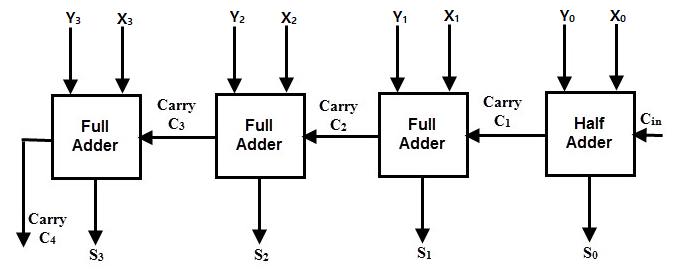
1. 10주차 예비보고서 20141196 김성희

**4-Bit Adder/Subtractor (이진 병렬 가/감산기)**

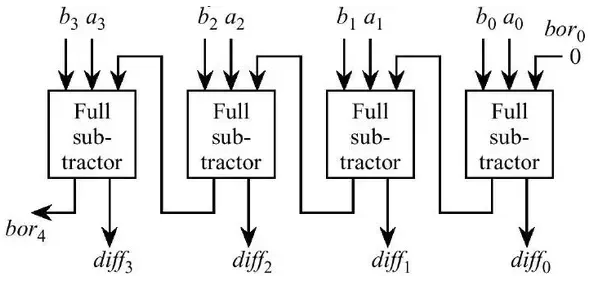


**전가산기 하나당 Cout=XY+Cin(XⓧY), S=XⓧYⓧCin4bit 이진수 두개의 합과 차를 구하는 회로다. 최종 carry인 C4를 구하기까지 순서대로 4개의 전가산기(FA)를 거쳐야 한다.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **X** | **Y** | **Cin** | **S** | **Cout** |
| **0** | **0** | **0** | **0** | **0** |
| **0** | **0** | **1** | **1** | **0** |
| **0** | **1** | **0** | **1** | **0** |
| **0** | **1** | **1** | **0** | **1** |
| **1** | **0** | **0** | **1** | **0** |
| **1** | **0** | **1** | **0** | **1** |
| **1** | **1** | **0** | **0** | **1** |
| **1** | **1** | **1** | **1** | **1** |

**-전가산기(FA) input output 진리표-**

**a-b = (a3-b3, a2-b2, a1-b1, a0-b0)**

****

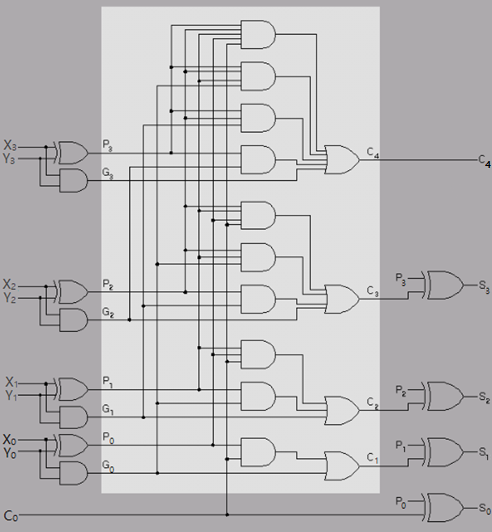
**전가산기 하나당 borout=a’b+bborin+a’borin(=a’b+borin(a’ⓧb) ), diff = aⓧbⓧborin4bit 이진수 두개의 합과 차를 구하는 회로다. 최종 carry인 C4를 구하기까지 순서대로 4개의 전가산기(FA)를 거쳐야 한다.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **a** | **b** | **borin** | **diff** | **borout** |
| **0** | **0** | **0** | **0** | **0** |
| **0** | **0** | **1** | **1** | **1** |
| **0** | **1** | **0** | **1** | **1** |
| **0** | **1** | **1** | **0** | **1** |
| **1** | **0** | **0** | **1** | **0** |
| **1** | **0** | **1** | **0** | **0** |
| **1** | **1** | **0** | **0** | **0** |
| **1** | **1** | **1** | **1** | **1** |

