6주차 예비보고서

전공: 컴퓨터공학과 학년: 2학년 학번: 20211522 이름: 김정환

**1.**

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

가산기는 입력을 받아 수의 덧셈을 수행하여 합과 자리올림을 출력하는 회로이다. 이는 입력의 개수에 따라서 반가산기(Half Adder)와 전가산기(Full Adder)로 나눌 수 있다. 우선 반가산기는 1 Bit 두 개의 입력을 통해 합과 자리 올림을 출력하는 회로이다. XOR Gate 하나와 AND Gate 하나로 구성할 수 있으며, 각각 합인 S는 XOR Gate를 거쳐 출력되고, 자리 올림인 C는 AND Gate를 거쳐서 출력된다. 이들의 논리식은 두 입력을 A, B라고 했을 때, S=A⊕B=A’B+AB’로 나타내고, C=AB로 나타낼 수 있다. 나타낸 논리식을 바탕으로 꼭 XOR Gate가 아닌, AND, OR, NOT Gate만으로도 나타낼 수 있다.

도표, 라인, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

위와 같은 방식으로 나타낼 수 있다. 다음으로 전가산기는 3개의 입력을 통해서 합과 자리 올림을 출력하는 회로이다. 이는 두 개의 반가산기와 하나의 OR Gate로 구성된다. 여기서 합인 S는 두 번의 XOR Gate를 거쳐서 출력되고, 자리 올림인 C는 XOR Gate 한 번, 그 후에 AND 두 번과 OR Gate 하나를 거쳐 출력된다. 이를 회로로 표시하면, 아래와 같다.

도표, 폰트, 상징, 스크린샷이(가) 표시된 사진

자동 생성된 설명

그리고 S와 C를 각각 논리식으로 표현하면, 우선 1 Bit 수의 입력 2개를 각각 A, B라 하고, 자리 올림 입력을 D라 하면, S=(A⊕B)⊕D, C=(A⊕B)D+AB로 나타낼 수 있다. 이 역시 AND, OR, NOT으로 XOR Gate를 대체하여 나타내면서 표시도 가능하다.

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

**2.**

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

감산기는 입력을 받아 수의 뺄셈을 수행하여 차와 빌림 수를 출력하는 회로이다. 이 역시도 입력의 개수에 따라서 반감산기(Half Subtractor)와 전가산기(Full Subtractor)로 구분한다. 우선 반감산기는 1 Bit 두 개의 입력을 통해서 차와 빌림 수를 출력하는 회로이다. XOR Gate 하나와 NOT Gate 하나, AND Gate 하나로 구성할 수 있다. 차는 두 입력 모두 XOR Gate를 거쳐서 출력되고, 빌림 수는 한 입력만 NOT Gate를 거친 후에 AND Gate를 거쳐서 출력된다. 이를 나타낼 때 차를 D라 하고, 빌림 수를 K, 두 입력을 A, B라 하면, D=A⊕B=A’B+AB’로 나타내고, K=A’B로 나타낼 수 있다. 나타낸 논리식을 바탕으로 꼭 AND, OR, NOT Gate만으로도 표현 가능하다. 회로도는 아래와 같다.

도표, 라인, 스케치, 기술 도면이(가) 표시된 사진

자동 생성된 설명

다음으로 전감산기는 전가산기와 유사한 방식으로 구현되는데, 이 역시 3개의 입력, 즉 두 수와 빌림 수가 입력으로 들어오며 차와 빌림 수를 출력하는 회로이다. 전가산기와 비슷한 방식으로 전감산기 두 개와 하나의 OR Gate로 구현할 수 있다. 이를 논리식으로 나타내면, 두 입력을 A, B 그리고 입력으로 들어온 빌림 수를 K라 하면, 차인 D=(A⊕B)⊕K와 같이 나타낼 수 있고, 출력하는 빌림 수인 O=K(A⊕B)’+A’B로 나타낼 수 있다. XOR은 AND, OR, NOT Gate로 대체할 수 있으므로 이 방법으로 다른 식으로도 같은 역할을 만들 수 있다. 우선 위의 방식에 따른 논리회로도는 아래와 같다.

