**컴퓨터공학 설계 및 실험Ⅱ**

10주차 예비보고서

서강대학교 공학부 컴퓨터공학 전공

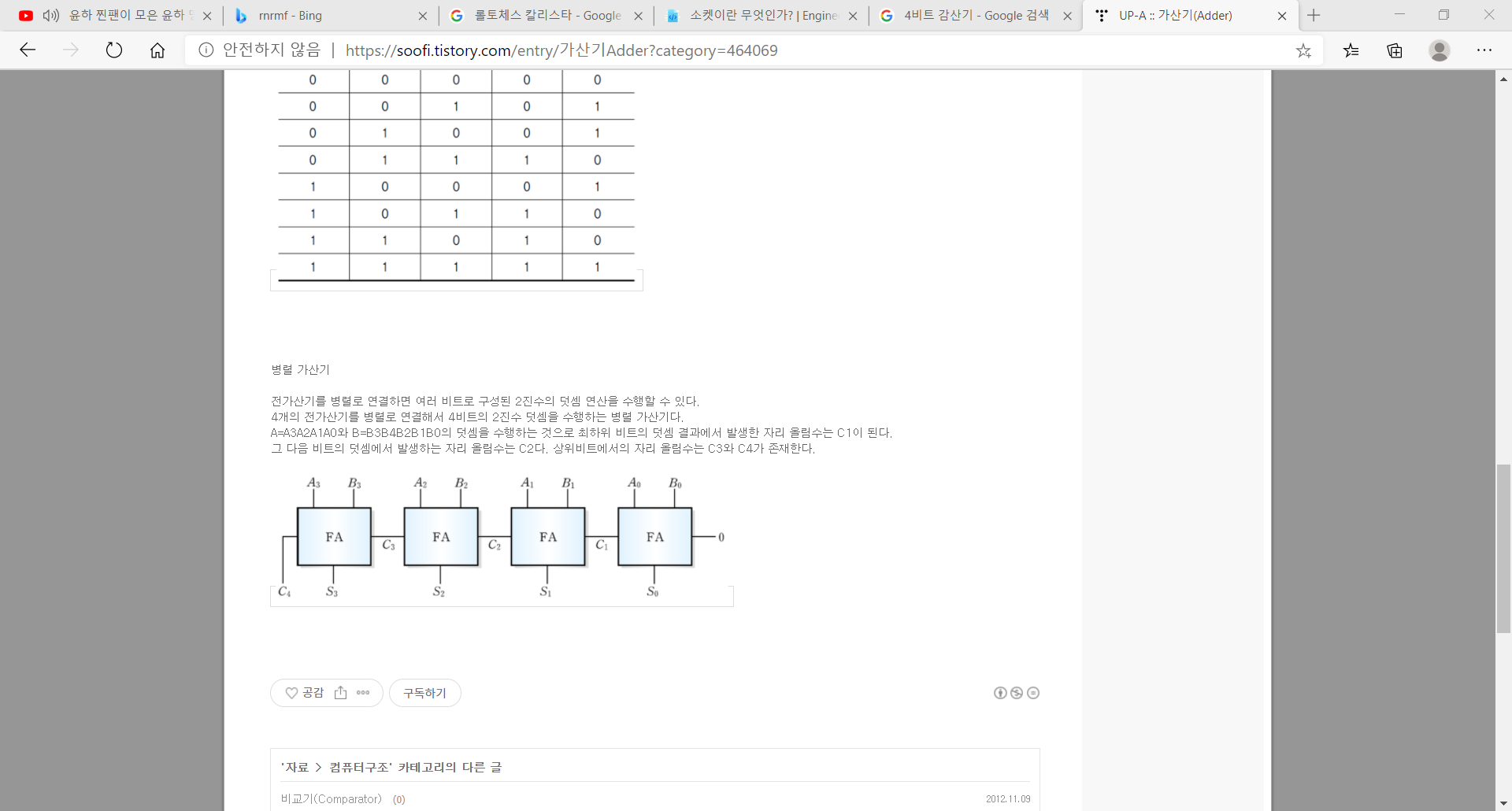
20171646 박태윤

**1. 4-Bit Adder 및 Substractor 이진 병렬 연산 기능에 대하여 조사하시오.**

- 4-Bit Adder

|  |  |
| --- | --- |
| 테이블이(가) 표시된 사진  자동 생성된 설명 | 테이블이(가) 표시된 사진  자동 생성된 설명 |

위의 그림은 전 가산기(Full Adder)의 회로도와 진리표를 나타낸다. A와 B는 계산하고자 하는 1비트의 입력을 나타내며, C0는 이전 자리에서 발생한 Carry를 나타낸다. 계산을 통해 합(S)와 다음 자리 계산을 위해 넘겨주는 올림수(C)가 출력된다.

