9주차 결과보고서

전공: 컴퓨터공학과 학년: 3학년 학번: 20212022 이름: 이예준

**1.**

텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

\*2 to 4 Decoder Simulation 결과 (Active High)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Input | | Output | | | |
| a | b | x | y | z | w |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 |

\*2 to 4 Decoder 진리표 (Active High)

위의 Decoder의 Simulation 결과를 보게 되면 Input a와 b가 2진수의 역할을 하고,

Output x, y, z, w가 10진수의 역할을 한다. 따라서 이 Decoder는 2진수를 10진수로

변환하는 Decoder라고 해석할 수 있다.

위의 진리표를 이용해 아래와 같은 카르노맵을 만들 수 있다.

|  |  |  |
| --- | --- | --- |
| A B | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |

Output x의 카르노맵

|  |  |  |
| --- | --- | --- |
| A B | 0 | 1 |
| 0 | 0 | 1 |
| 1 | 0 | 0 |

Output y의 카르노맵

|  |  |  |
| --- | --- | --- |
| A B | 0 | 1 |
| 0 | 0 | 0 |
| 1 | 1 | 0 |

Output z의 카르노맵

|  |  |  |
| --- | --- | --- |
| A B | 0 | 1 |
| 0 | 0 | 0 |
| 1 | 0 | 1 |

Output w의 카르노맵

카르노맵을 보게 되면 모든 Output들은 Input들이 AND gate로 조합되어 이루어져 있는

것을 볼 수 있다. 이 경우와 같이 출력 1일 때에 Output이 유효한 것을 Active High라고 한다.

스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명

\*2 to 4 Decoder Simulation 결과 (Active Low)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Input | | Output | | | |
| a | b | x | y | z | w |
| 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 |

\*2 to 4 Decoder 진리표 (Active Low)

위의 Decoder의 Simulation 결과를 보게 되면 첫번째 Decoder와 마찬가지로

이 Decoder는 2진수를 10진수로 변환하는 Decoder라고 할 수 있다.

위의 진리표를 이용해 아래와 같은 카르노맵을 만들 수 있다.

|  |  |  |
| --- | --- | --- |
| a b | 0 | 1 |
| 0 | 0 | 1 |
| 1 | 1 | 1 |

Output x의 카르노맵

|  |  |  |
| --- | --- | --- |
| a b | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 1 | 1 |

Output y의 카르노맵

|  |  |  |
| --- | --- | --- |
| a b | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |

Output z의 카르노맵

|  |  |  |
| --- | --- | --- |
| a b | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 0 |

Output w의 카르노맵

카르노맵을 보게 되면 모든 Output들은 Input들이 OR gate로 조합되어 이루어져 있는

것을 볼 수 있으며, 이 조합을 NAND gate로 변환할 수도 있다. 따라서 Active High에서 AND gate를 그냥 NAND gate로 바꿔주면 Active Low를 구현할 수 있다.

이 경우와 같이 출력 0일 때에 Output이 유효한 것을 Active Low라고 한다.

**2.**

스크린샷, 텍스트, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

\*4 to 2 Encoder Simulation 결과

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Input | | | | Output | |
| x | y | z | w | a | b |
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 |

\*4 to 2 Encoder 진리표

위의 Encoder의 Simulation 결과를 보게 되면 Input x, y, z, w가 10진수의 역할을 하고,

Output a와 b가 2진수의 역할을 한다. 따라서 이 Encoder는 10진수를 2진수로

변환하는 Encoder라고 해석할 수 있다.

