**10주차 예비보고서**

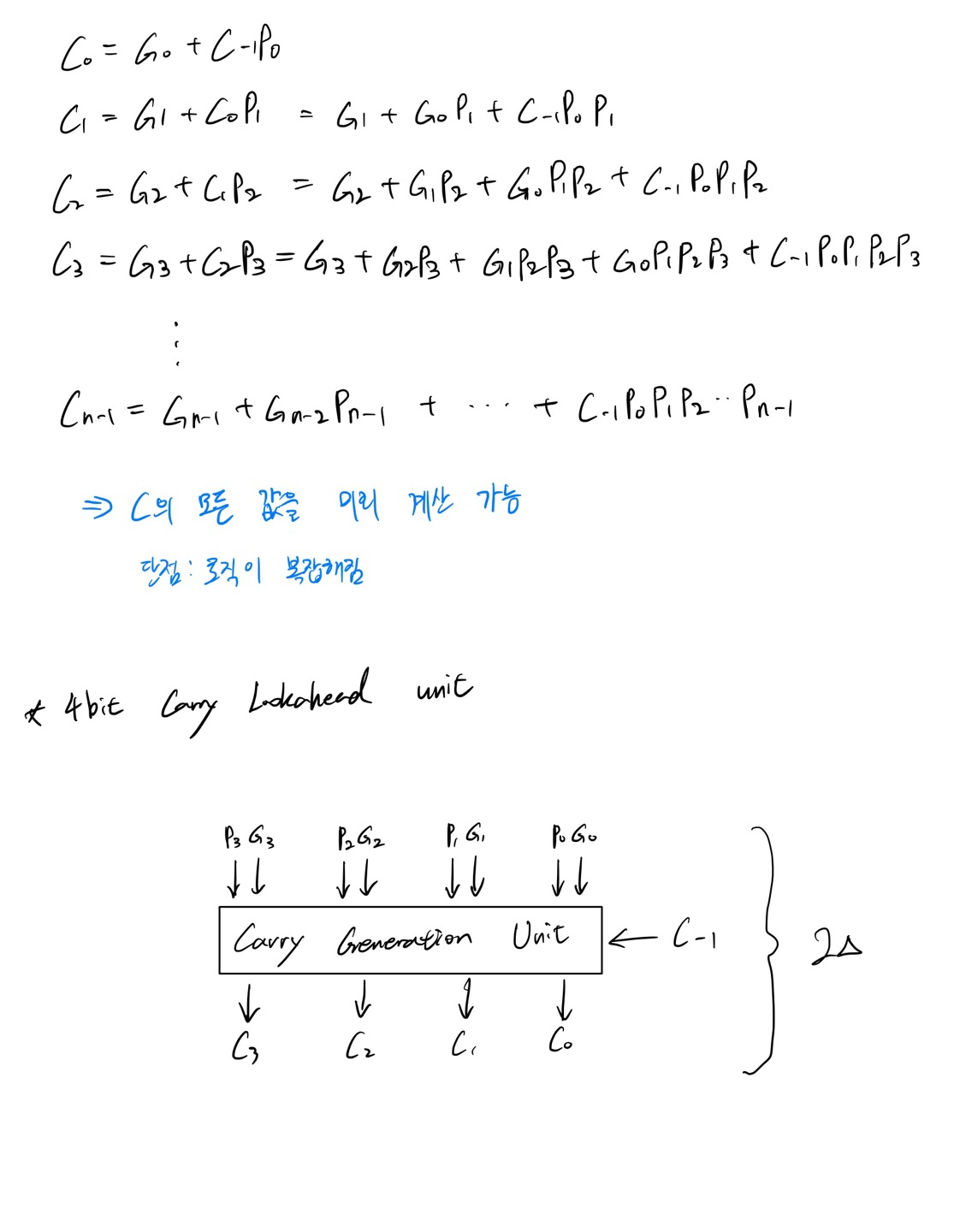
**전공: 경영학과 학년: 3학년 학번: 20190963 이름: 한다현**