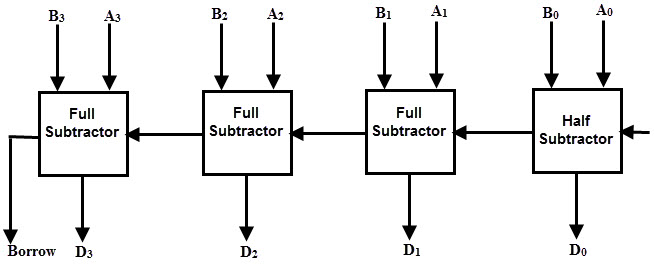


그림과 같이 전 가산기 4개를 병렬로 연결하여 4-bit Adder를 만들 수 있다. 은 Carry를 나타내는데, 한 비트에서 계산이 이루어지고 Carry가 생긴다면 다음 자릿수 비트 계산시에 고려해 줄 수 있도록 다음 전 가산기에 넘겨주는 방식이다. 이를 통해 4-bit 두 수의 합을 구할 수 있다.

- 4 Bit Subtractor

|  |  |
| --- | --- |
| 테이블이(가) 표시된 사진  자동 생성된 설명 | 테이블이(가) 표시된 사진  자동 생성된 설명 |

가산기와 다르게 감산기에서는 Carry대신에 Borrow가 존재한다. Br0는 이전 자리에서 발생한 빌림수를 의미하며 전감산기에서 계산을 통해 다음 비트 계산을 위해 넘겨주는 빌림수 Br과 뺄셈을 한 최종 결과 D가 출력된다.

