6주차 예비보고서

전공: 아트&테크놀로지학과 학년: 4학년 학번: 20191048 이름: 김도솔

**1.**

전가산기(Full Adder)와 반가산기(Half Adder)는 디지털 논리 회로에서 이진 덧셈 연산을 수행하는 회로이다.

1. **전가산기 (Full Adder):** 전가산기는 반가산기와 달리 연산에 Carry 값을 포함하는 덧셈 회로이다. 세 개의 입력(A, B, Carry-In)을 더하고 Sum과 Carry를 출력하며, AND, OR, XOR 게이트로 이루어진다. 전가산기는 Carry를 추가로 처리할 수 있기 때문에 이진 덧셈에서 더 복잡한 연산이 가능해 실질적인 기초 연산회로로 사용된다. 전가산기는 트랜지스터 수준의 회로 또는 여러 게이트의 조합 등 여러 방법으로 구현이 가능하다. 한 가지 예시로 반가산기 두 개와 OR 게이트를 사용해 전가산기를 구현할 수 있다. 우선 두 개의 입력 A, B를 첫 번째 반가산기에 연결하고, 출력된 Sum을 두 번째 반가산기의 입력으로 연결한다. 다음으로 두 번째 반가산기의 다른 입력에 Carry-in을 연결하면 두 번째 반가산기가 최종 Sum을 출력하고, 두 반가산기의 자리올림 수 출력을 OR 연산하면 Carry-out이 된다.

Boolean Function:

진리표와 회로도:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Input | | | Output | | 회로도 |
|  |  |  |  |  |
| 0 | 0 | 0 | 0 | 0 | 블랙이(가) 표시된 사진  자동 생성된 설명 |
| 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 |

1. **반가산기 (Half Adder):** 반가산기는 가장 기본적인 덧셈 회로로, 두 개의 입력(A, B)을 더하고 Sum과 Carry을 출력한다. 반가산기는 전가산기와 달리 덧셈 연산의 입력으로 자리올림(Carry-In)을 고려하지 않아 단순히 XOR과 AND 게이트로 구성된다. 두 개의 입력 A, B를 XOR 게이트에 연결해 Sum을 출력하고, AND 게이트에 연결해 Carry를 출력한다.

Boolean Function:

진리표와 회로도:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input | | Output | | 회로도 |
|  |  |  |  |
| 0 | 0 | 0 | 0 |  |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

**2.**

전감산기(Full Subtractor)와 반감산기(Half Subtractor)는 디지털 논리 회로에서 이진 뺄셈 연산을 수행하는 회로이다.

1. **전감산기 (Full Subtractor):** 전감산기는 반감산기와 달리 자리빌림(Borrow-In) 입력을 고려하는 이진 뺄셈 회로로, 더 복잡한 연산이 가능해 완전한 감산기의 기능을 갖췄다고 볼 수 있다. 세 개의 입력(A, B, Borrow-In)을 받아 연산을 수행하고 차이(Difference)와 빌림(Borrow)을 출력하며, XOR, AND, OR 게이트로 구성된다. 전감산기도 전가산기와 마찬가지로 트랜지스터 수준의 회로 또는 여러 게이트의 조합 등 여러 방법으로 구현이 가능하다. 한 가지 예시로 반감산기 두 개와 OR 게이트를 사용해 전감산기를 구현할 수 있다. 우선 두 개의 입력 A, B를 첫 번째 반감산기에 연결하고, 그 Difference를 두 번째 반감산기의 입력으로 연결한다. 다음으로 두 번째 반감산기의 다른 입력에 Borrow-in을 연결하면 두 번째 반감산기가 최종 Difference를 출력하고, 두 반감산기의 Borrow 출력을 OR 연산하면 Borrow-out이 된다.

Boolean Function:

진리표와 회로도:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Input | | | Output | | 회로도 |
|  |  |  |  |  |
| 0 | 0 | 0 | 0 | 0 |  |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

1. **반감산기 (Half Subtractor):** 반감산기는 두 개의 입력(A, B)에서 뺄셈 연산을 수행하고 Difference와 Borrow를 출력한다. 반감산기는 전감산기와 달리 뺄셈 연산의 입력으로 자리빌림(Borrow-In)을 고려하지 않으며, XOR, AND, NOT 게이트로 구성된다. 두 개의 입력 A, B를 XOR 게이트에 연결해 Difference를 출력하고, 한 입력에 NOT을 취한 후 다른 입력과 함께 AND 게이트에 연결해 Borrow를 출력한다.

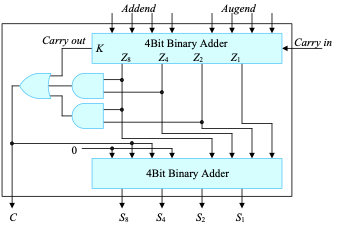
Boolean Function:

1. 진리표와 회로도:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input | | Output | | 회로도 |
|  |  |  |  |
| 0 | 0 | 0 | 0 |  |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 |

**3.**

**BCD 가산기**는 4비트 BCD 코드를 가산하는 논리 회로이다. BCD는 Binary Coded Decimal의 약자로 0부터 9까지의 10진수를 4비트 2진수(0000~1001)로 표현한 것이다. BCD 가산기는 보통 4개의 4비트 BCD 입력(A, B)와 자리올림 입력(Carry-In)을 받고, 그 결과를 4비트 BCD 출력(Sum)과 자리올림 출력(Carry-Out)으로 출력한다. BCD 코드는 2진수와 다르게 표현할 수 있는 범위가 0에서 9로 정해져 있다. 따라서 2진수 합의 결과가 1001(9) 이하인 경우에는 그대로 두고, 1010(10) 이상인 경우에는 6을 더해 결과를 보정해야 한다. 이때 발생한 자리올림 수는 상위 BCD 유닛의 최하위 비트로 보낸다. BCD 가산기는 주로 10진수 숫자를 다루는 디지털 시스템 및 응용 분야에서 사용되며, 주요 응용 분야는 계산기, 디지털 시계, 금융 애플리케이션, 측정 장치 등이 있다.



