력 신호를 할당하여 관찰할 것.

(참고)

실험 A 동기식 상향 카운터
(Rising edge triggered synchronous up counter with asynchronous active low reset)

FPGA 실험 보드(EP4CE6)의 PUSH0_Q switch를 누를 때마다 발생되는 클럭 신호를 이용하여 mod-10 동기식 상향 카운터 구현.

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

entity mod_10_counter is
    port (      CLK: in std_logic;
                ssd: out std_logic_vector(6 downto 0));
end entity;
-----------------------------------------------
architecture my_counter of mod_10_counter is

      필요하면 이곳에 signal 선언

begin
    -- counter process
    mod_10_up_counter: process(CLK, RESET)
    begin

        1. 비동기 RESET 신호를 먼저 처리
        2. CLK 입력시 0부터 9까지 계수하여 적절한 signal에 저장하여
          bcd_to_ssd process에 넘겨줌

    end process;

    -- seven segment decoding process
    bcd_to_ssd: process(count)
    begin

        mod_10_counter process로부터 넘겨받은 signal을 SSD에 출력

    end process;
end architecture ;
```

실험 B 병렬 데이터 적재가 가능한 동기식 상하향 카운터

(Presettable rising edge triggered synchronous up/down counter with asynchronous active low reset)

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

entity mod_10_up_down_counter_with_parallel_load is
    port (CLK, RESET: in  std_logic;
                   S: in  std_logic_vector(1 downto 0);
                   D: in  std_logic_vector(3 downto 0);
                 ssd: out std_logic_vector(6 downto 0));
end entity;
-----------------------------------------------
architecture my_counter of mod_10_up_down_counter_with_parallel_load is

      필요하면 이곳에 signal 선언

begin
    -- counter process
    mod_10_up_down_counter: process(CLK, RESET)
    begin

       1. 비동기 RESET 신호를 먼저 처리
       2. CLK 입력시 S1,S0의 값에 따라 parallel load, up count 혹은 down
          count하여 적절한 signal에 저장하여 bcd_to_ssd process에 넘겨줌

    end process;

    -- seven segment decoding process
    bcd_to_ssd: process(count)
    begin

       mod_10_counter process로부터 넘겨받은 signal을 SSD에 출력

    end process;
end architecture ;
```

실험 C 클럭 분주(clock frequency division)

FPGA 실험 보드(EP4CE6)에서 공급되는 50MHz의 클럭 신호를 분주하여 1 Hz의 클럭 신호를 만든 후, 이를 이용하여 실험 B의 동기식 상하향 카운트가 1초에 1번씩 카운트 되도록 구현.

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

entity mod_10_up_down_counter_with_parallel_load_freq_divider is
    port (CLK50MHz, RESET: in  std_logic;
                        S: in  std_logic_vector(1 downto 0);
                        D: in  integer range 0 to 15;
                      ssd: out std_logic_vector(6 downto 0));
end entity;
-----------------------------------------------
architecture my_counter of mod10_counter is

    필요하면 이곳에 signal 선언

begin

    -- frequency division process
    freq_divider: process(CLK50MHz)
    begin

    CLK50MHz의 클럭 신호를 분주하여 1Hz의 CLK1Hz 출력신호를 만들어 mod_10_counter
    process로 넘긴다.

    end process;

    -- counter process
    mod_10_up_down_counter: process(CLK1Hz)
    begin

    1. 비동기 RESET 신호를 먼저 처리
    2. CLK 입력시 S1,S0의 값에 따라 parallel load, up count 혹은 down
       count하여 적절한 signal에 저장하여 bcd_to_ssd process에 넘겨줌

    end process;

    -- seven segment decoding process
    bcd_to_ssd: process(count)
    begin

      mod_10_counter process로부터 넘겨받은 signal을 SSD에 출력

    end process;
end architecture ;
```