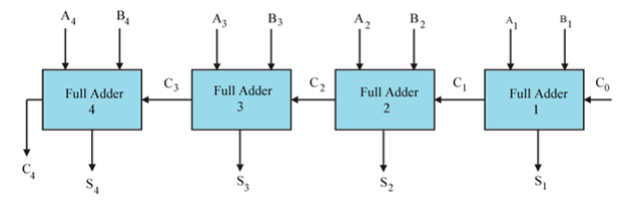
10주차 예비보고서

전공: 컴퓨터공학과 학년: 2학년 학번: 20231561 이름: 심소현

**1.**

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

이진 병렬 덧셈기는 여러 비트를 동시에 처리하기 위해 Full Adder를 병렬로 연결하는 방법이라고 볼 수 있다. 예를 들어, 두 개의 4비트 이진수를 더하려면 4개의 Full Adder를 병렬로 연결하여 각 비트의 덧셈을 동시에 처리해야 한다. 아래의 그림과 같이 연결하여 결과물은 C4S4S3S2S1이라고 볼 수 있다.

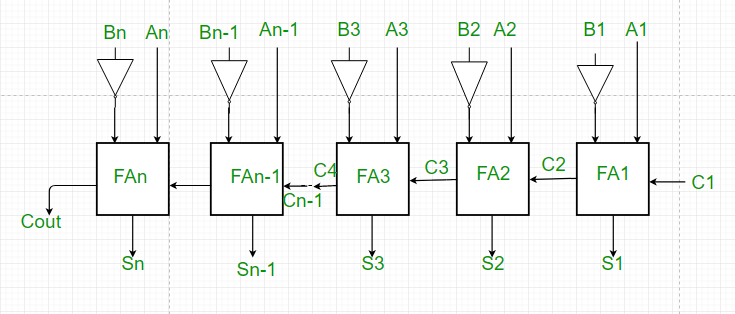


................

**2.**

A와 B는 각각 더하는 비트이며 C는 캐리 입출력이다. 첫 번째 덧셈기의 캐리 출력은 두 번째 덧셈기의 캐리로 이어지는 등 이전의 캐리 발생은 다음의 캐리 입력이 된다.

이진 병렬 감산기는 두 개의 이진수에 대해 여러 비트 길이를 가진 차이를 각 비트쌍을 병렬로 처리하여 결과를 구하는 회로이다. 회로의 구성은 아래와 같다.

병렬 감산기에서도 Full Adder를 사용하는데, 2의 보수 덧셈 원리를 사용한다. 이진수 계산에서 단순 감산의 결과는 빼는 값을 2의 보수로 만들어 더한 결과와 같다. 따라서 빼는 값을 1의 보수로 만들고 1을 더해 2의 보수로 만들면, 이를 피감수에 더해 Full Adder를 사용할 수 있게 된다. 2의 보수로 만든 후부터는 병렬 덧셈기와 작동 방식은 같다.

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

**2.**

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

Look ahead carry는 예측한 자리올림수라고 직역할 수 있다. 일반적인 Adder의 연결에서는 캐리를 계산하고 이를 다음 Adder에 입력하여 계산을 시작하는 방법을 사용하여 연산의 속도가 느리다. 그러나 Look ahead carry를 사용하면 모든 캐리를 예측 계산하여 속도의 지연을 줄일 수 있다. 이를 위해 신호를 발생시켜 캐리의 발생과 전파를 분석한다. 따라서 Look ahead carry는 Adder의 연결을 효율적으로 만들 수 있기 때문에 큰 숫자의 덧셈, 여러 비트의 덧셈에 효과적이다.

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

**3.**

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

XOR을 활용한 2’s complement 가감산에서는 우선 2’s complement, 2의 보수에 대해서 알아야 한다. 2의 보수는 이진수에서 모든 비트를 반전시킨 뒤 1을 더하면 만들 수 있다. XOR 연산은 비트를 반전시키는 것에 유용하므로, XOR 연산을 사용하면 2의 보수를 구하기 유리해 진다.

예를 들어, 5 (0101)의 2의 보수를 구하려면 우선 모든 비트를 반전시켜 1010으로 만든다. 이때 XOR을 사용할 수 있다. 여기에 1을 더해 1011으로 만들면 이것이 5의 2의 보수이다. 이는 감산기에서 사용하며 오버플로우 상황에서도 유용하게 사용할 수 있다.

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

**4.**

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

BCD 연산은 십진수를 이진수로 표현하는 방법이다. 10진수의 각 자릿수의 수를 4비트 이진수로 표현해 나타내는 것이다. 이 BCD 연산의 덧셈에서 각 자릿수의 수에 10 이상의 값이 나타나면 0110을 더해 자릿수를 올려야 한다. 이를 보정이라고 하며, 9+1을 예시로 들자면 9+1은 1010이 된다. 이때 0110을 더해 0001 0000으로 만들면 보정까지 처리 되어 BCD 연산이 제대로 수행된 것이다. BCD 연산은 사용하는 비트의 수가 많아 비효율적일 수도 있다.

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

**5.**

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

ALU는 Arithmetic Logic Unit으로, CPU에서 산술 및 논리 연산을 수행하는 요소라고 할 수 있다. ALU는 덧셈, 뺄셈, 곱셈, 나눗셈과 같은 산술 연산을 하고, AND, OR과 같은 논리 연산까지 수행한다. shift 연산, 비교 연산도 지원하여 다양한 계산을 돕는다. ALU는 연산 결과를 결과 상태에 따라 플래그 레지스터에 저정한다. 오버플로우 플래그가 이와 같은 경우라고 볼 수도 있다.

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

**6.**

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

플래그 레지스터에 대하여 추가적으로 조사하였다. 플래그 레지스터는 연산 결과의 상태를 저장하는 곳으로, 이후의 명령에 영향을 끼칠 수 있다. 연산의 결과가 음수인지 캐리가 발생했는지 오버플로우가 발생했는지 등을 저장하고 이후 명령에서 조건 판단에 사용하는 것이다.

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