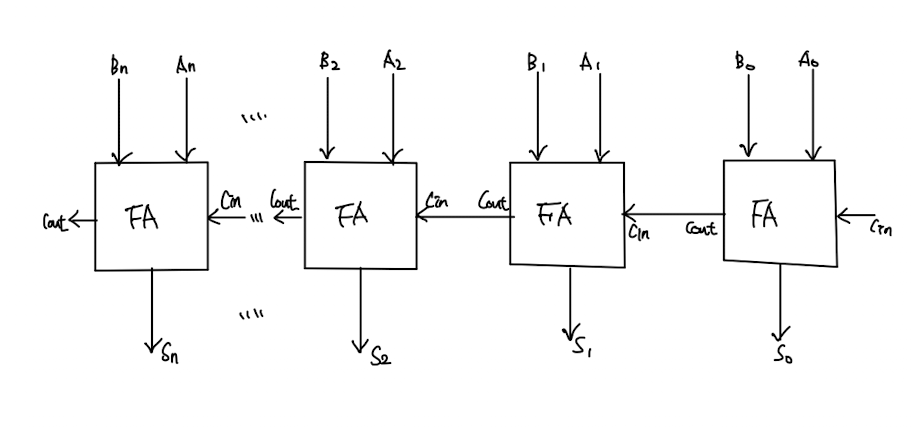
10주차 예비보고서

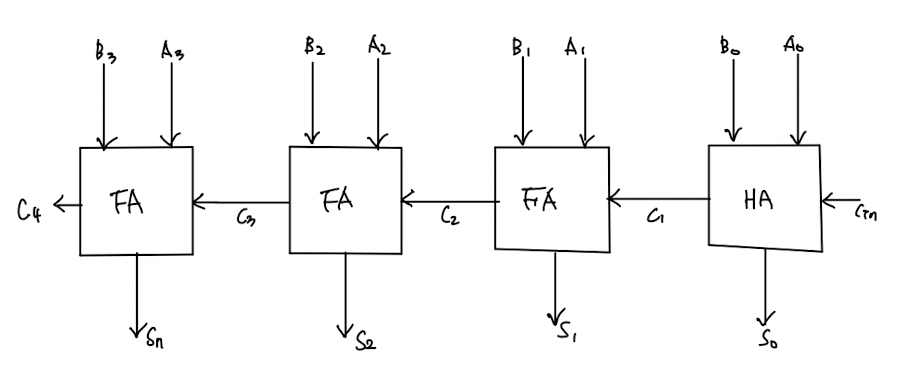
전공: 심리학과 학년: 3학년 학번: 20190345 이름: 김동현

1.

N비트의 이진수를 더하기 위해서는 1비트 당 1개의 full adder가 필요하다. 이러한 full adder를 병렬로 연결하여, 작은 자리의 비트부터 더하기 시작한다. 아래 그림은 n비트의 이진수를 더하는 n-bit adder의 회로도이다.



이전 자리의 carry ouput값이 다음 자리의 carry in값으로 전달되는 모습을 볼 수 있다. 이러한 n-bit adder를 활용하여 4-bit adder를 다음과 같이 구성할 수 있다.



4비트의 이진수 A3A2A1A0와 B3B2B1B0를 병렬로 더하는 회로로, 각각의 비트의 값과 초기 Cin값 0을 입력한다. 그 결과 두 이진수의 합 S3S2S1S0와 carry out 값이 출력된다. 하지만 이러한 병렬 가산기는 비트 수가 증가할수록 delay가 비례하여 증가한다는 단점이 있다. 이는 이전 가산기의 carry값이 계산되어야지 adder의 계산이 가능하기 때문에 앞선 adder들의 연산을 기다려야하기 때문이다.

N비트의 이진수를 빼기 위해서는 1비트 당 1개의 full subtractor가 필요하다. 이러한 full subtractor를 병렬로 연결하여 n비트의 감산기를 구현할 수 있다. 아래 그림은 4비트 감산기의 회로도이다.