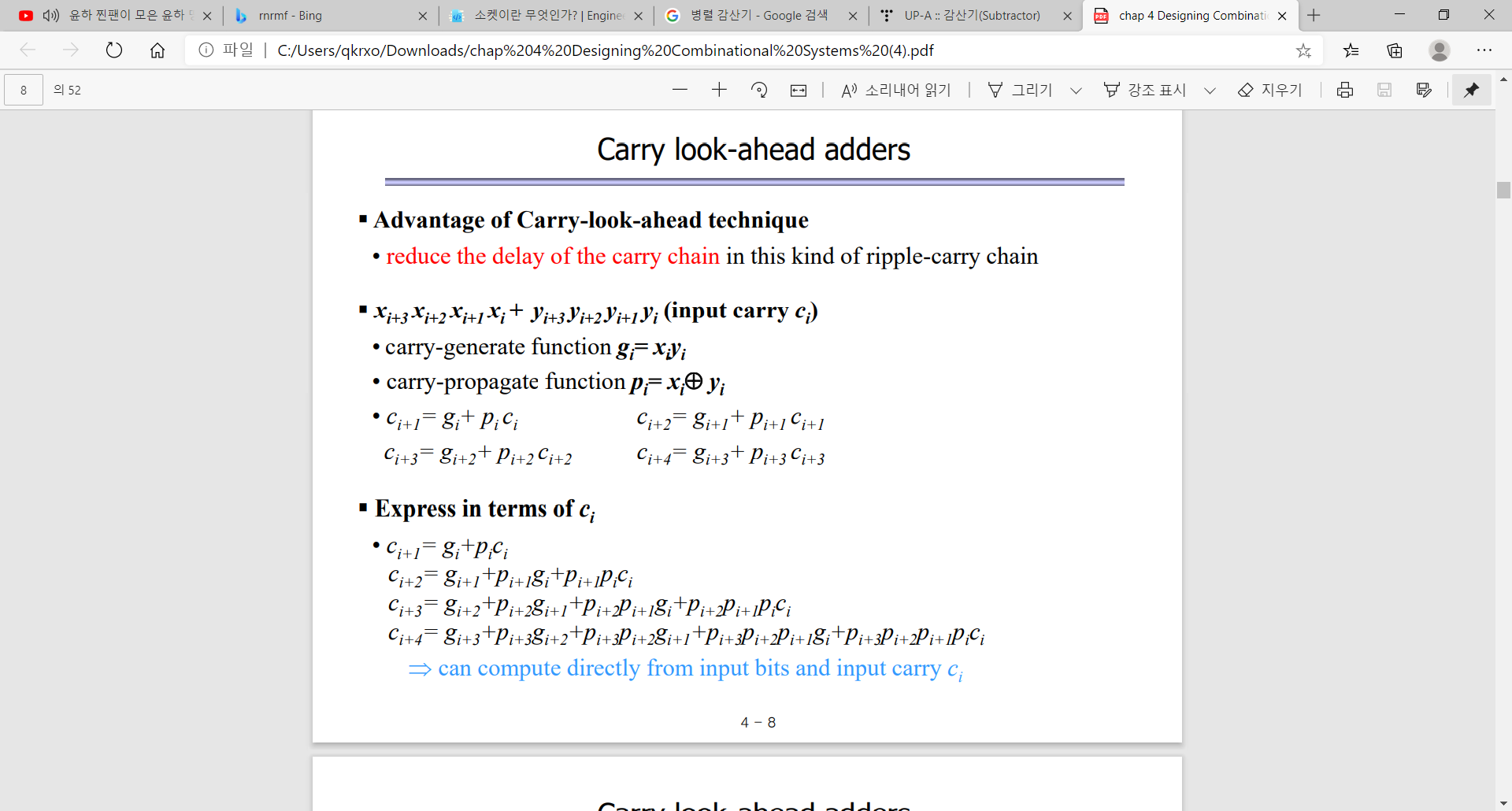


위의 그림과 같이 전감산기 4개를 병렬로 연결하여 4-bit Subtractor를 만들 수 있다.

**2. Look ahead carry 대하여 조사하시오.**

위에서 살펴본 4-bit Adder는 ripple carry adder로 이전 자리 비트에서 Carry를 넘겨주기 전 까지는 해당 비트에서 계산을 못하기 때문에 bit가 커질수록 계산이 매우 느려진다는 단점이 존재한다. 이를 해결하기 위해 자리올림수를 예측할 수 있는 방식이 존재하는데 이를 Look ahead carry라고 한다.

Look ahead adder에서 carry-generate function과 carry-propogate function이 존재한다. 각 함수는 다음과 같은 식을 나타낸다.

****

이런 방식으로 Carry를 기존의 연산을 기다리지 않고 빠르게 계산할 수 있다. 예를 들어 처음에 입력으로 Carry인 c0이 들어온다 가정하면 c1 = g0 + p0c0, c2 = g1 + p1c1 = g1 + p1(g0 + p0c0)이고 이러한 방식을 각 자릿수마다 적용하면 모든 Carry를 gi, pi, c0로 표현할 수 있다. 즉, 모든 Carry를 xi와 yi로 표현할 수 있기 때문에 이전 자리에서 발생한 Carry와 관계없이 모든 자리에서 발생하는 Carry를 계산할 수 있는 것이다.

