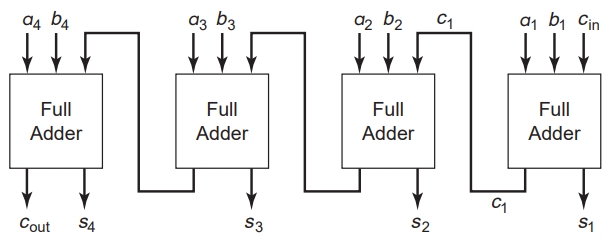
10주차 예비보고서

전공: 기계공학과 학년: 3학년 학번: 20191820 이름: 김형준

**1.**

- 4bit Binary Parallel Adder

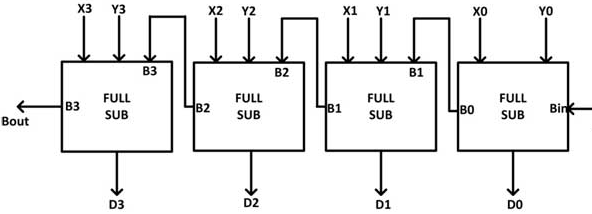


4bit Binary Parallel Adder(이진 병렬 가산기)는 두 개의 4비트 이진수를 입력으로 받아서 덧셈 연산을 수행하는 조합 회로이다. 4bit Ripple Carry Adder이라고도 하며, 4개의 1bit 전가산기(Full Adder)를 병렬로 연결하여 만들 수 있다. 이진 병렬 가산기를 구성하는 각각의 전가산기는 이전 자릿수의 올림수 출력을 다음 자릿수의 올림수 입력으로 연결하는 방식으로 연결되어 있다. 전가산기는 3 input, 2 output 구조로, 입력으로 , , 을 받아서, 출력으로 합()과 올림수()를 연산하는 회로이다. 출력 값들의 논리식은 다음과 같다.

이때, LSB의 전가산기의 입력캐리()는 하위비트에서 들어오는 올림수가 없기 때문에 항상 0이 된다. 또한, MSB의 출력캐리()은 두 이진수(부호가 없는 경우)의 합이 4bit을 넘을 때 1을 갖고 (Overflow 발생), 넘지 않는 경우 0을 갖는다.

이진 병렬 가산기는 상위 비트의 합과 올림수를 계산하기 위해서는 하위 비트의 올림수가 입력으로 들어와야 하므로, 상위 비트의 전가산기는 하위비트의 전가산기가 올림수를 연산할 때까지 대기해야 한다. 따라서 이진 병렬 가산기는 LSB부터 MSB의 순서로 차례대로 연산이 수행되며, 두 이진수의 비트수가 늘어날수록 연산시간이 선형적으로 증가한다.

- 4bit Binary Parallel Subtractor



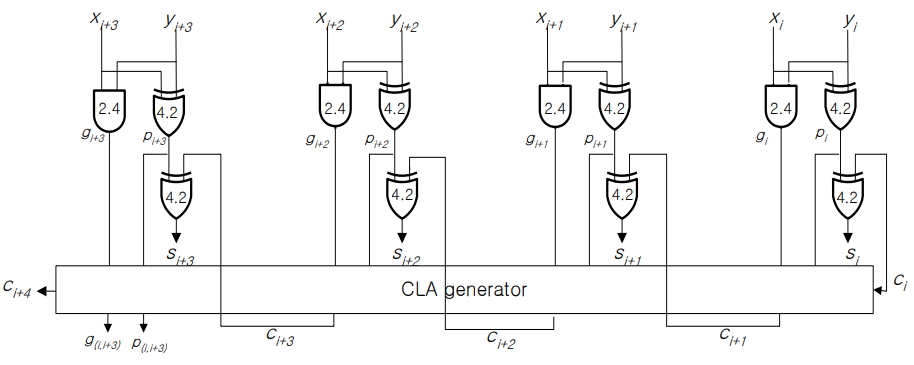
4bit Binary Parallel Subtractor(이진 병렬 감산기)는 두 개의 4비트 이진수를 입력으로 받아서 뺄셈 연산을 수행하는 조합 회로이다. 4개의 1bit 전감산기(Full Subtractor)를 병렬로 연결하여 만들 수 있다. 이진 병렬 감산기를 구성하는 각각의 전감산기는 이전 자릿수의 빌림수 출력을 다음 자릿수의 빌림수 입력으로 연결하는 방식으로 연결되어 있다. 전감산기는 3 input, 2 output 구조로, 입력으로 , , 를 받아서, 출력으로 차()과 빌림수()를 연산하는 회로이다. 출력 값들의 논리식은 다음과 같다.