텍스트, 안테나이(가) 표시된 사진

자동 생성된 설명

4비트의 이진수 A3A2A1A0와 B3B2B1B0를 병렬로 빼는 회로로, 각각의 비트 값이 입력으로 주어진다. Borrow값은 이전 값으로부터 값을 빌려오는지 여부를 나타내는 함수로, 앞선 감산기의 borrow out값이 다음 감산기의 borrow in값이 된다. 그 결과 D3D2D1D0의 값이 a-b의 값으로 출력되며, 최종 borrow값 역시 출력된다. 이러한 병렬 감산기 역시 비트 수가 증가할수록 delay가 비례하여 증가한다는 단점이 있다. 이는 병렬 가산기와 마찬가지로 이전 연산들을 통해 borrow값이 입력으로 들어와야 연산이 가능하기 때문이다.

2.

앞선 병렬 가산기, 병렬 감산기의 단점을 보완하기 위해 look ahead carry adder가 등장하였다. Look ahead carry는 carry값을 미리 계산하여 비트수가 늘어날수록 delay가 증가하는 문제점을 해결하기 위한 방법이다. CLA로 불리기도 하며, 다음과 같은 논리식을 통해 carry값을 예측한다.

=

= (generator)

= (propagate)

4비트 가산기에서 활용되는 Carry값과 Sum값을 나타내면 다음과 같다.

=

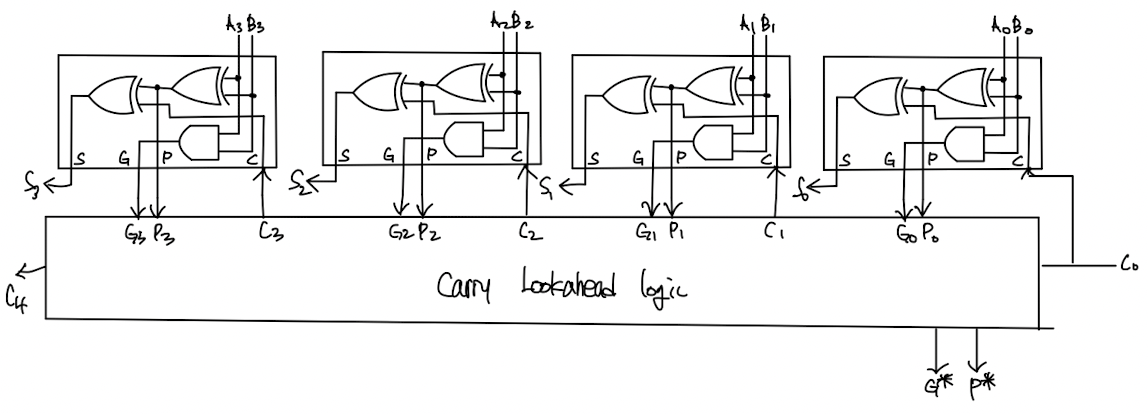
=

=

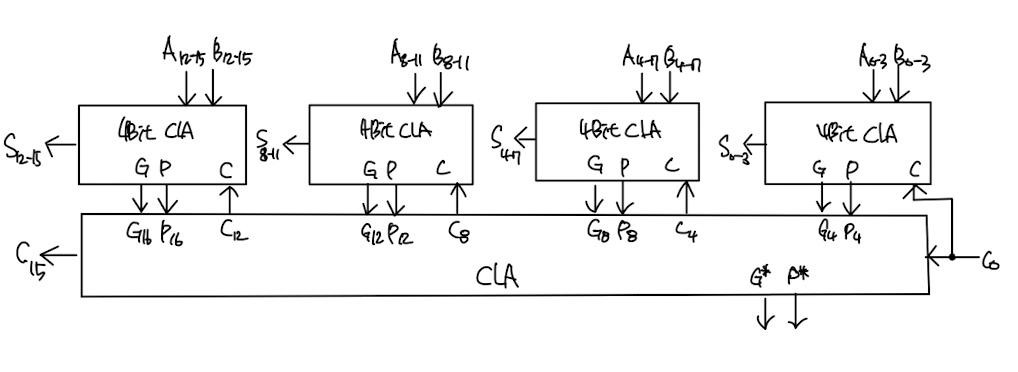
=

=

이와 같은 식을 활용하여 다음과 같은 CLA 가산기를 만들 수 있다.



