실험물리학 2

7주차 예비 레포트

<디지털 논리회로의 응용-가산기, 비교기, 멀티플렉서, 디멀티플렉서>

이름: 김나현

학번: 20191286

분반: 2분반

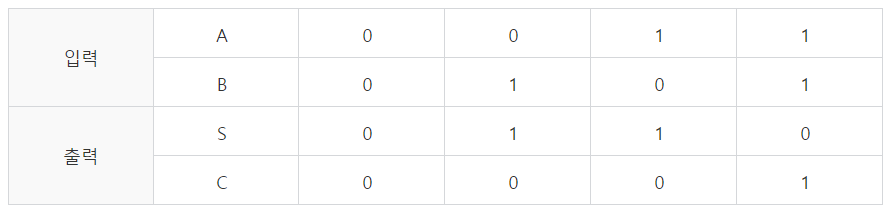
담당 교수님: 정명화 교수님

담당 조교님: 소현경 조교님

제출일자: 2020년 11월 4일 수요일

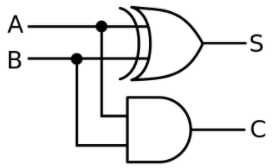
1. 실험 목표
2. 반가산기와 전가산기의 원리를 이해한다.
3. 비교기의 원리를 이해하고 이를 응용한 회로를 구성할 수 있다.
4. 멀티플렉서와 디멀티플렉서의 원리를 이해한다.
5. 실험 이론
6. 가산기

가산기란 여러 비트로 된 두 수를 더하는 덧셈 연산을 수행하는 논리게이트로, 게이트에 의해 출력되는 불 대수의 값이 해당 시점에서의 회로 상태에 무관하게 입력 값에 의해서만 정해지는 논리 회로인 조합 논리 회로로 연산되고, memory를 갖지 않는다는 특징이 있다. 이때, 조합 논리 회로에 대해 자세히 설명해보면 입력 값과 내부 상태에 의해 출력이 정해지는 순차 논리 회로와 달리 시간에 흐름과 무관하여 항상 규정된 입력 변수의 조합에 대해서는 같은 규정된 출력이 나오도록 논리 게이트를 연결한 것이다.

이러한 가산기에는 반가산기(half adder)와 전가산기(full adder)로 구분할 수 있고, 반가산기는 1비트 2진수로 나타낸 수를 합하여 그 결과로 1비트의 합과 1비트의 자리올림(carry)를 발생시키는 회로이고 전가산기는 자릿수가 많은 2진수의 덧셈에서 어떤 자리의 덧셈을 할 때 낮은 자리부터 올림수를 고려한 2진 1자리 가산기이다.

<그림 a> 반가산기 진리표의 예시

위의 <그림 a>는 가장 간단한 반가산기의 진리표를 나타낸 것으로, 0과 0을 입력받으면 두 수를 더해도 0이 되고 carry 또한 0이지만 1과 1을 입력받으면 두 수를 더한 결과는 0과 carry는 1로 두 경우에서 sum이 똑같이 0으로 나왔다고 해도 carry의 출력을 달리 해서 구분할 수 있게 된다. 따라서, A와 B의 입력이 둘 다 1일 때만 carry가 1이 되므로 carry=AB라는 식으로 결정되고, 1비트의 합은 두 개의 입력 중 하나만 1일 때 1이므로 앞선 실험에서 공부했던 XOR 게이트가 필요하고 이는 sum=A’B+AB’라는 식으로 결정된다는 것을 알 수 있다. 따라서, AND 게이트와 XOR 게이트를 사용하여 반가산기 회로를 구성해보면 아래 <그림 b>와 같이 된다. 이때, 출력 S는 sum, C는 발생하는 carry를 의미한다.

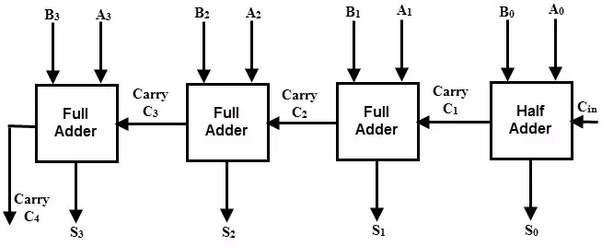


<그림 b> AND 게이트와 XOR 게이트를 활용한 반가산기 회로의 예시

전가산기는 이진수 덧셈을 수행할 때, 두 개의 한 자릿수 이진수 입력과 함께 하위 자리올림수(carry-in bit)를 포함하여 덧셈을 수행하는 회로이다. 아래 <그림 c>의 진리표에서는 Z가 하위 자리올림수 Cin에 해당하고 출력 C가 Cout에 해당된다.

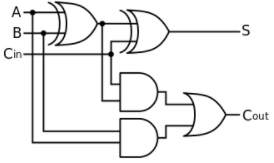
<그림 c> 전가산기 진리표

아래 <그림 d>를 보면 이와 같은 전가산기의 동작을 더욱 잘 이해할 수 있는데 A3A2A1A0와 B3B2B1B0이라는 네 자리 2진수를 더하고 싶을 때, 처음 1비트 2진수 A0, B0가 입력 되었을 때는 하위 자리올림수(Cin)이 없으므로 반가산기를 사용하여 더하면 되고, 그 결과 생긴 S0과 Carry, 즉 C1 중 C1은 전가산기의 하위 자리올림수로 들어가 A1과 B1을 더할 때 이용되어 S1과 C2를 발생시키고 이러한 회로를 2번 더 반복하면 최종적으로 C4와 S3, S2, S1, S0을 얻게 된다. 따라서 A3A2A1A0과 B3B2B1B0을 더한 결과는 carry를 C4로 갖는 S3S2S1S0이 된다.



