전공: 컴퓨터공학 학년: 2학년 학번: 20161603 이름: 신민준

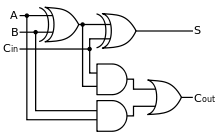
1. 전 가산기 및 반 가산기에 대해 조사하시오(예시 포함)

반 가산기(Half-adder)는 덧셈 연산을 수행하는 가산기 논리 회로의 일종이다. Half adder는 이진수에서 한 자리 수를 연산한 후, carry out에 따라 자리올림수를 출력한다.

반 가산기의 회로도는 다음과 같다.



전 가산기(Full adder)는 이진수에서 한 자리만을 연산하는데, 이 때 하위 digit의 자리올림 입력을 포함한 결과를 내놓는다. 하위 자리올림수를 상위의 자리올림수 입력으로 연결하는 cascading design을 사용하면 임의의 자리수를 가지는 이진수의 덧셈이 가능해진다. Full adder 하나는 두 개의 half adder와 하나의 OR gate로 구성된다. 이 Full adder의 회로도는 다음과 같다.



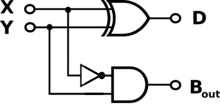
이러한 Full adder는 의 예처럼 여러 게이트들의 조합을 한 방법으로 구현될 수 있다.

결과값의 예를 들자면 이 가능할 것이다.

또한, half adder 2개를 사용해 full adder를 구현할 수도 있다. 먼저, 와 를 하나의 half adder에 연결하고, 그 출력을 두번째 half adder에 연결한다. 이 두번째 half adder에는 을 연결하면 이 출력은 가 되며, 은 두 half adder의 Carry out 값을 OR 연산함으로써 얻을 수 있다.

1. 전 감산기 및 반 감산기에 대해 조사하시오(예시 포함)

감산기(Subtractor)는 adder와 같은 방법으로 구현이 가능하다. 반 감산기(half subtractor)는 두 개의 비트 간의 뺄셈을 계산하는 논리 회로이다. 입력으로는 minuend 와 subtrahend , 그리고 출력으로는 차 와 borrow out 이 설정된다. 이 borrow-out 신호는 감산기가 다음 digit에서부터 값을 빌려와야 할 때 1이 된다. 즉, 이다. 중요한 점은 half subtractor는 를 구현하는 것이라는 점이다. 로 borrow out이 계산되기 때문에, 는 고려하지 않는다. 이는 뺄셈이 non-commutative하다는 특성에서 기인한다. 이 half-subtractor를 구현한 논리 회로도는 다음과 같다.



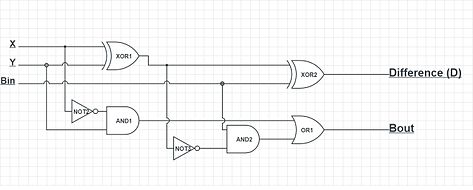
전 감산기(Full subtractor)는 세 개의 입력 비트를 받아 뺄셈을 수행한다. 이 때 사용되는 입력 신호는 다음과 같다:

출력되는 신호는 다음과 같다:

이 때 difference는 의 형태가 되므로, 이를 binary function으로 나타내면 의 예로 나타낼 수 있다.

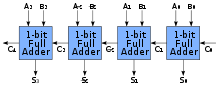
위 공식을 따라 의 경우라면, 이라는 결과가 나올 것이다.

해당 예시를 논리 회로도로 나타내면 다음과 같이 나타낼 수 있다.



1. BCD 가산기에 대해 조사하시오

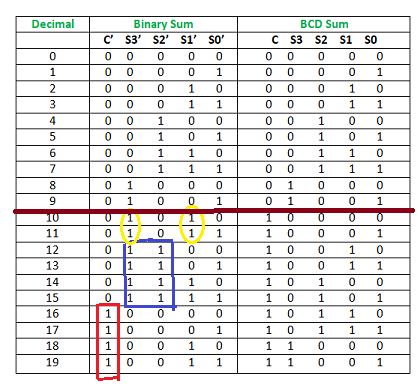
BCD는 binary coded decimal의 약자이다. 두 4비트 숫자 A와 B가 있을 때, BCD로 표현한다면 그 값은 0000(10진수로 0)부터 1001(10진수로 9)까지 표현될 수 있을 것이다.