뿐만 아니라 4비트의 CLA adder를 2 level로 연결하여 16비트 CLA adder를 구성할 수 있다.

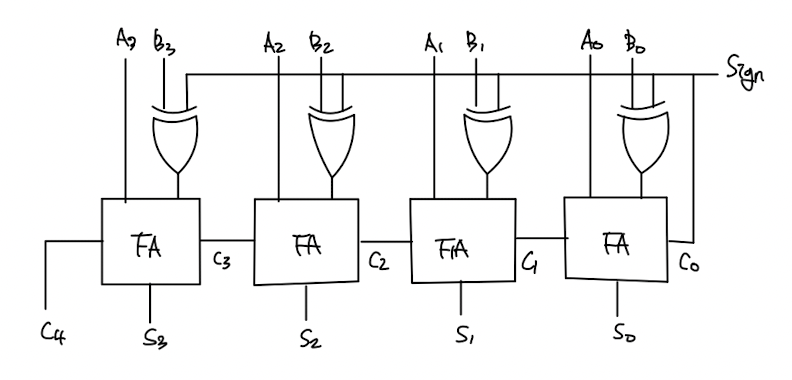


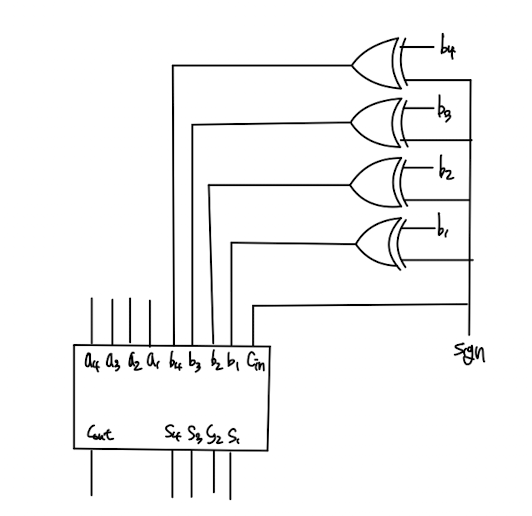
이때, G\*와 P\*는 다음과 같은 식으로 구성된다.

이러한 carry look ahead 회로는 이전 비트 계산을 기다릴 필요가 없어 delay 시간을 줄일 수 있다. 이는 회로가 빠르게 작동 가능하도록 하였다. 하지만 carry를 미리 구하기 위한 별도의 회로를 구성하므로, 비용과 파워가 증가한다는 단점이 있다. 뿐만 아니라 비트의 개수가 증가하면, Propagate값과 generator값 연산 역시 오랜시간이 걸릴 수 있다.

3.

2’s complement는 2의 보수로, 다음과 같은 과정을 통해 연산된다. 이진수의 모든 자리수를 반전시킨다. 0은 1이 되며, 1은 0이 된다. 이후 1을 더한다. XOR은 다음과 같은 특성을 가진다. 어떤 수 x와 1의 xor 연산은 x’ 이며, 어떤수 x와 0의 xor 연산은 x이다. 이러한 xor와 2의 보수의 특성을 활용하여 가감산기를 만들 수 있다.





B3B2B1B0를 더하느냐 빼느냐에 따라 연산이 달라지기 때문에 이를 결정하는 sign값과 xor연산하여 덧셈과 뺄셈 여부를 결정한다. Sign값이 1이라면 뺄셈이, 0이라면 덧셈이 수행된다. Sign값이 1이라면 b의 값이 1의 보수로 반전되어 나타나며 carry in 값으로 1이 입력되어 결과적으로 2의 보수가 된다. (2의보수 = 1의보수 +1) 이를 통해 뺄셈이 가능해진다. 반면 sign값이 0이라면 b의 값이 그대로 입력되어 full adder에 의해 가산 기능을 수행한다.