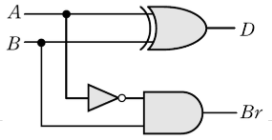
<그림 d> 4비트 전가산기

전가산기 회로를 나타낸 <그림 e>을 보면, 전가산기에서 S는 A, B, Cin 중 홀수 개의 입력이 1이어야 1의 출력을 가지므로 XOR 게이트를 두 번 거치면 되고, Cout은 세 가지 입력 중 2개 이상이 1이어야 1이므로 앞선 S을 구할 때 활용하였던 결과들을 이용하여 AND 게이트 두 번과 OR 게이트 한 번만 더 거치면 된다. 따라서, S=A⊕B⊕C라는 식으로 정리할 수 있고, Cout은 Cout=(A•B)+Cin•(A⊕B)라는 식으로 간단히 정리할 수 있다.



<그림 e> AND, OR 게이트와 XOR 게이트를 활용한 전가산기 회로의 예시

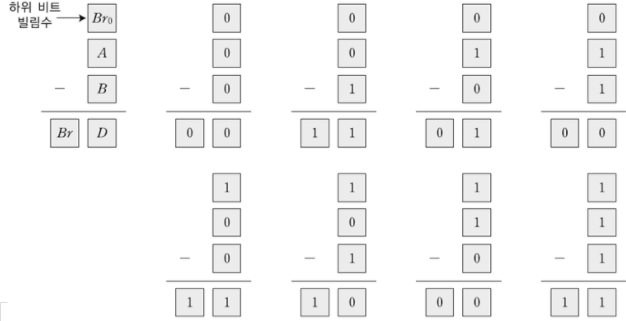
본 실험에서는 감산기 회로 또한 다루는 것 같기 때문에 이에 대해서도 간략히 설명을 해보면 감산기 또한 반감산기와 전감산기로 나뉘고 두 개의 회로 모두 두 개 이상의 입력에서 하나 입력으로부터 나머지 입력들을 뺄셈하여 그 차를 출력하는 조합 논리 회로이다. 가산기에서의 합은 감산기에서 차에 해당되며, 덧셈 과정에서 올림수(carry)가 발생하였던 가산기와 달리 감산기에서는 빌림수(borrow)가 발생된다.

반감산기는 1비트 2진수 두 개를 입력 받아 출력으로 차(D, difference)와 빌림수(Br)를 발생하는 회로로 입력 변수 A와 B가 있다고 하면 똑같이 0과 1을 각각 입력한다고 해도 A, B에 0, 1을 입력하는 것과 1, 0을 입력하는 결과가 다르게 나타난다. A에서 B를 빼는 반감산기에 1과 0을 입력하면 차는 1이 되고, Br는 0이 되지만 0과 1을 입력하면 차는 여전히 1이지만 Br가 1이 된다. Br가 1이라는 것은 두 수의 차가 음수라는 것을 의미하고 따라서 두 가지 입력에 대해서는 4가지의 서로 다른 결과를 얻을 수 있다.

<그림 f> NOT, AND, XOR 게이트를 활용한 반감산기 회로 예시

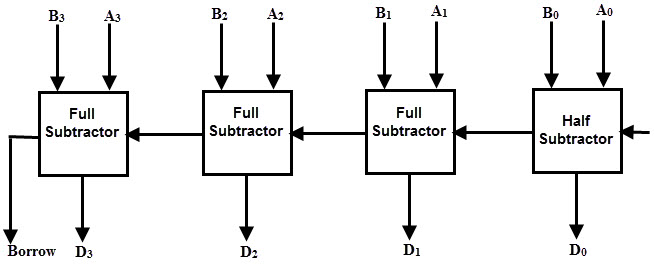
<그림 g> 반감산기 진리표

두 입력 간의 차이를 구하는 단순한 반감산기와 달리, 하위 비트에서 요구하는 빌림수에 의한 뺄셈까지 수행하는 전감산기 회로는 입력으로 해당 자리수의 뺄셈을 할 두 개의 A, B 입력과 아랫자리에서 요구하는 빌림수를 입력 받고 그 결과로 빌림수(Br)와 차(D)를 출력하는 회로이다.



