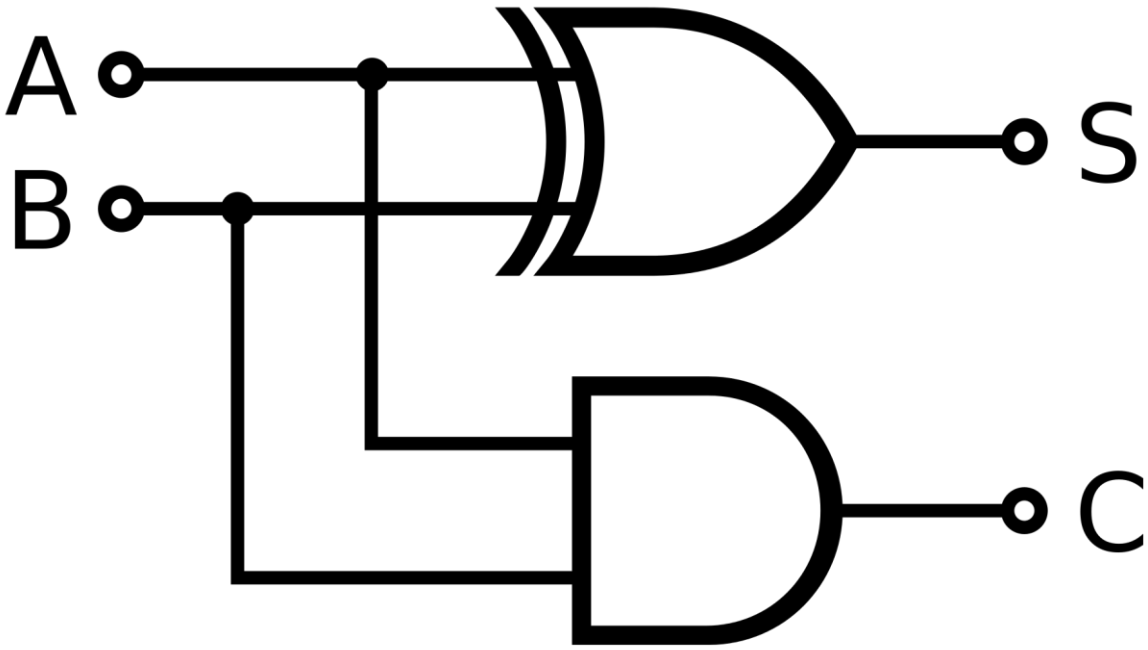
6주차 예비보고서

전공: 컴퓨터공학 학년: 2학년 학번: 20191629 이름: 이주헌

1. 전가산기 및 반가산기에 대해 조사하시오.

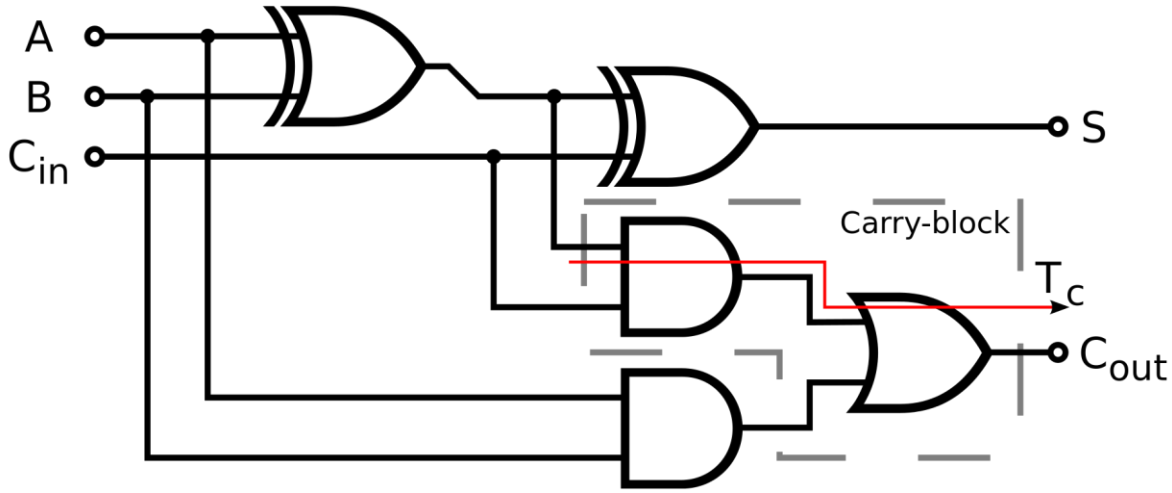
가산기(Adder)는 두 수를 더해주는 기기를 말한다. 보통 가산기 회로는 CPU의 산술 논리 장치(Arithmetic Logic Unit)에서 찾을 수 있지만, CPU의 제어부에서 주소 값을 계산하는 등의 경우에도 사용된다.

