20210815 정재현

bitNor(x,y)

NOR 자체가 OR게이트에 NOT을 붙인 것인데, ~(X|Y)를 전개해보면 ~X&~Y가 나오기 때문에 전개만으로도 Nor함수를 &와 ~로 나타낼 수 있었고, 해당 연산을 리턴하도록 함수를 만들었다.

isZero (x)

!연산자는 0일 경우 0을 출력하고, 나머지의 경우에는 1(참)을 출력하는 성질이 있다.

addOk(x,y)

x+y를 하였을 때 overflow가 일어나는지는 나타내준다. 먼저 overflow에 대해 짚고 넘어가보면 우리가 연산을 할 때 상식적으로 일어나는 것의 외로 양수와 양수를 더했는데 음수가 나오거나, 음수와 음수를 더했는데 양수가 나오는 등의 결과를 말한다. 함수에서는 x의 sign을 나타내는 MSB와 y의 sign을 나타내는 MSB를 변수로 만들고 추가적으로 이 둘의 합을 sign을 나타내도록 만들었는데 이때 >>31을 이용하여 MSB자리만 나오도록 만들었다. 먼저 x의 msb와 y의 msb를 xor 연산을 하게 되면 둘의 부호가 다를 경우 1을 나타내주기 때문에 overflow가 일어나지 않는다. 반대로 부호가 동일하다면 overflow가 일어날 가능성이 있다. 그래서 추가적으로 뒷부분을 구성했는데, x+y의 MSB와 x의 MSB 그리고 y의 MSB에 대해 xor 연산을 해주어 부호가 같은지 판단한다. 이때, 부호가 다르게 되면 결과가 overflow가 일어나기 때문에 iszero의 값이 0이 되어야 하므로 이 부분에 !을 해주어 overflow가 일어나면(부호가 다르면) 0이 되도록 조정을 해주고, 결과를 리턴하도록 함수를 만들었다.

absVal(x)

x값이 입력되었을 때 음수이면 절대값을 취하여 양수로 출력 해줘야한다. 기본적으로 음수를 양수로 바꾸기 위해서는 2의 보수를 사용 해야하는데 (1의 보수 ~와 +1) 이부분을 구성하기 위해서 xor에 0과 x가 들어갈 때 x가 나오고, xor에 1과 x가 들어가면 ~x가 나오는 성질을 이용하였다. 먼저 x>>31을 이용하여 부호를 알아내고, x+sign에 sign과 xor 연산을 하도록 하였다. X가 양수일 경우 sign=0 이므로 xor(0,x)이므로 x가 그대로 나오고, sign=1(음수)일 경우, x+1에 xor 1 연산을 하도록 하였는데 결국 ~(X+1)과 같은데 전개하면 ~x-1이므로 2의 보수로 나타내어 음수가 양수로 표현되도록 리턴하는 함수를 만들었다.

logicalShift(x, n)

logical shift는 먼저 arithmetic shift가 기본이 되어 이를 구현하기 위해서 MSB에 관계없이 bit를 shift하면 빈자리는 0이 되도록 구성을 해야한다. 먼저 x에 대해 n만큼 >> shift를 해주는데, 이때 MSB 값에 의해 이동하여 빈 bit자리에는 MSB 값이 채워지게 된다. 이를 해결하기 위해서 &로 쉬프트 한 값이면서 빈자리에 0이 들어가 &연산에 의해 0으로 바꿔지도록 만들었다. 해당하는 수를 만들 때에도 arithmetic shift가 기본으로 사용되기 때문에 이를 이용하여 오히려 1이라는 숫자를 만들고 shift를 하여 msb로 오게한다. 그 뒤에 n비트 이동하여 n-1 비트만큼 빈 자리였으므로 1로 덮어주기 위해 >> shift로 n-1을 하려고 했으나 이때 n이 0이 될 수 있기 때문에 n을 이동시키고 나서 다시 << shift를 사용하여 msb를 포함하여 n비트가 1이 되고 나머지는 0이 된다. 이때 ~를 사용하여 반전을 시켜주고 &연산은 기본적으로 AND(1,x)=x, AND(0,x)=0이므로 logical shift를 위해 이동시켜서 msb가 들어간 비트 부분은 and(0,x)로 0이 되도록 하고 나머지는 AND(1,x)연산을 하여 원래 x가 들어가도록 하여 logical shift의 결과를 리턴하도록 구현하였다.