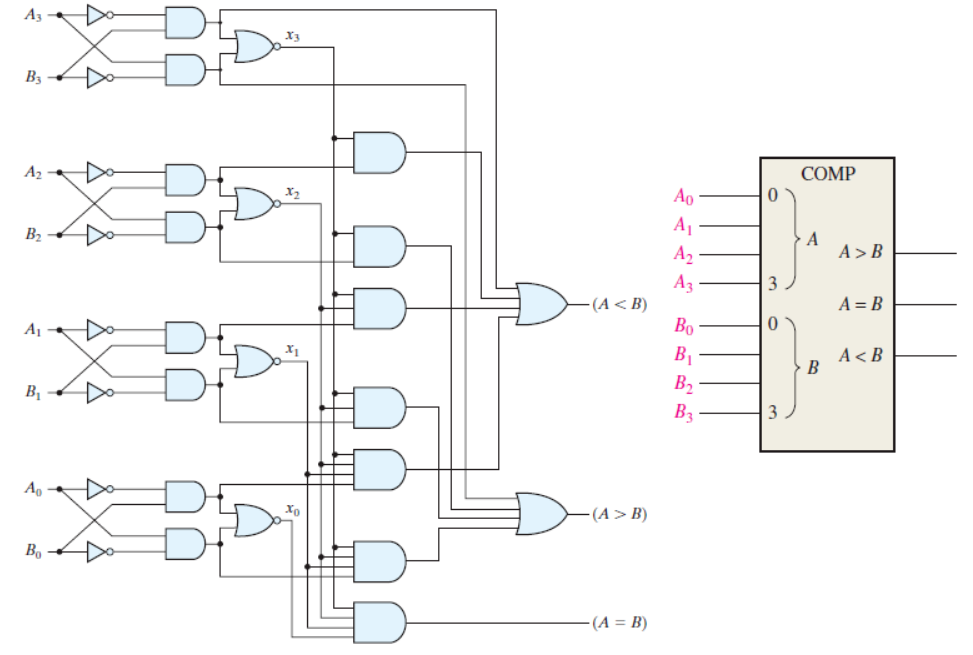
<그림 h> 전감산기에서 수행되는 8가지 뺄셈 계산

아래 <그림 i>를 보면 하위 자릿수에서 A0와 B0를 빼는 연산 과정에는 이보다 하위 자릿수가 존재하지 않기 때문에 단순히 반감산기 회로를 이용하여 계산하면 되고 그때 발생한 빌림수는 다음 상위 비트의 차를 계산하는 전감산기 회로의 입력으로 들어가서 최종적으로 A3A2A1A0과 B3B2B1B0의 차를 구하면 빌림수 한 개와 D3D2D1D0이라는 출력을 얻을 수 있게 된다.

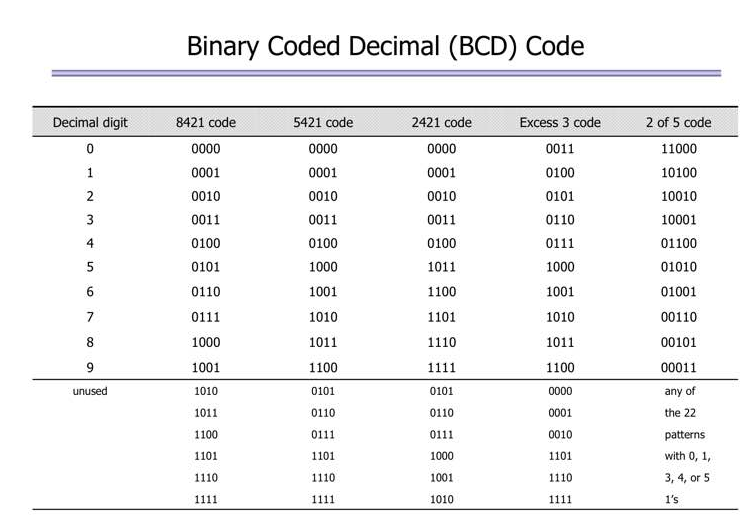


<그림 i> 4비트 전감산기

1. 비교기

비교기 회로란 두 가지 A, B의 입력에 대해 두 수의 크기를 비교하여 A>B, A=B, A<B의 상태에 따라 각기 다른 출력을 하는 것이다. 그러나 기본적인 1비트 비교기보다는 2비트 이상의 비교기에 대해 설명해보겠다. 우선, 4비트 2진수 A=A3A2A1A0, B=B3B2B1B0에 대해서 두 수의 크기를 비교해보면 A>B, A=B, A<B 중 하나로 결정되고, xi=Ai’Bi’+AiBi=A⊙B (i=0, 1, 2, 3)라고 할 때 A=B일 때 해당 출력이 1이 되려면 x3x2x1x0이어야 하고 A>B일 때 1이 되려면 A3B3’+X3A2B2’+x3x2A1Ba’+ x3x2x1A0B0’, A<B일 때 1이 되려면 (A3B3)’+x3A2’B2+x3x2A1’B1+x3x2x1A0’B1이어야 하므로 이를 회로로 나타내면 아래 <그림 j>와 같게 된다.

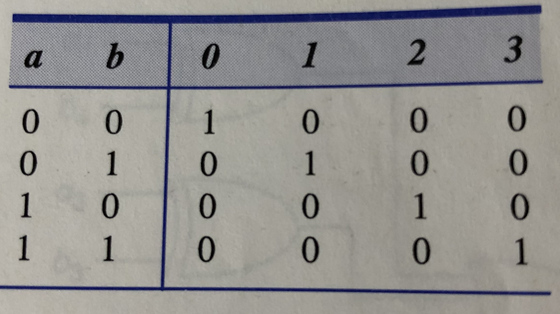
<그림 j> 4비트 비교기 논리 회로와 기호

1. 2진화 10진수 (binary coded decimal, BCD)