반가산기(Half-adder)는 2개의 입력을 받아 2개의 출력을 내는 가산 회로로, A와 B, 각 1비트의 입력을 받아 두 값을 더한 결과를 S 출력으로 보내고, 만약 1비트 범위를 벗어나는 수가 만들어진다면 C 출력을 논리적 참으로 하는 회로이다. 다음은 XOR 게이트와 AND 게이트를 이용하는 반가산기의 회로와 진리표이다.



A	B	C	S
T	T	T	F
T	F	F	T
F	T	F	T
F	F	F	F

2개의 입력을 받아 2개의 출력을 내는 반가산기와 달리, 전가산기(Full-adder)는 3개의 입력을 받아 2개의 출력을 내는 가산 회로이다. 전가산기는 반가산기의 모든 입출력을 가지고 있지만, 다른 가산 회로의 오버플로우 “받아올림” 비트를 받을 수 있는 입력 하나를 더 가지고 있다. 다음은 전가산기의 회로도와 진리표이다.

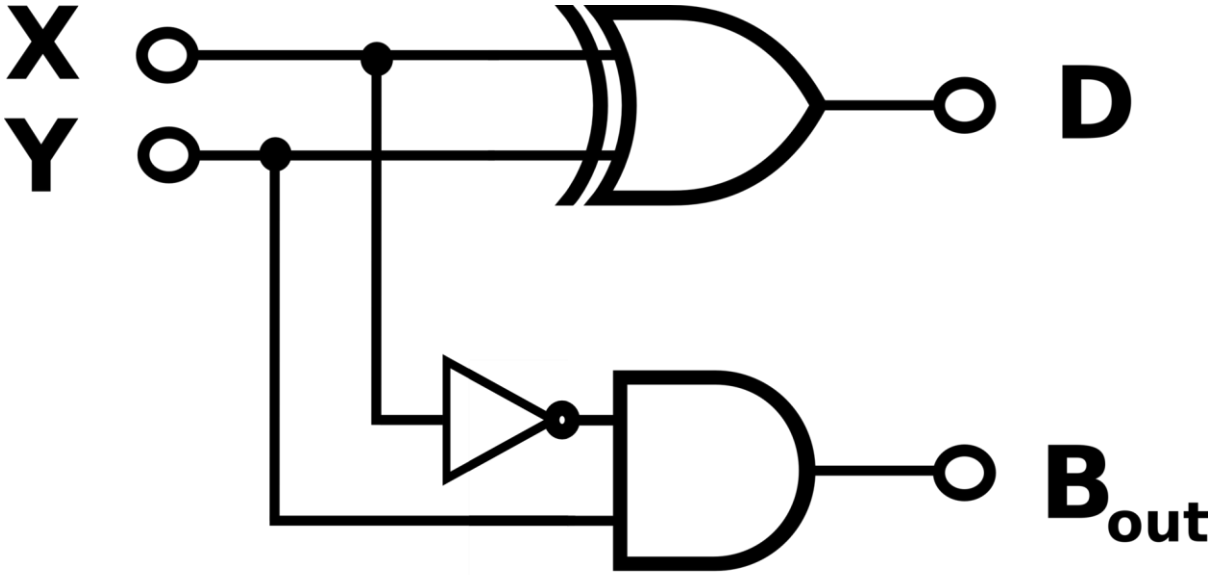


A	B	C_in	C_out	S
T	T	T	T	T
T	T	F	T	F
T	F	T	T	F
T	F	F	F	T
F	T	T	T	F
F	T	F	F	T
F	F	T	F	T
F	F	F	F	F

2. 전감산기 및 반감산기에 대해 조사하시오.