이때, LSB의 전가산기의 입력 빌림수()는 하위비트에서 들어오는 빌림수가 없기 때문에 항상 0이 된다. 또한, MSB의 출력 빌림수()은 두 이진수(부호가 없는 경우)의 차가 음수인 경우 1을 갖고 (Overflow 발생), 음수가 아닌 경우 0을 갖는다.

이진 병렬 감산기는 상위 비트의 차와 빌림수를 계산하기 위해서는 하위 비트의 빌림수가 입력으로 들어와야 하므로, 상위 비트의 전감산기는 하위비트의 전감산기가 빌림수를 연산할 때까지 대기해야 한다. 따라서 이진 병렬 감산기는 LSB부터 MSB의 순서로 차례대로 연산이 수행되며, 두 이진수의 비트수가 늘어날수록 연산시간이 선형적으로 증가한다.

**2.**

Look Ahead Carry는 Ripple Carry Adder에서 발생하는 Gate Delay를 줄이기 위해 사용하는 방법이다. 이 방법을 사용해 만든 가산기를 Carry Look Ahead Adder(CLA)라고 한다.



4bit Carry Look Ahead Adder는 4bit Ripple Carry Adder처럼 두 개의 4비트 이진수를 입력으로 받아서 덧셈 연산을 수행하는 조합 회로이고, 4bit Ripple Carry Adder에서 Carry를 연산하는 부분이 그림과 같이 4bit CLA generator로 교체된 구조로 되어있다.

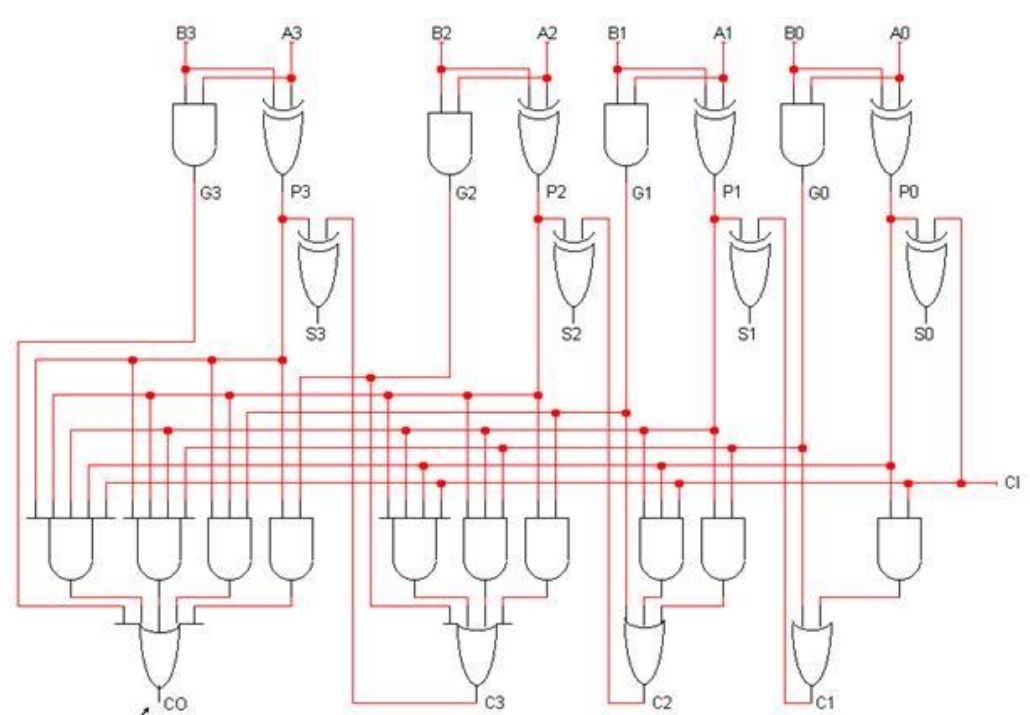
Carry Look Ahead Adder에서는 불필요한 Gate Delay가 생기는 것을 막기 위해 다음과 같은 (carry-propagate function)과 (carry-generate function)를 정의해 모든 자릿수의 캐리를 한 번에 미리 계산한다.

