Digital Logic Design I

이수용(한국과학기술연구원 휴먼로봇연구센터)

기계저널 독자는 대부분 기계공학을 전공하는 분들이지만, 학부과정에서 전자공학 개론을 선택하여 수강한 경험이 있을 것이다. Analog 회로이론뿐만 아니라 Digital Logic은 전자공학의 가장 근본이 되는 것이다. 그러나 전자공학 개론을 수강하지 않았더라도, 'AND', 'OR', 'NOT' 등은, programming할 때 조건문에서, 또는 internet에서 원하는 자료를 검색할 때에 누구나 사용하고 있을 것이다. 다만, Digital Logic에서는 'NAND', 'XOR' 등 흔히 접하지 않는 몇 가지 논리를 더 배우는 것뿐이다. 예를 들어 C 언어를 사용한 프로그램에서 다음과 같은 source code를 살펴보면,

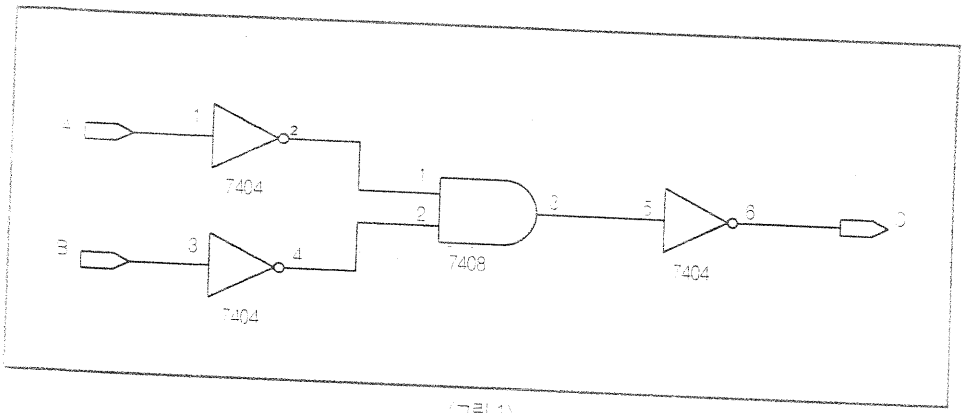
```
if ( (A==0) && (B!=1) )
        C=0;
else
        C=1;
```

Digital Logic에 관한 내용이므로, 모든 변수는 1 (High) 또는 0 (Low) 값만 갖는다고 가정한다. A의 값이 0이고, B의 값이 1이 아닌 경우 C의 값은 0이 되고, 이상의 조건들이 만족하지 않을 때는 C의 값은 1이 되는 내용이다. 따라서 복잡한 논리라도 프로그램 언어를 이용하여, 단순한 여러 식의 조합으로 표현할 수 있다. 그러나 이를 하드웨어로 구현하기 위하여 그림 1과 같이 구성하여야 한다.

A와 B신호를 각각 NOT gate를 거친 후 AND시킨 결과를 다시 한번 NOT gate를 거친 출력이 C가 되는 것이다. 그림 1에서 NOT gate는 7404, AND gate는 7408이라고 적혀 있는 것은 digital logic IC로 가장 많이 사용되고 있는 Texas Instrument 社의 TTL Logic series 부품번호이다. 즉 7404라는 IC와 7408이라는 IC를 사용하여 그림 1과 같이 배선하면 A와 B에 5V(high) 또는 0V(low)를 가하였을 때 C에 논리 결과에 해당되는 5V나 0V의 전압으로 출력된다. 이와 같은 예는 두 개의 입력으로 이루어졌으므로, 간단히 논리표(표 1)를 만들어 볼 수 있다. 결과를 보면 이 logic은 C=A OR B와 같다는 것을 쉽게 알 수 있을 것이다. 즉, 하나의 digital logic에 대하여 다양한 방법으로 복잡하게, 또는 아주 간단하게 표현할 수 있는 것이다.

똑같은 로직을 OR gate인 7432를 사용하여 그림 2와 같이 하나의 IC로 구현할 수도 있다.