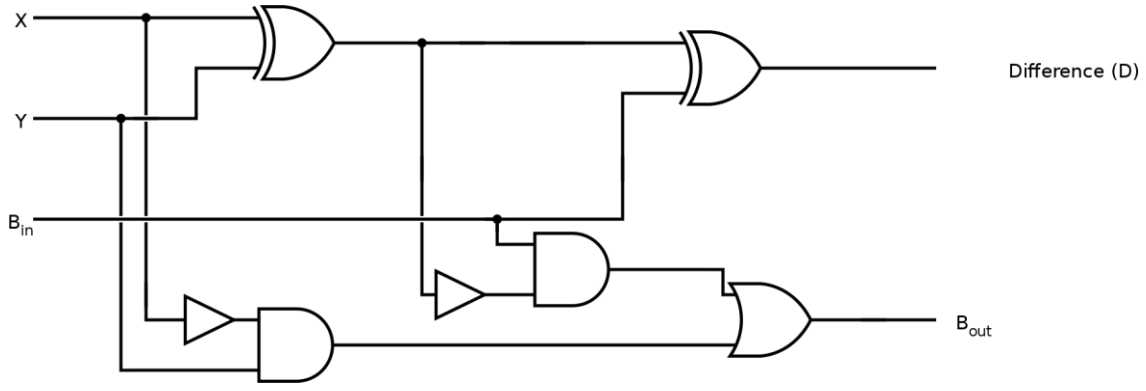
감산기(subtractor)는 가산기와 반대로 두 수를 빼는 기기를 말한다. 감산기 역시 CPU의 산술 논리 장치에서 찾을 수 있으며, 가산기와 마찬가지로 전감산기와 반감산기 두 종류가 존재한다.

먼저 반감산기(Half-subtractor)는 반가산기와 마찬가지로 두 개의 입력을 받아 두 개의 출력을 내는 장치이다. X와 Y는 입력이고, X - Y를 계산한 결과를 D로 출력하고, 만약 연산하는 데 받아내림이 필요하다면 B 출력을 논리적 참으로 설정하는 회로이다. 다음은 XOR, AND 게이트와 NOT 인버터를 사용하는 반감산기의 회로도와 진리표이다.



X	Y	B	D
T	T	F	F
T	F	F	T
F	T	T	T
F	F	F	F

전감산기(Full-subtractor) 역시 전가산기와 마찬가지로 받아내림 반가산기의 모든 입력값에 추가적으로 받아내림 연산의 여부를 입력으로 받는 논리 회로이다. 이 회로는 받아내림의 여부까지 고려하여 뺄셈을 할 수 있다. 다음은 전감산기의 회로도와 진리표이다.

