실험 6 Flip-Flop / Three-State Buffer

실험 목표

플립플롭과 three-state buffer의 동작 특성을 확인하고, VHDL을 통해 구현하고 고찰함으로써 플립플롭과 three-state buffer의 전반적인 이해를 도모한다.

실험 부품

~~74LS74 (D F/F)~~
~~74LS76 (JK F/F)~~
~~74LS244 (Three-State Buffer)~~
Breadboard
저항 1k x 2
LED x 2
Jump wire
~~Clock pulse generator~~
FPGA 실험 보드(EP4CE6)
USB Blaster Ⅱ
Quartus Ⅱ

관련 이론

플립플롭(flip-flop)은 쌍안정 멀티바이브레이터(bistable multivibrator)를 일컫는 것으로 0과 1의 두 개의 안정된 상태의 출력을 갖는다. 이때 두 출력은 항상 반전된 상태를 나타내며, 출력의 한쪽을 Q라 하면 다른 한쪽의 출력은 Q'이 된다. 플립플롭은 기억소자, 주파수 분주기, 계수기 등 많은 분야에 응용되며 일반적으로 플립플롭은 그 입력 회로의 구성에 따라 RS 플립플롭, JK 플립플롭, D 플립플롭 등으로 구분된다.
Three-state buffer (tri-state buffer)는 데이터 입력과 제어 입력을 가지며 3가지 출력 상태를 가지는 논리 소자이다. 제어 입력에 따라 입력이나 'High Z'가 출력된다. 여럿의 논리소자 출력이 병렬로 연결되어 있을 때 어느 한 순간에는 하나의 논리소자에서만 출력이 나와야하므로 나머지 소자는 'High Z'를 유지해야 한다.

1. SR Latch

그림 6-1에서 S와 R은 각각 set과 reset을 의미하며, S=1, R=0인 상태로 하면 Q=1, Q'=0인 상태로 되고, S=0, R=1인 상태로 하면 Q=0, Q'=1인 상태로 나타난다. 결국 S=1일 경우에는 Q 출력을 1로, R=1일 경우에는 Q 출력을 0으로 만든다.

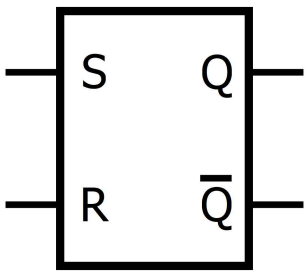


그림 6-1 SR Latch

INPUTS		OUTPUTS	
S	R	Q	Q'
0	0	No change	
0	1	0	1
1	0	1	0
1	1	Not allowed	

표 6-1 SR Latch의 진리치표

2. D 플립플롭

D 플립플롭(data flip-flop)은 clock 신호의 active edge 때의 D 입력을 기억하는 플립플롭이다. Positive-edge triggered D flip-flop에 대한 표시 기호를 그림 6-2에 나타내었고 상태천이표(state transition table)를 표 6-2에 나타내었다.

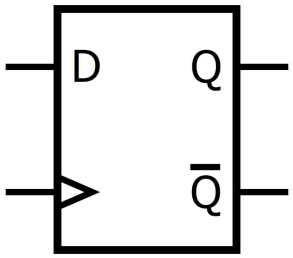


그림 6-2 Positive-edge triggered D flip-flop에 대한 표시 기호

INPUTS		OUTPUTS	
D	**CLK**	**Q**	**Q'**
0	없음	No change	
0	↑	0	1
1	없음	No change	
1	↑	1	0

표 6-2 Positive-edge triggered D flip-flop의 상태천이표

3. JK 플립플롭

JK 플립플롭의 동작은 SR 플립플롭과 유사하다. 입력 J, K는 각각 SR 플립플롭의 S, R 입력과 같은 역할을 한다. 다만 JK 플립플롭에서 J=K=1인 때에는 출력이 반전된다. 다음은 JK 플립플롭의 기호와 상태천이표이다.

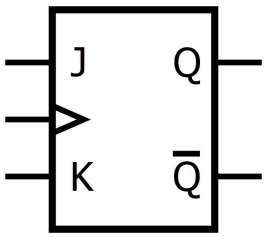


그림 6-3 Positive-edge triggered JK flip-flop에 대한 표시 기호

INPUTS			OUTPUTS	
J	K	CLK	Q	Q'
0	0	↑	No change	
0	1	↑	0	1
1	0	↑	1	0
1	1	↑	Toggle	

표 6-3 Positive-edge triggered JK flip-flop의 상태천이표

4. 플립플롭의 트리거 방식

1) Positive edge triggered: Clock signal이 Low에서 High로 변화할 때 내부 동작의 변화가 일어난다.
2) Negative edge triggered: Clock signal이 High에서 Low로 변화할 때 내부 동작의 변화가 일어난다.