텍스트이(가) 표시된 사진

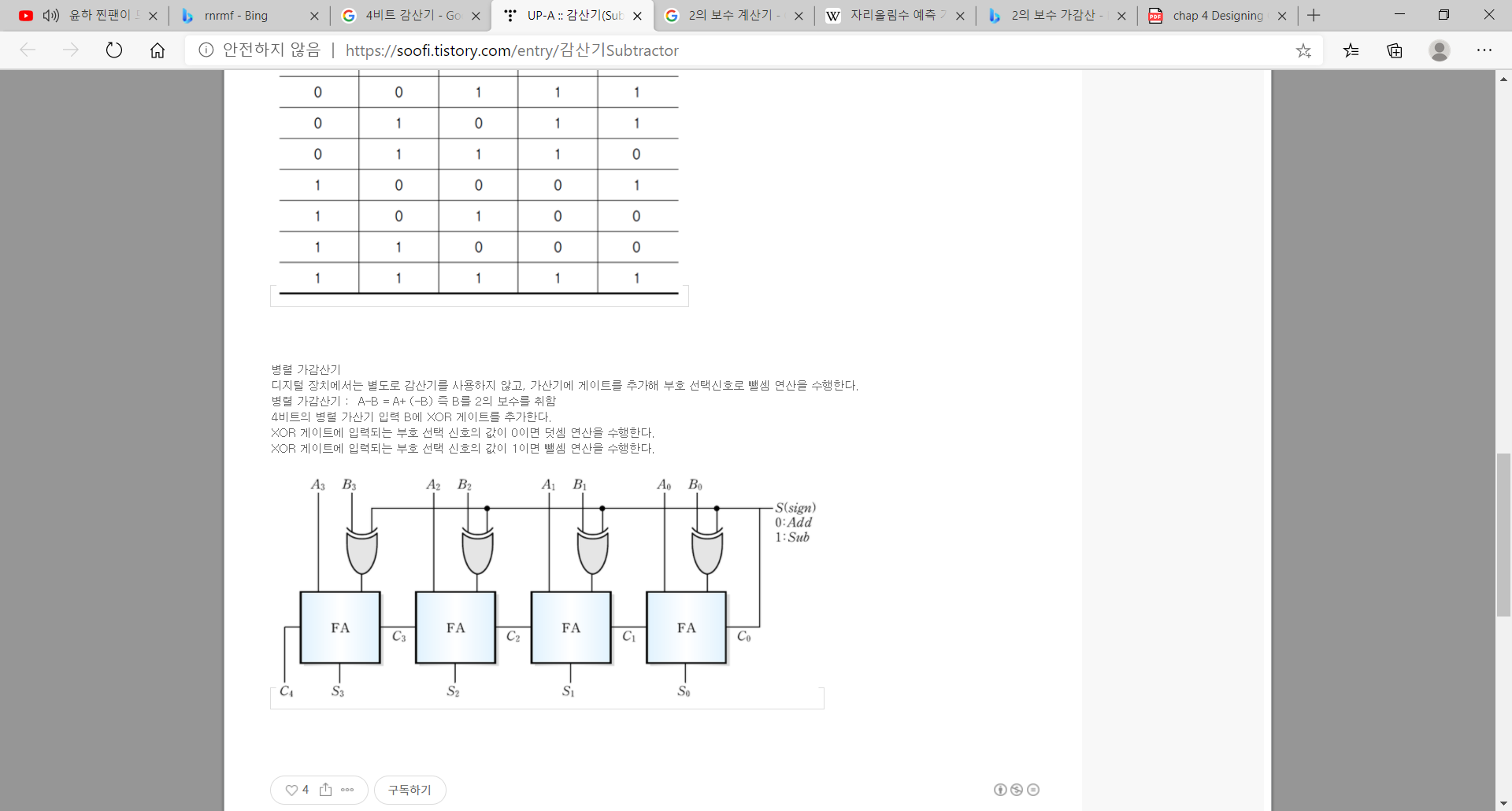
자동 생성된 설명

**3. XOR을 활용한 2’s complement 가감산에 대하여 조사하시오.**

XOR을 활용하여 2’s complement 계산은 부호 선택 신호를 이용하여 진행된다. 예를 들어, 4-bit인 A(A3 A2 A1 A0), B(B3 B2 B1 B0)을 가지고 A-B 혹은 A+B 계산을 한다 가정하면 부호 선택 신호는 다음과 같은 역할을 한다.

|  |  |  |
| --- | --- | --- |
| Bn | S | Bn’ = (Bn⊕S) |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