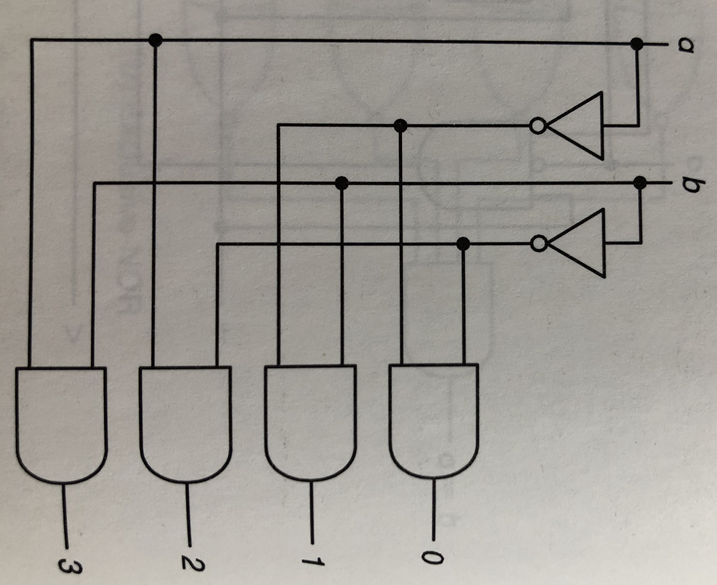
<그림 k> BCD code

2진화 10진수, 다른 말로 이진화 십진법은 이진수 네 자리를 묶어 한 개의 한 자리 10진수를 표현하는 방법으로, 숫자의 십진 출력을 요구하는 전자 회로나 마이크로프로세서 등에서 많이 이용된다. 하지만 예를 들어 BCD를 이용해서 세 자리 십진수를 표현하려면 12 비트가 필요한데 이는 저장장치에서 많은 부분을 차지해서 비효율적일 뿐만 아니라 산술연산에서도 이진수보다 훨씬 복잡하기 때문에 제한된 영역에서만 이용된다. 위의 <그림 k>는 0부터 9까지의 숫자를 표현하는 여러 가지 BCD code를 나타낸 것이다. 8421 code를 사용하면 일반적으로 10진수를 2진수로 바꾸는 데에 사용되는 방법과 다르지 않기 때문에 어렵지 않게 이해할 수 있고 5421이나 2421에서는 예를 들어 5를 표현하는 방법이 5421 code에서는 1000과 0101이, 2421 code에서는 0101과 1011이 있지만 둘 중 한 가지의 표현만을 사용하고 다른 하나는 사용하지 않는다는 특징이 있다. Excess 3 code는 일반적인 10진수 2진화 방법에서 3을 표현하는 방식이 0이 되고 12를 표현하는방식이 9가 되는 code이고 2 of 5 code는 앞에서 설명한 code들과는 다르게 네 자리 비트가 아니라 5 비트에서 2 비트만이 1인 표현 방법으로 0부터 9까지의 숫자를 표현하는 방법이다.

1. 디코더(decoder)와 인코더(encoder)

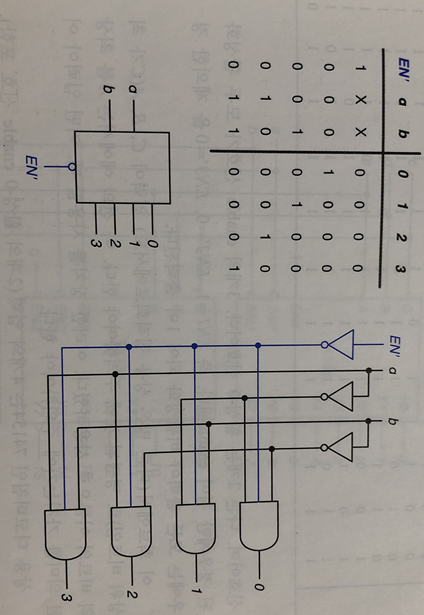
이진 디코더란 활성화되었을 때, 코딩된 입력 신호에 기초하여 여러 개의 출력 중에서 하나를 선택하는 장치이다. 대부분의 경우 입력은 n 비트의 이진수이며 2n의 출력이 존재한다. 2개의 입력을 가지는 디코더의 진리표가 아래 <표 a>와 같을 때 입력은 이진수로 취급이되고 선택된 출력은 1로 활성화되었다. 이 예시에서 출력은 활성-1이고, 즉 활성화된 출력은 논리 1이며 비활성화된 출력은 논리 0이라는 의미이다. 이 디코더는 출력에 대한 AND 게이트와 입력을 반전시키기 위한 NOT 게이트로 구성이 된다. <그림 a’>은 블록도로 입력 0에 대한 출력은 a’b’, 입력 1에 대한 출력은 a’b이며 입력 2와 3에 대한 출력은 각각 ab’, ab이다.

<표 a> 활성-1 디코더 진리표



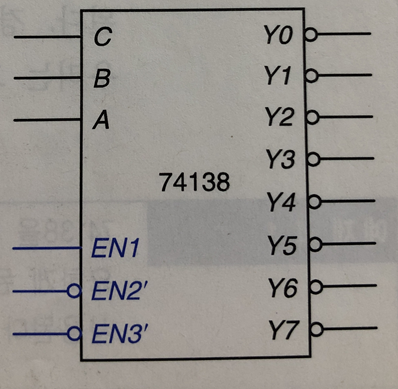
<그림 a’> 활성-1 디코더

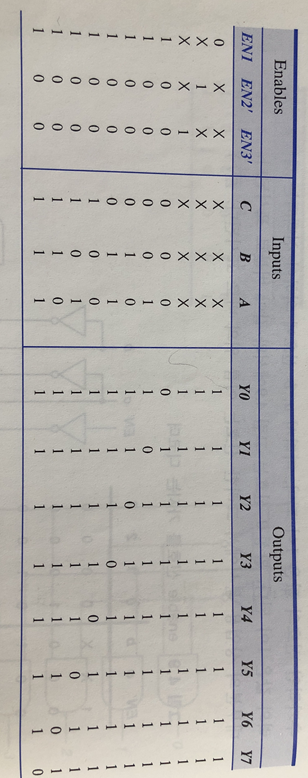
반면, 활성-0 출력을 가지는 디코더는 입력 조합에 따라 하나의 논리 0값을 갖고, skj지 출력은 논리 1이 된다.

대부분의 디코더는 하나 이상의 enable 입력을 가지고 있다. 이 입력이 활성화되는 경우 디코더는 앞에서 기술된 대로 동작하게 된다. 이 입력이 비활성화된 경우에는 디코더의 모든 출력이 비활성화 된다. 하나의 enable 신호를 가지는 디코더뿐만 아니라 대부분의 시스템에서 이 신호는 활성-0 신호이다. 아래 <그림 b’>는 활성-1 출력과 활성-0 enable 입력을 가지는 디코더의 진리표, 블록도, 회로이다.

