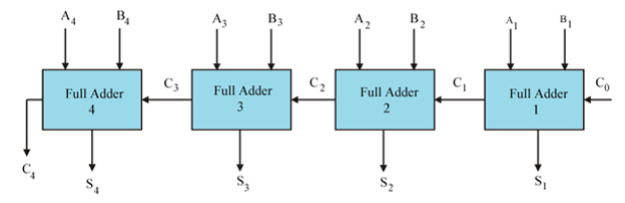
**10주차 발표 준비**

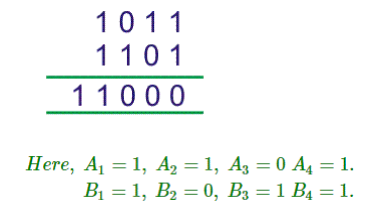
**4 bit binary parallel adder**

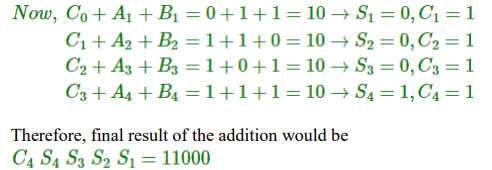
Full Binary Adder는 두 개의 이진 숫자에서 동일한 자리수의 비트와 이전 비트 덧셈에서 나온 캐리(Carry)를 더하는 연산을 수행합니다. 그러나 단일 전체 덧셈기는 한 번에 한 비트만 덧셈할 수 있습니다. 여러 비트를 동시에 더하려면, 추가할 비트 수만큼 여러 개의 전체 덧셈기를 연결해야 합니다. 이 방식은 여러 비트를 동시에 처리하기 위해 전체 덧셈기를 병렬로 연결하는 방법을 사용하며, 이를 이진 병렬 덧셈기 (Binary Parallel Adder)라고 부릅니다.

예를 들어, 두 개의 4비트 이진 수를 더하려면, 4개의 전체 덧셈기를 병렬로 연결하여 각 비트의 덧셈을 동시에 처리해야 합니다.

<https://www.sdbidoon.com/document/digital-electonics-and-microprocessor-04-04-2020.pdf>

위 회로의 정당성을 4비트 이진수 두 개를 더하는 예를 들어 살펴보겠습니다. 두 이진수 1011과 1101을 더한다고 가정해 봅시다. 여기서 이전 캐리가 없으므로, C0 = 0입니다.



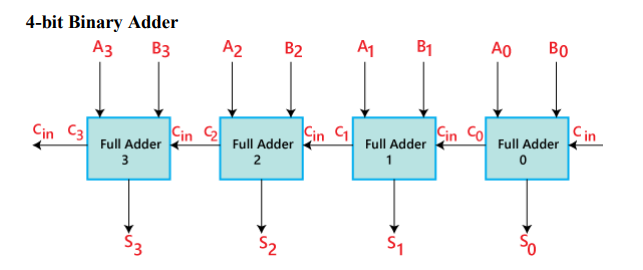


2.3 이진 덧셈기

이진 덧셈기는 두 개의 이진수에 대해 덧셈 연산을 수행하는 논리 회로입니다. 이진 덧셈기는 풀 애더(Full-Adder) 회로를 사용하여 구성됩니다. 풀 애더들은 직렬로 연결되며, 첫 번째 애더의 출력 캐리는 두 번째 애더의 입력 캐리로 처리됩니다.

2.3.1 N-비트 병렬 덧셈기

풀 애더는 캐리 입력을 가진 두 개의 단일 비트 이진수를 더하는 데 사용됩니다. 디지털 계산에서는 단일 비트 이진수뿐만 아니라 두 개의 N-비트 이진수를 더할 필요가 있습니다. 이를 위해 N-비트 병렬 덧셈기를 사용해야 합니다. N-비트 병렬 덧셈기를 얻기 위해 풀 애더를 N개 직렬로 연결합니다. 첫 번째 애더의 캐리 출력은 두 번째 애더의 캐리 입력으로 처리됩니다.



'A'와 'B'는 각각 피가산수(augend)와 가산수(addend) 비트이며, 아래첨자 번호로 정의됩니다. 아래첨자는 오른쪽에서 왼쪽으로 시작하며, 최하위 비트는 아래첨자 '0'으로 정의됩니다.

C0, C1, C2, C3는 풀 애더를 사용하여 연결된 캐리 입력입니다. C4는 마지막 풀 애더에서 생성된 캐리 출력입니다.

첫 번째 애더의 Cout(캐리 출력)은 두 번째 풀 애더의 Cin(캐리 입력)으로 연결됩니다.

S0, S1, S2, S3는 가산수와 피가산수 비트의 합을 생성하는 합 출력입니다.