위와 같은 와 를 사용하면 합과 캐리를 구하는 식은 =, 이 되고,

모든 자릿수에 대한 캐리를 를 재귀적으로 전개하여 나타내면 다음과 같이 , , 에 대해서 나타낼 수 있다.

위와 같은 식을 사용해 CLA generator 내부에서 2단계로 구성된 AND-OR 게이트를 통과해 2Δ만에 모든 자리의 캐리를 계산할 수 있다.

이를 논리회로로 나타내면 아래와 같다.



이러한 방법을 통해 Carry Look Ahead Adder는 모든 자릿수의 캐리를 한번에 미리 계산할 수 있어 상위 비트의 가산기가 하위 비트의 캐리값이 계산되는 것을 기다릴 필요가 없으므로 모든 자리의 연산이 동시에 진행된다.

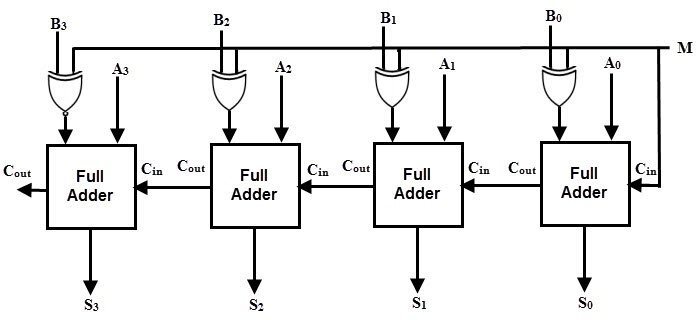
따라서 불필요한 Gate Delay가 사라졌으므로 Ripple Carry Adder보다 효율적인 연산이 가능하다.

그러나, 연산하는 이진수의 비트수가 늘어날수록 CLA Generator의 내부 회로가 복잡해지는 단점이 있어서 큰 비트수의 CLA의 경우, 4bit 또는 8bit 단위로 끊어서 복층으로 설계한다. (hierarchical CLA)

**3.**

이진법의 뺄셈은 2‘s complement(2의 보수법)을 사용하여 다음과 같이 덧셈으로 바꿀 수 있다.

이 성질을 이용해 Ripple Carry Adder에 입력되는 이진수의 비트 수만큼 XOR 게이트를 추가하여 선택 신호(M)로 덧셈과 뺄셈을 선택해서 연산하는 회로를 Ripple Carry Adder-Subtractor라고 한다.



위는 4bit Ripple Carry Adder-Subtractor의 회로도 이고, 그림에서 M이라고 표시되어있는 선택 신호가 LSB의 입력캐리와 추가된 XOR 게이트의 입력에 각각 연결한 구조로 되어있다. 이때, 입력된 두 이진수에 대해 M의 값이 0이면 덧셈을, 1이면 뺄셈을 수행한다.

M=0인 경우, LSB의 입력캐리의 값이 0이되고, 각각의 입력값 가 XOR 게이트를 그대로 통과하게 되므로() 회로의 결과값은 이 되어 두 이진수의 합을 계산하게 된다. 반면에 M=1인 경우 LSB의 입력캐리의 값이 1이되고, 각각의 입력값 가 XOR 게이트를 지나면서 반전되므로() 회로의 결과값은 이 되어 2의 보수법에 의해 두 이진수의 차를 계산하게 된다.

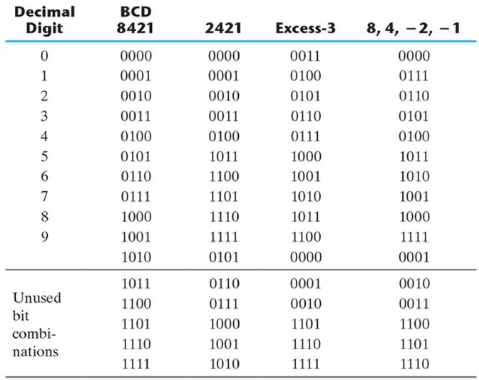
**4.**

BCD(Binary Coded Decimal)는 이진화 십진수라고 불리며, 4비트의 2진수로 십진수 한자리를 나타내는 기수법을 의미한다.

