10주차 예비보고서

전공: 아트엔테크놀로지 학년: 3학년 학번: 20191172 이름: 함승우

1. 4-Bit Adder 및 Subtractor 이진 병렬 연산 기능

4-bit Binary parallel Adder는 하나의 칩으로 설계된다. 두 개의 4 bit binary를 add하기 위한 논리 회로이다. 이미 진행한 실습 주차에서 Full adder라는 것을 배웠다. 이는 두 개의 입력을 받고 이전 자리올림과 합(Sum)과 궁극적인 자리올림(Carry)를 구하는 논리 회로이다.

그래픽이(가) 표시된 사진

자동 생성된 설명(Full adder)

A와 B와 cin(이전 자리 올림) 중, 1이 두 개인 경우 10이 output으로 출력되어야 한다. 이 때, Carry는 올림 수니까 1에 해당하고, Sum은 0에 해당한다. 또한, 1이 세 개인 경우 11이 output으로 출력되어야 한다. 이 때, Carry는 올림 수니까 1에 해당하고, Sum은 1에 해당한다. 병렬 4 – bit Binary Parallel Adder는 이 Full adder를 사용해 4 비트를 계산함에 있다. 4 비트 수 2개가 입력이 되고, 이 4bit 수가 각각 a3a2a1a0, b3b2b1b0일 때,

a0와 b0는 이전 자리올림수가 없기 때문에, Half adder를 사용하면 된다. a0와 b0가 더 해져서, sum값과 c1값을 출력한다. 그 후, a1, b1, c1 값을 Full adder를 사용하여 연산을 진행한다. 나머지 a2, b2, a3, b3 연산도 동일하게 진행한다.

\*a0, b0에서 Half adder를 사용하기도 하지만, cin값을 0으로 준 Full adder를 사용하기도 한다.

도표, 라인, 평면도, 기술 도면이(가) 표시된 사진

자동 생성된 설명

다음은, 4 bit binary parallel Subtractor이다.

도표, 라인, 기술 도면, 평면도이(가) 표시된 사진

자동 생성된 설명

앞서 관찰한 adder처럼 full subtractor를 4개 병렬 연결한 형태가 바로 4 bit subtractor이다.

2. Look ahead carry 대하여 조사하시오.

위 연산들을 Ripple Carry Adder라고 하는데, 4 bit로 설명이 되어있다. 하지만, Full Adder를 병렬로 연결한다면 연산할 수 있는 bit는 늘어난다. 즉, 16 bit, 32 bit를 연산할 수 있다는 것이다. 32 bit 연산을 한다면, RCA는 a31, a30, a29 ~ a0, b31, b30, b29 ~ b0을 연산을 하게 되는데, 이 때 a31, b31 연산은 이전까지의 모든 step이 다 끝나야 연산을 진행할 수 있다. Carry 값을 받고 연산을 해야 때문에 시간이 오래 걸린다는 단점이 있다. 이를 해결한 회로가 Carry Lookahead Adder(CLA)이다.

스케치, 도표, 그림, 기술 도면이(가) 표시된 사진

자동 생성된 설명

위 그림처럼, CLA generator가 필요한 carry 값을 바로 전달하기 때문에 delay를 줄일 수 있다. 구현을 하기 위해서는, 식을 정리해서 표현을 해보자.

텍스트, 폰트, 스크린샷, 화이트이(가) 표시된 사진

자동 생성된 설명

가산기에서, S값은 x ⊕ y ⊕ C로 구현을 했고, C = xy + c(x ⊕ y)로 구현을 하였다.

따라서, 이 XOR 연산과 and 연산을 정리하여, Carry 값에 대한 식을 세워 보면 위 그림과 같다. 이를 이전의 Carry 값들과 상관없이 x와 y 만으로 식을 표현한다면, 아래와 같다.

텍스트, 폰트, 스크린샷, 화이트이(가) 표시된 사진

자동 생성된 설명

직접적으로, 미리(Ripple Carry Adder에 비해) 계산해서 각각의 adder에 전달된다.

