4 장

멀티플렉서/디멀티플렉서

1. 실험 목표

▣ 멀티플렉서 (Multiplexer)와 디멀티플렉서 (Demultiplexer)의 구조와 기본 동작 원리를 이해한다.

▣ 멀티플렉서와 디멀티플렉서의 혼합 구성법을 이해하고 이를 응용한다.

▣ 조합회로 설계에 활용할 수 있는 프로토타입 설계 기법을 이해한다.

2. 실험 이론

가) 멀티플렉서 (Multiplexer: MUX)

멀티플렉서는 여러 소스로부터 입력되는 데이터 중 한 개의 입력을 선택하여 하나의 출력단을 통해 한 목적지로 연결하는 소자이다. 이를 위하여 멀티플렉서는 여러 개의 데이터 입력선과 하나의 출력선, 그리고 입력 데이터 중에서 어떤 데이터를 출력할 것인지를 선택하는 데이터 선택 입력을 가지고 있다. 멀티플렉서는 입력 데이터들 중에서 하나를 선택하여 출력하므로 데이터 선택기 (Selector)라고도 한다.

[그림 4-1]은 4:1 멀티플렉서를 나타낸다. I0, I1, I2, I3 까지의 4 개의 입력선은 각각 AND 게이트의 한 입력이 되고 선택선 S0 와 S1 은 특정한 AND 게이트를 선택하는데 사용되는데, 이 선택선의 비트 조합에 따라 출력이 결정된다.