위의 진리표를 이용해 아래와 같은 카르노맵을 만들 수 있다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| xy zw | 00 | 01 | 11 | 10 |
| 00 | X | 0 | X | 0 |
| 01 | 1 | X | X | X |
| 11 | X | X | X | X |
| 10 | 1 | X | X | X |

Output a의 카르노맵

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| xy zw | 00 | 01 | 11 | 10 |
| 00 | X | 0 | X | 1 |
| 01 | 0 | X | X | X |
| 11 | X | X | X | X |
| 10 | 1 | X | X | X |

Output b의 카르노맵

Encoder의 진리표를 보게 되면 Decoder의 Input과 Output이 뒤바뀐 형태인 것을

볼 수 있다. 또한 하나의 Input에 대해 오직 하나의 1만 존재하는 경우만 상정해 만든

Encoder이므로 2개의 이상의 1이나 모두 0인 경우를 고려하지 않고 모두 Don’t care로

둔다. 따라서 위의 카르노맵을 그릴 때에서도 하나의 1만 존재하는 경우에 대해서만

논리식을 계산했다.

**3.**

위의 Encoder는 각 Input signal마다 하나만 1일 때의 4가지 경우일 때에만 Output을

출력하게 구현되어 있다. 따라서 모든 Input signal이 0이거나 2개 이상의 1이 있는

Input일 때의 경우들은 없다고 가정하고 그런 상황에 대해서는 고려하지 않고 구현한다. 이런 경우들을 고려하는 Encoder를 Priority Encoder라고 한다.

**4.**

오직 하나의 signal만 1일 때 출력을 낼 수 있는 한계를 극복하기 위해 만든 Encoder를

Priority Encoder라고 한다. 물론 모든 경우의 수에 대한 서로 다른 Output을 내는 것은 어렵기 때문에 각 Input signal에 대해 우선 순위를 정해 여러 signal이 1이 되어도 우선 순위가 높은 signal에 대한 Output을 낸다. 또한 모든 signal이 0인 경우도 대응하기 위해 새로운 Output V를 하나 만든다.

아래의 진리표의 Input signal의 우선순위는 x>y>z>w 순이다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Input | | | | Output | | |
| x | y | z | w | a | b | V |
| 0 | 0 | 0 | 0 | X | X | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | X | 0 | 1 | 1 |
| 0 | 1 | X | X | 1 | 0 | 1 |
| 1 | X | X | X | 1 | 1 | 1 |

\*Priority Encoder 진리표

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| xy zw | 00 | 01 | 11 | 10 |
| 00 | X | 0 | 0 | 0 |
| 01 | 1 | 1 | 1 | 1 |
| 11 | 1 | 1 | 1 | 1 |
| 10 | 1 | 1 | 1 | 1 |

Output a의 카르노맵

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| xy zw | 00 | 01 | 11 | 10 |
| 00 | X | 0 | 1 | 1 |
| 01 | 0 | 0 | 0 | 0 |
| 11 | 1 | 1 | 1 | 1 |
| 10 | 1 | 1 | 1 | 1 |

Output b의 카르노맵

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| xy zw | 00 | 01 | 11 | 10 |
| 00 | 0 | 1 | 1 | 1 |
| 01 | 1 | 1 | 1 | 1 |
| 11 | 1 | 1 | 1 | 1 |
| 10 | 1 | 1 | 1 | 1 |

Output b의 카르노맵

따라서 위의 카르노맵을 이용해 회로도를 그리면 아래와 같다.

도표, 라인, 기술 도면, 폰트이(가) 표시된 사진

자동 생성된 설명

\*Priority Encoder 회로도

**5.**

스크린샷, 디스플레이, 텍스트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

\*BCD to Decimal Decoder Simulation 결과

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | | | | **Output** | | | | | | | | |
| w | z | y | x | i | h | g | f | e | d | c | b | a |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

위의 Decoder는 8421 BCD 코드를 10진수로 출력하는 Decoder이다. Output이 0~9까지만

할당되어 있기 때문에 9보다 큰 수가 나오는 경우는 고려하지 않는다. 모든 Output은

하나의 Input 패턴에만 대응하며, 단 Input패턴이 0일 때는 어떤 Output에도 대응하지

않는다. 위의 진리표를 이용해 아래와 같은 카르노맵을 만들 수 있다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| wz yx | 00 | 01 | 11 | 10 |
| 00 | 0 | 1 | 0 | 0 |
| 01 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 |

