6주차 예비보고서

전공: 컴퓨터공학과 학년: 4학년 학번: 20191583 이름: 김태곤

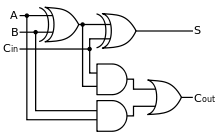
**1. 전 가산기 및 반 가산기에 대해 조사하시오(예시 포함).**

**1) 전 가산기**

전가산기(Full Adder)는 이진수의 덧셈을 수행하는 논리 회로이다. 전가산기는 세 개의 입력 비트를 받아, 두 개의 출력을 생성한다. 세개의 입력 비트는 두개의 더해지는 비트 A, B와 이전 자리에서 올라온 캐리 비트 이고, 두개의 출력 비트는 합과 새로운 캐리 을 나타낸다. 이 구조는 더 큰 이진수 덧셈을 가능하게 하기 위해 여러 개의 전가산기를 연결하여 사용할 수 있다.

전가산기는 일반적으로 두 개의 반가산기와 하나의 OR 게이트로 구성된다. 반가산기는 두 개의 입력 비트를 더하는 기본적인 논리 회로로, 합과 캐리를 출력한다. 첫 번째 반가산기는 A와 B의 합을 계산하고, 두 번째 반가산기는 첫 번째 반가산기의 합과 을 더한다. 마지막으로, 두 번째 반가산기의 캐리와 첫 번째 반가산기의 캐리를 OR 게이트로 결합하여 최종 을 생성한다.

예시))



****

****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A(in) | B(in) |  |  | S |
| 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 |

**2) 반 가산기**

반가산기(Half Adder)는 두 개의 이진수 비트를 더할 때 사용되는 기본적인 논리 회로이다. 이 회로는 두 개의 입력 비트를 받아, 합과 캐리 두 개의 출력을 생성한다. 반가산기는 단순한 덧셈 연산에 사용될 수 있지만, 두 개 이상의 비트 덧셈이나 이전 연산에서 발생한 캐리를 처리할 수는 없다. 반가산기는 기본적인 덧셈 연산을 수행하는 데 사용되며, 더 복잡한 덧셈 작업을 위한 전가산기의 구성 요소로도 활용된다.

**예시))**



|  |  |  |  |
| --- | --- | --- | --- |
| A(in) | B(in) | C | S |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

**2. 전 감산기 및 반 감산기에 대해 조사하시오(예시 포함).**

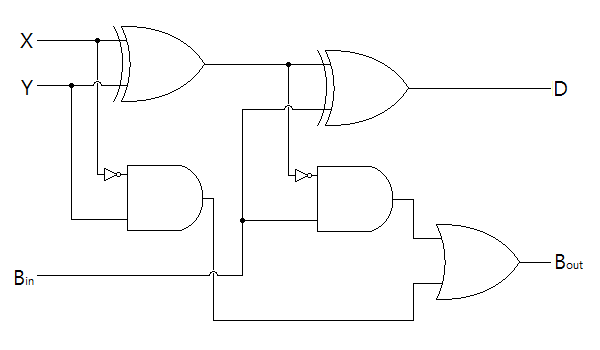
**1) 전 감산기**

전감산기(Full Subtractor)는 두 이진수의 차를 계산하는 데 사용되는 디지털 논리 회로이다. 전감산기는 세 개의 입력과 두개의 출력을 생성한다. 현재 자리의 빼지는 수, 현재 자리에서 빼는 수, 그리고 이전 자리 계산에서 발생한 이전 빌림(Borrow-in)을 받아, 현재 자리의 계산 결과인 차(Difference)와 현재 자리 계산에서 발생한 새로운 빌림(Borrow-out) 두 개의 출력을 생성한다. 전감산기의 논리적 구성은 보통 XOR, AND, NOT 게이트를 사용하여 구현된다. 차는 A, B, 그리고 Borrow-in의 XOR 연산으로 계산된다. 새로운 빌림은 더 복잡한 논리를 사용하여 계산되며, 일반적으로 (A의 보수 AND B) OR (B AND Borrow-in) OR (A의 보수 AND Borrow-in)의 논리 연산으로 이루어진다. 이는 A가 B보다 작아서 다음 자리수로부터 빌림이 필요한 경우, 또는 이전 자리에서 빌림이 발생했고 A와 B가 같은 경우 새로운 빌림이 발생한다는 것을 의미한다. 전감산기는 복잡한 이진수 뺄셈 연산에서 중요한 역할을 하며, 더 큰 수의 뺄셈을 위해 여러 개의 전감산기를 연결하여 사용할 수 있다.