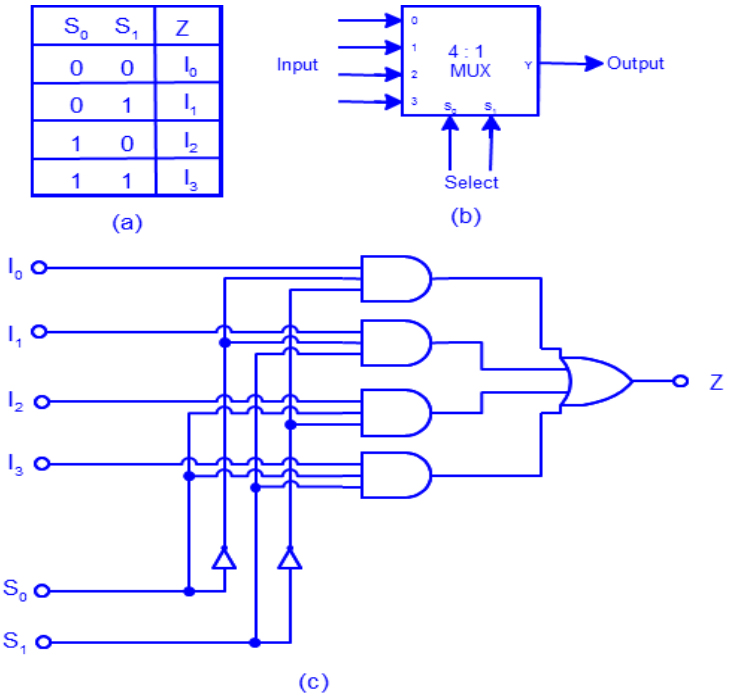


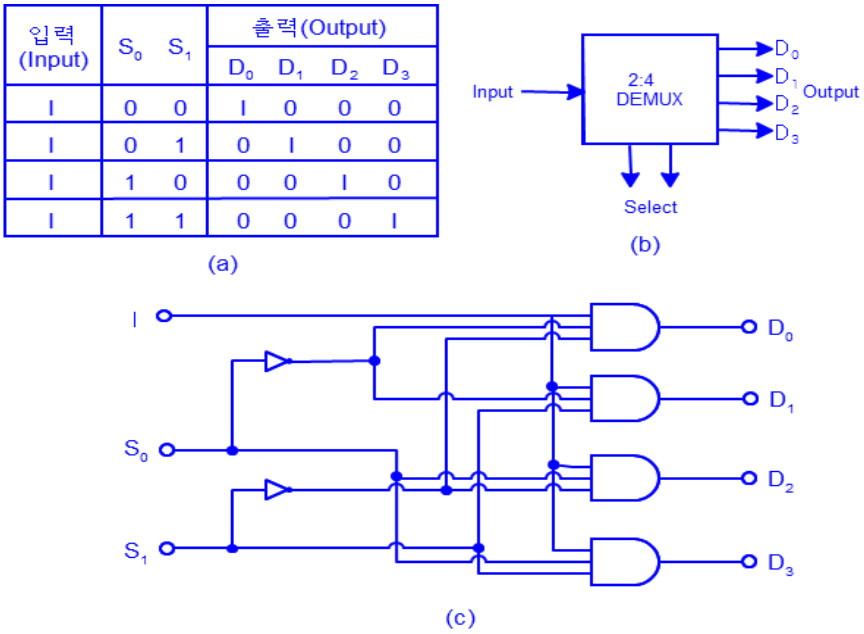
그림 4-1. 4:1 멀티플렉서의 (a) 진리표, (b) 블록 다이어그램, (c) 회로도.

이 멀티플렉서는 4 개의 입력을 제공하므로 입력 선택선은 2 개가 되어야 그 비트 조합으로 출력을 결정할 수 있다. 즉 선택선을 각각의 AND 게이트의 입력으로 연결하여서 선택선의 비트 조합으로 각각의 입력 AND 게이트를 선택하는 것이다. 만약 S1=0, S0=0 이면 입력 I0 에 연결된 AND 게이트의 값을 보면 S 의 두 입력은 '1'이 되어 I0 가 출력에 연결된다. 따라서 I0 의 값이 0 이면 출력에는 0 이, I0 의 값이 1 이면 출력은 1 이 된다. 나머지 3 개의 AND 게이트는

하나의 입력이 '0'이 되므로 출력은 모두 '0'이 된다. 따라서 OR 게이트의 출력은 I0 와 같게 된다. 즉, OR 게이트는 하나를 선택하여 입력선이 이진 정보를 출력선에 전달하여 주기 때문에 데이터 선택기 (Selector)라고도 부르는 것이다.

나) 디멀티플렉서 (Demultiplexer: DEMUX)

디멀티플렉서 멀티플렉서의 동작과 반대 기능을 수행한다. 디멀티플렉서는 한 개의 입력선으로 들어온 데이터를 여러 개의 출력선 중에 한 개를 선택하여 제공하여, 입력 데이터를 다수의 출력으로 분배 가능하다. 이러한 이유로 디멀티플렉서를 데이터 분배기라고도 한다.



입력 (Input)	S_0	S_1	출력(Output)			
			D_0	D_1	D_2	D_3
I	0	0	I	0	0	0
I	0	1	0	I	0	0
I	1	0	0	0	I	0
I	1	1	0	0	0	I

(a)

(b)

(c)

그림 4-2. 2:4 디멀티플렉서의 (a) 진리표, (b) 블록 다이어그램, (c) 회로도.