<그림 b’> enable 신호를 가지는 디코더

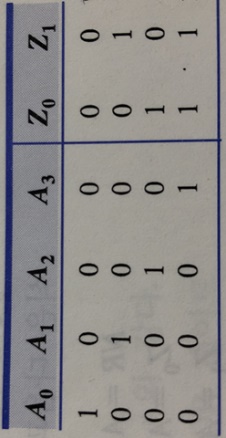
이 회로에서 enable 신호가 반전되어 각 AND 게이트에 입력되고 있는 것에 주목하면 EN’이 1인 경우, 각 AND 게이트의 입력에는 0이 나타나고 그 결과 모든 AND 게이트의 출력은 0이 된다. 반대로 EN’이 0인 경우 enable 신호를 갖지 않는 회로에 추가적인 신호의 값은 1이 되며 따라서 a, b 신호에 의하여 선택되는 출력의 값이 1이 된다.

디코더 74138에 대한 진리표 <표 b>와 블록도 <그림 c’>을 보면 이 칩은 활성-0 출력과 3개의 enable 신호를 가지고 있으며 이 3개의 enable 신호 중 하나(EN1)은 활성-1이며 다른 2개는 활성-0 신호이다. 따라서 EN1=1, EN2’=EN3’=0인 경우를 제외하고 모든 경우에서는 출력이 비활서오하 되어 1이 출력된다.

<그림 c’> 74138 디코더

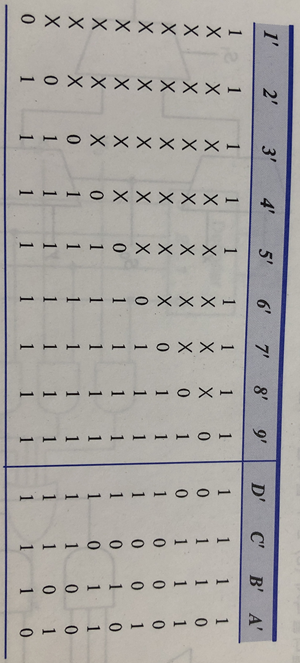
<표 b> 74138 디코더 진리표

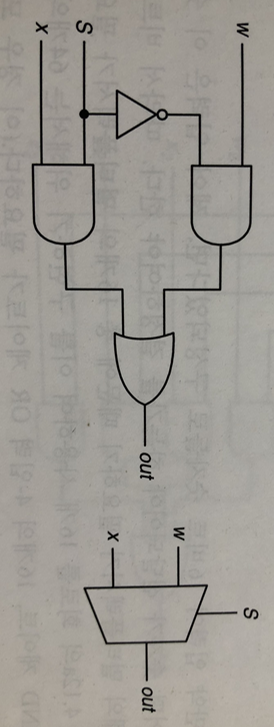
이러한 디코더는 여러 장치 중에서 하나를 선택하는 용도로 많이 사용된다. 이 경우 각 장치는 고유한 주소를 가지며 장치의 주소는 디코더에 입력되고, 하나의 출력만이 활성화되어 활성화된 입력에 연결된 장치만이 사용가능하게 되는 것이다. 경우에 따라서는 하나의 디코더로 다수의 장치가 선택되기도 한다. 디코더의 다른 용도로는 논리함수를 구현하는 것이다. 디코더의 각 활성-1 출력은 최소항에 해당되므로 논리함수를 구현하기 위해서는 적당한 출력을 OR 게이트로 연결하면 된다. 활성-0 출력 디코더의 경우, OR 게이트를 NAND 게이트로 변경하면 되고, 같은 입력을 가지는 하나 이상의 함수를 구현하는 경우에도 하나의 디코더와 함수 수만큼의 OR 또는 NAND 게이트를 사용하면 된다.

이진 인코더는 이진 디코더의 반대되는 동작을 하는 블록이다. 이진 인코더는 여러 개의 장치가 컴퓨터에 연결된 선에 1을 가함으로써 신호를 보내는 것과 같은 일을 할 때에 유용하다. 이 경우, 인코더는 장치의 번호를 생성하는데 <표 c>의 진리표를 보면 A0, A1, A2, A3의 4개의 입력 중 하나의 입력만이 논리 1의 값을 가지는 인코더의 동작을 기술한다.

<표 c> 4-라인 인코더

A0=1인 경우, 0번 장치를 표시하기 때문에 Z0=Z1=0이 된다. 마찬가지로 A2=1인 경우에는 2번 장치를 표시하기 위하여 Z0=1, Z1=0이 된다. 만약 항상 하나의 입력만이 논리-1의 값을 가진다면 <표 c>의 Z0, Z1의 식은 Z0=A2+A3, Z1=A1+A3이 된다. 그러나 이와 같이 구성된 경우에는 0번 장치가 선택된 경우와 아무 장치도 선택되지 않은 경우를 구별할 수 없으므로 이 같은 문제를 해결하기 위하여 한 개의 입력을 추가적으로 사용할 수 있다. 새로운 입력, N은 어떤 장치도 활성되지 않았다는 것을 의미하고 출력 NR은 어떤 입력도 활성화되지 않았다는 것을 표시한다.

만약 하나 이상의 입력이 동시에 1이 될 수 있다면 입력 간의 우선순위를 정해야 하고, 입력의 우선순위는 내림차순이나 오름차순으로 정렬되어 가장 크거나 가장 작은 입력 번호에 대해 최고의 우선순위를 배정하게 된다. 상용 BCD 74147 인코더는 9개의 활성-0의 입력을 받아서 4개의 활성-0 출력을 생성하고 이 인코더의 진리표는 다음 <표 d>와 같다.



