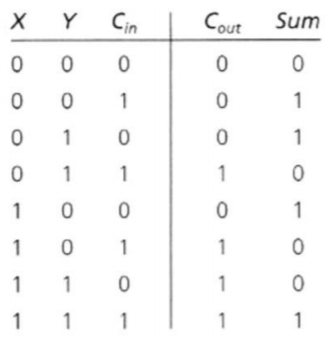
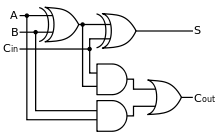
6주차 예비보고서

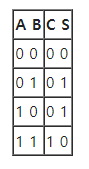
전공: 컴퓨터공학과 학년: 3학년 학번: 20191612 이름: 윤기웅

**1.**

1) 전 가산기 (Full Adder): 반 가산기와 다르게 아래의 자리에서 생긴 캐리 값까지 포함시켜서 세 개의 비트를 더하는 논리회로이다. 즉, 입력되는 값이 X,Y,으로 3개이고 출력값은 S,C로 2개이다. 전 가산기에 대한 표를 만들어보면 S=, 라는 식이 만들어지게 된다. S는 주어진 3개의 입력 중에서 1의 개수가 홀수이면 1이 되고 은 주어진 3개의 입력 중에서 1의 개수가 적어도 2개 이상이 되어야 한다

. 

2) 반 가산기: Half Adder 라고도 불리며 들어오는 입력 a,b에 대해서 기본적인 덧셈 연산을 실행시킨다. 전 가산기와는 다르게 1비트만 연산해서 최하위 비트에서만 사용 가능하다. 들어오는 입력을 각각 A, B라고 가정하면 , 즉, S는 A,B 둘 중 하나만 1일 때 1이 되고 둘 다 0이거나 1인 경우에 0을 출력한다. C는 A,B가 모두 1인 경우에만 1을 반환한다.

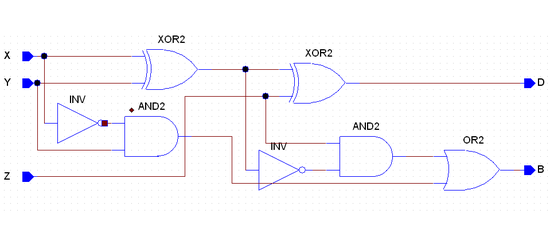


**2.**

**1) 전감산기 :**

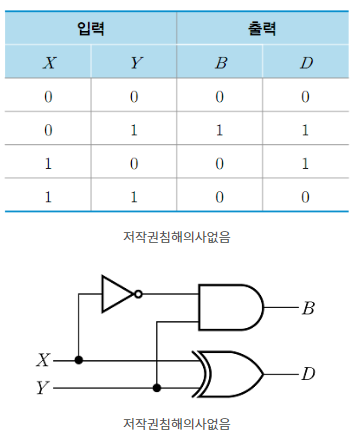
가산기와 다르게 뺄셈 연산을 하는 논리회로이다. 이전 계산의 바로우를 포함하여 뺄셈을 할 수 있도록 설계되었으며 반감산기가 2개와 OR 게이트 하나로 전 감산기를 만들 수 있다. 입력으로는 X,Y,B1이 들어오고 출력으로는 B0,D가 발생한다. 입력으로 들어온 바로우는 앞의 자리 연산에서 발생한 것으로 X,Y에 의한 뺄셈이 일어나지 않더라도 B1이 있는 경우에 뺄셈 연산을 수행해주어야 한다는 것이다. 만약 입력이 X,Y,B1 순서대로 0,0,1이면 B0, D는 모두 1이 된다. ,

****

****

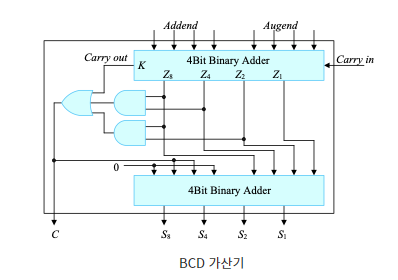
**2)반 감산기 :**

바로우가 발생하게 되고 이는 원래 음수여야 하지만 여기서는 양수로 표현한다. 입력은 X,Y로 두 개가 있고 출력은 바로우인 B와 X,Y의 뺄셈 자리에 있는 결과값 D가 있다. 그래서 만약 입력값 두 개가 X,Y순서대로 1,0인 경우 바로우는 필요없고 1-0=1이라서 D가 1이 된다. 아래의 표를 보면 B=X'Y, D= X(xor)Y의 연산을 하면 된다.



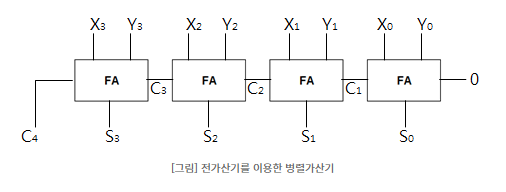
3.