두자리 이상의 10진수를 BCD 코드로 나타내면, 각 자릿수를 4비트의 2진법 코드로 각각

나타낸다. BCD 코드에는 아래와 같이 여러가지 종류가 있으며, 각각의 4비트의 값 중

0~9를 나타내는 10가지의 값은 사용되고, 나머지 6개의 값은 사용되지 않는다.



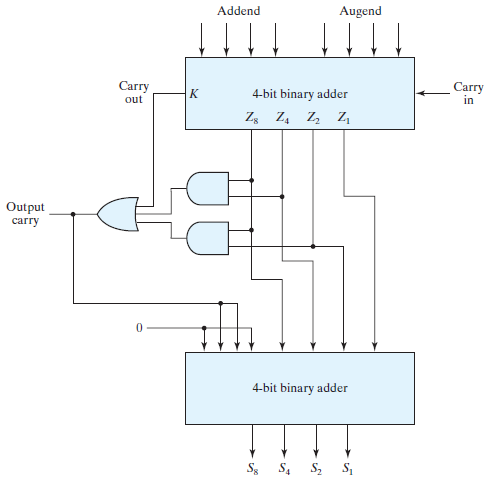
BCD 연산은 이러한 BCD 코드로 나타낸 십진수의 두 수를 더하는 연산이다.

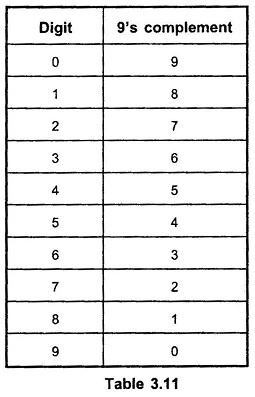
BCD 코드에 대해 BCD 연산을 수행하는 가산기를 BCD 가산기라 하며, 한자리 수의 BCD 코드의 덧셈을 수행한다.

BCD 가산기의 덧셈은 ‘BCD값 + BCD값 + 아랫단으로부터의 캐리’의 형태로 2진수 덧셈 방법대로 더하는 방식으로 진행되며, 합의 최대값은 9+9+1=19이다.

이때, 합이 9이하인 경우는 그대로 한자리 수의 BCD 코드로 나타낼 수 있으므로 별도의 변환없이 그대로 나타내고 (결과값에 0을 더함),

합이 10 이상인 경우 합을 한자리의 BCD 코드로 나타낼 수 없으므로 2진수의 합에 (6)을 더한 다음 발생한 캐리를 앞의 자릿수의 BCD코드의 최하위 비트에 캐리로 올려준다.



[BCD 가산기의 회로도]

BCD 코드의 뺄셈은 9의 보수법을 사용해 수행하는데, 빼는 수를 9의 보수법 상의 음수로 변환한 다음, 두 개의 BCD 코드의 덧셈을 수행한다.

그 다음 캐리가 발생했다면 결과값에 캐리를 더하고, 발생하지 않았다면 결과값의 9의 보수를 구한다.

예를들어 (8)-(2)=(6)의 경우,

(8)+(7)=(캐리) (5)가 되고, 캐리가 발생했으므로 결과값에 더하면 (5)+(1)=(6)이 된다.

또한, (8)-(8)=(0)의 경우,

(8)+(1)=(캐리) (9)가 되고, 캐리가 발생하지 않았으므로 결과값을 9의 보수로 변환하면 (0)이 된다.