<표 d> 74147 우선순위 인코더

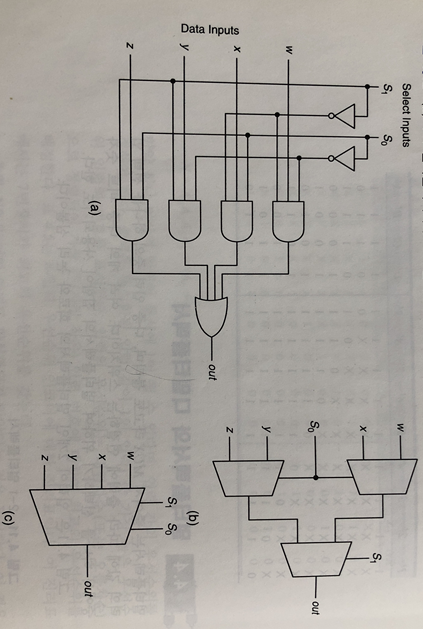
1. 멀티플렉서(multiplexer)와 디멀티플렉서(demultiplexer)

멀티플렉서란 다중 입력 중 하나를 선택 입력 값에 따라 출력에 연결하는 스위치이며 여러 개의 다중 비트 숫자 중에서 하나를 선택하기 위하여 멀티플렉서의 집합이 사용되기도 한다. 아래 <그림 d’>을 보면 출력(out)은 S=0인 경우에는 w와 같으며 S=1인 경우에는 x와 같다.

<그림 d’> 2-1 멀티플렉서의 회로와 논리 심볼

4-1 멀티 플렉서는 AND 게이트와 OR 게이트를 사용하여 구현이 가능하거나 2-1 멀티플렉서 3개를 이용하여 구현이 가능하다. 출력(out)은 (S1, S0)=(0,0)인 경우에 w와 같으며, (S1, S0)=01인 경우에는 x와 같고, 마찬가지로 (S1, S0)=10, 11인 경우에는 출력은 각각 y, z와 같다. 이 회로는 각 선택 입력에 AND 게이트가 하나씩 붙어있기 때문에 디코더 회로와 유사하다. 어떤 멀티 플렉서는 enable 입력을 가지고 있으며 이 경우에는 enable 신호가 활성화되지 않으면 출력은 항상 0이 된다.

멀티플렉서는 논리 함수 구현에 사용될 수 있으며 멀티플렉서를 이용한 가장 간단한 구현 방법은 선택 입력을 디코더처럼 사용하고 데이터 입력에 0 또는 1을 연결하는 것이다.



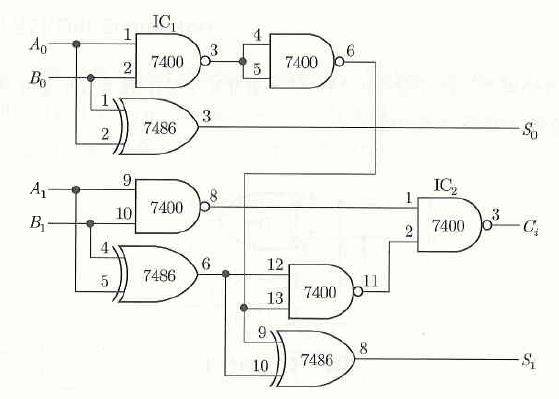
<그림 e’> 4-1 멀티플렉서와 논리 심볼

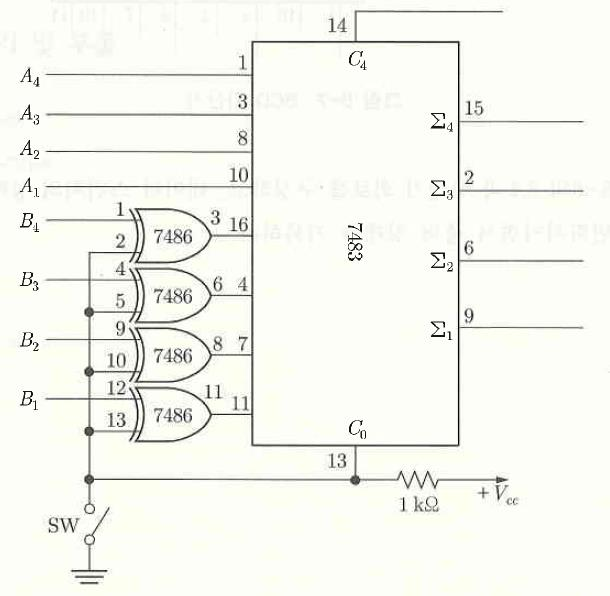
디멀티플렉서는 멀티플렉서의 반대되는 동작을 수행하며, 신호를 한 장소에서 여러 장소 중 하나로 배선한다.

1. 실험 장비 및 재료
2. 실험 장비
3. NI ELVIS
4. 오실로스코프: PHILIPS 60 MHz Digital Storage Oscilloscope PM3335
5. 함수발생기: EZ FG-8002
6. DC power supply
7. 실험 재료
8. 저항
9. 7404, 7408, 7420, 7432, 7447, 7483, 7485, 7486, 74139, 74151
10. 7-segment
11. 실험 방법
12. 가산기와 감산기

가산기는 컴퓨터에서뿐만 아니라 수치 데이터를 처리하는 디지털 시스템에서 매우 중요한 요소이므로 가산기의 기본 동작을 이해하는 것은 디지털 시스템을 공부하는 데에 중요한 기초가 된다.