덧셈인 A+B를 할 때 부호 선택 신호(S)는 0을 나타내며, 뺄셈인 A-B를 할 때 부호 선택 신호는 1을 나타낸다. 즉, XOR게이트를 통해 값을 Inversion시켜 계산을 진행한다.



위의 그림에서 처음 전가산기를 보면 뺄셈을 계산할 때 XOR게이트에 의해 B0가 Inversion되어 B0’가 입력으로 들어가고 S가 1을 나타내므로 처음 들어가는 Carry C0는 1을 나타내는 것을 확인할 수 있다. 따라서 전체적인 계산은 A-B = A+B’+1이 된다.

**4. BCD 연산에 대하여 조사하시오.**

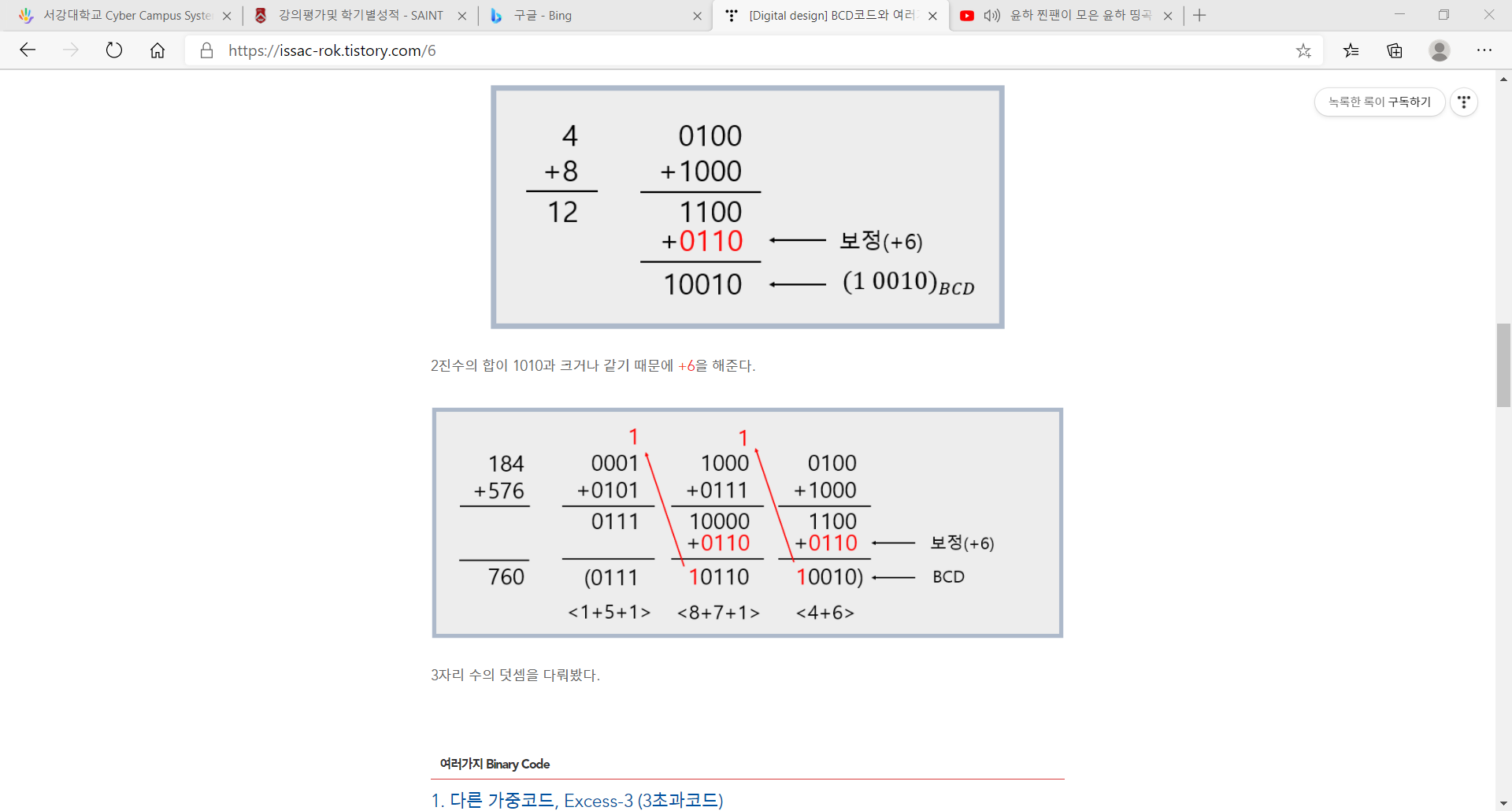
BCD(Binary Coded Decimal)이란 십진수의 각 자릿수를 4비트의 이진수로 표현한 것을 의미한다. 4비트당 0부터 9까지 표현을 한다.