5. 타이밍도(timing diagram)

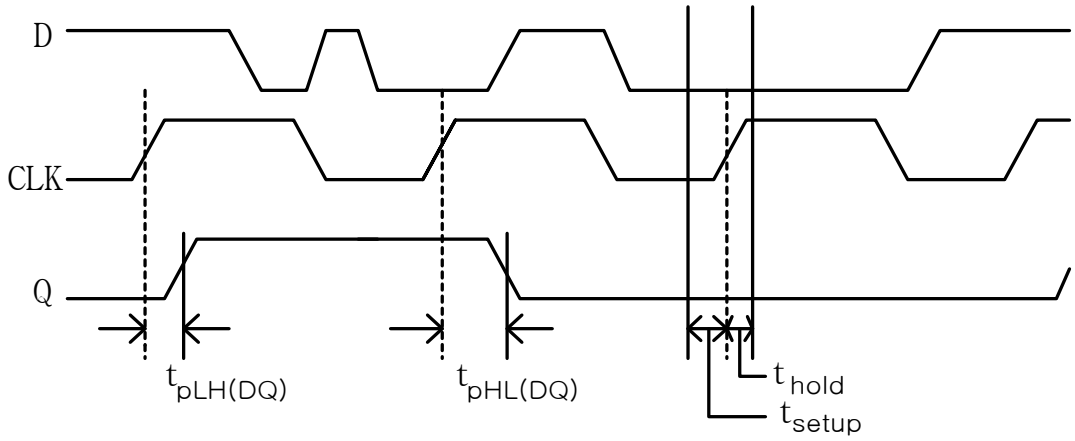


그림 6-4 Positive edge triggered D 플립플롭

1) t_{pLH}: Clock signal(CLK)이 Low→High로 변한 후 Q 출력이 Low→High로 변할 때까지 걸리는 전파 지연 시간(propagation delay time).

2) t_{pHL}: CLK signal이 Low→High로 변한 후 Q 출력이 High→Low로 변할 때까지 걸리는 전파 지연 시간.

3) t_{setup}(setup time): CLK이 Low→High로 변화되기 이전에 입력 D에 안정된 입력이 가해지고 있어야 하며, 이때의 최소한의 준비 시간을 말한다. 만약 CLK과 D 신호가 동시에 변화하면 올바른 D 입력값이 기억되지 못한다.

4) t_{hold}(hold time): CLK이 Low→High로 변화한 후에도 입력 D의 데이터 값이 얼마동안 안정되게 유지되어 있어야 하며, 이때 유지되어야 하는 최소한의 시간 간격을 말한다.

*결국 $t_{setup}+t_{hold}$ 시간 만큼 D 입력 신호가 일정하게 유지되어야만 D 입력값이 저장된다.

6. 스위치 바운싱(switch bouncing)과 SR Latch

기계적 접점을 가지는 스위치는 접점이 붙거나 떨어지는 순간에 바운싱(bouncing) 현상이 나타난다. 이것은 접점이 붙거나 떨어지는 짧은 순간에 접점이 고속으로 수차례 on/off 되는 현상을 말하며 기계적인 스위치라면 반드시 발생하는 현상이다. 바운싱은 매우 짧은 순간(수 ms 이내)에 나타나며, 이 바운싱에 의해 수차례의 엣지(edge)가 발생할 수 있다. 이러한 바운싱 현상을 제거하여야 의도한 대로 회로를 동작시킬 수 있으며, 이것을 디바운싱(debouncing)이라 한다.

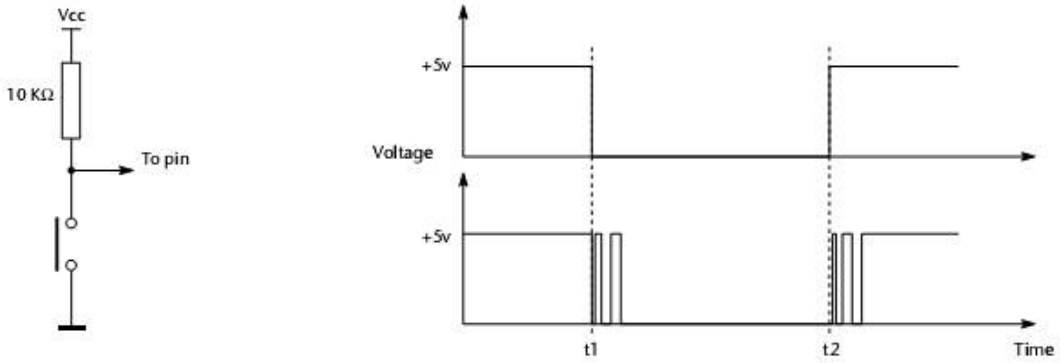


그림 6-5 일반적인 스위치 회로 및 바운싱 현상

7. 스위치를 이용한 펄스 발생 회로

SPDT 스위치와 SR latch를 이용하면 그림 6-6에 나타낸 것과 같은 펄스 발생 회로를 구현할 수 있다. 스위치의 바운싱은 SR latch에 의해 제거되고 깨끗한 펄스 파형을 얻을 수 있다.