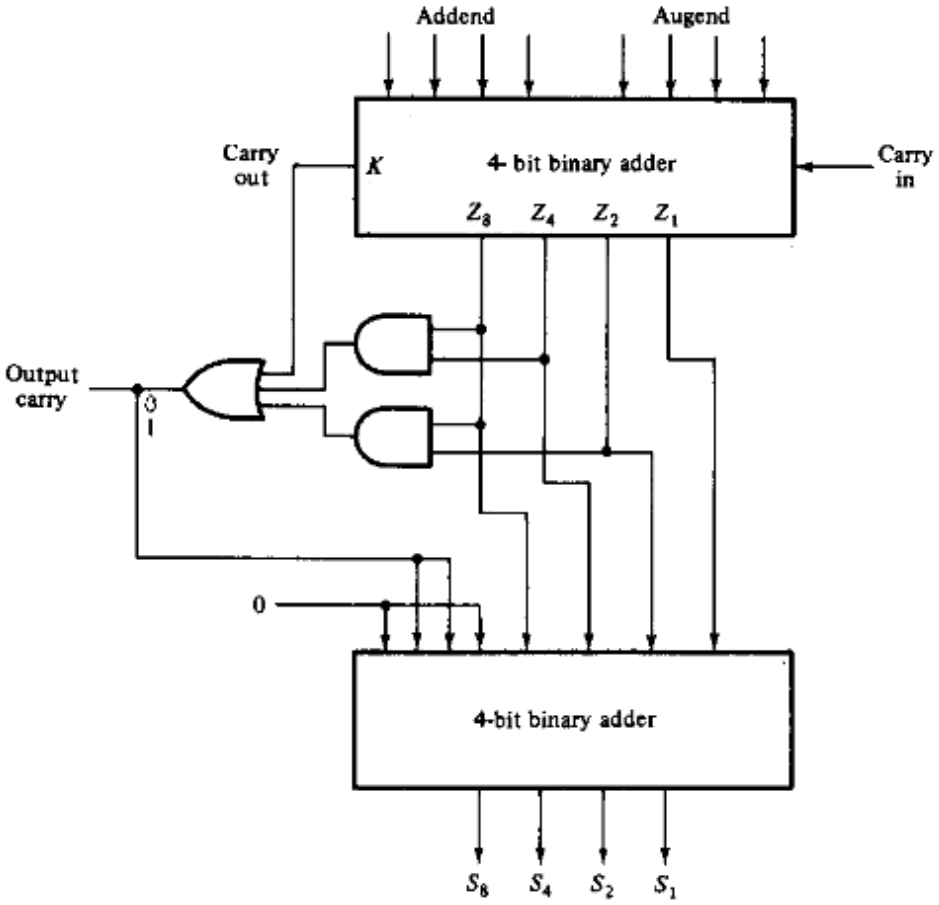


X	Y	B_in	B_out	D
T	T	T	T	T
T	T	F	F	F
T	F	T	F	F
T	F	F	F	T
F	T	T	T	F
F	T	F	T	T
F	F	T	T	T
F	F	F	F	F

3. BCD 가산기에 대해 조사하시오.

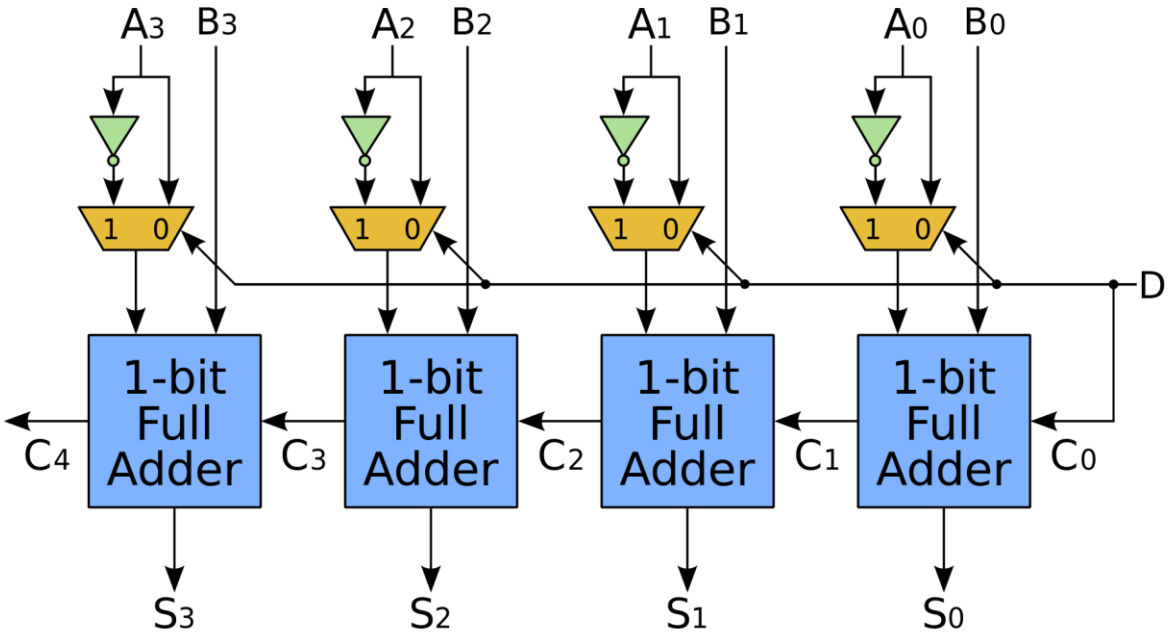
BCD는 binary-coded decimal의 약자로, 십진수 숫자를 이진수로 나타낸 것을 말한다. 즉, 0부터 9까지의 수가 위치할 수 있는 십진수 자릿수 하나를 1니블(4비트) 이진수로 나타낸 것이다. 따라서 각 자릿수는 0000부터 1001 사이의 숫자만 가질 수 있다. 이러한 숫자 표현 방식은 십진수 기반 시스템을 설계하는 데 자주 사용되었다.