4.

BCD코드는 0부터 9까지의 정수를 표현하기 위한 코드이다. 두자리수 이상의 정수를 표현하기 위해서는 각자리수를 따로 표현해야 한다. 그렇기 때문에 덧셈 연산을 수행한다면, 10 이상의 결과값이 나왔을 때, BCD코드에 맞는 표현으로 맞추기 위한 보상회로가 필요하다. 즉, 7+3의 연산을 수행했을 때, 10으로 표현하는 것이 아닌 1과 0으로 표현해야 한다. 아래 표는 두 수를 합했을 때 나오는 10진값에 대한 2진합 결과와 bcd합의 결과를 보여준다. K와C는 각각 2진합과 bcd값의 표현 범위를 넘어서 발생한 carry값이다.

테이블이(가) 표시된 사진

자동 생성된 설명

BCD합에서 carry가 발생하는 경우는 다음과 같다. 우선 2진합의 k값이 1인 경우 c값은 항상 1이 된다. 뿐만 아니라 다음과 같은 함수로 표현가능하다.

F(Z8,Z4,Z2,Z1)=

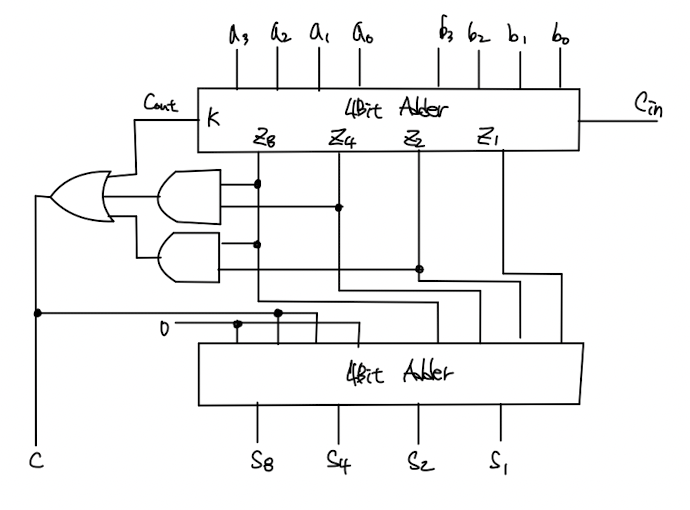
이를 카르노맵을 통해 논리식을 간소화하면 다음과 같다.

텍스트, 낱말맞추기게임이(가) 표시된 사진

자동 생성된 설명

F(Z8,Z4,Z2,Z1)=Z8Z4+Z8Z2

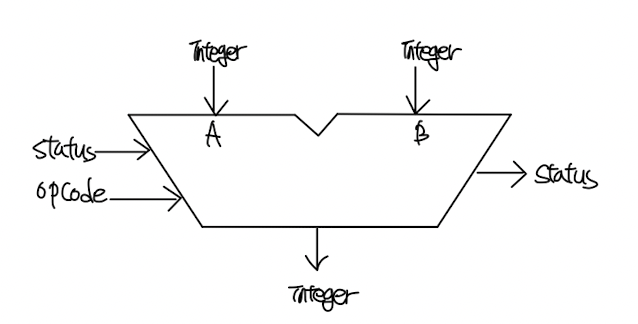
즉 C=K+Z8Z4+Z8Z2와 같이 표현 가능하다.



두 비트의 합을 구할 때, C값이 1이 되면, BCD의 상위 자릿수로 캐리가 입력된다. 이는 실제로 연산을 수행할때, 두 합의 결과가 BCD의 범위에서 벗어나면 6(0110)을 더하여 연산을 수행한다. 예를 들어 8+5 = 1000(2)+0101(2)=1101(2) 이지만 1101(2)이 BCD 범위를 벗어나므로, 1101(2)+0110(2) = 1 0011(2)= 1 3으로 표현할 수 있다.

