6주차 예비보고서

전공: 컴퓨터공학과 학년: 3학년 학번: 20212022 이름: 이예준

**1.**

가산기는 가산 회로라고도 하며 덧셈을 수행하는 조합 논리회로이다.

하지만 덧셈 외에도 뺄셈, 곱셈, 나눗셈을 할 수 있는데 그 방법은

뺄셈은 보수를 취해 더하면 구할 수 있고, 곱셈은 덧셈을 반복하면 구할 수 있고,

나눗셈은 뺄셈을 반복하면 된다. 가산기의 종류로는 반가산기와 전가산기가 있다.

-반가산기는 2개의 입력값을 받아 합(Sum)과 캐리(Carry)를 산출하기 위한

논리 조합 회로이다. 예를 들어 0과 0 또는 0과 1을 더할 때 합은 0 또는 1이 되고

캐리는 산출되지 않는다. 하지만 1과 1을 더할 때는 합은 0이 되고 캐리가 산출된다.

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

\*반가산기의 진리표

이 진리표를 이용해 논리식을 구하면 , 이렇게 구할 수 있다.

(S는 합을 구하고, C는 캐리를 구한다)

도표, 스케치, 라인, 폰트이(가) 표시된 사진

자동 생성된 설명

\*반가산기의 회로도

위의 회로도를 보게 되면 합(Sum)은 XOR 게이트로 구현했고,

캐리(Carry)는 AND 게이트로 구현했다.

예시)

도표, 스케치, 라인, 폰트이(가) 표시된 사진

자동 생성된 설명도표, 스케치, 라인, 폰트이(가) 표시된 사진

자동 생성된 설명

**0**

**1**

**1**

**1**

**0**

**1**

**1**

**0**

도표, 스케치, 라인, 폰트이(가) 표시된 사진

자동 생성된 설명도표, 스케치, 라인, 폰트이(가) 표시된 사진

자동 생성된 설명

**1**

**0**

**0**

**0**

**0**

**0**

**0**

**1**

-전가산기는 반가산기 2개를 응용하여 캐리(Carry)까지 입력으로 받아

3개의 비트를 입력 받아서 더하는 논리회로이다.

예를 들어 입력을 0과 1로 받고 캐리를 1로 받으면 합은 0이 되고 캐리가 산출된다.

또한 입력을 1과 1로 받고 캐리도 1로 받는다면 합은 1이 되고 캐리도 산출된다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 입력 | | | 출력 | |
| X | Y | C(in) | S | C(out) |
| 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 |

\*전가산기의 진리표

이 진리표를 이용해 논리식을 구하면 , 이렇게 구할 수 있다.

도표, 스케치, 평면도, 기술 도면이(가) 표시된 사진

자동 생성된 설명

\*전가산기의 회로도

위의 회로도를 보게 되면 반가산기 두개와 OR게이트로 구현한 것을 볼 수 있다.

예시)

3(0011) +6(0110) = ?

도표, 스케치, 평면도, 기술 도면이(가) 표시된 사진

자동 생성된 설명도표, 스케치, 평면도, 기술 도면이(가) 표시된 사진

자동 생성된 설명

**0**

**1**

**1**

**0**

**0**

**1**

**0**

**1**

**0**

**1**

도표, 스케치, 평면도, 기술 도면이(가) 표시된 사진

자동 생성된 설명도표, 스케치, 평면도, 기술 도면이(가) 표시된 사진

자동 생성된 설명

**0**

**1**

**1**

**0**

**0**

**0**

**1**

**1**

**0**

**1**

3(0011) +6(0110) = 9(1001)

**2.**

위에서 설명했듯이 감산은 가산기에서 입력에 보수를 취해서 구하는 방법도 있지만

감산기를 통해서도 뺄셈을 연산할 수 있다. 가산기에서는 캐리(Carry)가 산출되지만

감산기에서는 빌려오는 수(Borrow)가 산출된다.

-반감산기는 2진수 한자리 두개의 비트를 빼서 그 차를 산출하는 논리 조합회로이다.

입력 X,Y의 차(Difference)와 빌려오는 수(Borrow)를 산출한다.

예를 들어 X가 1이고 Y가 0이라면 차는 1이 되고 빌려오는 수는 산출되지 않는다.

하지만 만약 X가 0이고 Y가 1이라면 차는 1이 되고 빌려오는 수가 산출된다.