**예시))**



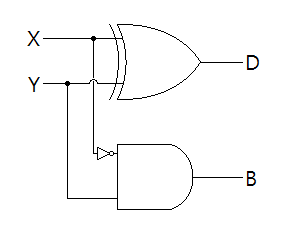
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| X(in) | Y(in) | B(in) | B(out) | D |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

**2) 반 감산기**

반감산기(Half Subtractor)는 두 개의 이진수 비트를 빼는 기본적인 디지털 논리 회로이다. 반감산기는 두 개의 입력과 두개의 출력을 생성한다. 현재 자리의 빼지는 수와 현재 자리에서 빼는 수를 받아, A 와 B의 뺄셈 결과인 차와 현재 자리의 뺄셈에서 빌림이 발생했는지 나타내는 빌림(Borrow)를 출력한다. 차는 입력 A와 B의 XOR 연산으로 계산된다. 빌림은 입력 B가 A보다 클 경우 발생한다. 즉 A의 부정 연산과 B의 AND 연산으로 계산된다.

Borrow = AB

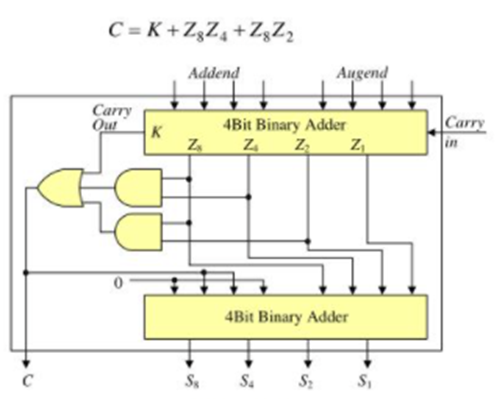
**예시))**



|  |  |  |  |
| --- | --- | --- | --- |
| X(in) | Y(in) | B | D |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 |

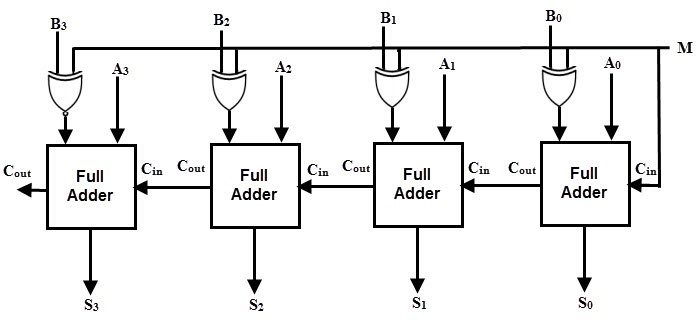
**3. BCD 가산기에 대해 조사하시오.**

BCD(Binary-Coded Decimal) 가산기는 이진수로 표현된 십진수(BCD)를 더하는 데 사용되는 디지털 논리 회로이다. BCD는 각 십진수를 4비트 이진수로 표현하는 방식으로, 0부터 9까지의 숫자를 0000부터 1001까지로 나타낸다. BCD 가산기는 이러한 BCD 형식의 숫자를 정확하게 더하고, 필요한 경우 적절한 십진수 조정을 통해 결과를 BCD 형식으로 유지한다. BCD 가산기의 기본 원리는 두 BCD 숫자를 더하고, 결과가 BCD 형식을 벗어나는 경우(즉, 10 이상이 되는 경우) 십진수 조정을 수행하여 결과를 다시 BCD 형식으로 변환하는 것이다. 이를 위해 BCD 가산기는 일반적으로 두 BCD 숫자를 이진수로서 더하는 기본 이진 덧셈과, 덧셈의 결과가 BCD 형식을 벗어나는 경우 십진수 조정을 통해 결과를 BCD 형식으로 변환하는 십진수 조정 단계를 수행한다.