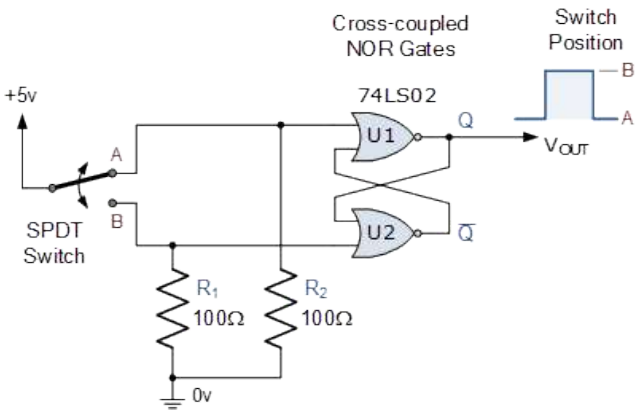


그림 6-6 SR Latch를 이용한 스위치 디바운싱(debouncing) 회로

8. VHDL Process Statement

Process statement 자체는 concurrent statement이지만 그 내부에는 *if-then-else* 혹은 *case* 등과 같은 sequential statements를 포함한다.

1) Process statement의 문법적인 구조

```
[process_name]:    process (sensitivity_list)
                   begin
                          -- statements.
                   end process;
```

(1) *process_name*: (선택사항) process statement의 이름

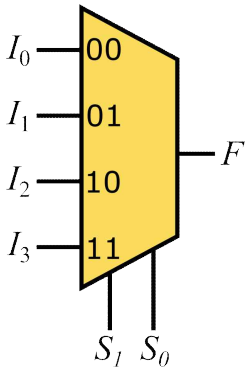
(2) *sensitivity_list*: (강제사항) 여기에 나열된 어떤 신호든지 그 값이 변하면 이 process statement가 실행됨.

(3) begin과 end 사이에 있는 statements들은 순차적으로 수행된다.

(4) Process statement 안에 있는 signals들은 그 process statement 안의 모든 statements의 실행이 끝나야만 그 값들이 변한다.

2) Process statement의 예

(1) 4-to-1 MUX using *if-elsif-else*




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

entity multiplexer_4to1 is
    port (I0,I1,I2,I3 : in std_logic;                    -- multiplexer inputs
                  S   : in std_logic_vector(1 downto 0); -- select lines
                  F   : out std_logic);                  -- multiplexer output
end multiplexer_4to1;

architecture prototype of multiplexer_4to1 is
begin
    process (I0,I1,I2,I3,S)
    begin
        if (S = "00") then
            F <= I0;
        elsif (S = "01") then     -- notice the syntax “elsif” without spaces
            F <= I1;
        elsif (S = "10") then
            F <= I2;
        else
            F <= I3;
        end if;
    end process;
end prototype;
```

(2) BCD-to-7 segment decoder using *case*

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

entity bcd_decoder is
    port(code : in std_logic_vector(3 downto 0); -- coded inputs
         leds :  out std_logic_vector(6 downto 0)); -- signal outputs
end bcd_decoder;

architecture prototype of bcd_decoder is
begin
    process (code)
        begin
            case code is
                                        --abcdefg, Common Cathode SSD
                when "0000" => leds <= “1111110" ; -- 0
                when "0001" => leds <= “0110000" ; -- 1
                when "0010" => leds <= “1101101" ; -- 2
                when "0011" => leds <= “1111001" ; -- 3
                when "0100" => leds <= “0110011" ; -- 4
                when "0101" => leds <= “1011011" ; -- 5
                when "0110" => leds <= “1011111" ; -- 6
                when "0111" => leds <= “1110000" ; -- 7
                when "1000" => leds <= "1111111" ; -- 8
                when "1001" => leds <= “1111011" ; -- 9
                when others => leds <= null ; -- the default case.
        end case;
    end process;
end prototype;
```

(3) Positive edge-triggered D flip-flop

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

entity dff is
    port (d, clk : in std_logic;      -- data and clock inputs
                q : out std_logic);   -- dff output
end dff;

architecture prototype of dff is
begin
    process (clk)
    begin
        if (clk'event and clk='1') then
            q <= d;     -- d is assigned to q on rising edge of the clock
        end if;         -- otherwise, does not change.
    end process;
end prototype;
```

(4) Positive edge-triggered D flip-flop with asynchronous set and reset signals

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

entity dffsr is
    port (  d          : in std_logic;  -- data and clock inputs
            s, r, clk  : in std_logic;  -- clock and control
             q         : out std_logic);     -- dff output
end dffsr;

architecture prototype of dffsr is
begin
    process (clk, s, r)
    begin
        if (s='1') then
            q <= '1';
        elsif (r='1') then
            q <= '0';
        elsif (clk'event and clk='1') then
            q <= d;     -- d becomes q on rising edge of clock
        end if;         -- otherwise, does not change.
    end process;
end prototype;
```

