6주차 예비보고서

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

1.

도표, 스케치, 라인, 기술 도면이(가) 표시된 사진

자동 생성된 설명

가산기는 Adder로 두 수를 더 해주는 것이다, 가산기에는 전가산기와 반가산기가 있는데,

A,B라는 input이 있을 때, 1 bit의 input이 있을 때 두 값을 더한 값을 S와 C를 출력한다. S와 C는 sum과 carry이다. 반 가산기란 2개의 input 값을 더해 합을 Sum, 자리올림 값을 Carry로 나누어 출력하는 회로이다. 이전 까지는 1과 1을 더했을 때 1이라는 출력을 갖는다고 했었다. 하지만, 반가산기에서는 덧셈을 한 후에 자리 올림을 고려하여 이를 C(carry)값으로 출력한다.

예를 들어, input X, Y가 있다고 할 때 X에 1을 Y에 1을 입력한다고 할 때, Sum값은 0이 되고, Carry 값은 자리올림에 의해서 1이 된다. 자리 올림이 일어나지 않는, X가 1이고, Y가 0인 경우에는 Sum값이 1, Carry값은 0이 된다.

반 가산기의 진리표는 다음과 같다.

|  |  |  |  |
| --- | --- | --- | --- |
| X | Y | S | C |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

진리표에서 볼 수 있듯이 S(Sum)은 input 값이 동일할 때 0을 출력하고, Input 값이 서로 다를 떄 1을 출력하는 것을 볼 수 있다. 따라서 Sum의 출력은 XOR의 연산으로 표현 가능하다. C(carry)는 input값이 1이어야만 1을 출력하는 것을 볼 수 있다. 따라서 이는 AND 연산으로 표현 가능하다.

스케치, 도표, 그림, 라인이(가) 표시된 사진

자동 생성된 설명

2개의 입력을 받고, Sum과 Carry 값을 출력하는 반가산기와 비슷하게 전가산기도 Sum, Carry 출력을 갖지만, 3개의 출력을 받는다는 점에서 다르다. 전가산기는 C\_in(Carry input)값을 추가로 가지고 있다. 뒷자리에서 올라온 C\_in을 포함해서 1 bit 크기의 2진수 3자리를 더하여 S(Sum)과 Ci+1(carry + 1)을 구한다.

예를 들어, A가 1, B가 1이라고 가정해보자, 또한 이전 단계에서의 자리올림 (Carryinput)값이 0이라고 해보자. 이 경우에는 Sum에는 A와 B가 1로 같은 비트이기 때문에 Sum은 0이 되고, 자리올림 (Carry)는 A와 B가 1로 동일하고, 이전 단계에서의 자리 올림도 1이기 때문에 1이다.

전가산기를 진리표는 다음과 같다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| X | Y | Ci | S | C |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |

마지막으로 전 가산기의 Sum은 두 input이 모두 같고, Cin이 1인 경우에만 1이되고, Carry는 a, b가 서로 다르고, Carry가 1일 때다. 추가로, 전가산기를 유심히 보면, 반가산기의 형태가 2개 보인다.

2. 반 감산기, 전 감산기

도표, 라인, 그림, 스케치이(가) 표시된 사진

자동 생성된 설명

감산기는 두 개의 이진 수를 뺄셈 연산을 수행하기 위한 논리 회로이다.

우선 반 감산기는 2개의 input 2개의 output을 갖는데, 2진수 1자리의 두 개 비트를 뺄셈하고 그 차를 출력한다. 이 때 출력 값은 D(difference)와 B(Borrow) 두 가지가 있는데 각각은 차와 빌려오는 수이다. Borrow는 입력 A가 B보다 작은 경우만 발생한다. 따라서,

예를 들어, 두 가지의 input X, Y가 각각 1, 0일 때 D는 1에 해당한다. 하지만, X가 Y보다 크기 때문에 Borrow가 발생하지 않아 B는 0이다. 하지만 각각 0, 1일 때는 마찬가지로 D는 1에 해당하지만, X가 Y보다 작기 때문에 Borrow값이 발생해 B에서 1이 출력된다.

|  |  |  |  |
| --- | --- | --- | --- |
| X | Y | D | B |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |

이 진리표를 가지고 D와 B의 수식을 구하면 D는 XOR(X, Y)이지만, B는 X’Y이다.

텍스트, 바퀴, 마스크, 클립아트이(가) 표시된 사진

자동 생성된 설명

전감산기도 반감산기와 마찬가지로 뺄셈 연산을 하고 Difference와 Borrow 값을 출력하지만, 이전 자릿수에서 빌림을 했는지에 관한 입력 값이 추가로 있어, 세가지의 input 값이 있고, 두 가지의 output이 있는 형태이다.

예를 들어, 10과 01을 전감산기를 통해서 뺄셈을 구현한다고 해보자. (이 둘은 모두 2진수이다). 이때 10에서 01을 뺄 때에는 맨 뒤에 자릿수부터 진행하는데, 맨 뒷자리 수 에서 Borrow가 발생한다. Borrow in이 1이되고, 최종 D값은 1이 되는 것을 알 수 있다.