|  |  |  |  |
| --- | --- | --- | --- |
| **Decimal** | **BCD** | **Decimal** | **BCD** |
| 0 | 0000 | 10 | 0001 0000 |
| 1 | 0001 | 11 | 0001 0001 |
| 2 | 0010 | 12 | 0001 0010 |
| 3 | 0011 | 13 | 0001 0011 |
| 4 | 0100 | 14 | 0001 0100 |
| 5 | 0101 | 15 | 0001 0101 |
| 6 | 0110 | 16 | 0001 0110 |
| 7 | 0111 | 17 | 0001 0111 |
| 8 | 1000 | 18 | 0001 1000 |
| 9 | 1001 | 19 | 0001 1001 |

위의 표와 같이 표현할 수 있다. 이렇게 표현된 BCD를 가지고 연산을 할 수 있는데, 일반적인 이진수 연산과 차이점을 보인다.

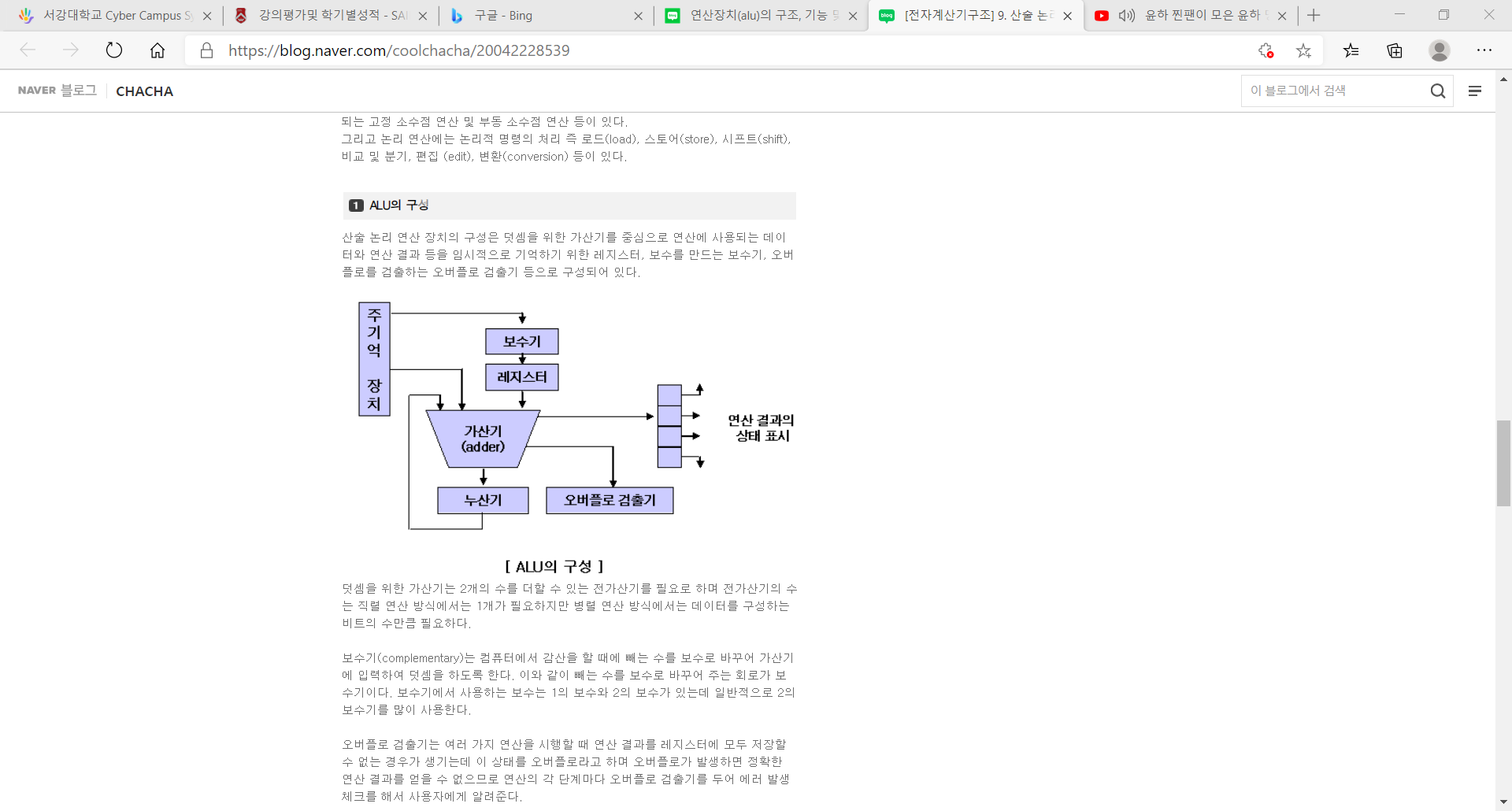
각 자릿수마다 따로 연산을 해주면 되는데, 이 때 연산의 결과가 9이하인 경우는 그대로 결과값으로 사용할 수 있다. 예를 들어, 31+62을 진행한다고 하면 BCD코드에서 0011 0001 + 0110 0010 = 1001 0011(93)으로 표현할 수 있다. 하지만 어떤 자릿수에서 연산의 결과로 10이상인 수가 나온다면 결과값에 6(0110)을 더해주어야 한다. 예를 들어, 18+29은 BCD로 0001 1000 + 0010 1001 = 0011 10001인데, 10001에 6(0110)을 더해주면 10111이고, 현재 5-bit로 이는 자리올림수가 발생한 것이기 때문에 1을 다음 자리로 옮겨주면 최종적으로 결과는 0100 0111(47)이 된다.

같은 방법을 적용하여 다음과 같이 3자릿수 BCD 연산을 진행할 수 있다.