실험 순서

실험 A D 플립플롭 (74LS74)

Positive-edge triggered D flip-flop이 두 개 내장된 IC 소자인 74LS74의 기본 동작을 분석한다. 이 IC 소자에는 Data와 Clock 입력 외에도 active low의 preset(SD) 입력과 clear(RD) 입력이 있다. 실험 보고서의 표 6-5의 순서에 따라 이 소자의 동작을 분석하시오. (분석 결과에 따라 빈칸을 0 또는 1로 채우시오.)

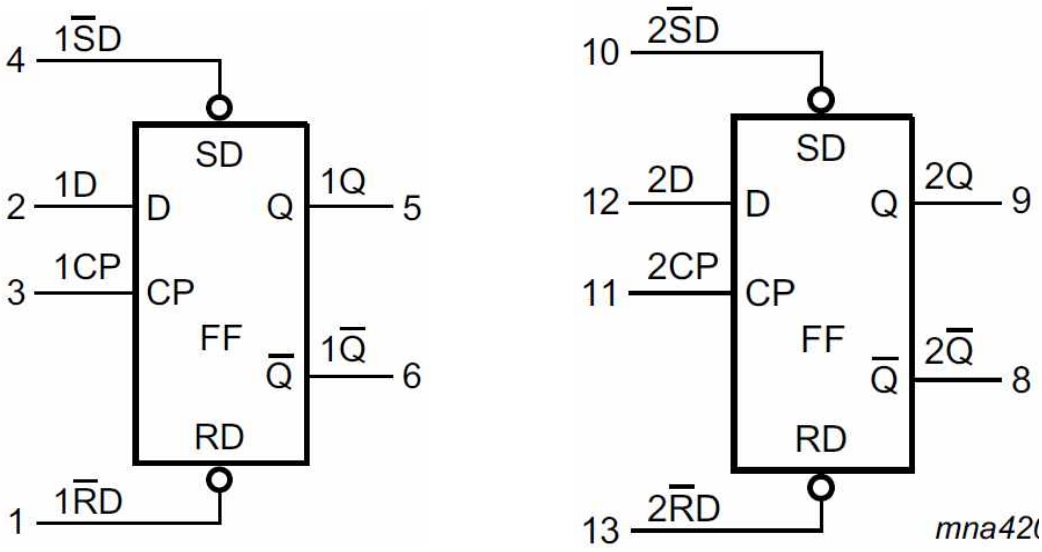


그림 6-8 74LS74

INPUTS				OUTPUTS	
$\overline{RD}$ (clear)	$\overline{SD}$ (set)	D	CP(clock)	Q	$\overline{Q}$
1	0	X	X		
0	1	X	X		
1	1	0	0, 1, ↓		
1	1	1	0, 1, ↓		
1	1	0	↑		
1	1	1	↑		

표 6-5

실험 B J-K 플립플롭 (74LS76)

Negative-edge triggered J-K flip-flop이 두 개 내장된 IC 소자인 74LS76의 기본 동작을 분석한다. 이 IC 소자에는 J, K, CP(clock pulse) 입력 외에도 active low의 SD(set) 입력과 CD(clear) 입력이 있다. 실험 보고서의 표 6-6의 순서에 따라 이 소자의 동작을 분석하시오. (분석 결과에 따라 빈칸을 0 또는 1로 채우시오.)

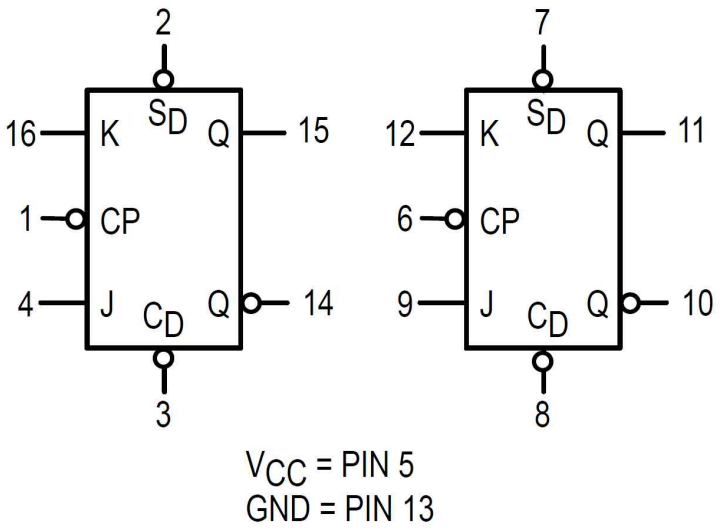


그림 6-9 Dual J-K Flip-Flops with Preset and Clear

INPUTS					OUTPUTS	
$\overline{CD}$ (clear)	$\overline{SD}$ (preset)	J	K	$\overline{CP}$ (clock)	Q	$\overline{Q}$
1	0	X	X	X		
0	1	X	X	X		
1	1	0	0	↓		
1	1	0	1	↓		
1	1	1	0	↓		
1	1	1	1	↓		

