6주차 예비보고서

전공 : 국제한국학과 학년 : 4학년 학번 : 20181202 이름 : 김수미

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

**1) 전 가산기 (Full Adder)**

- 값의 덧셈을 연산하는 논리회로이다. 전 가산기는 입력 값 뿐만 아니라 carry값 또한 덧셈 연산에 포함시킬 수 있으며 다양한 연산 회로의 구현에 사용되는 기초 연산 회로이다. 출력은 sum(S)와 carry(C)로 이루어지며, 덧셈 연산에서 발생하는 carry를 carry out(), 이전 Bit 자릿수에서 올라와서 현재 Bit 자릿수의 덧셈 연산에 포함시켜야 하는 carry를 carry in()이라고 한다.

- : input(입력값 2개, )에서 1이 홀수 개이면 Sum = 1

- : input(입력값 2개, )에서 1이 2개 이상이면 = 1

- 예시 :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Input | | | Output | | Logic Schematic 예시 |
|  |  |  |  |  |  |
| 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 (Half Adder)**

- 반 가산기는 2개의 입력과 출력을 가진다. 입력은 2개의 1-bit number이며, 출력은 sum(S)와 carry(C)로 이루어진다. 2개의 1bit수를 더해서 1bit로 표현할 수 있는 범위를 넘으면 이 때 carry가 발생하는데, 반 가산기에서 발생하는 carry값은 다음 자릿수 비트의 덧셈에 영향을 주지 않는다. 즉 반 가산기에서 Carry는 두 input이 모두 1일때만 1이 되며 Sum은 두 input의 값이 서로 다를 때(0, 1 또는 1, 0) 1이 된다. 두 Input이 모두 1인지를 체크하기 위해서는 AND 게이트를 활용할 수 있으며 두 input의 값이 서로 다른지의 여부는 XOR 게이트를 이용해 확인할 수 있다.

- ,

- 예시 :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input | | Output | | Logic Schematic 예시 |
|  |  |  |  |  |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |

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

**1) 전 감산기(Full Subtractor)**

- 가산기가 덧셈 연산에 사용되는 논리 회로였다면, 감산기는 뺄셈 연산에 사용되는 논리회로이다. Borrow를 연산에 포함시키지 않는 반 감산기와 달리, Borrow 발생 여부를 입력에 포함시키기 때문에 완전한 Subtractor의 기능을 갖춘다. 출력은 difference(D)와 borrow(B)로 이루어진다.

- ,

- 예시 :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Input | | | Output | | Logic Schematic 예시 |
|  |  |  |  |  |  |
| 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 감산기(Full Subtractor)**

- 2개의 1bit number 입력을 가지며, 출력은 difference(D)와 borrow(B)로 이루어진다. 각 Bit 자릿수에 맞게 뺄셈 연산을 수행하며, A-B 를 수행한다고 할 때 B의 값이 A보다 크면 앞의 Bit 자릿수에서 Borrow를 가져온다.

- Boolean Function : ,

- 예시 :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input | | Output | | Logic Schematic 예시 |
| X | Y | B | D |  |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 |

**3. BCD 가산기에 대해 조사하시오.**

BCD 가산기란, BCD 코드로 표현된 10진숫자 2개를 입력으로 받아 덧셈을 수행하여 그 결과를 BCD 코드로 출력하는 회로를 의미한다. 여기서 BCD 코드란 Binary Coded Decimal Code를 의미하며, 이는 십진수를 자리 단위로 이진수로 변환한 표현이다. BDC 코드는 2진수와 달리 표현하는 수의 범위가 0에서 9이므로 BCD계산에 있어서 일반 2진수 계산에 따로 보정을 더해줘야 한다.

|  |
| --- |
| BCD 가산기 |
|  |

**4. 병렬 가감산기 에 대해 조사하시오.**

**1) 병렬 가산기** : 전 가산기 여러 개를 병렬 연결한 회로이다. 즉 n개의 전 가산기(FA)로 구성되어 있으며, n Bit의 2진수 A,B의 덧셈 연산을 수행하는 논리회로이다. 병렬 가산기를 이용할 때 발생하는 전파지연(delay)를 줄이기 위해 뒤에서 살펴볼 Carry Look Ahead Adder가 주로 사용되곤 한다.

|  |
| --- |
| 병렬 가산기 |
|  |

**2) 병렬 가감산기** : 병렬 가산기의 입력을 Sign 값과 XOR 하여 다시 전 가산기의 입력으로 사용함으로써 덧셈과 뺄셈을 모두 가능하게 하는 회로이다.

병렬 가감산기의 세부적인 구조를 살펴보면, 맨 아래 입력단의 캐리 입력을 Sign 값으로 간주하고, 각 자리의 B입력과 부호를 XOR 게이트의 입력으로, XOR 게이트의 출력을 전 가산기의 두번째 입력으로 받는다.

|  |
| --- |
| 병렬 가감산기 |
|  |

이 때 XOR 게이트의 동작은 Sign 값이 0이면 원래의 B값이 그대로 출력되고, Sign 값이 1이면 B의 보수 값이 출력된다. 여기서 출력되는 값은 다시 전 가산기의 입력으로 들어간다.

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

Ripple Carray Adder 는 계산을 Serial하게 수행하는 반면 Carry Look-Ahead Adder는 계산을 Parallel하게 수행한다. 따라서 Carry Look-Ahead Adder를 사용하면 Ripple Carray Adder를 사용할 때 보다 발생하는 Delay를 줄일 수 있다.

Ripple Carray Adder에서는 Carry가 다음 비트의 add연산으로 전달되면서 계산이 순차적으로(Serial하게) 진행되지만, Carry Look-Ahead Adder에서는 Carry에 대한 식을 풀어서 계산함으로써 모든 비트의 Carry를 이전 비트로부터의 Carry전달이 필요 없이 한번에(Parallel하게) 계산할 수 있게 되고, 그로 인해 delay시간을 줄일 수 있는 것이다.

예를 들어 일 때, Ripple Carray Adder 에서 각 비트 자릿수 의 을 살펴보자.

, , ,

이 때 이 들을 정리하여 Carry Look-Ahead Adder 에서 사용하는 형식으로 바꾸면

=

와 같이 을 를 이용하여 표현할 수 있는 것이다.

**6. 기타이론**

**1) 전파 지연과 Logical 게이트**

|  |  |  |  |
| --- | --- | --- | --- |
| Gate 종류 | Delay(ns) | Gate 종류 | Delay(ns) |
| NOT gate | 1 | NAND gate | 1.4 |
| Buffer gate | 2 | NOR gate | 1.4 |
| AND gate | 2.4 | XOR gate | 4.2 |
| OR gate | 2.4 | XNOR gate | 3.2 |

이와 같이 다양한 감산기, 가산기를 사용하는 이유는 논리 회로에서 발생하는 전파지연을 최소화시키거나, 값이 비싼 게이트의 사용을 최소한으로 줄이는 등의 방법을 추구해 효율적인 회로를 구성하기 위함이다.