Output a의 카르노맵

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| wz yx | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 1 |
| 01 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 |

Output b의 카르노맵

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| wz yx | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 1 | 0 |
| 01 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 |

Output c의 카르노맵

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| wz yx | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 1 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 |

Output d의 카르노맵

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| wz yx | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 1 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 |

Output e의 카르노맵

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| wz yx | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 0 | 0 | 1 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 |

Output f의 카르노맵

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| wz yx | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 0 | 1 | 0 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 |

Output g의 카르노맵

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| wz yx | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 1 | 0 | 0 | 0 |

Output h의 카르노맵

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| wz yx | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 0 | 1 | 0 | 0 |

Output i의 카르노맵

**6.**

Encoder와 Decoder는 모두 데이터를 한 형식에서 다른 형식으로 변환하는데 사용되는 장치이다. Encoder는 데이터를 암호화하는데 사용되며, Decoder는 Encoder로 암호화된 데이터를 해독하는 역할을 한다. 일반적으로 두 장치는 네트워크와 같은 통신 시스템에서 자주 사용된다. 예를 들면 문자를 변환하는 방법이 있는데 사용자가 입력한 문자나 기호를 컴퓨터가 사용할 수 있는 형태로 바꾸어 주는 인코딩과 그 정보를 다시 사람이 이해할 수 있는 형태로 바꿔주는 디코딩이 있다. 디코딩을 할 때 정해진 기준을 바탕으로 하는데, 그런 기준을 문자열세트 또는 문자셋이라고 한다.

**7.**

텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

\*4 to 1 line Mux Simulation 결과

텍스트, 스크린샷, 소프트웨어, 웹 페이지이(가) 표시된 사진

자동 생성된 설명

\*4 to 1 line Mux Verilog 코드

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Select** | | **Input** | | | | **Output** |
| x | y | a | b | c | d | t |
| 0 | 0 | 0 | X | X | X | 0 |
| 0 | 0 | 1 | X | X | X | 1 |
| 0 | 1 | X | 0 | X | X | 0 |
| 0 | 1 | X | 1 | X | X | 1 |
| 1 | 0 | X | X | 0 | X | 0 |
| 1 | 0 | X | X | 1 | X | 1 |
| 1 | 1 | X | X | X | 0 | 0 |
| 1 | 1 | X | X | X | 1 | 1 |

\*4 to 1 line Mux 진리표

선택 line x와 y값에 따라 Input중 하나를 선택해 Output으로 연결해준다.

Input은 총 4개가 있으며(a, b, c, d) Input은 오직 하나만 선택된다.

x와 y가 0,0일 때는 a값이 선택되어 출력되고, 0,1일 때는 b값이 선택되어 출력되고,

1,0일 때는 c값이 선택되어 출력되고 마지막으로 1,1일 때는 d값이 선택되어 출력된다.

선택되는 Input signal외에는 고려하지 않기 때문에 Don’t care로 표시해준다.

**8.**

스크린샷, 디스플레이, 텍스트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

\*1 to 4 line Demux Simulation 결과

텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

\*1 to 4 line Demux Verilog 코드

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Select** | | **Data** | **Output** | | | |
| x | y | t | a | b | c | d |
| X | X | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 |

\*1 to 4 line Demux 진리표

선택 line x와 y값에 따라 데이터 t를 특정 Output로 출력한다. 데이터 t가 0일 때는

선택 line에 관계없이 어느 Output에도 연결되지 않는다. x와 y가 0,0일 때는 a가 1로

출력되고, 0,1일 때는 b가 1로 출력되고, 1,0일 때는 c가 1로 출력되고 마지막으로

1,1일 때는 d가 1로 출력된다. 이때 데이터 t를 Enable Signal로 보고, 선택 line x, y를 Input으로 본다면 2 to 4 Decoder와 같은 기능을 한다고 볼 수 있다. 따라서 2 to 4 Decoder들을 조합해 4 to 16 Decoder를 만들 수 있듯이, 1 to 4 line Demux을 이용해 4 to 16 Decoder를 만들 수 있다.