입력 변수 'A'와 'B'의 입력은 서로 다른 소스 레지스터에서 가져옵니다. 예를 들어, 입력 변수 'A'의 비트는 레지스터 'R1'에서, 입력 변수 'B'의 비트는 레지스터 'R2'에서 가져옵니다.

두 입력 변수를 더한 결과는 세 번째 레지스터나 소스 레지스터 중 하나에 저장됩니다.

**4 bit binary parallel adder**

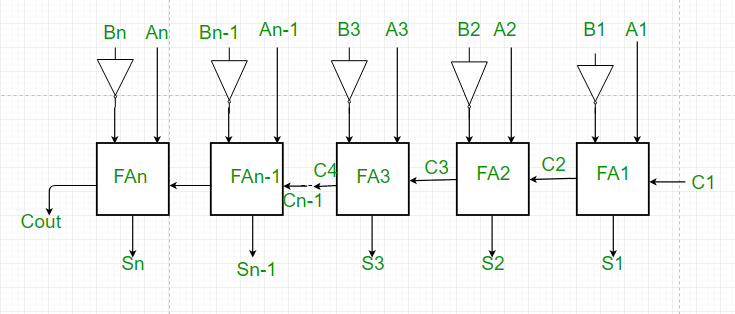
병렬 감산기 (Parallel Subtractor)는 두 개의 이진수에 대해 여러 비트 길이를 가진 산술 차이를 구할 수 있는 디지털 회로로, 각 비트 쌍을 병렬로 처리하여 차이를 계산합니다. 병렬 감산기는 여러 가지 방법으로 설계될 수 있으며, 그 중 일부 방법은 다음과 같습니다:

반감산기(Half Subtractor)와 풀 감산기(Full Subtractor)의 조합

모든 풀 감산기(Full Subtractor) 사용

모든 풀 애더(Full Adder) 사용, 빼는 수를 보수(complement)로 입력

이러한 방법을 통해 병렬 감산기는 효율적으로 두 이진수의 차이를 계산할 수 있습니다.



병렬 감산기의 동작 원리

그림에서 볼 수 있듯이, 병렬 이진 감산기는 모든 풀 애더(Full Adder)를 사용하여 빼는 수(subtrahend)의 보수(complement)를 입력으로 처리하는 방식으로 구성됩니다. 이 연산은 피감수(minuend)와 빼는 수(subtrahend)의 2의 보수를 더하는 것이 빼기 연산과 같다는 원리를 기반으로 합니다.

동작 과정:

1의 보수 구하기 : 먼저, 빼는 수 B의 1의 보수는 NOT 게이트를 사용하여 구합니다. 즉, B의 각 비트를 반전시켜 1의 보수를 구합니다.

2의 보수 구하기 : 그 후, 1의 보수에 1을 더하여 B의 2의 보수를 계산합니다. 이 2의 보수는 빼는 수로 사용됩니다. 이때 발생하는 캐리(Carry)는 풀 애더에서 처리됩니다.

덧셈을 통한 뺄셈 수행 : 이제 피감수 A와 빼는 수 B의 2의 보수를 더하는 방식으로 산술적 뺄셈이 수행됩니다. 각 비트에 대해 풀 애더가 A의 비트와 B의 2의 보수 비트를 더하고, 발생한 캐리는 다음 풀 애더로 전달됩니다.

마지막 비트와 캐리 출력 : 마지막 풀 애더(FAn)는 마지막 비트의 입력 An과 Bn의 2의 보수를 더하여 결과를 구합니다. 이때 마지막 캐리 비트(Cout)는 출력으로 생성됩니다.

요약:

병렬 감산기는 풀 애더를 사용하여 2의 보수 덧셈을 통해 두 이진수의 뺄셈을 수행합니다.

1의 보수를 구한 후 1을 더하여 2의 보수를 만들고, 이를 피감수에 더함으로써 산술적 차이를 계산합니다.

각 풀 애더는 입력과 이전 애더의 캐리 출력을 받아 계산을 수행하며, 최종적으로 마지막 캐리 비트(Cout)가 출력됩니다.

<https://www.geeksforgeeks.org/parallel-adder-and-parallel-subtractor/>

**BCD Adder**

BCD 덧셈기는 두 개의 이진화된 10진수(BCD, Binary-Coded Decimal) 숫자를 더하는 회로입니다. BCD는 각 숫자를 4비트 이진 코드로 표현하는 방식으로, 0부터 9까지의 숫자는 각기 고유한 4비트 값으로 나타냅니다. BCD에서는 가장 중요한 비트(MSB, Most Significant Bit)가 항상 0이기 때문에, 각 숫자는 0에서 9까지의 범위로 제한됩니다.

