6주차 예비보고서

전공: 경영학과 학년: 3학년 학번: 20190963 이름: 한다현

**1.**

논리 회로의 종류에는 조합 논리회로와 순차 논리회로가 있는데, 전 가감산기와 반 가감산기는 조합 논리회로에 속한다. 반 가산기는 두 개의 입력과 두 개의 출력으로 이루어져 있는 조합 논리 회로이고 두 개의 단일 비트 이진수가 더해지도록 설계되어 있으며 두 개의 출력은 carry와 sum이라고 불린다. 이 때 sum이란 두 2진수의 합을 의미하고 carry는 자리 올림을 의미한다. 반 가산기의 예로, 반가산기의 sum은 두 이진수 중 하나의 값이 1일 때 1을 출력하고 둘 다 0이거나 둘 다 1일 때는 1을 출력한다. Carry는 두 이진수가 모두 1일 때만 1을 출력하고 나머지 경우에는 0을 출력한다. 즉 Sum은 XOR 게이트의 출력이고 carry는 AND 게이트의 출력이라고 할 수 있다. 입력 A와 B가 (0,0), (0,1), (1,0) 인 경우에는 자리 올림이 발생하지 않기 때문에 carry는 0이 되고 sum의 값은 각각 0, 1, 1이 된다. 반면 A와 B가 모두 1인 경우에는 자리 올림이 발생하기 때문에 carry의 값은 1이 되고 sum의 값은 0이 된다. 전 가산기는 반 가산기의 단점을 보완하기 위해 개발되었는데 전 가산기는 기존의 단일 2진수 A, B와 더불어 carry C도 더할 수 있다. 이 때 C는 이전 단계에서 발생한 carry를 의미한다. 즉 전 가산기는 3개의 입력과 두 개의 출력(sum, carry)로 이루어져 있다. 전 가산기는 입력되는 1의 수가 0, 1, 2, 3이 될 수 있는데, 입력 중 1의 개수가 0일 때는 자리 올림도 발생하지 않고 0의 합은 0이기 때문에 sum과 carry 모두 0이 된다. 1의 개수가 1인 경우에는 자리 올림이 발생하지 않고 0+0+1은 1이 되기 때문에 carry = 0, sum = 1이 된다. 1의 개수가 2인 경우에는 자리올림이 발생하기 때문에 carry는 1이 되고 sum은 0이 된다. 마지막으로 세 입력 모두 1인 경우에는 자리 올림이 발생하기 때문에 carry는 1이 되고 1을 추가로 더하기 때문에 sum도 1이 된다.

**2.**

반 감산기와 전 감산기 역시 조합 논리 회로의 한 종류이다. 감산기는 가산기와 반대로 단일 비트인 두 이진수의 뺄셈을 계산한다. 반 감산기도 반 가산기와 마찬가지로 두 개의 입력과 두 개의 출력으로 이루어져 있는데, 두 출력은 sum과 carry가 아닌 difference와 borrow이다. Difference는 두 이진수의 차이를 의미하고 borrow는 자리 내림을 의미한다. 즉 borrow는 A-B의 식을 계산할 때 A<B인 경우에 상위 비트에서 1을 빌려오는 것이라고 할 수 있다. 반 감산기의 예를 살펴보면 두 입력 A, B가 있을 때 A와 B가 (0,0)이면 자리 내림도 발생하지 않고 0-0 = 0 이므로 difference와 borrow 모두 0이 된다. A와 B가 (0,1)일 때는 B가 A보다 크기 때문에 자리 내림이 발생하고 0과 1의 차이가 1이기 때문에 difference와 borrow 모두 1이 된다. A와 B가 (1,0) 일 때는 A가 B보다 크기 때문에 자리 내림이 발생하지 않고 1과 0의 차이가 1이기 때문에 difference는 1, borrow는 0이 된다. 마지막으로 A와 B가 모두 1인 경우에는 1-1 = 0이기 때문에 자리 내림이 발생하지 않고 차이가 0이기 때문에 Difference와 borrow모두 0이 된다. 전 감산기도 전 가산기와 비슷하게 기존의 입력 A, B에 더해 이전 감산기의 borrow를 추가로 입력한다. 전 감산기의 출력은 반 감산기와 동일하게 borrow와 difference이다. 전 감산기도 반 감산기와 동일한 원리로 작동하는데 입력 중 1의 개수가 짝수일 때는 difference가 0이 되고 1의 개수가 홀수일 때는 difference가 1이 된다. 몇 가지 경우만 예로 들어 보면 입력 A, B, C가 (0,1,1)일 때 A가 B보다 작기 때문에 borrow는 1이 되고 입력 중 1의 개수가 2개이기 때문에 difference는 0이 된다. A, B, C가 (0,1,0)일 때는 A가 B보다 작기 때문에 borrow는 1이 되고 1의 개수가 1개이기 때문에 difference는 1이 된다. A,B,C가 (1,1,1)인 경우에는 A와 B의 차이는 0이 되고 이를 다시 C와 비교하면 C가 A와 B의 차이보다 크기 때문에 borrow가 1이 되고 차이는 1이 된다.