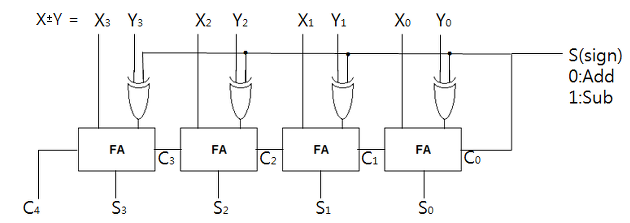
**-전감산기(FA) input output 진리표-**

**Look ahead carry**

**이름 그대로 carry를 내다본다는 의미이다. 이진 병렬 가산기와는 달리 순차적으로 C2, C3, C4를 구하지 않고 바로 C2, C3, C4를 구해낸다. 방법의 착안은 전가산기의 S와 Cout의 논리식에 XⓧY와 XY 그리고 Cin의 곱과 합으로 이루어져 있는 것에서 시작된다.  
Pi = XiⓧYi Gi = XiYi  
Si+1 = PiⓧCi Ci+1 = Gi + PiCi 이며  
C1 = G0+P0C0  
C2 = G1+P1C1 = G1+P1(G0+P0C0) = G1+P1G0+P1P0C0C3= G2+P2C2= G2+P2(G1+P1G0+P1P0C0) = G2+P2G1+ P2P1G0+ P2P1P0C0C4 = G3+P3C3= G3+P3(G2+P2G1+ P2P1G0+ P2P1P0C0) = G3+P3G2+ P3P2G1+ P3P2P1G0+ P3P2P1P0C0이다.  
따라서 기존의 전가산기는 C1를 구하는 데에 3개의 게이트, C2, C3, C4를 구하는 데에 각각 2개의 게이트를 거치기 때문에 총 7개의 게이트를 거쳐야 결과가 나오지만, Look ahead carry 가산기는 C1,2,3,4가 동시에 구해지므로 총 3개의 게이트를 거치는 시간 안에 구할 수 있다.  
(Critical path(걸리는 시간에 영향을 미치는 계산들)은 carry를 구하는 것이다.)**

****

**XOR를 활용한 2’s complement 가감산**



**전감산기가 전가산기에서 파생된 이유는 수체계로 2’s complement를 이용하고 있기 때문이다. X-Y이 X+NOT(Y)+1과 동일하기 때문이다. 위 전가/감산기의 그림에서 XOR gate 부분이 NOT(Y)에 해당한다. (1bit -> XOR=NOT) 그리고 C0에 1을 넣어주는 부분이 NOT(Y)+1의 +1 부분이다.**

**-Y가 NOT(Y)+1과 같아진 상세한 이유는 다음과 같다.  
…, 11110 = -2, 1111 = -1, 0000 = 0, 0001 = 1, 0010 = 2, … 와 같이 2’s complement 수 체계에서는 음수 부분은 1111을 -1로 시작하여 1씩 감소시킨다. (-2 = -1-1 = 1111-1 = 1110) NOT(0)이 0이 아닌 -1이고, NOT(1)이 -1이 아닌 -2인 것처럼 NOT을 취했을 때 원래 기대하던 결과값보다 1이 더 작은 값이 나오므로 +1을 취해서 기대하던 결과값을 만들어 낸 것이다. (즉 요약하면 NOT(Y) = -Y-1 이므로 -Y를 구하기 위해 NOT(Y) + 1을 해주는 것이다.)**

**BCD연산**

**- 우리가 흔히 알고 있는 10진수의 수의 덧셈을 2진수를 통해 구현한 연산이다. 예를 들어서 6+9=15는 첫째 자리의 1과 둘째 자리의 5의 조합이다. 마찬가지로 이를 2진수를 통해 구현하면 다음과 같다. 0101 + 1001 = 0001 0101.**