아래 <그림 1>의 회로를 구성하고 진리표를 작성하고, <그림 2>의 회로를 구성하고 SW의 상태에 따라 진리표를 작성한다. 서로 다른 두 회로의 진리표를 비교함으로써 가산기와 감산기의 차이점에 대해 이해해본다.



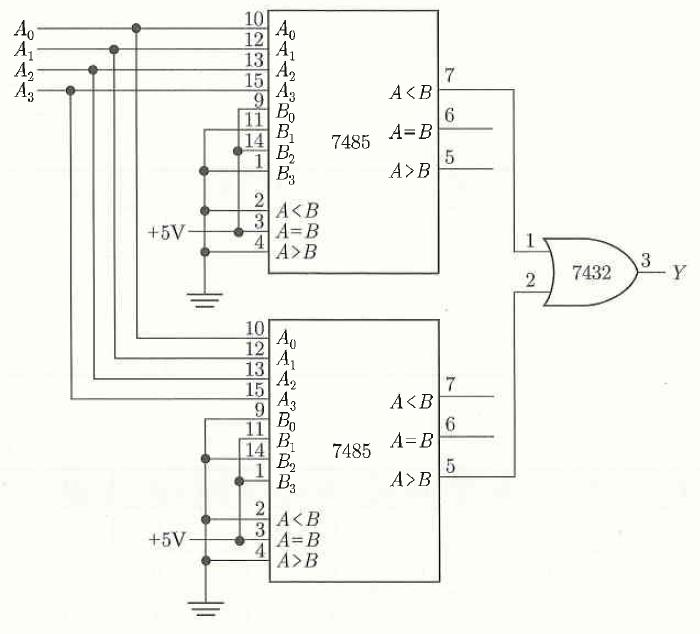
<그림 1> 2비트 병렬 가산기

<그림 2> 4비트 가, 감산기

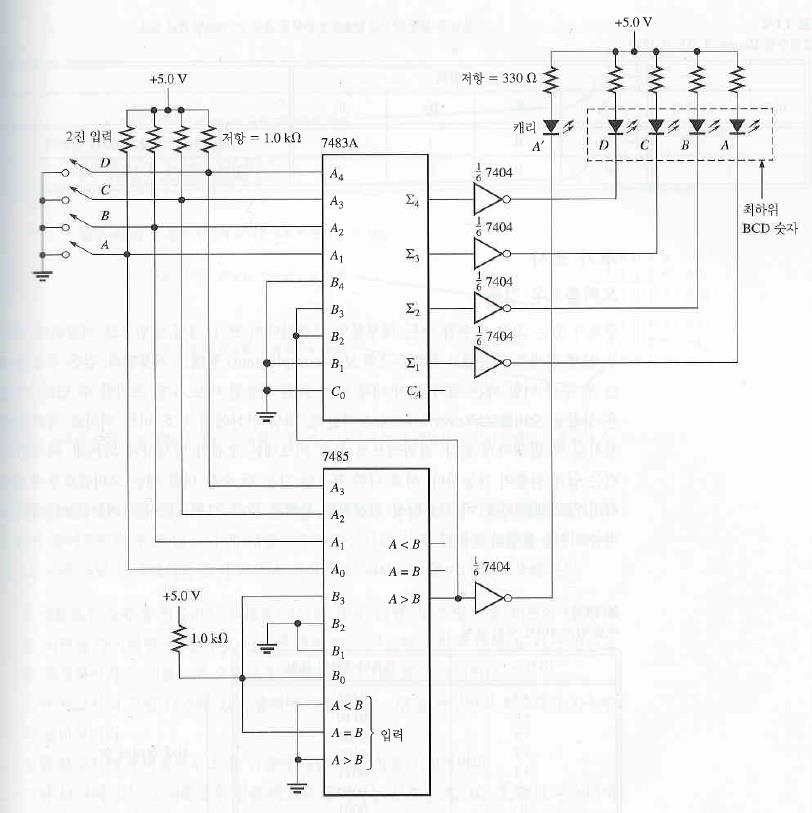
1. 비교기

비교기의 기능 중 가장 기본적인 것은 두 2진수의 크고 작음을 결정하기 위해 두 수의 크기를 비교하는 것이다. 이때, XNOR 게이트를 기본적인 비교기로 사용할 수 있지만 본 실험에서는 XNOR 게이트를 이용한 기본적인 회로는 생략하고 4비트 비교기인 7485 소자를 바로 다루도록 하였다.

아래 <그림 3>의 회로를 구성하고 진리표를 작성한다. 또한 아래 <그림 4>와 같은 회로를 구성하고 진리표를 작성한다. 이때, <그림 4>의 회로는 2진수를 2진화 10진수(binary coded decimal, BCD)로 변환해주는 회로이고 본 실험에서는 이 회로에서 가산기의 입력 부분인 B1~B4 단자에 인가해주는 입력이 회로에서 어떠한 역할을 하는지 진리표를 통해 이해하는 것이 매우 중요하다. 또한 BCD는 디코더에 관한 실험에서 중요하기 때문에 개념을 확실히 익히도록 한다.