**5. ALU의 기능에 대하여 조사하시오.**

ALU(Arithmetic Logic Unit)은 산술 논리 장치라고도 하며 덧셈, 뺄셈과 같은 산술 연산과 두 수의 크기를 비교하고 판단하는 것과 같은 논리연산을 계산하는 기능을 하는 디지털 회로이다. ALU는 레지스터 부분과 내부 장치 부분으로 나뉘는데, 그 중에서도 내부 장치 부분은 가산기, 보수기, 시프터, 오버플로우 검출기로 구성된다. 가산기(Adder)는 실질적인 산술연산을 수행하는 회로이며 보수기(Complementer)는 보수를 만들어주는 논리 회로이며, 시프터(Shifter)는 2진수의 각 자리를 왼쪽 또는 오른쪽으로 이동시켜주는 회로이고 마지막으로 오버플로우 검출기는 결과값이 해당 레지스터의 용량을 초과했을 때 검출해주는 회로이다. ALU의 구성을 그림으로 표현하면 다음과 같다.



ALU는 다양한 산술 마이크로 동작과 논리 마이크로 동작을 수행하는 디지털 회로이기 때문에 한 마이크로 동작을 선택하기 위해 많은 선택선을 가진다. 선택 변수들을 ALU 내에서 디코드 되어 선택 변수가 n개가 존재한다면 2^n개의 서로 다른 마이크로 동작을 수행할 수 있다.

ALU에서 논리 연산에서는 논리적 명령의 처리 즉 로드(load), 스토어(store), 비교 및 분기, 편집(edit), 변환(conversion)등이 있다. 로드와 스토어, 비교 및 분기는 논리 조작에 속하며 스토어는 같은 정보를 시간이 지난 후 다시 받아오기 위해 연산 장치에서 기억 장치로 옮겨 놓는 것을 의미하고, 로드는 기억 장치로부터 컴퓨터 내부 연산 장치로 가져오는 것을 말한다. 비교는 조건에 맞추어 두 개 데이터의 대소관계를 판별하는 것을 뜻하며 분기는 명령의 실행 순서를 변경하는 것을 말한다. 편집은 zone 10진수를 pack 10진수로 변환하거나 그 역으로 변환하는 기능을 하며 출력을 위한 유효숫자 왼쪽의 0을 지우거나 소수점 또는 콤마를 삽입하는 기능을 의미한다. 마지막으로 변환은 2진수를 10진수로 혹은 그 역으로 변환하거나 코드를 서로 변환하는 기능을 뜻한다.

**6. 기타 이론.**

레지스터(Register)란 CPU가 요청을 처리하는 데 필요한 데이터를 일시적으로 저장하는 기억장치이다. 즉, 명령을 처리하기 위해서 주소와 명령의 종류를 저장할 수 있는 기억 공간이 레지스터이다. 레지스터는 공간이 작지만 CPU와 직접 연결되어 있으므로 연산 속도가 메모리보다 굉장히 빠르다는 특징을 지니고 있다. 레지스터는 플립-플롭의 집합이라고도 할 수 있는데, 플립-플롭이란 1bit의 정보를 저장할 수 있는 회로를 의미하며 이전 상태를 계속 유지한 채 저장을 할 수 있다.

레지스터는 크게 General Purpose Register와 Special Purpose Register로 나눌 수 있다. General Purpose Register는 외부메모리에 쓰거나 읽을 때 데이터가 들어 있는 주소를 가르키는 값을 넣어두는 Adderess Register와 외부 메모리에서 읽어온 값을 임시 저장하는 Data Register, 외부 메모리에서 읽어온 명령어를 저장하는 Instruction Pipeline Register등으로 구성된다. Special Purpose Register는 현재 실행되고 있는 프로세스의 주소를 가리키는 Program Counter, 현재 사용되고 있는 프로세스의 스택 여역에서 마지막 데이터가 Push된 곳의 주소를 가리키는 Stack Pointer, jump명령을 수행했을 경우 돌아갈 곳의 주소를 가리키는 Linked Register, MCU의 현재 상태를 나타내는 Status Register등으로 구성된다.