BCD란 biniary coded decimal code의 줄임말이고 BCD 코드로 만들어진 10진수 숫자를 2개 입력 받고 더하기를 수행해서 그 결과를 다시 BCD 코드로 출력하는 회로이다. 숫자의 표현범위가 0부터 9까지 모두 가능해서 결과값을 이진수 변화하면서 보정을 해줘야 한다.

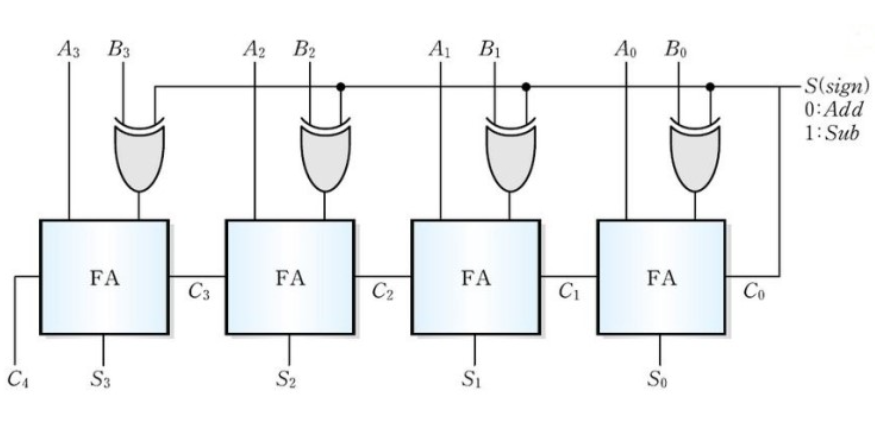


4.

병렬 가산기는 전 가산기를 병렬적으로 여러 개 연결한 회로를 의미한다. 만약 n개의 전 가산기로 이루어져 있으면 n Bit의 2진수 a,b를 더한다는 뜻이다. carry look ahead를 사용하면 전파의 지연을 막을 수 있다.



위의 병렬 가산기를 변형해서 병렬 가감산기를 만드는 것이 가능하다. 맨 아래의 입력의 캐리를 부호로 정하고 각각의 B입력과 부호를 xor게이트에 입력으로 정한다. 그리고 xor게이트의 출력을 전 가산기의 입력으로 정해준다. xor 게이트에서 부호가 0이면 덧셈을 의미하여 원래 B값이 출력되고 부호가 1이면 뺄셈을 의미해서 B와 반대되는 값을 내보낸다. 그리고 맨 아래의 캐리가 1이면 뺄셈을 의미해서 뺄셈을 하는 경우 2의 보수 뺄셈이 실행된다.

****

5. carry look ahead adder는 ripple carry adder보다 지연 속도를 줄일 수 있는데 계산을 하는 경우에 이전 비트의 carry전달에 상관 없이 한 번에 계산을 할 수 있기 때문이다. ripple carry adder는 계산을 할 때 순차적으로 진행이 되어서 이전의 연산 값이 필요해서 속도가 상대적으로 느리다. 즉, n번째 가산기는 n-1개의 가산기가 완료될 때 까지 기다려야 한다. 연산의 양이 많아지면 n이 커지고 그러면 더 많은 시간이 연산에 필요하다. 그러나 carry look ahead adder는 더 향상된 성능을 보이고 시간복잡도는 logn이다. 연산을 할 때 G,P를 만들어 낸다. G는 자리 올림수 생성이고 P는 자리올림수 전파라고 불린다. G(A,B)=A⋅B, P(A,B)=A⊕B의 식으로 만들 수 있고 다시 G,P를 이용해서 sum, carry를 만들 수 있다. Si=Pi⊕C, Ci+1=Gi+PiCi을 보면 sum은 A,B,C의 xor 연산으로 볼 수 있고 carry는 G, P를 모두 사용해서 만들어야 한다.

6.

가산기는 컴퓨터 연산에 필요한 장치로 사용되고 디지털식 장치의 연산에 쓰이는 기초 소자이다. 사람은 10진법의 표현이 편하고 컴퓨터는 2진법의 숫자 표현이 편하다. 그래서 아스키 코드를 사용하거나 BCD 코드를 사용해서 사람들이 사용하기 편하게 해주고 있다. 요즘은 아스키 코드가 더 대중적으로 쓰이긴 하지만 BCD 또한 이것만의 특징 때문에 사용되기도 한다. 8421코드라는 것이 있는데 이는 각 자리의 가중치가 각각 8,4,2,1이라서 붙은 이름이다. 이는 하드웨어 알고리즘의 관점으로 보았을 때 다른 알고리즘보다 구현하기가 쉽다. 자리수에 맞는 변환기 하나만 필요해서 입력을 받고 보여주는 것에 효율적이다. 아스키 코드의 방식보다 차지하는 크기가 작고 정확한 소숫점 표현이 가능하다.