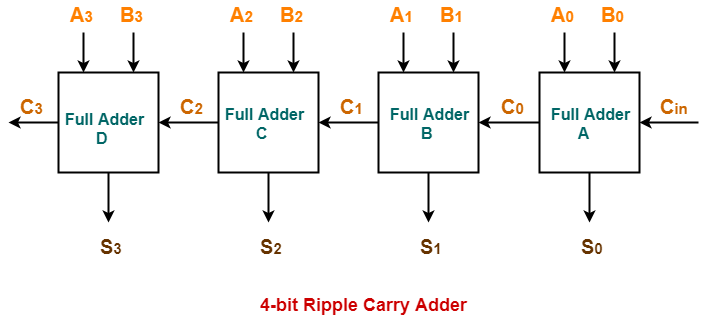
전공: 컴퓨터공학 학년: 2학년 학번: 20161603 이름: 신민준

1. 4-bit adder 및 subtractor 이진 병렬 연산 기능에 대하여 조사하시오.

이진 병렬 감산/가산기는 1-bit adder/subtractor를 서로 연결해 n-bit의 수에 대해서 연산을 수행하는 논리 회로이다.



위 그림은 그러한 4-bit의 두 수의 합을 연산하는 4-bit ripple carry adder를 보이고 있는데, 이 때 각 carry(subtractor의 경우, borrow) 값은 다음 bit를 연산하는 full adder로 넘겨지게 된다.

Binary parallel subtractor는 2’s complement를 바탕으로 사용되므로, subtrahend에 해당하는 bit에 inverter를 사용해 값을 바꾸고 연산하고자 하는 두 값을 합치는 방식으로 구현된다. 이 방식으로 구현한 binary parallel subtractor는 아래의 회로도와 같은 모습이 된다.

담장, 녹색, 텍스트, 건물이(가) 표시된 사진

자동 생성된 설명

그러나, Ripple Carry Adder의 구조를 가진 경우, 구조가 간단해서 구현하는 과정의 어려움은 없지만, 특정 위치의 bit를 계산하기 위해서는 그 전 위치에 있는 adder의 실행이 끝나야 한다는 단점이 있다. 따라서 위 구조는 계산하는 bit의 개수에 따라 실행 시 gate delay가 bit 수와 비례해 의 delay 시간을 갖게 된다.

1. Carry Lookahead에 대하여 조사하시오.

Carry Lookahead adder는 간단한 n-bit adder인 ripple-carry adder에서 단점이 되는 실행시간을 효과적으로 줄인 논리 회로이다. Ripple-carry adder에서는 각 full adder들이 그 전의 full adder가 실행되어 carry bit이 생성될 때 까지 기다려야 하기 때문에, 계산하는 bit의 수에 따라 비례적으로 gate delay가 늘어나는데, CLA는 이 carry bit들을 계산하는 데 걸리는 시간을 최대한 줄여 전체적인 gate delay를 줄이고자 한다.

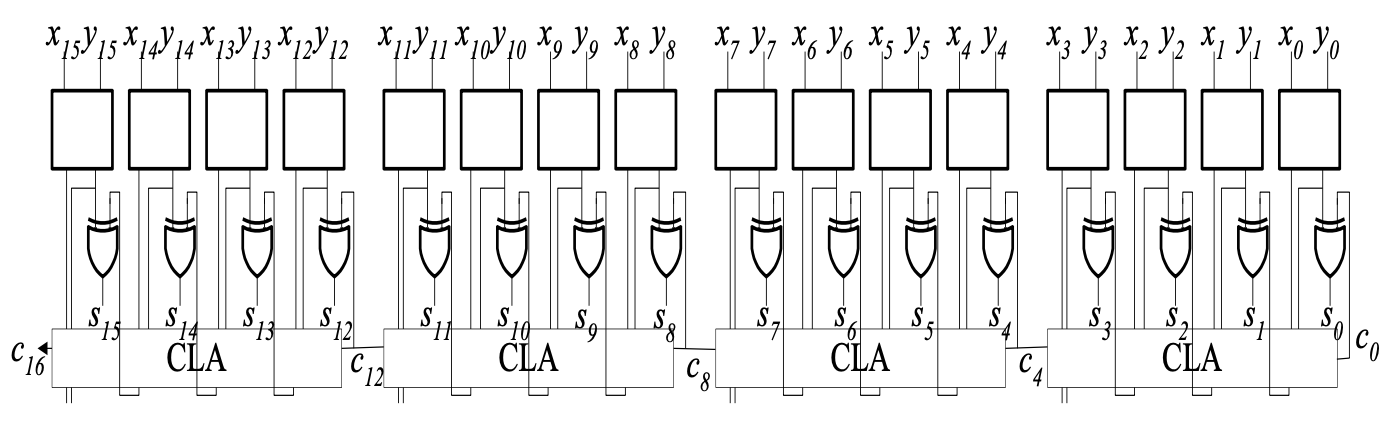
CLA에서는 두 함수, carry-generate function 와 carry-propagate function 를 사용해 carry bit을 제공한다. 예를 들어, 의 식으로 각 bit의 carry를 나타낼 수 있으므로, 이를 풀어서 4bit에 대한 결과로 나타내면