도표, 텍스트, 평면도, 기술 도면이(가) 표시된 사진

자동 생성된 설명

\*1 to 4 line Demux를 이용한 4 to 16 Decoder 회로도

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | | | | **Output** | | | | | | | | | | | | | | | |
| X | Y | Z | W | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 | D10 | D11 | D12 | D13 | D14 | D15 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

\*4 to 16 Decoder 진리표

`timescale 1ns / 1ps

module Decoder(input X,Y,Z,W,

output D\_0, D\_1, D\_2, D\_3, D\_4, D\_5, D\_6, D\_7, D\_8, D\_9, D\_10, D\_11, D\_12,

D\_13, D\_14, D\_15

);

wire enable0, enable1, enable2, enable3;

Demux DM0(1, X, Y, enable0, enable1, enable2, enable);

Demux DM1(enable0, Z, W, D\_0, D\_1, D\_2, D\_3);

Demux DM2(enable1, Z, W, D\_4, D\_5, D\_6, D\_7);

Demux DM3(enable2, Z, W, D\_8, D\_9, D\_10, D\_11);

Demux DM4(enable3, Z, W, D\_12, D\_13, D\_14, D\_15);

endmodule

\*4 to 16 Decoder Verilog 코드

코드를 보게 되면 Demux는 위에서 설명한 1 to 4 line Demux와 같은 코드이다.

DM0는 맨 앞에 있는 Demux으로 이 장치의 Output에 의해 2 level의 어떤 Demux의 Output을 출력할지 결정한다. DM1부터 DM4는 2 level에 있는 Demux로 괄호 안에 있는 변수는 순서에 맞게 배정한다. wire는 enable값들을 저장하기 위한 변수이다.

**9.**

이번 실험에서 Encoder와 Decoder, Mux와 Demux의 작동방식에 대해 Simulation결과와

FPGA 보드 구현 결과를 통해 확인할 수 있었다. 이번실험에서 구현한 Decoder는 2진수를

10진수로 변환하고, Encoder는 10지수를 2진수로 변환했다. 또한 Mux는 선택line에 따라

Input중 하나를 선택해 Output을 출력했고, 반대로 Demux는 선택line에 따라 Data를

여러 Output중 하나를 선택해 연결해주었다. 위에서 Decoder와 Encoder의 응용에 대해서

설명했는데 Mux와 Demux에 대해서는 설명하지 않았다. 따라서 이번 실험의 논의사항은

Mux와 Demux의 응용에 대한 것이다. Mux는 입력되는 여러 신호 중 하나를 선택해서

출력하는 회로이다. 예를 들어 메인 컴퓨터에 여러 단말기가 연결되어 있을 때, 서로 다른

통신회선으로 연결하면 비용이 높고 관리가 어렵기 때문에 각각의 단말기를 묶어

하나의 통신회선에 연결하고 Mux를 통해 단말기를 선택하는 방법이 더 효율적이다.

Demux는 반대로 하나의 입력신호를 선택에 따라 여러 Output을 제공하는 회로이다.

Mux와 반대로 Demux는 메인 컴퓨터에서 하나의 통신회선을 통해 여러 개의 단말기중

하나를 선택해 데이터를 보낼 때, Demux를 이용해 전송한다.

**10.**

Mux는 디지털 논리회로 뿐만 아니라 다양한 분야에서도 사용된다. 그 중 하나로 통신분야가

있는데, 하나의 회선을 분할하여 개별적으로 독립된 신호를 동시에 송수신할 수 있는 채널을

구성하는 기술이 그 예이다. 대표적인 방식으로는 시간간격을 분할하여 다중화하는

시분할 다중화 방식(TDM) 그리고 주파수를 분할하여 다중화하는 주파수 분할 다중화

방식(FDM)이 있다.