피감수 X가 감수 Y보다 작기 때문에 빌려오는 수가 산출된다.

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

\*반감산기의 진리표

이 진리표를 이용해 논리식을 구하면 , 이렇게 구할 수 있다.

도표, 라인, 기술 도면, 스케치이(가) 표시된 사진

자동 생성된 설명

\*반감산기의 회로도

위의 회로도를 보게 되면 차(Difference)는 XOR게이트 1개로 구현했고,

빌려오는 수(Borrow)는 NOT게이트 1개와 AND게이트 1개로 구현한 것을 볼 수 있다.

회로도를 보게 되면 반가산기와 차이점은 NOT게이트가 하나 추가되었다는 것이다.

예시)

도표, 라인, 기술 도면, 스케치이(가) 표시된 사진

자동 생성된 설명도표, 라인, 기술 도면, 스케치이(가) 표시된 사진

자동 생성된 설명

**1**

**0**

**1**

**0**

**1**

**1**

**1**

**0**

도표, 라인, 기술 도면, 스케치이(가) 표시된 사진

자동 생성된 설명도표, 라인, 기술 도면, 스케치이(가) 표시된 사진

자동 생성된 설명

**0**

**0**

**0**

**0**

**0**

**1**

**1**

**0**

-전감산기는 전가산기와 같이 3개의 입력을 받게 되는데 전가산기와 달리

캐리(Carry)가 아니라 빌려오는 수(Borrow)를 받아 뺄셈연산을 하게 된다.

예를 들어, 입력 X, Y가 각각 0과 1이고 B(in)가 1이라면 차는 0이 되고 B(out)가 산출된다. 피감수 X가 감수 Y와 이전 자리의 빌림수 B(in)의 합보다 작기 때문에

B(out)가 산출된다. 또한 X, Y가 각각 1과 0이고 B(in)가 0이라면 차는 1이 되고

B(out)가 산출되지 않는다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 입력 | | | 출력 | |
| X | Y | B(in) | D | B(out) |
| 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 |

\*전감산기의 진리표

이 진리표를 이용해 논리식을 구하면 ,

이렇게 구할 수 있다.

도표, 스케치, 평면도, 기술 도면이(가) 표시된 사진

자동 생성된 설명

\*전감산기의 회로도

위의 회로도를 보게 되면 반감산기 두개와 OR게이트로 구현한 것을 볼 수 있다.

예시)

6(0110) - 3(0011) = ?

**1**

**0**

도표, 스케치, 평면도, 기술 도면이(가) 표시된 사진

자동 생성된 설명도표, 스케치, 평면도, 기술 도면이(가) 표시된 사진

자동 생성된 설명

**1**

**1**

**0**

**0**

**0**

**0**

**0**

**1**

**1**

**1**

**1**

**1**

**1**

**0**

도표, 스케치, 평면도, 기술 도면이(가) 표시된 사진

자동 생성된 설명도표, 스케치, 평면도, 기술 도면이(가) 표시된 사진

자동 생성된 설명

**0**

**0**

**1**

**0**

6(0110) - 3(0011) = 3(0011)

**3.**

BCD는 Binary Coded Decimal의 약자로 이진코드의 십진화를 의미한다.

십진수의 각 자릿수를 이진수의 4bit로 표현한다.

예를 들어 15를 4bit 이진법으로 표현한다면 1111이지만, BCD코드로 표현한다면

0001 0101로 표현할 수 있다. BCD 코드에서는 한자리 수인 0~9까지만을 숫자로 표현하고

그이상의 4비트코드인 10~15(1010, 1011, 1100, 1101, 1110, 1111)에 대해서는

사용하지 않는다.

라인, 도표, 스케치, 화이트이(가) 표시된 사진

자동 생성된 설명

\*BCD코드로 표현한 953

BCD 가산기란 위에서 설명한 BCD코드를 통해 십진수의 덧셈을 하는 논리 조합회로이다.