처럼 나타내어지게 된다. 각 carry bit들은 2 gate delay 내에 계산될 수 있으므로, 이를 사용해 각 full adder를 연산하면 실행 시간이 확연하게 빨라질 수 있다.

텍스트, 지도이(가) 표시된 사진

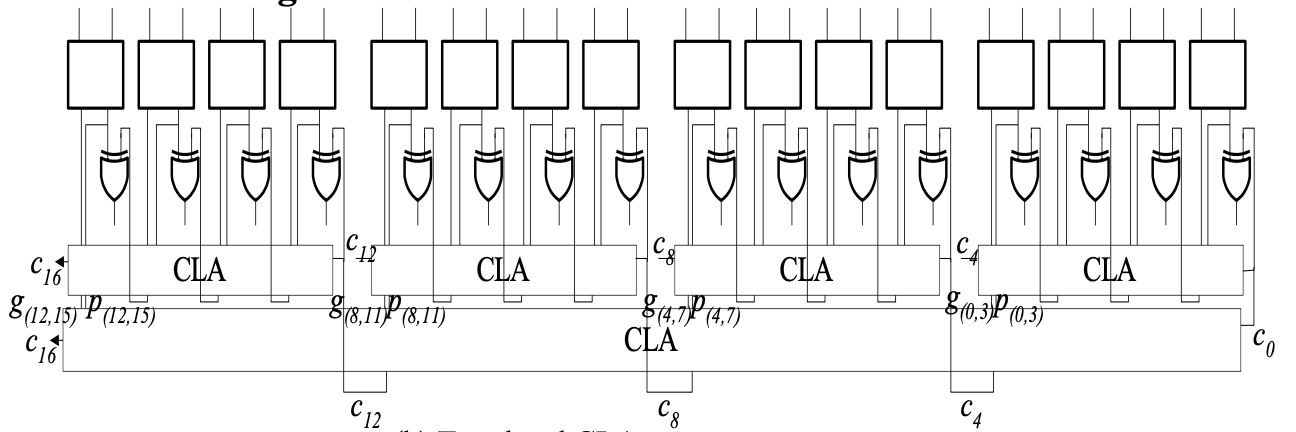
자동 생성된 설명

위 그림과 같이 CLA 회로는 나타내어질 수 있는데, 이 경우, 총 gate delay는 가 된다.



더 많은 수의 bit을 계산하는 경우, 위 그림처럼 여러 개의 CLA를 cascading 한 디자인으로 구현할 수 있는데, 이 때도 RCA와 같은 단점을 갖게 된다. 그렇다면 다중의 level을 가진 CLA를 디자인해서 구현하면 gate delay를 똑같이 줄일 수 있다.

예를 들어, 16bit를 계산하는 calculator를 구현한다면, 두 개의 CLA를 two-level 형식으로 다음과 같이 구현할 수 있다.



1. XOR를 활용한 two’s complement 가감산에 대하여 조사하시오.

Two’s complement 시스템에서 감산을 구현하는 방법으로, 빼는 값을 2’s complement 시스템 상의 음수값으로 변환하고 변환한 값과 뺄셈 당하는 값을 서로 더하는 방법이 있다. 따라서, 이 과정을 진리표로 나타내면 다음과 같다.

|  |  |  |
| --- | --- | --- |
| Select signal | Function | comment |
| 0 | A+B | 덧셈 |
| 1 | A+B’+1 | 뺄셈 |

이 때, A, B는 의 형태를 갖는 binary input이고, 는 이 계산기의 출력값이다. 이 과정을 간단하게 다이어그램으로 그린 것이 아래의 그림이다.

개체이(가) 표시된 사진

자동 생성된 설명

XOR연산은 다음과 같은 특성이 있다.