**1.**

**4-bit adder는 4개의 전가산기로 이루어져 있으며 가수 비트들과 피가수 비트들이 동시에 입력된다. 즉, 가수를 A, 피가수를 B라고 한다면 첫번째 전가산기에는 A0와 B0가 입력되고, 두번째 전가산기에는 A1과 B1이 입력되며, 세번째 전가산기에는 A2와 B2가 입력되고, 네번째 전가산기에는 A3와 B3가 동시에 입력된다. 또한 각각의 전가산기의 출력 자리 올림수는 다음 전가산기의 입력 자리올림수로 사용된다. 즉, 첫번째 전가산기에 입력되는 carry를 C0라고 한다면 두번째 전가산기에 입력되는 carry는 첫번째 전가산기에서 출력된 C1이 되고, 세번째 전가산기에 입력되는 carry는 두번째 전가산기에서 출력된 C2가 되고, 마지막 전가산기에 입력되는 carry는 세번째 전가산기에서 출력된 C3가 되며 최종적으로 출력되는 carry는 C4가 된다. 그리고 전가산기에서 출력되는 4개의 출력 S0, S1, S2, S3는 각각의 전가산기에 입력된 두 입력인 A0~A3과 B0~B3의 합이 된다. 4-bit subtractor의 이진 병렬 연산 기능도 이와 유사하다. 4-bit adder와 subtractor는 보통 하나의 회로를 사용하는데, 4비트 이진수인 A와 B의 차를 구하고자 할 때 연산 방법은 B의 2의 보수를 구한 뒤 A에 더하는 방식이기 때문에 감산기가 아닌 전가산기 4개가 필요하다. 이러한 회로에서 가산과 감산을 구분하기 위해서 제어 입력 C가 필요하다. C가 0일 때 회로는 가산기로 동작하고, C가 1일 때 회로는 감산기로 동작한다. C가 1일 때 회로가 감산기로 동작하는 이유는 2의 보수를 구할 때, 1의 보수를 먼저 구한 뒤 최하위 비트에 1을 더해주어야 하기 때문이다. 따라서 B의 1의 보수를 인버터로 구현하고, 제어 입력인 1을 더해주면 2의 보수를 구할 수 있다. 이렇게 구현된 4비트 감산기는 4비트 가산기와 마찬가지로 A0~A3과 B0~B3을 각각의 전가산기에 입력 받고 carry를 다음 전가산기의 입력 carry로 전달하며 각각 S0~S3를 출력한다.**

**2.**

**1번에서 조사한 4 비트 병렬 가산기는 첫번째 전가산기에서 출력된 carry가 다음 전가산기에 입력되고 다시 그 전가산기에서 출력된 carry가 다음 전가산기에 입력되는 방식이다. 이렇듯 carry가 순차적으로 입력되기 때문에 지연 시간이 발생하는데, 이를 전달지연이라고 한다. 이 캐리전파지연 시간을 최소화하기 위해 병렬 가산기에서 가장 널리 사용되는 방법이 look ahead carry이다. Look ahead carry를 사용하는 병렬 가산기는 carry propagate와 carry generate 변수를 사용하는데 Pi = Ai** ⊕ **Bi이고 Gi = Ai\*Bi이다. 이를 활용해 출력의 합과 캐리를 표현하면 Si = Pi** ⊕ **Ci-1, Ci = Gi +Pi\*Ci-1이다. 이제 C0부터 Cn-1까지를 위의 식을 적용하여 정리하면 다음과 같다.**



**정리된 C0 ~ Cn-1을 보면 알 수 있듯이 가장 처음 가/감산기에 입력되는 C-1의 값만 있으면 모든 carry의 값을 동시에 계산할 수 있다. 즉, look ahead carry 방법을 사용하면 이전에 설계했던 병렬 가/감산기와 달리 이전 전가산기에서 출력된 carry를 기다릴 필요 없이 모든 carry가 동시에 전파된다는 것이다. 따라서 look ahead carry 기술은 각각의 전가산기에 입력되는 모든 캐리를 동시에 계산할 수 있고, 병렬 가/감산기의 캐리 전파 속도를 높이며 출력되는 모든 합(S)이 모두 같은 전파지연시간을 갖는다는 특징이 있다.**