한 자리의 덧셈에서 결과가 0~9까지라면 그대로 표현하고, 10~15는 BCD로 표현하지

않기 때문에 10부터는 그 결과값에 6(0110)을 더해서 발생한 올림 수를

상위 BCD 코드의 최하위 bit로 보낸다

도표, 라인, 평면도, 텍스트이(가) 표시된 사진

자동 생성된 설명

\*BCD가산기의 회로도

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 이진 합 | | | | | BCD합 | | | | | 십진수 |
| K | Z8 | Z4 | Z2 | Z1 | C | S8 | S4 | S2 | S1 |  |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 2 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 3 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 4 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 5 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 6 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 7 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 8 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 9 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 10 |
| 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 11 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 12 |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 13 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 14 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 15 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 16 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 17 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 18 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 19 |

\*BCD가산기의 진리표

위 진리표를 보면 캐리가 발생하는 경우는 C=K+Z8Z4+Z8Z2로 K=1(16~19), Z8Z4=11(12~15), Z8Z2=11(10~11)임을 확인할 수 있으며 결과값에 0110을 더하고 있음을 볼 수 있다.

**4.**

전가산기 여러 개를 병렬로 연결하여 덧셈과 뺄셈이 모두 가능한 회로를

병렬 가감산기라고 한다. 연결되어 있는 전가산기의 개수에 따라 연산할 수 있는 비트 수도

달라진다.

도표, 라인, 평면도이(가) 표시된 사진

자동 생성된 설명

\*병렬 가감산기 회로도

위의 회로도에서 S가 0이라면 XOR게이트에서 나오는 값은 Y 그대로이기에 덧셈을

수행하고, 1이라면 Y값이 NOT이 되기 때문에 뺄셈을 수행한다. 한 전가산기에서 연산을

할 때 그 결과값에서 캐리(Carry)가 나온다면 그 다음 전가산기에 입력된다. 이 과정을 연속적으로 수행하여 연산한다. 뺄셈을 하기 위해 S에 1이 입력되는데, 이때 그 1은

C0로 입력된다. 그 이유는 어떤 수의 보수를 구하는 과정에서 모든 bit를 반전시킨 뒤

1을 더해줘야 하기 때문이다.

**5.**

위에서 설명한 전가산기같은 경우를 Ripple Carry Adder라고 하는데, 구현이 쉽다는 장점이

있지만 전가산기를 이용해 덧셈연산을 할 때, 앞 bit 전가산기 연산이 끝나 캐리(Carry)를

얻기 전에는 다음 비트의 계산을 수행할 수 없어 연산하는 숫자가 커질수록 연산에 시간이

많이 소요되는 단점이 있다. 이를 해결하기 위해 모든 자리의 덧셈을 한번에 할 수 있도록

한 것이 Carry Look-Ahead Adder(CLA)라고 한다.

입력 X, Y에 대해 CLA는 다음의 연산을 수행해 G와 P를 구한다.

G는 Generation이라고 불리며, 반드시 캐리(Carry)가 생기는 경우,

즉 X와 Y가 모두 1인 경우를 확인한다.

P는 Propagation이라고 불리며, 추가로 캐리(Carry)가 생기는 경우, 즉 X+Y의 결과가 1이고

이전 bit에서 캐리(Carry)가 발생해 새로운 캐리(Carry)가 생기는 경우를 확인한다.

G와 P를 활용하여, 가산기의 합(Sum)과 캐리(Carry)를 아래와 같이 구할 수 있다.

위의 C에 대한 식을 전개하면 아래와 같다.

결과적으로 모든 carry가 위와 같이 기존 연산에 구애받지 않고 처리되기 때문에

전가산기에 비해 빠르게 연산을 수행할 수 있다.

하지만 Ripple Carry Adder보다 더 많은 논리 게이트와 회로가 필요하다는 단점

또한 존재한다.

텍스트, 스크린샷, 폰트, 흑백이(가) 표시된 사진

자동 생성된 설명

\* Carry Look-Ahead Adder의 회로도

**6.**

전파지연시간은 신호가 입력되어서 출력될 때까지의 시간을 말하며, 게이트의 동작속도이다. 각 집적회로의 디지털 논리군에 따라 전파지연시간에 차이가 있다.

기본회로로 NOR게이트를 사용하는 ECL이 가장 전파지연시간이 짧고, 그 다음으로

NAND게이트를 기본회로로 사용하는 TTL이 전파지연시간이 짧다. 그 다음은

단상 트랜지스터인 NMOS와 PMOS를 같이 사용하는 CMOS가 짧으며, 마지막으로

NMOS 트랜지스터가 사용되는 MOS가 가장 전파지연시간이 길다.