[BCD 가산기의 회로도]

**4.**

**병렬 가감산기**는 병렬 처리 방식을 사용해 덧셈과 뺄셈을 동시에 수행할 수 있는 디지털 논리 회로이다. 병렬 가감산기를 설명하기에 앞서, 먼저 병렬 가산기의 개념을 이해해야 한다. 병렬 가산기는 전가산기 여러 개를 병렬로 연결해 n bit의 2진수 덧셈 연산이 가능한 회로이다. 병렬 가감산기는 앞서 설명한 병렬 가산기의 입력 B를 부호(Sign)와 XOR 연산해 전 가산기의 입력으로 사용함으로써 Sign bit가 0일 때는 회로가 병렬 가산기로 동작하고, Sign bit가 1일 때는 회로가 병렬 감산기로 동작할 수 있도록 한다. 이 가감산기는 두 개의 이진 입력을 동시에 가산하거나 감산할 수 있어 다양한 응용 분야에서 사용된다. 주요 응용 분야로는 산술 논리 장치(ALU), 컴퓨터 프로세서, 디지털 시계, 계산기 등이 있다.

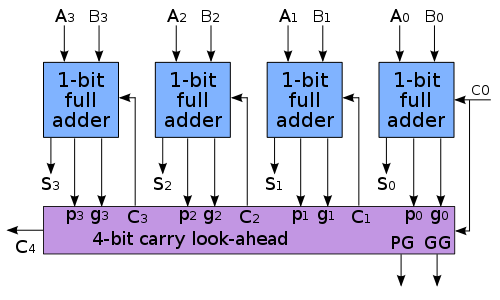
도표, 라인, 그래프, 디자인이(가) 표시된 사진

자동 생성된 설명

[병렬 가감산기 회로도]

**5.**

**Carry Look-Ahead Adder (CLA)**는 이름에서 알 수 있듯이 특별한 예측 논리에 따라 모든 자리올림 수 출력을 한 번에 계산하는 가산기이다. 따라서 하위 자릿수의 자리올림이 계산되기까지 기다리지 않고 각 비트 별로 자리올림을 미리 계산하는 방식으로 작동한다. 이는 각 비트의 자리올림을 병렬로 계산한다는 점에서 Ripple Carry Adder보다 Delay 발생이 적어 더 빠른 연산이 가능하고, 입력 비트 수에 관계 없이 고정된 연산 속도를 가진다. 또한 입력 비트 수와 관계 없이 동일한 수의 논리 게이트를 사용하여 자리올림을 계산하기 때문에 자리올림에 필요한 저장 공간이 비교적 적다는 특징이 있다. 따라서 고속 연산이 필요한 경우나 입력 비트 수가 많은 경우에 유용하게 사용된다.



[4 bit CLA]

반면 **Ripple Carry Adder (RCA)**는 하위 자릿수의 자리올림 계산이 상위 자릿수의 자리올림 계산에 영향을 미치는 방식으로 작동하는 가산기이다. 따라서 한 비트의 연산 지연이 다른 비트에도 연쇄적으로 영향을 미치기에 비트가 추가될 때마다 연산 시간이 증가한다. 즉, 비트 수가 많아질수록 연산 속도가 느려질 수 있다. 또한 각 비트 별 자리올림을 저장하기 위한 공간이 필요하고, 비트 수가 증가할수록 더 많은 수의 논리 게이트가 요구된다는 특징이 있다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

[4 bit RCA]

**6.**

**인코더(Encoder)**는 여러 개의 입력 라인 중 하나의 활성화된 입력을 감지하고 해당 입력을 인코딩하여 하나의 출력으로 표현하는 논리 회로이다. 일반적으로 인코더는 입력 데이터의 형태를 변환하거나 축소하는 데 사용된다. 주로 우선순위 부여, 멀티플렉서(Multiplexer) 선택, 센서 및 스위치 상태 검출 등 다양한 응용 분야에서 사용된다.

ex) 4 to 1 라인 인코더: 4개의 입력 라인 중 하나의 라인만 활성화되어 있을 때 해당 라인을 2비트 이진수로 인코딩한다. 예를 들어, 입력 0010은 출력 10으로 인코딩된다.

**디코더(Decoder)**는 인코더와 반대되는 기능을 수행하는 논리 회로로, 하나의 입력을 받아 여러 개의 출력 중 하나를 활성화시키는 역할을 한다. 디코더는 주로 이진수 또는 다른 형태의 입력 데이터를 해석하고 출력 데이터로 변환하는 데 사용된다. 주로 주소 해석, 데이터 디멀티플렉싱, 제어 신호 생성 등 다양한 응용 분야에서 사용된다.

ex) 2 to 4 라인 디코더: 하나의 2비트 입력을 받아 4개의 출력 라인 중 하나만 활성화한다. 입력 01은 출력 0010을 활성화한다.