6주차 예비보고서

전공: 컴퓨터공학과 학년: 3학년 학번: 20191599 이름: 송경호

**1.**

**1-1) 전가산기**

전가산기는 덧셈 연산을 수행하는 논리 회로로 두 개의 이진수를 더하는 연산을 수행한다. 이 때 두개의 입력뿐만 아니라 carry값을 연산에 포함하는 것이 특징이다. 따라서 2개의 input과 3개의 output을 갖게 된다. 다음은 전가산기의 진리표와 회로이다.

**그래픽이(가) 표시된 사진

자동 생성된 설명**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | | | **Output** | |
| **A** | **B** | **Cin** | **S** | **Cout** |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

회로에서 S는 A와 B의 XOR연산의 결과를 다시 Cin과 XOR연산한 결과임을 알 수 있다. 진리표를 확인했을 때에도 A, B, Cin의 1의 개수가 홀수일 때 S가 1이되고 짝수일 때 S가 0이 되는 것을 알 수 있다. 또한 Cout은 A, B, Cin 중 1이 두개 이상일 때 1이 되는 것을 알 수 있는데 이는 A와 B의 AND연산과 A와 B의 XOR연산을 Cin과 AND연산한 결과를 OR연산을 하여 구현한다. 정리된 식은 다음과 같다.

**S = (A ⊕ B) ⊕ Cin,**

**Cout = (A ⊕ B)Cin + AB**

**1-2) 반가산기**

반가산기 역시 가산기로 덧셈 연산을 수행하는 논리회로이다. 전가산기와 차이점은 자리 올림 값을 받지 않는다는 점이다. 즉 전가산기와 달리 carry값을 연산에 포함되지 않는다. 따라서 2개의 input과 2개의 output이 발생한다. 다음은 반가산기의 진리표와 회로이다.

**로고이(가) 표시된 사진

자동 생성된 설명**

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | | **Output** | |
| **A** | **B** | **S** | **C** |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

반가산기는 전가산기에 비해 훨씬 간단한 회로를 갖고 있다. S는 A와 B의 XOR연산의 결과이고 C는 A와 B의 AND연산의 결과이다. 따라서 S는 A와 B의 값이 다르면 1이되고 같으면 0이된다. 또한 C는 A와 B의 값이 모두 1인 경우에만 1이된다. 이는 전가산기와 동일하게 S는 입력의 1의 개수가 홀수일 때, C는 입력의 합이 2이상일 때 1이되는 것으로도 볼 수 있다. 정리된 식은 다음과 같다.

**S = A ⊕ B**

**C = AB**

**1-3) 예시**

입력 A와 B가 모두 1인 경우

반가산기는 C = 1\*1 = 1 | S = 1 ⊕ 1 = 0이 된다.

전가산기는 Cin이 0일 경우 Cout = (1 ⊕ 1) \* 0 + 1\*1 = 1, S = (1 ⊕ 1) ⊕ 0 = 0

Cin이 1일 경우 Cout = (1 ⊕ 1) \* 1 + 1\*1 = 1, S = (1 ⊕ 1) ⊕ 1 = 1이 된다.

**2.**

**2-1) 전감산기**

전감산기는 뺄셈 연산을 수행하는 논리 회로로 하나의 이진수에서 다른 이진수를 빼는 연산을 수행한다. 감산기는 가산기와 그 형태가 비슷하여 마치 carry와 같이 두 개의 입력뿐만 아니라 borrow값을 연산에 포함시키는 것이 특징이다. 따라서 3개의 input과 2개의 output을 갖게 된다. 다음은 전감산기의 진리표와 회로이다.

**도표, 라인, 기술 도면, 평면도이(가) 표시된 사진

자동 생성된 설명**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | | | **Output** | |
| **A** | **B** | **Br0** | **D** | **Br** |
| 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 |

전감산기는 아래에 설명된 반감산기 두개를 연결하여 전가산기와 같이 A와 B의 연산 이전에 발생한 Br0에 대한 고려한다. 따라서 D는 A-B-Br0의 결과값을 갖게 된다. 이는 전가산기와 동일하게 A와 B의 XOR 연산 결과를 Br0와 한번더 XOR 연산을 수행하여 구한다. Br의 경우 A-B-Br0의 과정에서 음의 값이 발생하면 Br 역시 1을 갖게 되는 것이므로 A<B+Br0일 때 Br은 1이 된다. 이는 A와 B의 XNOR결과와 Br0의 AND결과 그리고 ~A와 B의 AND결과의 OR 연산으로 구할 수 있다. 정리된 식은 아래와 같다.