표 6-6

실험 C Three-State Buffer

Three-state buffer가 8개 내장된 74LS244의 동작을 분석한다. 그림 6-11과 같이 회로를 구성하고 실험 결과를 실험 보고서의 표 6-7의 순서에 따라 이 소자의 동작을 분석하시오. (빈칸에 0, 1, 또는 Hi-Z 가운데 하나의 값을 기록하시오.)

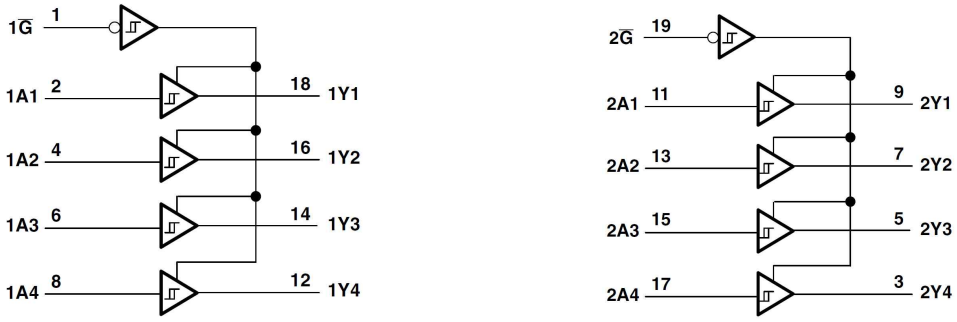


그림 6-10 74LS244 Octal Three-state Buffer IC Logic Diagram

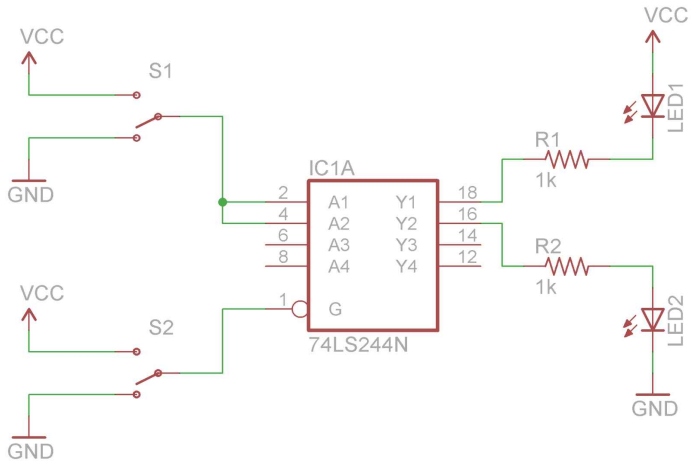


그림 6-11 Tri-state Buffer 실험 회로

INPUTS		OUTPUT	
S2 ($\overline{G}$)	S1 (A1, A2)	Y1	Y2
0	0		
0	1		
1	0		
1	1		

표 6-7

실험 D FPGA/VHDL을 이용한 D Flip-flop, JK Flip-flop 및 Three-State Buffer의 구현

1. D F/F의 구현

1) FPGA/VHDL을 이용하여 74LS74에 내장된 2개의 D F/F 가운데 하나의 D F/F의 기능을 구현하시오.

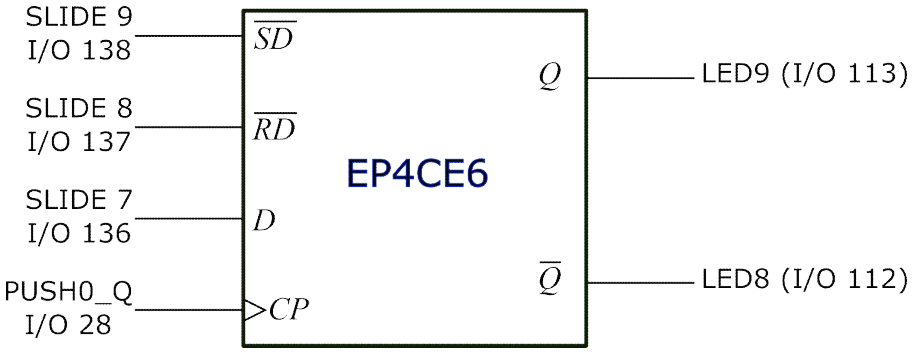


그림 6-12 1/2 74LS74를 VHDL/FPGA로 구현

2) Simulation을 통해 그 동작을 확인하시오. (그림 6-13 참조)

(1) 적절한 시점에서 $\overline{RD}$와 $\overline{SD}$ 신호를 논리값 '0'으로 만들어 Q 출력이 변하는지 확인할 수 있도록 simulation 파형을 작성한다.