<그림 3> 4비트 비교기를 이용한 입력 비교 회로

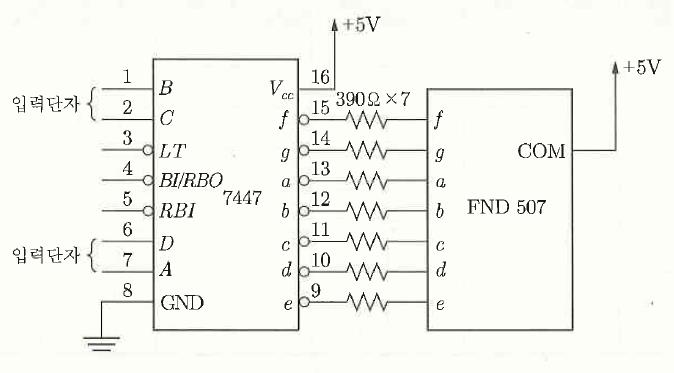


<그림 4> 2진, BCD 코드 변환기

1. 디코더

상태나 명령들을 그에 사응한 2진 정보로 변환하는 회로를 인코더(encoder)라고 하며, 역으로 주어진 2진 정보가 어떠한 상태 혹은 명령을 나타내는가를 해독하는 회로를 디코더(decoder)라고 한다. 본 실험에서는 대표적인 디코더 중 하나인 BCD to 7-segment decoder에 대해 다루도록 한다.

아래 <그림 5>의 회로를 구성하고 입력에 따른 7-segment의 출력 상태를 기록한다. 이때, 와단자에는 1의 입력을, 단자에는 입력이 모두 0인 경우에만 1을 넣고 나머지는 0 혹은 1 중 어떠한 입력이든 무관하게 넣는다.

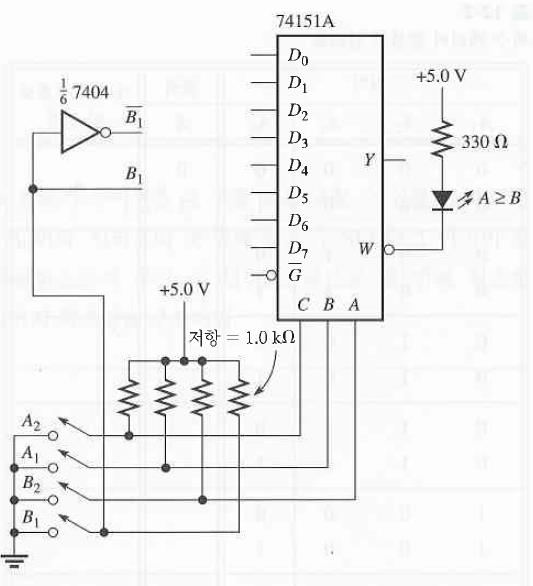


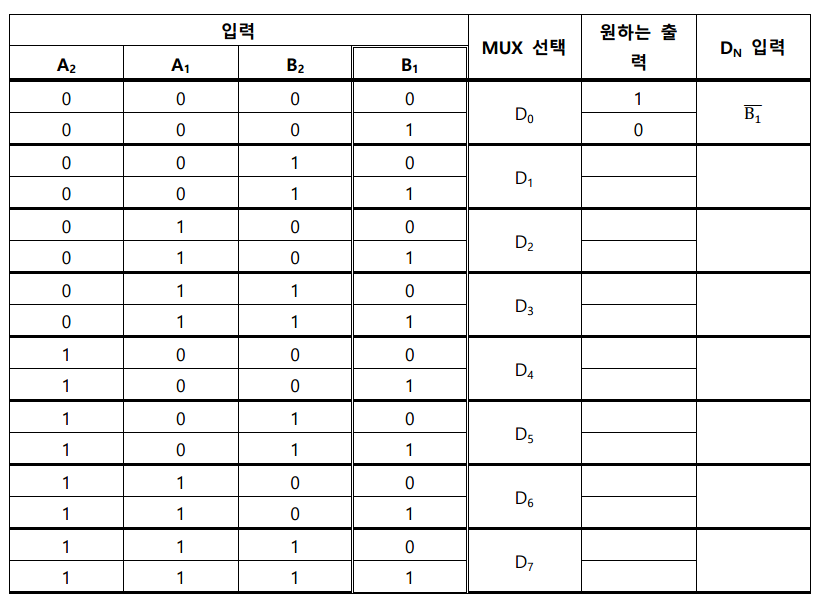
<그림 5> BCD to 7-segment decoder

1. 멀티플렉서, 디멀티플렉서

멀티플렉서(multiplexer)는 여러 개의 데이터 소스로부터 입력되는 디지털 정보들을 하나의 출력단을 통해 공통의 목적지로 전송하는 소자이다. 기본적인 멀티플렉서는 여러 디지털 입력 데이터 중 어떠한 데이터를 출력할 것인지를 선택하는 데이터 선택 입력, 다수의 입력선과 하나의 출력선을 가지고 있다. 멀티플렉서는 이러한 의미에서 데이터 선택기라고도 불린다. 반면, 디멀티플렉서(demultiplexer)는 하나의 선으로 디지털 정보를 받아서 여러 출력선으로 분배한다. 디멀티플렉서는 이러한 의미에서 데이터 분배기라고도 한다. 이번 실험에서는 시간 관계상 멀티플렉서와 관련된 회로만을 다루도록 한다.

아래 <그림 6>의 회로는 멀티플렉서 소자를 이용한 비교기 회로이다. A≥B일 경우에 Y 단자에 1이 W 단자에 0이 출력되도록 D0~D7 단자에 적절한 신호를 인가해야 한다. 아래 진리표를 완성하여 <그림 6>의 회로가 2비트 비교기로 작동할 수 있도록 D0~D7 단자에 0, 1, B1, 중에서 적절한 신호를 인가하여 회로를 완성하도록 한다.



<그림 6> 2비트 비교기

<표 1> <그림 6> 회로의 진리표

1. 참고문헌

-Earl Gates, 전기전자공학, 1판, 북스힐, 2018년, pg. 247-248, 280-289

-Alan B. Marcovitz, Introduction to logic design, 3판, McGraw-Hill Higher Education, 2009년, pg. 20-22, 197-218