**D = (A ⊕ B) ⊕ Br0**

**Br =** **~(A ⊕ B)Br0 + ~AB**

**2-2) 반감산기**

반감산기 역시 가산기로 뺄셈 연산을 수행하는 논리회로이다. 전가산기와 차이점은 빌림 수를 받지 않는다는 점이다. 즉 전가산기와 달리 borrow값을 연산에 포함되지 않아서 2개의 input과 2개의 output이 발생한다. 예를 들어 A가 0 B가 1일 때 A가 B보다 작으므로 Br는 1이 되고 D는 2 – 1 = 1이 된다. 다음은 반감산기의 진리표와 회로이다.

**도표, 라인, 그림, 스케치이(가) 표시된 사진

자동 생성된 설명**

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | | **Output** | |
| **A** | **B** | **Br** | **D** |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 |

전감산기는 반감산기 2개로 이루어진 만큼 반감산기는 전감산기에 비해 훨씬 간단한 회로를 갖고 있다. D는 반가산기의 S와 동일하게 A와 B의 XOR 연산을 통해 구한다. 전감산기와 반감산기의 D는 모두 전가산기와 반가산기의 S와 동일한 것을 알 수 있다. 또한 Br의 경우 A가 0이고 B가 1인 경우에만 1이 되므로 ~A와 B의 AND 연산을 통해 구할 수 있다. 정리된 식은 다음과 같다.

**D = A ⊕ B**

**Br = ~AB**

**2-3) 예시**

입력 A와 B가 0,1인 경우

반가산기는 Br = ~0 \* 1 = 1 | D = 0 ⊕ 1 = 1이 된다.

전가산기는 Br0이 0일 경우 Br = ~(0 ⊕ 1) \* 0 + ~0\*1 = 1, D = (0 ⊕ 1) ⊕ 0 = 1이 된다.

Br0이 1일 경우 Br = ~(0 ⊕ 1) \* 1 + ~0\*1 = 1, D = (0 ⊕ 1) ⊕ 1 = 0이 된다.

**3.**

**3-1) BCD 가산기란**

도표, 텍스트, 라인, 평면도이(가) 표시된 사진

자동 생성된 설명

우선 BCD란 Binary Coded Decimal의 약자로, 이진 코드로 표현된 10진수 숫자를 나타내는 방법이다. BCD는 0부터 9까지의 숫자를 4 bit를 통해 나타내는데, 10부터 15까지의 숫자를 나타내는 이진수인 1010부터 1111까지는 사용하지 않는다. BCD코드는 일반적인 2진수 혹은 16진수로 10진수를 나타내는 것과 달리 각 0부터 9까지의 숫자를 나타내는 4 bit가 10진수의 자릿수를 나타낸다. 예를 들어 999는 BCD 코드로 표현 시 1001,1001,1001로 표현되는 데 각 4 bit마다 좌측부터 100의 자리, 10의 자리, 1의 자리 숫자를 나타낸다.

BCD 가산기란 BCD 코드로 표현된 10진수 숫자를 2개를 입력 받아서 덧셈을 수행하여 그 결과를 BCD 코드로 출력해주는 가산기의 한 일종이다.

**3-2) 예시**

BCD 코드의 덧셈은 다음과 같이 이루어진다. 먼저 2진수의 덧셈 규칙에 따라 덧셈을 수행한다. 다음으로 연산 결과가 9 이하의 값이라면 해당 연산 결과를 사요아 9 이상의 값이라면 결과값에 0110을 더하여 사용한다.

987 + 123의 연산을 BCD 가산기를 통해 계산한다.