두 BCD 숫자(1니블 크기)를 더하기 위해서는 먼저 각 니블을 일반적인 이진수 가산기로 더한다. 만약 두 수를 더한 결과가 1니블보다 크거나 하나의 니블이 10 이상의 수를 나타내어 십진수 받아올림이 필요한 경우, 결과값에 0110을 더하여 2니블으로 나타낸다. 다음은 해당 알고리즘을 논리회로로 구현한 결과이다.



4. 병렬 가감산기에 대해 조사하시오.

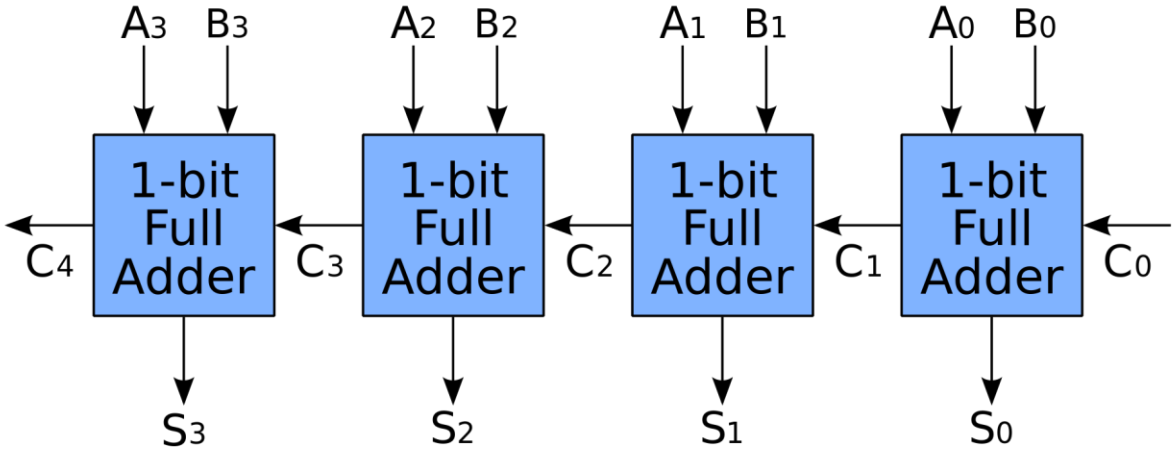
병렬 가감산기(Parallel adder-subtractor)는 기본적으로 덧셈과 뺄셈을 모두 할 수 있는 논리 회로를 말한다. 가산기와 감산기의 회로를 잘 비교해 보면, 가산기의 S 신호와 감산기의 D 신호의 출력 결과는 서로 같고, 받아올림과 받아내림 연산에만 인버터의 유무라는 차이점이 있다. 이로부터 다음과 같은 병렬 가감산기의 회로를 만들 수 있다.



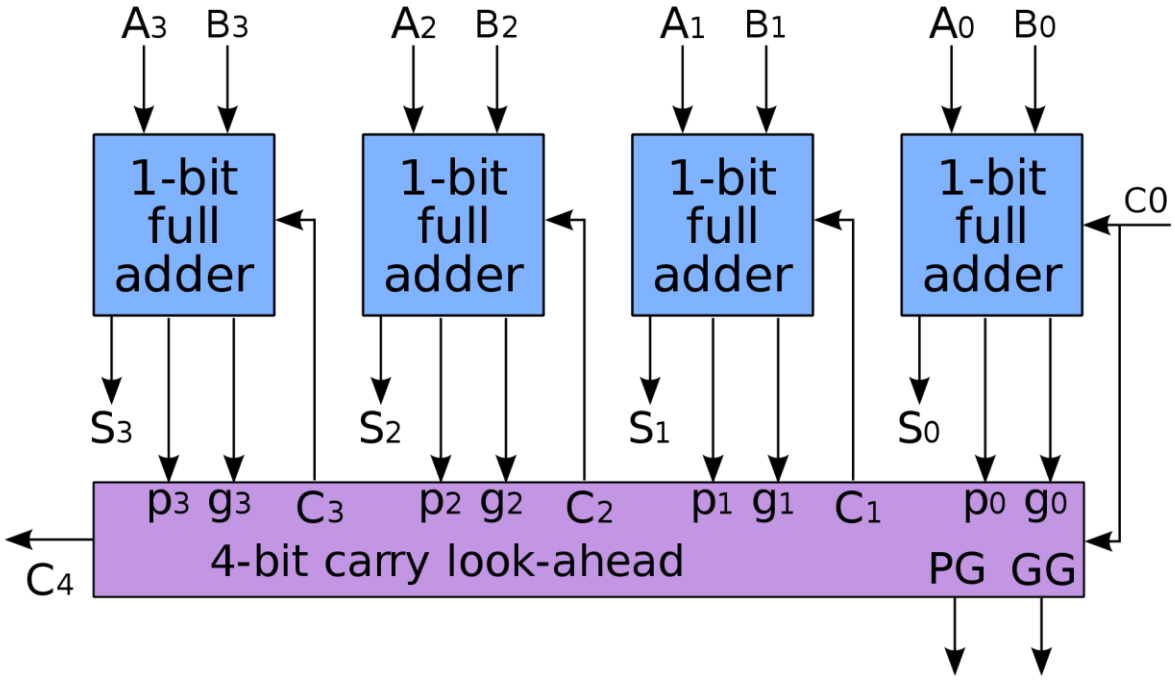
위 회로를 이용하여 덧셈을 하려면 D 입력을 0으로 하고, 뺄셈을 하려면 D 입력을 1으로 하면 된다. 만약 D 입력이 1이라면 입력 A의 1의 보수를 취한 다음 1을 더하여 2의 보수로 만든 뒤, B와 더하는 과정이 된다. 이는 A의 부호를 반전한 다음 B와 더하는 것과 같으므로, B - A를 구현한다고 할 수 있다.