그림 4-2 는 2:4 디멀티플렉서의 구조를 나타내며, 여기서 2 는 2 개의 선택 신호이고 4 는 출력의 수를 나타낸다. 즉 출력선이 4 개이므로 선택선은 2 개가 필요하게 된다. 디멀티플렉서도 위의 멀티플렉서와 같이 입력을 4 개의 출력선 중 하나에 연결하여 출력하는 구조이므로 AND 게이트를 이용하여 구현할 수 있다. S1=0, S0=1 이면 4 개의 AND 게이트 중 하나의 게이트(두 번째 게이트)만이 구동되므로 입력신호 I 는 출력선 D1 에 제공된다.

```
module  MUX_4_TO_1  ( I0, I1, I2, I3, Z, S);
            input                   I0, I1, I2, I3;
            input       [1:0]       S;
            output                  Z;
            reg                     Z;

            always  @ ( I0 or I1 or I2 or I3 or S) begin
                        case (S)
                                    2'b00: Z = I0;
                                    2'b01: Z = I1;
                                    2'b10: Z = I2;
                                    default: Z = I0;
                        endcase
            end
endmodule


module  DEMUX_1_TO_4  ( I, S, D0, D1, D2, D3);
            input                   I;
            input       [1:0]       S;
            output                  D0, D1, D2, D3;
            reg                     D0, D1, D2, D3;

            always  @ ( I or S ) begin
                        case (S)
                                    2'b00: begin
                                                D0 = I;
                                                D1 = 1'b0;
                                                D2 = 1'b0;
                                                D3 = 1'b0;
                                    end
                                    2'b01: begin
                                                D0 = 1'b0;
                                                D1 = I;
                                                D2 = 1'b0;
                                                D3 = 1'b0;
                                    end
                                    2'b10: begin
                                                D0 = 1'b0;
                                                D1 = 1'b0;
                                                D2 = I;
                                                D3 = 1'b0;
                                    end
                                    default: begin
                                                D0 = 1'b0;
                                                D1 = 1'b0;
                                                D2 = 1'b0;
                                                D3 = I;
                                    end
                        endcase
            end
endmodule
```

그림 4-3. 4:1 MUX 와 2:4 DEMUX 의 Verilog HDL 코드.

3. 예비보고서 작성

▣ 2:1, 4:1 멀티플렉서와 1:2, 2:4 디멀티플렉서의 구조를 NAND 게이트로 구성하시오.

▣ 멀티플렉서와 디멀티플렉서의 차이점과 특징에 대해 설명하시오.

▣ 멀티플렉서와 디멀티플렉서의 응용분야에 대해 조사하시오.

4. 실험에 필요한 기기

▣ 실험용 PC

▣ MAX PLUS II 소프트웨어 / Quartus II-Modelsim 소프트웨어

5. 실험 내용과 과정

① Altera Modelsim 을 실행하여 새로운 Project 를 생성하고, 실험에 사용할 module 및 testbench 에 대한 Verilog code 를 작성한다.
② Compile > Compile All 을 통해 test module 및 testbench 에 대한 컴파일을 마친다.
③ Simulate > Start Simulation 을 통해 시뮬레이션 다이얼로그를 띄운 후, testbench 에 대한 시뮬레이션을 시작한다.
④ 실험에서 사용되는 port 들을 클릭하여 Add Wave 를 통해 추가해주거나, 하단부의 command line 에서 'add wave *'을 통하여 등록한다.
⑤ Simulate > Run > Run all 혹은, 하단부의 command line 에서 'run –all'을 통해 시뮬레이션을 진행한다.
⑥ Waveform 화면을 통해 실험의 결과를 확인하고, command line 에서 'quit –sim'을 입력해주거나, Simulate > End Simulation 메뉴의 선택을 통해서 실험을 종료한다.

6. 실험 결과 보고서

① 2:1 멀티플렉서, 4:1 멀티플렉서를 이용하여 8×1 멀티플렉서를 구현하고 동작을 확인하시오.

② 1:2 디멀티플렉서, 2:4 디멀티플렉서를 이용하여 3:8 디멀티플렉서를 구현하고 동작을 확인하시오.

③ ①에서 구현한 8x1 멀티플렉서를 활용하여, 2-input / 2-bit 가산기를 구현하고 동작을 확인하시오. (힌트: 3 개의 8x1 멀티플렉서를 활용하시오.)

④ ①, ②에서 구현한 모듈이 Gate Level 혹은 Data Flow 모델에 기반으로 작성된 것이라면, Behavioral Level 로 구현하시오. 그 반대라면, Gate Level 혹은 Data Flow 모델의 형식으로 구현해보시오.