**3.**

BCD는 binary coded decimal의 약자로 십진수를 2진수로 변환하는 과정을 뜻한다. BCD 가산기는 BCD 수를 더하는 연산을 수행한다. 즉, 10진수를 2진수로 변환하여 덧셈을 수행하는데, 그렇기 때문에 두 입력 A, B가 있을 때, BCD 가산기의 범위는 0부터 18까지이다. 다만, 입력이 A, B와 CARRY가 있을 경우에는 0부터 19까지가 BCD 가산기의 범위이다. 입력이 A와 B가 있는 경우에 BCD 가산기의 동작 과정은 2진수로 변환된 A와 B를 계산한다. 예를 들어 A가 7이고 B가 8이라면 A는 0111, B는 1000이므로 A+B = 1111, 즉 15가 된다. 하지만 BCD 가산기에서는 1111로 표시하지 않고 1과 5를 나누어 1과 0101(=5)로 표시하여 출력한다. 즉 0111과 1000의 합은 10101로 출력된다.

**4.**

병렬 가산기는 1 비트보다 큰 두 이진수의 합을 계산할 수 있는 가산기이다. 병렬 가산기는 여러 개의 전 가산기로 이루어져 있는데, 첫 번째 가산기에서 A와 B가 입력되어 sum과 carry(C1)가 출력되면 C1이 다른 두 입력과 함께 두 번째 가산기의 입력이 되어 다음 계산이 수행된다. 두 번째 가산기에서 sum과 carry(C2)가 출력되면 다시 다른 두 입력과 C2가 입력되어 세 번째 가산기의 계산이 수행된다. 이러한 과정을 n번 반복하면 병렬 가산기의 결과가 출력된다. 즉 n비트 크기의 입력 A, B를 계산하기 위해서는 병렬 가산기에 N개의 전 가산기가 필요하다. 병렬 감산기도 마찬가지로 1 비트보다 큰 두 이진수의 차이를 계산할 수 있는 감산기이다. 병렬 감산기는 전 감산기와 반 감산기의 조합이나 전 감산기와 전 가산기 등으로 설계될 수 있다. 병렬 감산기도 마찬가지로 n개의 감산기/가산기의 조합으로 이루어져 있으며 이전 계산에서 발생한 carry를 다음 계산의 입력으로 사용하는 과정을 반복하면서 최종 결과를 출력한다.

**5.**

Ripple carry adder는 전 가산기의 연결로 구성되어 있다. 즉 4비트의 입력을 더하려면 4개의 전 가산기가 필요한데, 각각의 가산기는 하나의 AND와 하나의 OR 게이트로 이루어져 있다. 이러한 설계로 Ripple carry adder는 이전 전가산기의 carry가 입력되기를 기다려야 하는데 이 과정에서 딜레이가 발생한다. 따라서 Ripple carry adder는 비트의 수가 커질수록 실행시간이 늘어나는 단점이 있다. 이러한 단점을 극복할 수 있는 가산기가 Carry look ahead adder이다. Carry look ahead adder는 CLA generator가 존재하여 여러 개의 carry들을 더 적은 딜레이로 생성할 수 있다. 따라서 CLA는 ripple carry adder에 비해 딜레이 시간을 줄임으로써 더 빠르게 계산을 수행할 수 있다.

**6.**

논리 회로 중에는 디코더가 존재한다. 2개의 입력 A, B가 있는 디코더는 네 개의 출력을 나타낼 수 있다. 이를 D1, D2, D3, D4라고 할 때 D1은 A’\*B’의 출력, D2는 A’\*B의 출력, D3은 A\*B’의 출력, D4는 A\*B의 출력이다. A, B가 모두 0일 때는 D1이 1이 되고 모두 1일 때는 D4가 1이 된다. A가 0이고 B가 1일 때는 D2가 1이 되고 A가 1이고 B가 0일 때는 D3가 1이 된다. 이러한 특성을 가진 디코더는 여러 상황에 대해 출력 값을 다르게 할 수 있다는 특성이 있다. 이를 활용한 대표적인 예가 디지털 숫자이다. 입력이 4개가 있을 때 디코더가 표현할 수 있는 출력은 총 16개이다. 따라서 디코더를 사용하면 0부터 9까지 총 10개의 숫자를 4개의 입력만으로 구현할 수 있다.

**참조:**

**Field Programmable Gate Array and Applications**

(http://libproxy.sogang.ac.kr/\_Lib\_Proxy\_Url/https://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=1718987&lang=ko&site=ehost-live&ebv=EB&ppid=pp\_5\_12)