5. Carry Look-ahead Adder를 Ripple Carry Adder와 비교하여 설명하시오.

n비트 가산기를 만들기 위해서는 여러 개의 1비트 전가산기를 이용하여 구성하게 된다. 그러기 위해서는 각 전가산기의 받아올림 비트를 묶어야 할 필요가 생기게 되는데, 이를 해결하는 방법 중 가장 간단한 방법은 이전 가산기의 받아올림 출력을 다음 가산기의 받아올림 입력에 연결하는 것이다.



이렇게 각 가산기의 받아올림 입출력을 차례로 연결하여 구성한 가산기를 ripple carry adder라고 한다. 이 방법은 제일 구현이 간단하지만, 하나의 논리 소자의 출력과 다른 논리 소자의 입력을 연결하는 방식이므로 필연적으로 전파 지연이 커지게 된다. 이러한 단점을 보완한 것이 carry-lookahead adder이다.



Carry-lookahead adder는 전가산기를 서로 연결하는 것이 아니라, 어떠한 받아올림 계산 버스에 연결하여 동시에 계산값을 도출할 수 있도록 만든 회로이다. Ripple carry adder와 달리, 받아올림 예측 버스가 미리 받아올림 값을 예측하여 각 가산기에 입력하기 때문에 전파 지연을 크게 줄일 수 있다. 따라서, Carry-lookahead 가산기가 보통 ripple carry 가산기보다 빠르다.

6. 기타이론

Carry-lookahead 버스는 전파(propagate), 생성(generate), 받아올림(carry)의 세 단계로 구성된다. 전파 단계에서는 이 덧셈이 이전에 받은 받아올림을 **전파**하는지 여부를 검사하며, 생성 단계에서는 해당 덧셈이 새로 받아올림을 **생성**하는지 여부를 검사하고, 마지막으로 받아올림 단계에서는 전파된 받아올림과 생성된 받아올림을 합하여 다음 받아올림을 생성한다.

어떠한 비트 덧셈이 받아올림을 전파하려면 A와 B, 두 입력의 값이 달라야 한다는 조건이 있다. 즉, 덧셈 결과는 1비트 범위 안이지만, 받아올림에 의해 받아올림이 생긴다는 뜻이다. 반대로 어떤 비트 덧셈이 받아올림을 생성하려면 A와 B, 두 입력이 모두 논리적 참이어야 한다는 조건이 있다. 덧셈 결과 자체만으로 받아올림이 생성된다는 것이다. 따라서 다음과 같이 식을 만들 수 있다.

$$P = A \oplus B$$
$$G = A \cdot B$$
$$C_{n+1} = G + (P \cdot C_n)$$

n번째 자릿수의 받아올림 C_n을 구하려면 세 번째 식을 어떤 자연수 n에 대하여 재귀적으로 전개하면 되므로 모든 자릿수에 대하여 손쉽게 받아올림을 예측할 수 있다는 것을 알 수 있다.