따라서, adder&subtractor를 동시에 구현하는 방법은, signal line의 값을 두 번째 입력값 B와 XOR 연산을 시켜 그 값을 사용하는 것이다. 이 방법을 사용한 디자인은 다음과 같이 나타내어질 수 있다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

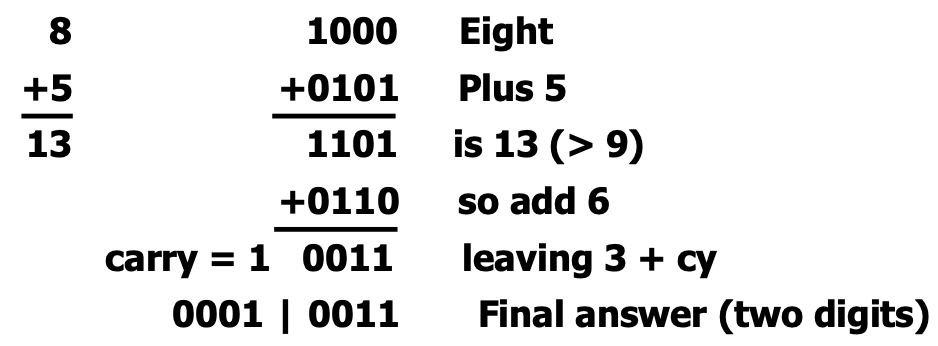
1. BCD 연산에 대하여 조사하시오.

Binary Coded Decimal은 0부터 9까지의 10진수 digit들 각각을 특정한 고유의 bit 시퀀스로 나타내는 binary encoding 방식이다. 이 때, 특별한 bit 패턴들을 사용해 에러나 오버플로우를 감지하기도 한다.

일반적으로 8421 code라고 불리는 형식의 BCD 코드를 받았을 때, 일반적인 binary adder를 사용해서 계산할 수 있다.

만약 1000(8)+0101(5)의 계산을 해야 한다면, binary adder의 결과는 1101(13)이 된다. 하지만, BCD 코드에서 9 이상의 값은 invalid codeword이므로, 13이라는 결과는 두 개의 digit으로 표현되어야만 한다.

따라서, 이 값을 정상적으로 BCD 형태로 출력하기 위해선, 6을 더하고 16으로 나눈 나머지는 결론적으로 해당 값에서 10을 뺀 것과 같으므로, 결과 값에서 6을 더하고, 더한 값을 다시 16으로 나눠 그 나머지를 두고, 몫인 1은 carry로 넘겨주게 된다. 따라서, 0001|0011의 형태가 되어 13이라는 값을 BCD 형태로 보여주는 것이다. 이 과정은 다음 그림에서 확인할 수 있다.



1. ALU의 기능에 대하여 조사하시오.

Arithmetic Logic Unit은 binary로 표현된 두 정수를 받아 산술연산과 논리연산을 계산하는 디지털 회로이다. ALU는 흔히 도식으로 다음과 같이 표현된다.

개체, 시계이(가) 표시된 사진

자동 생성된 설명

ALU는 컴퓨터 회로에서 가장 기본적인 구성요소로서, 컴퓨터의 CPU, FPU, 또는 GPU 등의 장치를 구성한다. 이 때, 한 개의 CPU, FPU, GPU가 여러 개의 ALU를 내장하고 있을 수도 있다.

ALU에서 입력받는 데이터들은 operand들이라 부르며, operand들과 함께 수행할 연산에 대한 데이터 코드와 함께 주어진다. ALU의 output에는 이렇게 입력받은 데이터들로 수행한 연산의 결과가 나온다.

추가적으로, ALU는 상태를 나타내는 input/output이 존재하는데, 이는 방금 전 수행한 연산에 대한 정보나 현재 연산 정보를 제공하는 데 사용된다.

기본적으로 general purpose ALU들은 다음과 같은 연산을 제공한다.

산술 연산: add, add with carry, subtract, subtract with borrow, negate, increment, decrement, pass through

논리 연산: AND, OR, XOR, NOT

Bit-shift 연산: arithmetic shift, logical shift, rotate, rotate through carry

1. 기타 이론.

* ALU가 정수 형태의 binary number를 계산하는 계산기라면, 반대로 부동소수점 숫자에 대해 작동하는 FPU(Floating-point Unit)도 존재한다. FPU는 math coprocessor라고도 하며, 컴퓨터 시스템에서 부동소수점 계산을 수행하기 위해 특별하게 고안된 장치인데, 일반적으로 덧셈, 뺄셈, 곱셈, 나눗셈, 제곱근, bit-shifting의 연산을 지원한다.
* ALU는 더 복잡한 연산을 수행할 수 있도록 디자인될 수 있지만, 이로 인해 증가하는 회로 복잡도, 비용, 전기 사용량, 크기로 인해 대부분의 경우 비효율적이다. 따라서, ALU는 상기한 것처럼 매우 빠른 시간 안에 해결이 가능한 간단한 문제들을 해결할 수 있는 기능만을 제공한다.