다음은, 전감산기의 진리표이다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| X | Y | Bi | B | D |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

3. BCD 가산기에 대해 조사하시오.

BCD는 Binary Coded Decimal의 약자로 이진수로 표현된 10진수라는 뜻이다. BCD는 이진수로 표현된 10진수를 더하는 데에 사용된다. BCD는 0000(0) ~1001(9)까지를 이진수로 표현한다. BCD는 이진 가산기와는 다르게 10부터 위 자리에 Carry가 발생한다. 두 BCD 숫자 (0000 ~ 1001)로 표현된 두 가지의 숫자를 더한다면, 각 비트들을 이진수 가산기로 더하는 과정이 필요하다. 이때 Carry가 되는, 자리 올림이 필요한 경우는 10 이상의 수를 나타내게 될 때이고, 이 경우에는 0110을 더해준다. 예를 들어

8 + 7을 해보자, 8은 BCD로 1111이고, 7은 0111이다. 1111+0111 = 1111이다. 또한 10을 넘었기 때문에 0110을 추가로 더 해줘 0001 0101이 된다.

4. 병렬 가감산기.

병렬 가감산기는 덧셈과 뺄셈을 모두 할 수 있는 논리 회로이다. 병렬 가감산기는 뺄셈 연산을 2의 보수 덧셈으로 구현해서 가산기와 감산기로 모두 동작이 가능하다. 이를 덧셈으로 사용하려면 입력 신호를 그대로 입력되게끔 하면 되고, 뺄셈으로 사용하려면 피가수는 그대로 두고 가수의 신호에 2의 보수를 취해주면 된다. 이전 전감산기와 전가산기를 살펴보면 더한 값(Sum)과 뺄셈 한 값(Difference)는 모두 XOR 연산을 통해서 구현이 가능했고, 전감산기와 전가산기의 Carry값과 Borrow 값의 관계를 살펴보면 둘이 서로 인버터의 유무만 존재하는 것을 확인할 수 있다. 병렬 가감산기는 전가산기에 XOR 게이트를 추가해 병렬로 배치하여 만들 수 있다.병렬 감가산기는 sign비트가 0이면 덧셈을, 1이면 뺄셈을 수행하게 된다.

5. Carry Look-Ahead Adder 를 Ripple Carry Adder와 비교하여 설명하시오.

Ripple Carry Adder는 여러 개, n개의 전가산기가 병렬 형태로 이어져 하나의 전 가산기의 carry 비트가 다음 전 가산기의 carry-in 값으로 입력이 되는 회로이다. 하나의 전가산기는 1 비트 덧셈만 가능하지만, n개를 병렬로 만들어 n비트의 덧셈을 구현 가능하다. 또한, 연산이 순차적으로 이루어져야만 한다. 그 이유는 하나의 전 가산기에서 출력한 Carry Out이 있어야만 다른 전 가산기의 input에 Carry in이 있기 때문에 이 회로는 연산이 순차적으로 이루어진다. Ripple Carry adder는 bit 수가 매우 큰 경우 carry의 연산 속도 때문에 전체적으로 연산 지연이 발생할 수 있다. 이 연산지연을 propagation delay 혹은 overflow라고 칭한다. 이 Delay를 보완하기 위해 계산을 동시에 수행하는 회로가 Carry Look Ahead Adder이다. 각 가산기에 입력 되는 a, b 비트의 XOR 값, 첫 input carry-in 값(가산기에 의해 출력된 값이 아닌)이 있다면 a & b 연산과 XOR 연산을 한꺼번에 받고, 그 결과를 다시 각 가산기로 보내 한꺼번에 Sum값을 연산할 수 있다.

6.

BCD(Binary Coded Decimal) code에 대해서 살펴보자. 여기에는 8421, 5421, 2421 코드 등이 있다. 8421 code는 강의 자료와 일반적으로 많이 쓰이는 code로 0000(0) ~ 1001(9)를 사용한다. 5421 code는 0101, 0110, 0111, 1101, 1110, 1111를 사용하지 않고, 2421 code는 0101, 0110, 0111, 1000, 1001, 1010을 사용하지 않는다.

|  |  |  |  |
| --- | --- | --- | --- |
| DECIMAL | 8421 | 5421 | 2421 |
| 0 | 0000 | 0000 | 0000 |
| 1 | 0001 | 0001 | 0001 |
| 2 | 0010 | 0010 | 0010 |
| 3 | 0011 | 0011 | 0011 |
| 4 | 0100 | 0100 | 0100 |
| 5 | 0101 | 1000 | 1011 |
| 6 | 0110 | 1001 | 1100 |
| 7 | 1000 | 1011 | 1110 |
| 8 | 1000 | 1011 | 1110 |
| 9 | 1001 | 1100 | 1111 |