(2) 적절한 시점에서 D와 CP 신호를 변화시켜 설계한 D F/F이 제대로 동작하는지 확인할 수 있도록 simulation 파형을 작성한다.

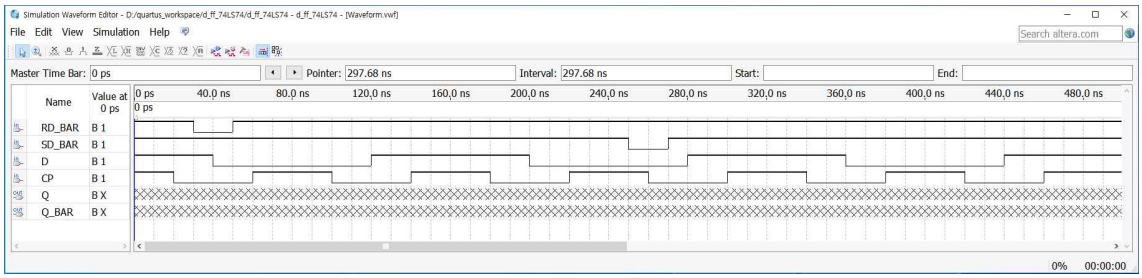


그림 6-13 1/2 74LS74의 Simulation Waveform 예

3) FPGA 소자에 프로그램하여 그 동작을 확인하시오. (그림 6-12 및 표 6-8 참조)

2. JK F/F의 구현

1) FPGA/VHDL을 이용하여 74LS76에 내장된 2개의 JK F/F 가운데 하나의 JK F/F의 기능을 구현하시오.

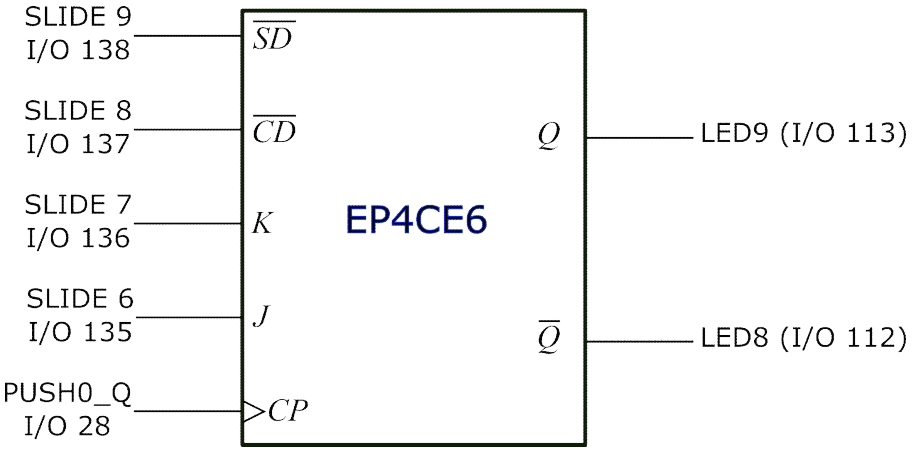


그림 6-14 1/2 74LS76을 VHDL/FPGA로 구현

2) Simulation을 통해 그 동작을 확인하시오. (그림 6-15 참조)

(1) 적절한 시점에서 $\overline{CD}$와 $\overline{SD}$ 신호를 논리값 '0'으로 만들어 Q 출력이 변하는지 확인할 수 있도록 simulation 파형을 작성한다.

(2) 적절한 시점에서 J와 K 및 CP 신호를 변화시켜 설계한 JK F/F이 제대로 동작하는지 확인할 수 있도록 simulation 파형을 작성한다.