BCD 덧셈기의 동작 원리:

BCD 덧셈기는 4개의 풀 애더(Full Adder) 회로를 직렬로 연결하여 구성됩니다. 각 풀 애더는 두 개의 BCD 숫자와 이전 단계에서 발생한 캐리(Carry-In)를 고려하여 덧셈을 수행합니다. 각 풀 애더는 합(Sum) 비트와 캐리 아웃(Carry-Out) 비트를 출력합니다. 이 캐리 아웃 비트는 다음 풀 애더의 입력 캐리로 전달됩니다.

동작 과정:

입력: 두 개의 BCD 숫자(각각 4비트씩)와 이전 단계에서 전달된 캐리 비트가 입력으로 들어옵니다.

풀 애더 동작: 각 풀 애더는 두 BCD 숫자와 이전 단계의 캐리 입력을 더하고, 합(Sum) 비트와 캐리 아웃(Carry-Out) 비트를 생성합니다.

출력: 각 풀 애더에서 생성된 합 비트는 최종적으로 BCD 덧셈기의 결과로 출력됩니다. 또한, 최종 풀 애더에서 생성된 캐리 아웃 비트는 덧셈 결과가 9를 초과할 경우 추가적인 보정을 수행하기 위해 사용될 수 있습니다.

보정: BCD 덧셈에서는 만약 덧셈 결과가 10 이상일 경우, 6(0110)을 더하여 보정해야 할 수도 있습니다. 이 보정은 각 풀 애더가 9를 넘는 값을 처리할 수 있도록 돕기 위해 추가적인 조치로 필요합니다.

요약:

BCD 덧셈기는 4개의 풀 애더를 직렬로 연결하여 두 BCD 숫자에 대해 덧셈을 수행합니다.

각 풀 애더는 입력된 BCD 숫자와 캐리 비트를 더한 후, 합 비트와 캐리 아웃 비트를 출력합니다.

덧셈 결과가 9를 초과하면 보정이 필요하며, 이 경우 추가적인 처리가 이루어질 수 있습니다.

BCD 덧셈기를 설계하는 과정은 몇 가지 중요한 단계를 거쳐야 합니다. 아래는 BCD 덧셈기를 설계하기 위한 주요 단계입니다:

1. 숫자 자릿수 파악 (Find Number of Digits)

설계할 덧셈기가 지원해야 할 BCD 자릿수를 결정합니다. BCD 숫자 하나는 4비트로 표현됩니다.

예를 들어, 2자리 BCD 숫자를 더하려면 각 숫자는 4비트이고, 총 2개의 4비트 숫자를 다뤄야 하므로 총 8비트가 필요합니다.

따라서, 추가하려는 BCD 자릿수에 따라 몇 개의 풀 애더가 필요한지 계산합니다.

2. 덧셈기 구조 설계 (Adder Structure)

\*\*풀 애더(Full Adder)\*\*를 직렬로 연결한 구조를 일반적인 덧셈기 구조로 선택합니다.

추가적인 풀 애더가 필요하면, 하나의 BCD 자릿수 덧셈과 이전 단계에서 발생한 캐리 비트를 처리하는 방식으로 구성됩니다.

각 풀 애더는 두 BCD 숫자 비트와 이전 단계에서 생성된 캐리 비트를 더하고, 합과 캐리를 출력합니다.

3. 풀 애더 회로 구현 (Full-Adder Circuit Implementation)

두 개의 4비트 BCD 숫자와 캐리 입력을 더할 수 있는 풀 애더 회로를 구현합니다.

풀 애더는 각 비트를 더한 후 합(Sum) 비트와 캐리 아웃(Carry-Out) 비트를 출력합니다.

각 단계에서 캐리 아웃 비트는 다음 단계로 전달되어 덧셈을 이어갑니다.

4. 풀 애더 연결 (Interconnect the Full-Adders)

풀 애더들을 직렬로 연결합니다.

각 단계에서 생성된 캐리 아웃은 다음 단계의 캐리 인으로 전달됩니다.

덧셈이 진행될 때마다 각 풀 애더는 이전 단계에서 전달된 캐리 비트를 더하여 최종 결과를 생성합니다.

5. BCD 보정 제공 (Provide BCD Correction)

두 BCD 숫자의 합이 9(1001)보다 클 경우, BCD 보정이 필요합니다. 이 경우 0110을 더하여 결과를 보정합니다.

보정이 필요한지 여부를 검사하는 로직을 구현하여, 덧셈 결과가 9를 초과하는지 확인합니다.

만약 덧셈 결과가 9보다 크다면, 6(0110)을 더해주고, 새로운 캐리 비트가 다음 단계로 전달됩니다.