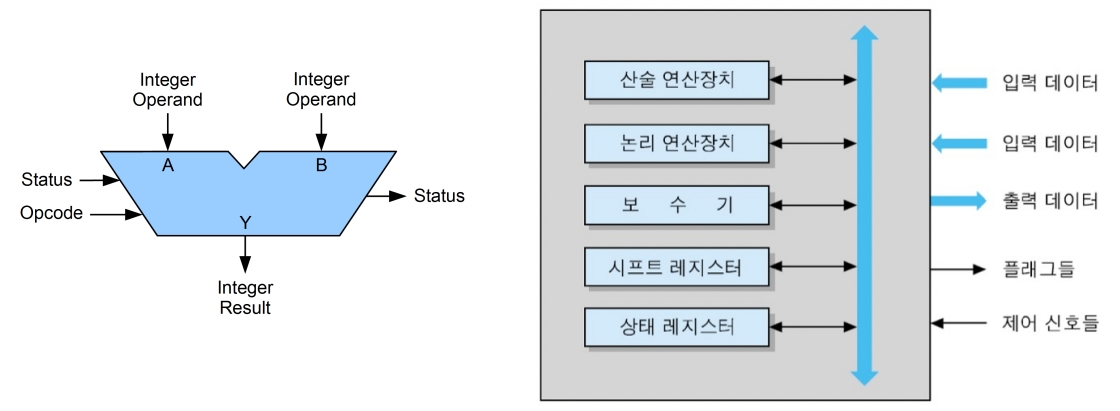
0~9의 숫자에 대해 대응되는 9의 보수의 값은 위쪽의 표와 같다.

**5.**

ALU (Arithmetic and Logic Unit)란 산술 논리 장치로 CPU 내부에 존재하며, 덧셈, 뺄셈, 곱셈, 나눗셈과 같은 두 숫자의 산술 연산과 논리합, 논리곱과 같은 논리 연산을 담당하는 장치이다.

ALU는 제어신호에 따라 레지스터 또는 기억장치로부터 데이터를 가져와 연산을 하며, 연산 결과를 상태 레지스터에 저장하게 된다.

ALU의 구성요소는 가산기, 시프터, 레지스터, 보수기, 오버플로 검출기, 누산기 등으로 구성되어있다.



보수기는 컴퓨터에서 감산을 할 때 빼는 수를 보수로 바꿔서 가산기에 입력하여 덧셈을 수행하게 만든다. 보수기에서 사용하는 보수는 1의 보수와 2의 보수가 있는데, 일반적으로 2의 보수기를 많이 사용한다.

오버플로 검출기는 여러가지 연산을 수행할 때 연산결과를 레지스터에 모두 저장할 수 없는 경우(오버플로 발생) 에러 발생 여부를 검사하여 사용자에게 알려준다.

시프터(Shifter)는 2진수의 각 자리를 왼쪽 또는 오른쪽으로 이동(Shift)해주는 회로이다.

레지스터는 어떤 데이터를 일시적으로 기억하는 장치로, 그 기능에 따라 여러가지 이름이 붙여진다. 대표적으로, 중앙 처리 장치 내의 제어 장치에 의해 사용되는 명령 레지스터와 프로그램 카운터가 있으며, 이들은 프로그램 실행의 제어를 위해 사용된다.

누산기(Accumulator)는 산술연산과 논리연산의 중간 값을 임시적으로 보관하기 위한 레지스터이다.

데이터 레지스터는 연산을 위한 데이터를 일시적으로 저장하는 레지스터이고, 저장 레지스터는 주 기억장치로 전송되는 데이터를 임시로 저장하는 레지스터이다.

**6.**

- FPU(Float Processing Unit)

ALU는 기본 정수에 대한 연산을 수행하는 장치이다. 그러나 CPU에서는 부동소수점 연산도 수행하는데, 이 경우 소수를 표현하기 위해 소프트웨어적인 방법으로 소수를 정수 데이터로 구현해 일반 정수보다 수천 번 이상의 계산을 더 해야하므로 마이크로 프로세서에 과도한 부담을 주어 전체적인 컴퓨터의 처리속도를 떨어뜨리게 된다.

고성능 컴퓨터는 이러한 부동소수점 연산으로 인해 성능저하를 막기 위해 소수 계산만을 전담하는 장치 FPU(Float Processing Unit)를 따로 장착하고 있다.

부동소수점 연산장치가 마이크로 프로세서에 내장된 형태를 FPU(Float Processing Unit)이라고 부르며, 별도의 마이크로 프로세서의 형태인 경우 Numeric Co-Processor라고 부른다.

FPU는 부동소수점 실수에 대한 사칙연산을 지원하며, 모듈에 따라 거듭제곱, 삼각함수 등의 다양한 수학계산을 할 수 있다.

현재 대부분의 PC에서는 FPU를 채택하고 있으며, 평상시에는 수행하지 않다가 그래픽 이미지 처리나 표현 등과 같은 특별한 일을 수행할 때에만 FPU가 작동한다.