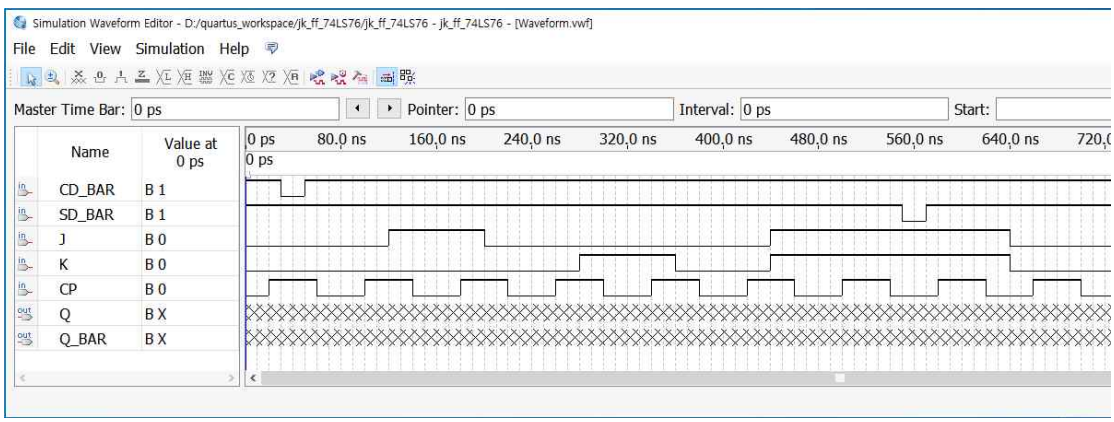


그림 6-15 1/2 74LS76의 Simulation Waveform 예

3) FPGA 소자에 프로그램하여 그 동작을 확인하시오. (그림 6-14 및 표 6-9)

3. Three-State Buffer의 구현

1) FPGA/VHDL을 사용하여 74LS244에 내장된 8개의 three-state buffer 가운데 하나의 three-state buffer의 기능을 구현하시오.

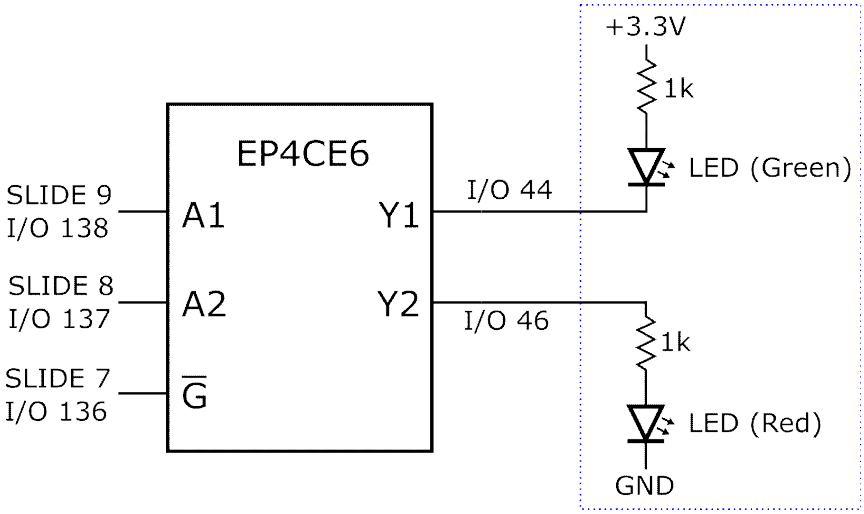


그림 6-16 1/8 74LS244를 VHDL/FPGA로 구현

2) Simulation을 통해 그 동작을 확인하시오. (그림 6-17 참조)

적절한 시점에서 A1, A2 및 $\overline{G}$ 신호를 변화시켜 Y1 및 Y2 출력이 제대로 변하는지 확인할 수 있도록 simulation 파형을 작성한다.

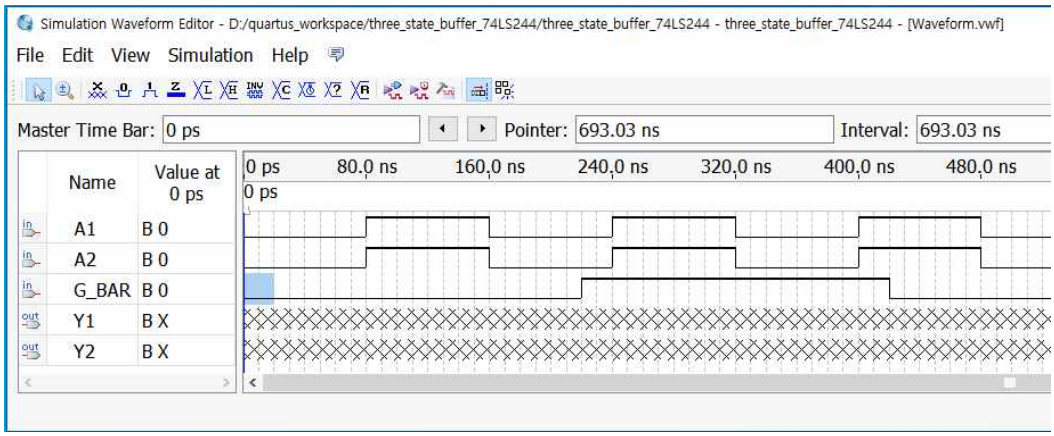


그림 6-17 1/8 74LS244의 Simulation Waveform 예

3) FPGA 소자에 프로그램하여 그 동작을 확인하시오. (그림 6-16 및 표 6-10)

실험 보고서

실험 결과

실습 A D 플립플롭 분석

(1) 74LS74 D 플립플롭의 기본 동작 분석 결과에 따라 빈칸을 0 또는 1로 채우시오.