**(4비트 크기의 2진수를 선정한 이유는 십진수를 각 자리마다 나타내면 0~9가 전부이고 이를 표현하기 위해서는 4비트면 충분하기 때문이다.)**

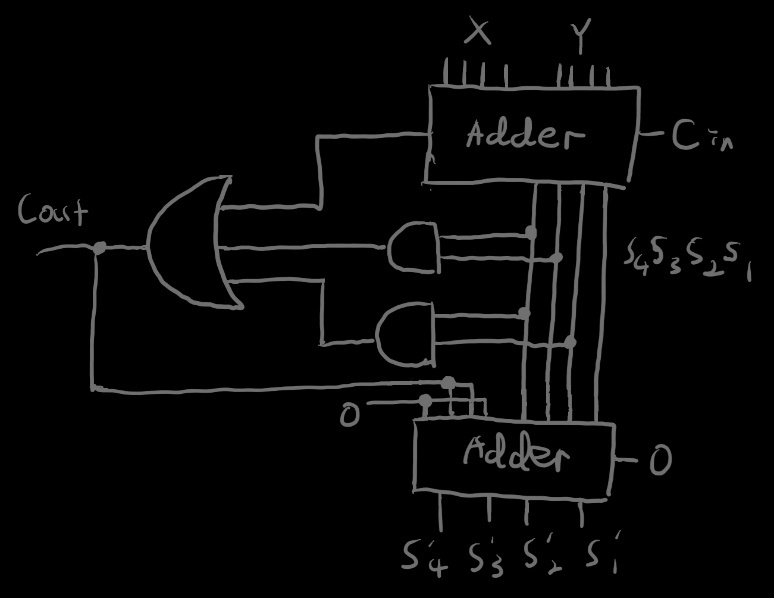
**- 다음 두자리수의 덧셈을 살펴보자**

**53 + 68 = 121을 BCD연산으로 푼다면 (0101 0011) + (0110 1000) = (0001 0010 0001)과 같이 계산할 수 있다.**

**- 이제 회로를 구현해보자.**

**1. 한 자리 수를 아무리 더해도 18을 넘길 수 없다.  
2. 0~9는 변화 없이 그대로 사용된다.  
3. 10이상의 결과는 두 개로 쪼개어진다.  
4. 두 개로 쪼개는 방식은 1을 앞으로 빼내고 원래 결과에서 10을 빼면 된다.**

**위 4가지 규칙을 만족하기 위해서는 4비트 이진 병렬 가산기를 통해 덧셈을 구한 뒤 모종의 회로를 통해 10이 넘는지 파악하고 만약 10이 넘는 다면 10을 빼고 앞에 0001을 추가로 덧붙여주고 아니라면 그대로 넘겨야 한다. 10이 넘는지 판별하는 방법은 4비트 가산기 한 개를 통해 나온 최종 carry가 1인지 보거나 결과값 중에서 4번째 자리(8421중 8의 자리)와 3번째 자리(4의 자리) 또는 2번째 자리(2의 자리)에 1이 있는지를 보면 된다. 즉 C4 +** **S4S3 + S4S2의 값이 1이면 10이상의 결과이고 0이면 10미만의 결과다. 그리고 10이 넘었을 때 결과값에서 10을 빼는 행위는 다음과 같다. S4 S3 S2 S1 의 결과에서 0110을 더하면 된다. (10은 0으로, 11은 1로, 12는 2로 바꾼다. 그리고 0은 1 0000으로도 표현할 수 있다. 앞의 1은 탈락되기 때문이다. 즉 1010에 어떤 값을 더해서 1 0000으로 만들어 주려면 0110을 더하면 된다.) 따라서 회로는 다음과 같이 나타낼 수 있다.**