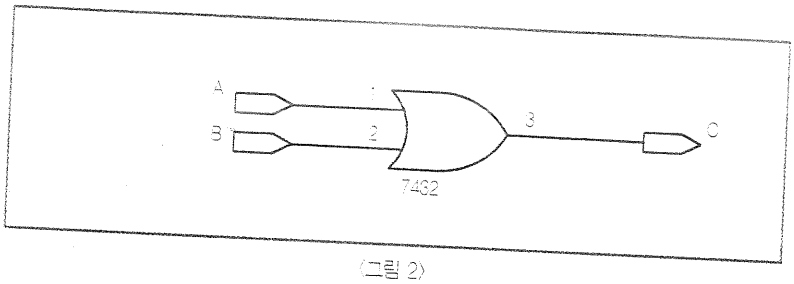
위의 예는 매우 단순한 경우이므로 몇 개의 IC만으로 충분하였으나, 보다 복잡한 logic은 당연히, 매우 많은 logic IC를 사용하여야 한다. 그렇다



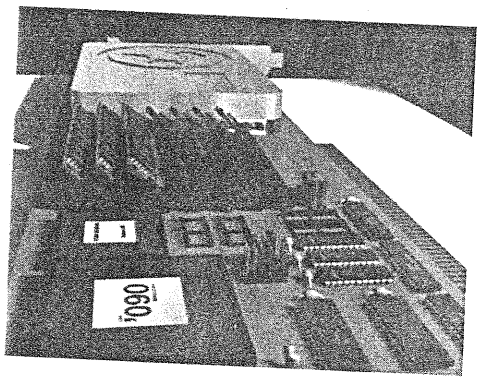
(그림 1)

면, 마이크로프로세서를 프로그램하여 복잡한 logic을 나타낼 수 있는데도 불구하고, 왜 digital logic을 소개하였는지 당연히 의문이 생길 것이다. 간단한 기계들의 작동을 살펴보면 수식 연산은 필요 없이 논리연산만으로 제어할 수 있는 경우가 많다. 예를 들어 담배 자판기의 경우 투입된 동전 또는 지폐의 금액(동전 또는 지폐 판독기로부터의 digital 입력)에 따라 살 수 있는 담배의 선택 버튼에 불이 켜지며, 눌려진 버튼에 해당되는 담배 저장 stack의 출구를 한 번 작동한 후, 거스름돈을 내주는 것이다(이 과정은 수식 연산 또는 counter logic을 이용한 논리 연산으로 볼 수 있다). 유료 주차장의 입구에 설치된 '만차' 램프는 들어온 자동차수에서 나간 자동차수를 뺀 값이(이 또한 counter logic을 이용한 논리 연산으로 가능하다) 주차장 수용 가능 자동차 수와 같거나 커지면 램프를 켜는 것이다. 마이크로프로세서를 사용하였다면 단 몇 줄의 프로그램으로 원하는 결과를 얻을 수 있으나 logic IC를 사용하면 약간 복잡한 회로가 될 것이다. 그러나 논리회로는 대부분 static mapping이므로 주어진 입력에 대하여 언제나 정해진 논리 결과만 출력한다. 마이크로프로세서와 같이 발진회로나 메모리가 필요 없으며, 보다 높은 신뢰도를 갖는 장점이 있다. 가격면에서는 마이크로프로세서의 단가가 성능은 향상되면서도 지속적으로 하락하고 있으므로 큰 비교가 되지 않을 것이다. 반면에 논리회로를 수정하고자 할 때 마이크로프로세서는 소프트웨어만 수정하면 가능하나, 논리회로는 하드웨어를 교체하여야만 한다. 하지만 이와 같은 단점들은 최근 FPGA(Filed Programmable Gate Array)와 같은 논리소자들의 발전으로 대부분 해결되고 있다. 즉 수백~수십만 개의 logic gate를 하나의 IC에 집적한 IC가 등장하여, 마이크로프로세서를 프로그래밍하는 것과 마찬가지로 쉽게 내부 논리를 몇 번이고 수정 가능하여졌다. 만약 logic IC를 사용하였다면 방 하나를 가득 채울 IC들이 필요하였던 것이 단 하나의 IC에 내장할 수 있는 것이다. 완제품 컴퓨터를 구입하지 않고, 부품을 모아 컴퓨터를 조립해본 사람들은 메인보드의 사양에 어느 회사의 어떤 chipset을 사용하였다는 것을 보았을 것이다. 예전 XT나 386 컴퓨터의 메인보드에는 메모리 외에도 많은 주변 IC들이 보였으나, 요즘 컴퓨터를 열어보면 CPU와 메모리 이외에는 몇 개의 큼직한 IC만 붙어 있는 것을 볼 수 있을 것이다. 이들이 모두 복잡한 논리 회로들을 집적한 chipset들이다. 그러나 이러한 IC들의 발전으로 마이크로프로세서를 완전 대치할 수는 없으며, 서로 상호 보완적인 역할을 하며, 그 용도에 따라 선택하여야 할 것이다. 다음 호에는 이러한 집적소자들과 이를 프로그래밍하는 방법에 대하여 소개하겠다.

〈이수용 위원; gemma@kistmail. kist.re.kr〉



〈그림 2〉



〈표 1〉

A	B	C
0	0	0
1	0	1
0	1	1
1	1	1