위 도식에서 볼 수 있는 4-bit adder를 사용하면, 결과는 0부터 18까지 나올 수 있을 것이다(가 0이라는 가정 하에) 하지만, carry가 존재한다고 한다면 결과는 0~19까지 가능할 것이다. 이 방법으로 A와 B를 더하면, 결과는 binary sum이 된다. 출력 결과를 BCD의 형태로 받기 위해선, BCD Adder를 설계해야 한다.

예를 들어, 입력으로 A=0111(=7), B=1000(=8)이 들어오면, 출력인 Y=1 0101이 되는 식이다. 일반적인 binary sum을 하면 출력이 1111(=15)가 되겠지만, 이를 BCD로 출력해 1 0101이 결과값이 나와야 한다. 이를 위해선, Binary Sum 이 9를 넘는 경우, 0110(=6)을 더해주어 의 값을 설정시켜야 한다.

이 과정의 진리표를 나타내면 다음과 같다.



따라서, 위 논리를 구현하면, 1. 일 때, 2. 일 때, 그리고 3. 일 때, 결과값에 0110을 더하는 회로도를 만들면 될 것이다. 따라서,

일 때 0110을 더할 수 있도록 4-bit adder 2개를 설정하면 될 것이다. 이를 구현하면 다음과 같다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

1. 병렬 가감산기에 대해 조사하시오

병렬 가감산기(Adder-subtractor)는 2진수 값을 더하거나 빼는 회로이다. 이는 Full adder의 회로의 두번째 input에 Control signal을 XOR 시킴으로서 구현이 가능하다.

2’s complement에서, 는 로 구현된다.

따라서, control input을 D라고 할 때, B의 각 비트와 D를 XOR하고, D를 첫번째 carry인 로 넣어준다면,

이므로, 다음과 같이 기능이 구현된다.

개체이(가) 표시된 사진

자동 생성된 설명

1. Carry Look-Ahead Adder를 Ripple Carry Adder와 비교하여 설명하시오

Ripple-carry adder는 일반적인 Full adder를 원하는 비트 수 만큼 각각의 과 을 연결시켜 carry-chain을 만들고, 이를 통해 계산 결과를 얻는 adder 구현 방식이다. 이 구현 방식은 단순하기에 디자인에 시간이 오래 걸리지 않는다는 장점이 있지만, n-bit RCA는 carry chain의 존재 때문에, 그 전의 full adder에서 carry의 계산이 끝날 때 까지 기다려야 하기에 총 계산을 마칠 때 까지 의 gate delay가 생기고, 이 delay가 꽤 큰 문제가 된다는 단점이 있다.

이러한 carry chain의 문제를 해결하기 위해 제시된 방안이 Carry-lookahead adder이다. 이 adder는 각 비트에 대응하는 P(propagate)와 G(generate) 신호를 만들고, 해당 비트에 대응하는 carry를 동시에 의 gate delay에 생성하여 실행 속도를 크게 개선한다. CLA에 대한 도식은 다음에서 볼 수 있다.

시계, 개체, 벽, 탑재이(가) 표시된 사진

자동 생성된 설명

1. 기타이론

* Full subtractor의 식 에서, 의 앞에 곱해지는 2는 계산하고자 하는 값이 binary이기 때문에 곱해지는 것이다. 만약 계산하고자 하는 진수가 10진수일 경우, 이 때엔 이 값이 10이 될 것이다.
* 다수의 CLA를 Cascading하게 연결하면 더 많은 bit들에 대응하는 Adder를 구현할 수 있다. 예를 들어, 다음과 같이 16bit CLA 4개와 두 level의 LCU(Lookahead carry unit)를 사용해 64-bit Adder를 구현할 수 있다.

시계, 개체, 탑재, 벽이(가) 표시된 사진

자동 생성된 설명