INPUTS				OUTPUTS	
$\overline{RD}$ (clear)	$\overline{SD}$ (set)	D	CP(clock)	Q	$\overline{Q}$
1	0	X	X		
0	1	X	X		
1	1	0	0		
1	1	1	0		
1	1	0	↑		
1	1	1	↑		

표 6-5

(2) 74LS74의 clear와 set 입력은 동기식인가 비동기식인가?

실습 B J-K 플립플롭 분석

(1) 7LS76A J-K 플립플롭의 기본 동작 분석 결과에 따라 빈칸을 0 또는 1로 채우시오.

INPUTS					OUTPUTS	
$\overline{CD}$ (clear)	$\overline{SD}$ (preset)	J	K	$\overline{CP}$ (clock)	Q	$\overline{Q}$
1	0	X	X	X		
0	1	X	X	X		
1	1	0	0	↓		
1	1	0	1	↓		
1	1	1	0	↓		
1	1	1	1	↓		

표 6-6

(2) 74LS76A의 clear와 preset 입력은 동기식인가 비동기식인가?

실습 C Three-state buffer 분석

74LS244 Three-state buffer의 동작 분석 결과에 따라 빈칸에 0, 1, 또는 Hi-Z 가운데 하나의 값을 기록하시오.

INPUTS		OUTPUT	
S2 ($\overline{G}$)	S1 (A1, A2)	Y1	Y2
0	0		
0	1		
1	0		
1	1		

표 6-7

실습 D FPGA/VHDL를 이용하여 구현한 1/2 74LS74 D F/F의 기능 확인.

1. 1/2 74LS74 D F/F을 VHDL로 구현하시오.

2. 그림 6-13을 참조하여 적절한 Simulation 입력 파형을 작성하고, Functional Simulation을 수행하시오.

3. 위 2의 Functional Simulation을 결과에 대해 입력 신호의 레벨이 변하는 모든 시점에서의 출력값에 대한 자신의 설명을 추가하시오.

4. 위에서 설계한 내용을 FPGA에 프로그램한 후, 아래의 표 6-8에 따라 그 기능을 시험하여 빈칸에 0 또는 1을 쓰시오.

INPUTS				OUTPUTS	
$\overline{RD}$ (clear)	$\overline{SD}$ (set)	D	CP(clock)	Q	$\overline{Q}$
1	0	X	X		
0	1	X	X		
1	1	0	0		
1	1	1	0		
1	1	0	↑		
1	1	1	↑		

표 6-8

실습 E FPGA/VHDL를 이용하여 구현한 1/2 74LS76A JK F/F의 기능 확인

1. 1/2 74LS76A J-K F/F을 을 VHDL로 구현하시오.

2. 그림 6-15를 참조하여 적절한 Simulation 입력 파형을 작성하고, Functional Simulation을 수행하시오.

3. 위 2의 Functional Simulation을 결과에 대해 입력 신호의 레벨이 변하는 모든 시점에서의 출력값에 대한 자신의 설명을 추가하시오.

4. 위에서 설계한 내용을 FPGA에 프로그램한 후, 아래의 표에 따라 그 기능을 시험하여 빈칸에 0 또는 1을 쓰시오.

INPUTS					OUTPUTS	
$\overline{CD}$ (clear)	$\overline{SD}$ (preset)	J	K	$\overline{CP}$ (clock)	Q	$\overline{Q}$
1	0	X	X	X		
0	1	X	X	X		
1	1	0	0	↓		
1	1	0	1	↓		
1	1	1	0	↓		
1	1	1	1	↓		

표 6-9

실습 F FPGA/VHDL를 이용하여 구현한 Three-State Buffer의 기능 확인

1. 1/8 74LS244 three-state buffer를 VHDL로 구현하시오.

2. 그림 6-17을 참조하여 적절한 Simulation 입력 파형을 작성하고, Functional Simulation을 수행하시오.

3. 위 2의 Functional Simulation을 결과에 대해 입력 신호의 레벨이 변하는 모든 시점에서의 출력값에 대한 자신의 설명을 추가하시오.

4. 위에서 설계한 내용을 FPGA에 프로그램한 후, 아래의 표 6-10에 따라 그 기능을 시험하여 빈칸에 0, 1 또는 Hi-Z로 채우시오.

INPUTS		OUTPUT	
$\overline{G}$	A1, A2	Y1	Y2
0	0		
0	1		
1	0		
1	1		

표 6-10

결론

본 실습을 통해 얻은 내용을 기록하시오.