3. XOR을 활용한 2‘s complement 가감산

4 bit 병렬 이진 가산기와 감산기를 한 번에 구현할 수 있다, 이는 4개의 Full adder를 사용할 수 있다. 컴퓨터에서는, A – B에 해당하는 식은, A + (-B)로 생각을 할 수 있기 때문이다.

도표, 스케치, 평면도, 기술 도면이(가) 표시된 사진

자동 생성된 설명

이 그림은 앞의 병렬 가산기와 비슷해 보이지만, M값과, XOR 연산이 추가되었다는 점에서 다르다.

M은 Minus 값으로, 덧셈의 경우에는 0, 뺄셈의 경우에는 1이 입력된다. 그 후, B0, B1, B2, B3 값과 XOR 연산을 진행하는데, 이 연산의 의미는, B를 음수로 만들기 위해서 2의 보수로 구하는 과정이다. M이 1이라면, B와 XOR 연산하면 B의 값이 반전되어 입력된다. 또한, Cin으로 M이 들어가면서, 2의 보수법을 구현하게 된다. M값이 0이라면, XOR 연산을 통해서 아무것도 반전되지 않고 출력이 된다. 덧셈과 뺄셈을 선택하여 구현할 수 있다.

4. BCD 연산

BCD는 Binary Code Decimal로 10진수를 2진수로 표현을 한 것이다. 앞선 예비보고서에 설명이 되어있지만, 주로 8421 code를 사용한다. 이는 수가 8, 4, 2, 1일 때 가중치값을 넣어주었다고 생각하면 된다. 예를 들어, 숫자 11인 경우에는, 0001 0001로 변환된다.

덧셈의 경우, overflow를 생각을 해야 되기 때문에, 그 덧셈의 결과값에 0110을 추가적으로 더해줘야 된다. 6과 5를 더하면, 0110 + 0101이다. 덧셈을 그냥 진행하면, 1011으로 원하는 BCD가 아니다. 이는 0~ 9까지만 표현해야 해야 하기 때문에 이 결과 값에 0110을 더해줘서 0001 0001값을 만들면, 원하는 결과를 얻을 수 있다. Decimal하게 표현하자면, 6을 더해줘서 Binary로 표현하면 된다.

텍스트, 도표, 라인, 평면도이(가) 표시된 사진

자동 생성된 설명

5. ALU의 기능

ALU는 Arithmetic Logic Unit으로 산술 연산과 논리연산을 수행하는 장치이다. ALU는 CPU의 구성요소 중 하나이다. CPU에서 연산을 담당한다. ALU는 덧셈, 뺄셈, 논리 연산, 시프트와 같은 산술 연산을 수행할 수 있다. 또한 이 덧셈과 뺄셈을 활용하여 곱셈과 나눗셈도 구현 가능하다. 곱셈은 덧셈의 반복으로 구현 가능하고, 나눗셈은 뺄셈의 반복으로 구현 가능하다. 이러한 기능을 구현하기 위해, 가산기, 뺄셈을 구현하기 위해서 보수기, 누산기, 플립플롭, 데이터 레지스터로 구성된다.

6.

글로켄슈필이(가) 표시된 사진

자동 생성된 설명

위 그림은 16 bit Carry Lookahead Adder이다. Ripple Carry Adder같이, adder를 16개 병렬로 이루어져 있지만, CLA generator를 이중으로 집어넣어서, C12, C8, C4, C0값들은 2- level CLA generator가 계산하게 되어, 이전에 배웠던 single level CLA 보다도 더 Delay를 줄였다.

텍스트, 폰트, 스크린샷, 화이트이(가) 표시된 사진

자동 생성된 설명

마찬가지로, level을 더 추가하여 더 많은 비트를 빠르게 계산할 수 있다.

텍스트, 도표, 폰트, 평면도이(가) 표시된 사진

자동 생성된 설명

Sum:

Carry:

Difference:

Carry:

= Ai⊕Bi⊕Ci

Carry= Ci+1= Ci(Ai⊕Bi) + AiBi

텍스트, 스크린샷, 소프트웨어, 컴퓨터이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 소프트웨어, 컴퓨터이(가) 표시된 사진

자동 생성된 설명