**3.**

**1번과 같이 4비트 병렬 가/감산기는 제어 입력 C의 값에 따라 가산 기능을 수행하기도 하고 감산 기능을 수행하기도 한다. 두 이진수 A, B가 있을 때, A+B를 연산할 때는 A와 B가 모두 그대로 전가산기에 입력되어야 하지만 A-B를 연산할 때는 A는 그대로 전가산기에 입력되지만 B는 2의 보수를 취한 후에 전가산기에 입력되어야 한다. 이러한 기능을 수행하는 논리 게이트가 XOR 게이트이다. B의 비트들이 각각의 전가산기에 입력되기 전에 XOR 게이트를 거쳐야 하는데, 이 때 XOR 게이트의 입력은 B의 비트 Bi와 제어 입력 C이다. C가 0인 경우, 즉 가산 기능을 수행해야 하는 경우에는 Bi의 값이 변하지 않고 그대로 입력되어야 한다. C가 0이면, XOR 게이트의 결과는 항상 Bi**⊕**0은 항상 Bi가 된다. Bi가 0일 때, 0**⊕**0은 0이고 Bi가 1일 때 1**⊕**0은 1이기 때문에 XOR 게이트의 결과는 항상 Bi와 동일한 것을 확인할 수 있다. 반대로, C가 1인 경우에는 가산 기능을 수행하는데, 이때는 Bi의 값이 Bi의 부정이 되어야 한다. 이 역시도 XOR 게이트를 사용하여 해결할 수 있는데, C가 1이고 Bi가 0일 때, 0**⊕**1은 1이 되고, Bi가 1일 때, 1**⊕**1은 0이 되므로 XOR 게이트의 출력 값이 Bi의 부정인 것을 확인할 수 있다. 따라서 Bi와 C가 입력인 XOR 게이트를 사용하면 C가 0일 때는 전가산기에 Bi가 입력되고, C가 1이면 전가산기에 Bi의 부정이 입력되는 것을 알 수 있다.**

**4.**

**BCD 연산 중에는 덧셈과 뺄셈이 있다. BCD의 덧셈부터 살펴보면, BCD의 덧셈기는 2개의 4비트 BCD 수와 자리올림 입력 C, 합을 의미하는 출력 Z, 그리고 자리 올림 출력 K로 구성되어 있다. BCD 덧셈을 수행하는 방법은 다음과 같다. 먼저 두 개의 4비트 2진수를 더한 후 그 값이 9보다 크면 합에 6을 더하고, 크지 않으면 그대로 출력한다. 예를 들어, 두 이진수 0101과 0011이 입력된 상황을 가정하면 0101은 10진수로 5를 의미하고, 0011은 10진수로 3을 의미한다. 따라서 두 이진수의 합인 1000은 8을 의미하고 이는 9보다 크지 않기 때문에 올바른 BCD 합이라고 할 수 있다. 반면 0111과 0101이 입력된 상황에서 0111은 10진수로 7을 의미하고 0101은 10진수로 5를 의미하기 때문에 두 이진수의 합인 1100은 10진수로 12를 의미한다. 이는 9보다 크기 때문에 6을 의미하는 이진수인 0110을 더해주어야 하고, 그 결과는 10010이다. 따라서 1의 자리올림 출력이 발생하고 합을 의미하는 Z는 0010이 된다. 다음으로 BCD의 뺄셈을 살펴보면, 뺄셈의 출력은 뺄셈의 결과인 S와 결과의 부호를 의미하는 C가 있다. 먼저 BCD의 뺄셈을 수행할 때는 9의 보수가 필요하다. 두 10진수 X, Y가 있을 때 X-Y를 수행하기 위해서는 Y에 9의 보수를 취해야 하는데 이 때 Y의 9의 보수 Y’는 9-Y이다. 즉, Y가 0일 때 Y’은 9이고, Y가 1일 때 Y’은 8이 되는 방식이다. 이제 BCD 뺄셈의 예를 들어보면, X가 5이고 Y가 2일 때 X-Y를 수행하려면 Y에 9의 보수를 취한 뒤 X에 더해주어야 한다. 2의 9의 보수는 7이므로 5를 의미하는 0101과 7을 의미하는 0111을 더하면 12를 의미하는 1100이 출력된다. 이는 9보다 큰 값이므로 다시 6을 더해주면 10010이 되는데, 이 때 제일 앞에 있는 1이 C의 값이 되고 이는 양수를 의미한다. 나머지 4비트 이진수 0010에 1을 더해주면 0011이 되고 이는 3을 의미하기 때문에 올바른 출력 값을 얻을 수 있다. 반면 X가 5이고 Y가 9일 때 X-Y를 수행하려면 Y에 보수를 취해줘야 하는데, 9의 보수는 0이다. 5를 의미하는 이진수 0101과 0을 의미하는 이진수 0000을 더하면 0101이 되는데, C의 값이 0이므로 음수를 뜻한다. 출력 S는 0101의 9의 보수인 0100이 되고 이는 -4를 의미하기 때문에 올바른 출력 값을 얻었다고 할 수 있다.**