도표, 라인, 그래프, 평면도이(가) 표시된 사진

자동 생성된 설명

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

**3.**

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

BCD 가산기는 0에서 9까지의 숫자를 더하는, 즉 십진수를 이진수로 나타낸 코드들의 합의 결과를 출력하는 회로이다. BCD 코드에서 0000부터 1001까지의 코드들을 사용하며, 우선 두 수의 합이 9이하일 때는 이진수의 덧셈을 그대로 수행하여 표기한다. 예를 들어 6을 나타내는 0110과 3을 나타내는 0011의 합을 표기하면 그대로 1001로 9가 된다. 그러나 9보다 큰 결과값이 나올 때는 그대로 표기할 수 없으므로 다른 방법이 필요하다. 이는 자리 올림을 표기함으로써 해결된다. 예를 들어 8과 9를 더하는 경우 각각 1000, 1001을 더하는 것인데, 이진수 자체는 더하면 10001로 한 자리 수가 늘어나면서 잘못된 표기를 하게 된다. 이러한 표기가 아닌 자리 올림으로 0001 0111로 표기를 하는 방식으로 17이라는 결과를 나타내게 된다. 이처럼 자리 올림이 있을 경우 올림한 만큼 나타내고, 뒷자리는 그대로 0~9 사이의 수를 표기하는대로 나타낸다. 회로의 구성은 아래와 같은 그림으로 확인할 수 있다.

도표, 기술 도면, 평면도, 스케치이(가) 표시된 사진

자동 생성된 설명

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

**4.**

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

병렬 가감산기는 가산기와 감산기의 기능을 모두 가지고 있는 회로를 의미한다. 이는 전가산기에 감산기의 기능을 더하기 위해서 XOR Gate를 추가로 연결하여 병렬로 배치한 회로이다. 또한 입력을 받을 때 sign, 즉 0일 때는 가산, 1일 때는 감산을 진행할 수 있도록 구성되어 있다.

도표, 라인, 평면도, 기술 도면이(가) 표시된 사진

자동 생성된 설명

위의 구조를 보면 각각의 전가산기의 입력에 A, B가 들어가면서 B에는 입력으로 가기 전에 S와 XOR Gate를 지나는 것을 볼 수 있다. 이 경우 S가 0이면 B가 그대로 출력되어 가산기로 수행되고, S가 1인 경우에는 B와 반대인 값이 들어가서 NOT의 효과가 나타나며 전감산기로 수행된다.

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

**5.**

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

Ripple Carry Adder는 각 전가산기를 병렬로 연결할 때 출력으로 나오는 자리 올림을 다음 전가산기에 자리 올림 입력으로 전달하는 식으로 연결하는 회로를 의미한다. 이 경우에는 가장 마지막에 자리 올림을 전달받는 전가산기에서 출력을 만들기 위해서는 앞에서의 연산으로부터 모두 전달받아야 하므로 지연 시간이 생기게 된다. 해당 지연 시간을 줄이기 위해서 고안된 방법 중 하나가 Carry Look-Ahead Adder이다. 이는 자리 올림을 따로 먼저 계산해주는 방식으로 구현된다. 이는 병렬로 연결한다고 하였을 때 각각의 자리 올림 수가 어떻게 연산되는지 식을 세워봄으로써 파악할 수 있다. 우선 첫 번째 자리 올림은 D=AB + (A⊕B)C로 식을 세워볼 수 있다. 여기서 A, B는 각각 숫자 입력, C는 입력으로 들어온 자리 올림이다. 이를 다음 전가산기로 이어가며 연산해보면 모두 각 전가산기별 AB, A⊕B와 처음 들어온 C로 구성되도록 정리할 수 있는 점을 볼 수 있다. 이를 이용해 carry lookahead 블록에 복잡한 논리회로를 포함하면, 회로 자체는 복잡하지만 bit 수가 커질수록 Ripple Carry Adder보다 더 빠른 회로를 얻을 수 있다.

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

**6.**

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

Carry-skip Adder라는 또 다른 Ripple Carry Adder의 지연 시간을 줄이기 위한 방법이 있다. 이는 마지막 자리 올림의 출력이 나오는 것을 조건이 충족될 때 바로 나오도록 만들어서 지연 시간을 줄인다. 이는 각각의 두 입력의 XOR 연산을 라고 할 때 이들 모두와 처음 입력으로 주어지는 자리 올림으로 마지막 자리 올림을 구할 수 있으므로 이를 통해서 지연 시간을 단축할 수 있는 방식이다.

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