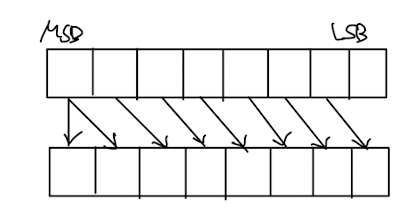
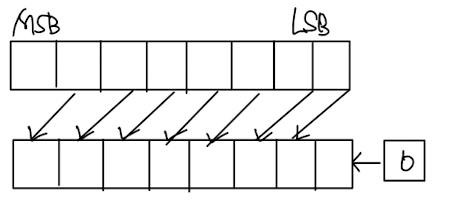
5.

ALU는 Arithmetic and Logical Unit으로, 이진수 정수에 대한 산술 및 bitwise 연산을 수행하는 회로이다. 회로는 입력으로 주어진 정수 a,b가 연산자 opcode에 의해 연산되어, 정수의 결과값을 출력하며 작동한다. Status도 입력과 출력으로 주어지는데, 이는 이전 연산 또는 현재 연산에 대한 정보를 전달하는 데 사용된다.

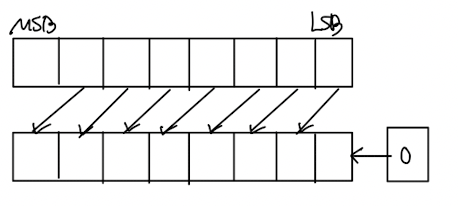


ALU는 다음과 같은 연산을 한다. 기본적인 산술 연산으로 덧셈, 뺄셈, 2의 보수, 곱셈, 나눗셈이 가능하며, bitwise 연산으로 and, or, xor, 1의 보수 연산이 가능하다. 뿐만 아니라 bit shift 연산이 가능하다.

Arithmetic shift는 비트가 방향에 따라 이동하는 것이다. 이때 왼쪽으로 이동하면 빈 자리에 0이, 오른쪽으로 이동하면 빈 자리에 기존 값 중 가장 왼쪽 값으로 채워진다.



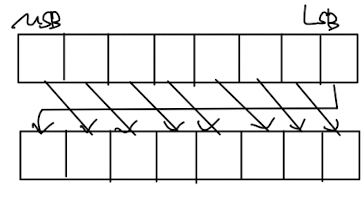
Logical shift는 논리 이동으로, 방향에 따라 비트가 이동한다. 앞선 arithmetic shift와 비교하면, 왼쪽으로 이동할 때는 동일하지만, 오른쪽으로 이동할 때에는 빈 자리가 0으로 채워진다는 점이다.

쇼지이(가) 표시된 사진

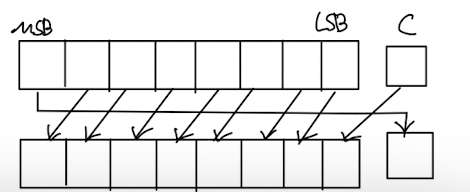
자동 생성된 설명

Rotate는 왼쪽으로 이동할 때, 가장 왼쪽의 비트가 가장 오른쪽의 비트가 되고 나머지 비트는 왼쪽으로 한칸씩 이동한다. 반면 오른쪽으로 이동할 때에는 가장 오른쪽의 비트가 가장 왼쪽의 비트가 되고 나머지 비트는 오른쪽으로 한칸씩 이동한다.

쇼지이(가) 표시된 사진

자동 생성된 설명

Rotate through carry는 rotate연산을 수행하되, 왼쪽으로 회전할 때에는 가장 오른쪽 값이 carry값으로 채워진다. 오른쪽으로 회전할 때에는 가장 왼쪽 값이 carry값으로 채워진다.

쇼지이(가) 표시된 사진

자동 생성된 설명

6.

Carry lookahead logic의 한 종류인 ic74182에 대해 조사하였다. Ic 74182는 다음과 같은 논리회로를 가지고 있다.

선화이(가) 표시된 사진

자동 생성된 설명

7. 참고문헌

“한권으로 읽는 컴퓨터 구조와 프로그래밍”, 조너선 스타인하트

Alan B. Marcvitz, Introduction to Logic Design, McGraw-Hill(2010)