1. **987** → 1001,1000,0111 / **123** → 0001,0010,0011

2. **7 + 3** → 0111 + 0011 = 1010 → 10이므로 0110을 더함 → 10000. 자리 올림 값을 제외하고 십진수로 표현 시 **0**

3. **8 + 2 + 1**(앞선 자리 올림 값) → 1000 + 0010 + 0001 = 1011 → 11이므로 0110을 더함 → 10001. 자리 올림 값을 제외하고 십진수 표현 시 **1**

4. **9 + 1 + 1**(앞선 자리 올림 값) → 1001 + 0001 + 0001 = 1011 → 11이므로 0110을 더함 → 10001. 자리 올림 값을 제외하고 10진수 표현 시 **1**

5.10진수의 표현을 모아주면 **1110**으로 결과 확인

**4.**

도표, 스케치, 라인, 기술 도면이(가) 표시된 사진

자동 생성된 설명

이름 그대로 숫자를 더하거나 빼는 데 사용되는 회로이며 여러 비트를 동시에 처리할 수 있어 빠른 계산이 가능하다. sign 비트에 따라 가산기로 작동될지 감산기로 작동될지 결정된다. S가 0일 때 가산기이며 1일 때 감산기이다. 그러나 중요한 것은 FA는 항상 전가산기로써 작용한다는 점이다.

입력하는 A0부터 A3와 B0부터 B3는 더하거나 빼고자 하는 입력이다. sign bit가 0이라면 XOR 연산의 결과는 Bi의 값이 그대로 출력되므로 A와 B가 그대로 전가산기에 입력된다. 즉 A와 B의 덧셈 연산이 수행된다.

그러나 sign bit가 1이라면 Bi의 보수가 전가산기에 입력된다. 이때 sign bit의 값 1이 C0의 값으로 사용되므로 입력 Bi에 대해 2의 보수를 취하는 형식이 발생한다. 이는 A와 B의 뺄셈 연산을 의미한다.

병렬 가산기를 이용할 때 전파지연이 발생하므로 Carry Look Ahead Adder가 주로 사용된다.

**5.**

Carry Look-Ahead Adder은 계산을 parallel하게 수용하는 반면 Ripple Carry Adder은 계산을 serial하게 수행한다는 점이 가장 큰 차이점이다. 따라서 Carry Look-Ahead Adder의 경우 회로가 더 복잡하지만 속도가 더 빠르다는 장점을 지닌다.

Carry Look-Ahead Adder은 여러 전가산기와 반가산기로 구성되며 각 전가산기는 두 비트를 더하고 carry bit를 생성한다. 이때 Carry Look-Ahead Adder의 핵심 아이디어는 입력 비트의 자리올림을 사전에 계산하는 것이다.

우선 G = A⦁B, P = A⊕B를 통해 G와 P를 구한다. 이때 G는 특정 비트 위치에서 carry가 발생하 것인지 확인하는 비트이며 P는 다음 비트로 carry를 전파할 필요가 있는지 판단하는 비트이다. 다음으로 Si = Pi⊕Ci, Ci+1 = Gi + (Pi⦁Ci)를 구하는데 이때 핵심은 Ci+1과 Ci의 점화식이다. 둘의 점화식을 통해 일반화가 가능하며 이는 C1을 통해 Cn을 구할 수 있다는 것을 의미한다. 때문에 지연시간이 발생하지 않아 수행시간이 줄어든다. 그러나 위에 언급한 연산들을 모두 수행할 수 있어야 하기 때문에 회로의 복잡도가 상승한다는 단점이 있다.

**6.**

**6-1) 전파 지연**

전파 지연이란 논리회로에 입력신호가 들어오는 순간부터 출력신호가 나가는 순간까지 걸리는 시간을 의미한다. (입력신호가 변경되는 순간 출력 신호가 최종 출력 수준의 50%에 도달하는데 걸리는 시간을 의미하기도 한다) 디지털 회로에서 이러한 전파 지연은 회로의 성능과 직결되는 특성이다. 각 회로는 전파 지연을 줄여 데이터의 전송을 빠르게 하는 것에 초점을 두어야 한다. 다음은 각 Gate의 전파지연 시간이다.

|  |  |  |  |
| --- | --- | --- | --- |
| NOT | 1 | NAND | 1.4 |
| Buffer | 2 | NOR | 1.4 |
| AND | 2.4 | XOR | 4.2 |
| OR | 2.4 | XNOR | 3.2 |

**6-2) 예측 자리 올림 수 장치**

**텍스트, 폰트, 라인, 번호이(가) 표시된 사진

자동 생성된 설명**

Look Ahead Carry Unit(LCU)라고도 부르며 가산기 장치의 계산 시간을 감소시키기 위해 사용한다. 앞서 말한 Carry Look Ahead Adder에도 이를 이용하여 Carry bit를 미리 구할 수 있었다. 위 그림은 16-bit Carry Look Ahead Adder이다.