**5.**

**ALU란 arithmetic logical unit의 줄임말로 산술 논리 장치를 의미한다. 컴퓨터 시스템은 주기억장치, 중앙처리장치를 의미하는 CPU, 입출력 제어기 등으로 구성되어 있으며 그 중 CPU는 다시 제어장치와 산술 논리 장치로 구성되어 있다. 이 산술 논리 장치가 바로 ALU이다. ALU는 레지스터와 논리 부분의 작업을 수행하며, 일반적으로 단일 ALU는 덧셈과 뺄셈 등의 산술 동작과 AND, OR, XOR 게이트의 논리동작 수행이 가능하다. ALU의 기능을 더 자세히 살펴보면, 고정 소수점 수의 덧셈과 뺄셈, AND나 OR와 같은 비트 논리 연산, 순환연산과 쉬프트 연산 등의 기능을 수행할 수 있다. 이 때 순환 연산이란 비트들의 집합의 자리를 이동시키는 동작 연산을 의미한다. 또한 ALU는 덧셈, 뺄셈, 자리 이동 연산의 수행을 통해 곱셈과 나눗셈의 기능을 수행하기도 하고, 고정 소수점 수의 연산이 아닌 부동 소수점 수의 연산 기능을 수행하기도 한다. 프로그램이 실행될 때, 컴퓨터의 메모리에 프로그램에 의해 처리될 프로그램 명령어와 데이터가 저장되고, 제어 장치에 의해 복호화 된 명령어가 ALU에 의해 실행되는 방식으로 프로그램이 실행된다. 컴퓨터는 자신의 명령어의 집합인 레퍼토리를 갖는데, 이 레퍼토리에 있는 명령어가 AUL가 수행할 산술, 논리, 비교 등의 동작을 결정한다.**

**6.**

**컴퓨터 구조를 설계할 때 영향을 미치는 중요한 개념이 워드이다. 워드(Word)란 주소를 지정하는 가장 작은 저장단위를 의미한다. 워드는 일반적으로 16 bit에서 64 bit 사이의 크기를 갖는데, 이는 컴퓨터 설계에 따라 달라질 수 있다. 대부분의 CPU 내부 레지스터는 워드 하나의 크기를 갖고 내부/외부버스는 한 번의 수행에서 워드 하나 크기의 데이터를 이동시킬 수 있다. 워드의 크기를 크게 하면 CPU 하드웨어와 버스를 구현하는 비용도 그만큼 증가한다는 단점이 있지만, 큰 워드 크기는 처리 속도를 높일 수 있다는 장점이 있다. 따라서 CPU를 설계할 때, 구현 비용, 처리 속도, 효율 등을 따져보고 적절한 워드 크기를 설정해야 한다.**

**참조:**

**디지털 논리 설계(황의철)**

**디지털 논리 회로 설계(이상범)**

**디지털 논리 및 컴퓨터 설계(McCalla)**

**논리회로와 컴퓨터 설계(Alan Marcovitz)**