**4. 병렬 가감산기에 대해 조사하시오.**

병렬 가감산기는 한 번의 연산으로 여러 비트의 숫자를 동시에 더하거나 빼는 기능을 수행할 수 있는 디지털 논리 회로이다. 이 회로는 주로 컴퓨터의 산술 논리 연산 장치 내부에 구현되어 있으며, 고속의 산술 연산을 가능하게 한다. 병렬 가감산기의 설계는 덧셈과 뺄셈을 모두 처리할 수 있도록 유연성을 갖추고 있다. 병렬 가감산기에서 덧셈은 전가산기를 통해 직접 수행된다. 각 전가산기는 두 개의 입력 비트와 이전 자리에서 올라온 캐리를 받아, 해당 자리의 합과 다음 자리로 넘어갈 캐리를 출력한다. 뺄셈의 경우, 2의 보수 방식을 사용할 때는 빼는 수의 각 비트를 반전시킨 후, 전체에 1을 더하여 2의 보수를 생성한다. 이렇게 생성된 2의 보수를 빼지는 수에 더함으로써, 본질적으로 뺄셈을 덧셈으로 전환하여 수행한다.



**5. Carry Look-Ahead Adder 을 Ripple Carry Adder 와 비교하여 설명하시오.**

Carry Look-Ahead Adder(CLA)와 Ripple Carry Adder(RCA)는 두 이진수의 덧셈을 수행하는 회로이지만, 캐리를 처리하는 방식에서 중요한 차이가 있다. 이 차이는 연산 속도와 회로의 복잡성에 영향을 미친다.

CLA는 캐리의 전달 지연을 줄이기 위해 설계된 고속 가산기이다. CLA는 각 비트 위치에서 캐리가 발생할 조건을 미리 계산함으로써, 실제 캐리가 이전 비트로부터 전달되기를 기다리지 않고 미리 다음 캐리를 예측한다. 이는 특정 위치에서 발생하는 조건을 나타내는 Generate와 해당 비트 위치의 덧셈 결과가 이전 캐리를 그대로 다은 비트로 전달하는 조건을 나타내는 Propagate 신호를 사용하여 수행되며, 이러한 신호들은 캐리를 빠르게 예측하여 각 비트의 덧셈을 동시에 수행할 수 있도록 한다.

RCA는 전가산기를 직렬로 연결하여 구성된 가장 기본적인 형태의 병렬 가산기이다. 각 전가산기는 한 비트의 덧셈을 수행하며, 이전 전가산기로부터의 캐리를 입력으로 받는다. 이 캐리는 가장 낮은 비트의 전가산기부터 시작하여 가장 높은 비트의 전가산기로 순차적으로 ripple 되며 전달된다.

CLA가 캐리의 순차적 전달을 기다리지 않고 미리 예측하기 때문에 CLA는 RCA에 비해 연산 속도가 빠르다. 다만 더 많은 논리 게이트와 정교한 설계가 요구되어 CLA거 RCA보다 더 복잡하다. 이러한 특징으로 인해 RCA는 구조가 단순하고 이해하기 쉬워 기초적인 교육용도나 단순한 디지털 시스템에서 선호되고 CLA는 고속 연산이 필요한 애플리케이션에 적합하다.

**6. 기타이론.**

Carry Look Ahead Adder와 Ripple Carry Adder 이외에도 여러 가산기가 존재한다. 그중 Carry Save Adder(CSA)는 다수의 이진수를 더할 때 사용되는 고속 덧셈 회로이다. 이 회로는 주로 세 개 이상의 이진수를 동시에 더하는 데 사용되며, 특히 곱셈 연산의 중간 단계에서 빈번하게 활용된다. CSA의 기본 원리는 덧셈 과정에서 발생하는 캐리를 즉시 다음 계산으로 전달하는 것이 아니라, 저장함으로써 여러 이진수의 합을 동시에 처리하는 데 있다. CSA는 크게 Save와 Carry 단계로 나누어진다. Save 단계에서는 여러 이진수를 더할 때, 각 자리의 비트 합과 캐리를 분리하여 계산한다. 이때, 비트 합은 즉시 다음 단계로 전달되는 반면, 캐리는 저장되어 별도로 처리한다. Carry 단계에서는 저장된 캐리들을 최종 덧셈 결과에 반영한다. 이를 위해 일반적으로 최종 단계에서 별도의 가산기를 사용하여 모든 저장된 캐리와 현재의 비트 합을 더한다. 이 방법은 병렬 처리를 가능하게 하여 덧셈 연산의 속도를 크게 향상시킨다. CSA는 주로 전가산기와 반가산기를 사용하여 구성된다. 이 구성 요소들은 각 자리수의 덧셈을 수행하며, 비트 합과 캐리를 각각 생성한다. CSA의 구성은 입력되는 이진수의 수와 필요한 연산 속도에 따라 다양하게 설계될 수 있다.