6주차 예비보고서

전공: 컴퓨터공학과 학년: 2학년 학번: 20231561 이름: 심소현

**1.**

.................

우선 가산기는 입력들의 합을 출력하는 논리 회로이다. 반가산기는 2개의 입력을 더하는 회로이며 Half Adder(HA)라고 한다. 이때 자리올림수(Carry out)과 합(Sum)으로 출력되는데 Sum은 XOR 연산의 값과 같고 Carry는 AND 연산의 값과 동일하다. 따라서 두 입력 비트의 값이 다르면 Sum에서는 1을 출력하고, 두 입력 비트의 값이 모두 1일 때만 Carry에서 1을을 출력한다. 반가산기는 Carry에 대한 입력을 따로 받지 않기 때문에 최하위 비트(LSB)에서만 사용할 수 있다. 반가산기 연산의 예시로는 입력이 A와 B로 주어진 상황에서 두 입력이 모두 1인 상황에 Sum은 0, Carry는 1의 값을 출력하는 것이다.

전가산기(Full Adder, FA)는 반가산기와 같은 입력 2개에 자리올림수의 입력을 추가하여 Carry와 덧셈의 결과를 출력하는 회로이다. 여기선 자리올림수의 입력은 이전 자리에서 전달된 Carry이다. 여기서 Sum의 출력은 두 개의 기존 입력을 XOR 연산한 값과 Carry 입력값을 XOR 연산하여 나타낸다. Carry 출력은 두 개의 기존 입력을 논리곱 출력을 받은 값과 Carry 입력과 두 개의 기존 입력의 XOR 연산값을 논리곱 출력한 값을 논리합으로 연산하여 나타낸다. 전가산기 연산의 예시로는 두 입력이 1, 0이고 Carry 입력이 1인 상황에서 Sum은 0, Carry 출력은 1인 상황이 있다.

................

**2.**

.......................

반감산기(Half Subtractor, HS)는 두 개의 비트 입력을 빼는 논리 회로이다. 출력은 차(sub)와 빌림(borrow)으로 나타난다. 두 입력이 A와 B라고 할 때, sub는 (A’\*B)+(A\*B’)의 값과 같고, borrow는 A’\*B의 값과 같다. 따라서 예시로는 0, 1의 입력이 주어졌을 때 sub는 1, borrow는 1의 출력이 나타날 수 있다.

전감산기(Full Subtractor, FS)는 바로 앞 낮은 수의 비트에 빌려 준 1을 고려하며 두 비트 입력의 뺄셈을 하는 조합 회로이다. 입력에는 두 개의 기존 A, B 입력과 빌림 입력(b)이 있고, 출력으로는 빌림(borrow)와 차(sub)가 있다. borrow는 A’\*B + A’\*b + B\*b로 계산되고, sub는 A ⊕ B ⊕ b 계산으로 이루어진다. 예시로는 A와 B가 각각 1, 0이고 b가 1일 때 borrow로 0, sub으로 0이 출력되는 경우가 있다.

.........................

**3.**

.......................

BCD(Binary Coded Deciaml) 가산기는 10진수로 코딩된 BCD 데이터를 더하는 논리 회로이다. 이는 각 10진수를 4비트의 이진수로 표현하고 더한 결과가 9를 넘으면 자릿수 초과를 방지하기 위하여 6을 추가적으로 더해 10진수의 결과를 만든다. 동작 과정은 우선 두 BCD 입력을 이진수로 변환하여 전가산기를 이용해서 더한다. 이때 결과가 10 이상이라면 6을 더하고 최종적으로 BCD 형식으로 저장한다.

.........................

**4.**

.......................

병렬 가감산기는 합과 차를 동시에 수행할 수 있는 회로이다. 가산기와 감산기의 기능을 하나로 합쳐 여러 데이터를 병렬로 처리한다. 이를 위하여 병렬 가감산기는 전가산기에 감산기를 더하는 방식으로 이루어져 XOR 게이트를 추가로 배치하였다. 동작 방식은 뺄셈을 수행할 때는 뺄셈할 값을 2의 보수로 변환하고 가산기를 통해 두 수를 더하는 것이다.

.........................

**5.**

.......................

우선 Ripple Carry Adder는 각 자리의 덧셈을 수행한 후 carry가 다음 자리로 전달되는 방식으로 작동된다. 구조는 단순하지만 앞자리에서의 carry의 전달을 기다려야 하기 때문에 속도가 느려진 수 있다. 비트의 수가 많아질수록 비효율적인 프로그램이다.

Carry Look-Ahead Adder는 자리올림을 더 효율적으로 수행한다. 자리올림을 미리 예측하여 병렬적으로 동시에 계산하기 때문에 더 빠른 덧셈이 가능하다. 다만 Ripple Carry Adder의 방식보다 구조적으로 더 복잡하다.

.........................

**6.**

.......................

기타적인 이론으로 2의 보수는 뺄셈 연산을 덧셈으로 변환하기 위한 방법이다. 주어진 이진수의 모든 비트를 반전시킨 값에 1을 더하면 구할 수 있다.

.........................