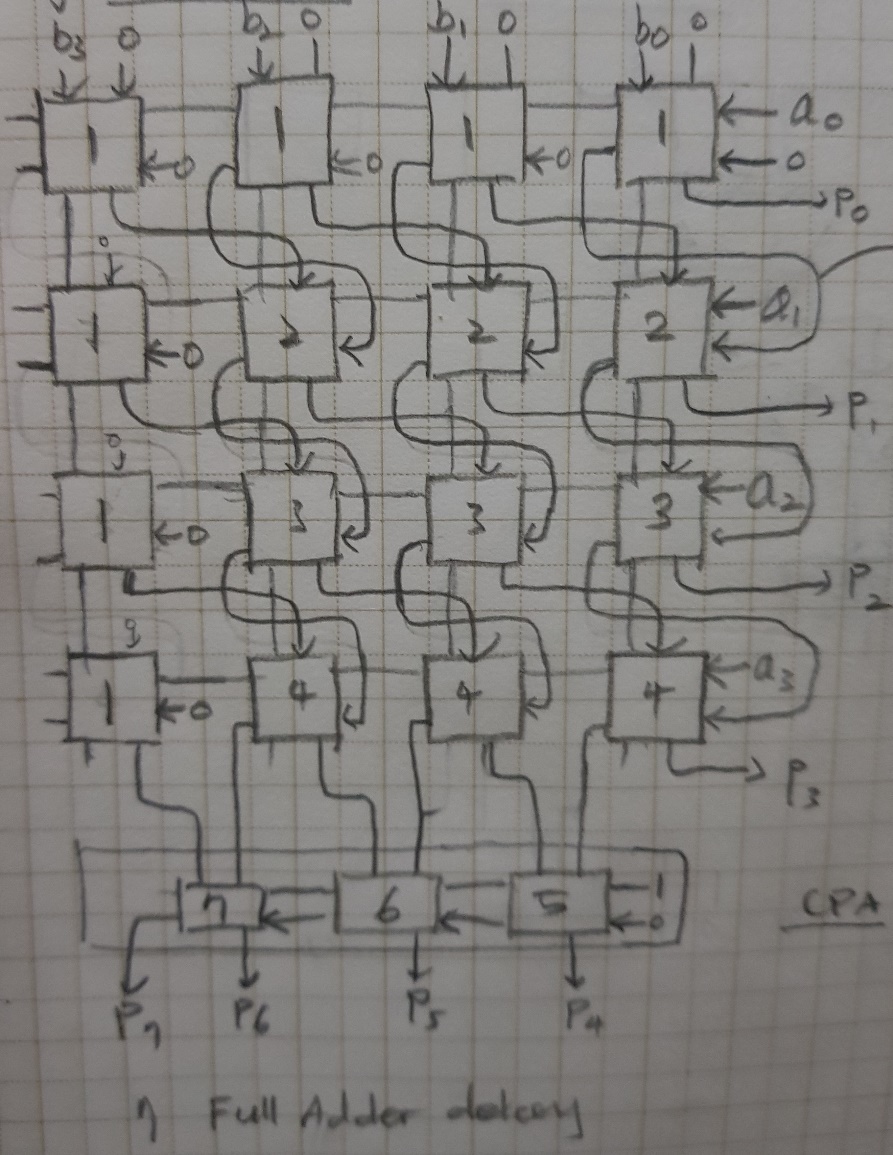
****

**ALU**

**Arithmetic Logic Unit의 약자로 산술논리 연산장치라고 한다. 중앙처리장치의 일부로서 산술연산과 논리연산을 담당하는 회로 집합이다. ALU는 크게 산술 연산 장치와 논리 연산 장치 그리고 shift register, complementer, status register로 나뉜다. 뒤에 3가지는 순서대로 쉬프트 연산용 레지스터, 2의 보수 취해 주는 부분(음수화), 연산 결과에 대한 상태를 저장하는 레지스터다.**

**ALU는 가산기, 감산기, AND, OR, Inverter, 제어신호를 위한 회로들이 포함 되어 있다.**

**추가적으로 곱셈회로는 전가산기를 2차원 배열형태로 정렬시켜서 구현할 수 있다.**

****

**결과 : P7 P6 P5 P4 P3 P2 P1 P0 7개의 전가산기를 거치는 만큼의 시간이 걸린다.  
작은 사각형은 전가산기를 뜻하며 사각형 안의 숫자는 1 전가산기 delay를 의미한다.**