10주차 결과보고서

전공: 심리학과 학년: 3학년 학번: 20190345 이름: 김동현

**1.**

4bit binary parallel adder는 4비트로 이루어진 2진수 2개를 더하는 회로이다. 4비트 2진수 2개를 각각 a(a[3]a[2]a[1]a[0]), b(b[3]b[2]b[1]b[0])라고 할 때, 두 수의 합은 s(s[3]s[2]s[1]s[0]) 으로 표현할 수 있다. 이때 입력값에 carry in값이 출력값에 carry out값이 추가된다. 아래 표는 i번째 비트의 덧셈에 관한 진리표이다. A[i]와 b[i]는 각각 2진수 a,b의 i번째 비트 값이다. c[i]는 이전 연산에서 계산된 carry값이다. I가 0일때에는 cin값이 carry값으로 주어진다. s[i+1]은 두 이진수의 합을 저장한다. c[i+1]는 a[i],b[i]의 합의 carry값으로 다음 비트의 carry in값으로 주어진다.

테이블이(가) 표시된 사진

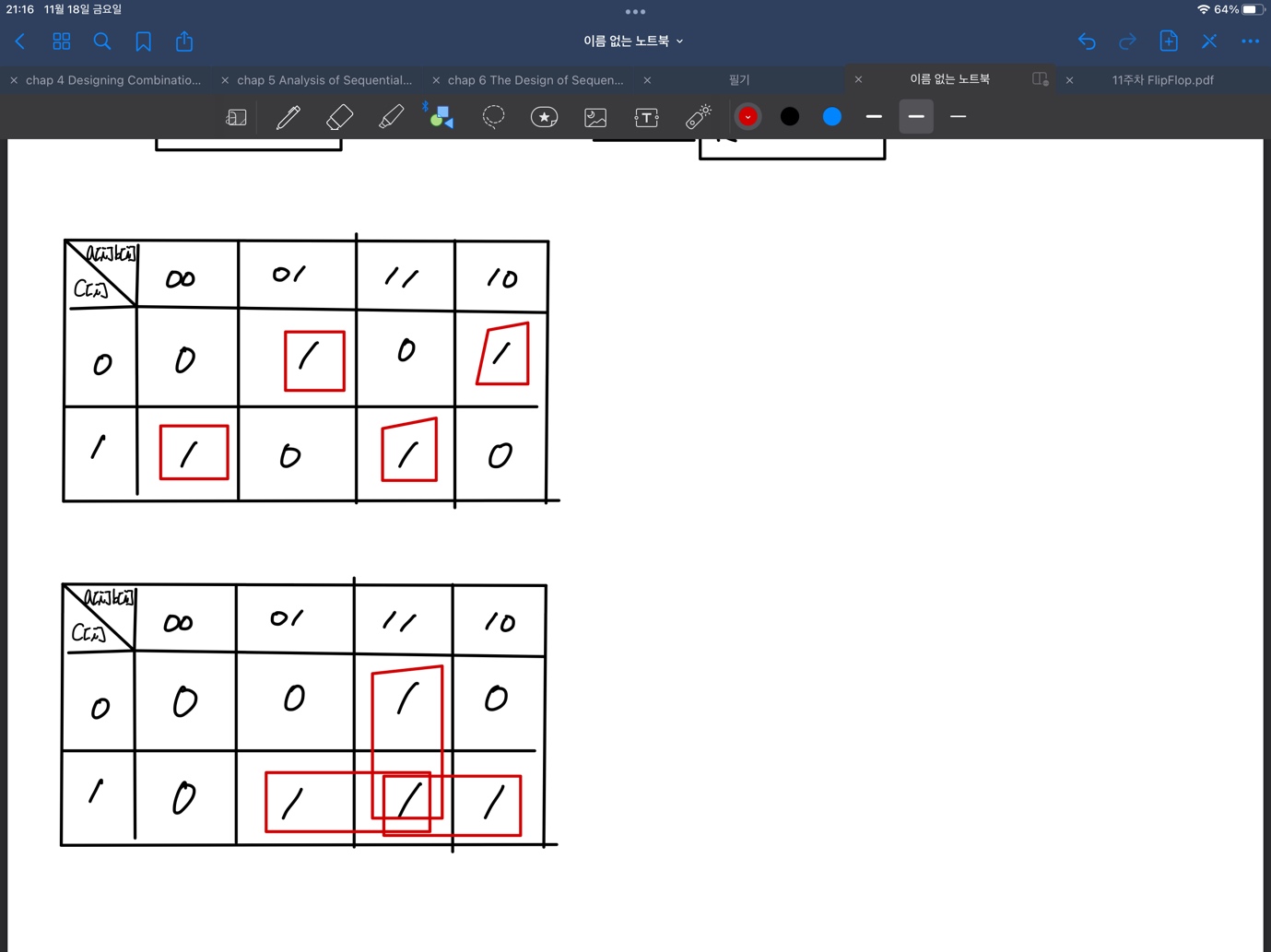
자동 생성된 설명

위의 표에서 s[i+1]과 c[i+1]의 값을 카르노맵으로 표현하면 다음과 같다.

텍스트이(가) 표시된 사진

자동 생성된 설명

S[i+1]=a[i]^b[i]^c[i]



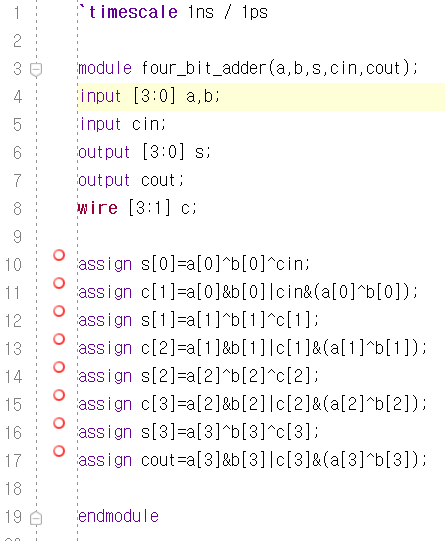
C[i+1]=c[i]&b[i]+c[i]&a[i]+a[i]&b[i]

=a[i]’&b[i]’&c[i]+a[i]’&b[i]&c[i]’+a[i]&b[i]’&c[i]’+a[i]&b[i]&c[i]+a[i]&b[i]

=c[i]&(a[i]’&b[i]+a[i]&b[i])+c[i]’&(a[i]’&b[i]+a[i]&b[i]’)+a[i]&b[i]

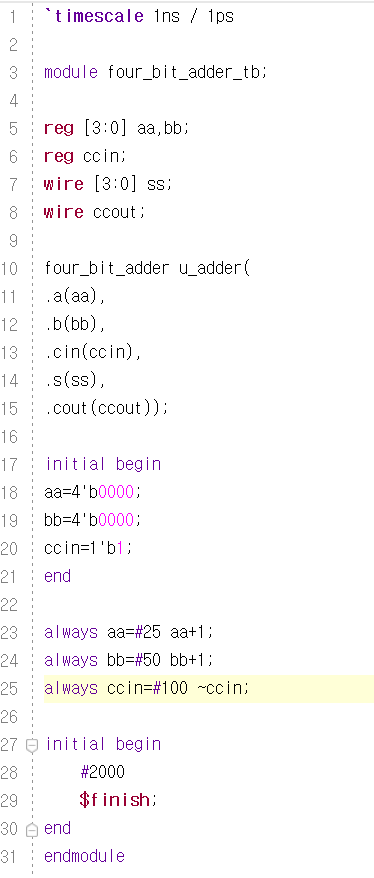
=a[i]&b[i]+c[i]&(a[i]^b[i])

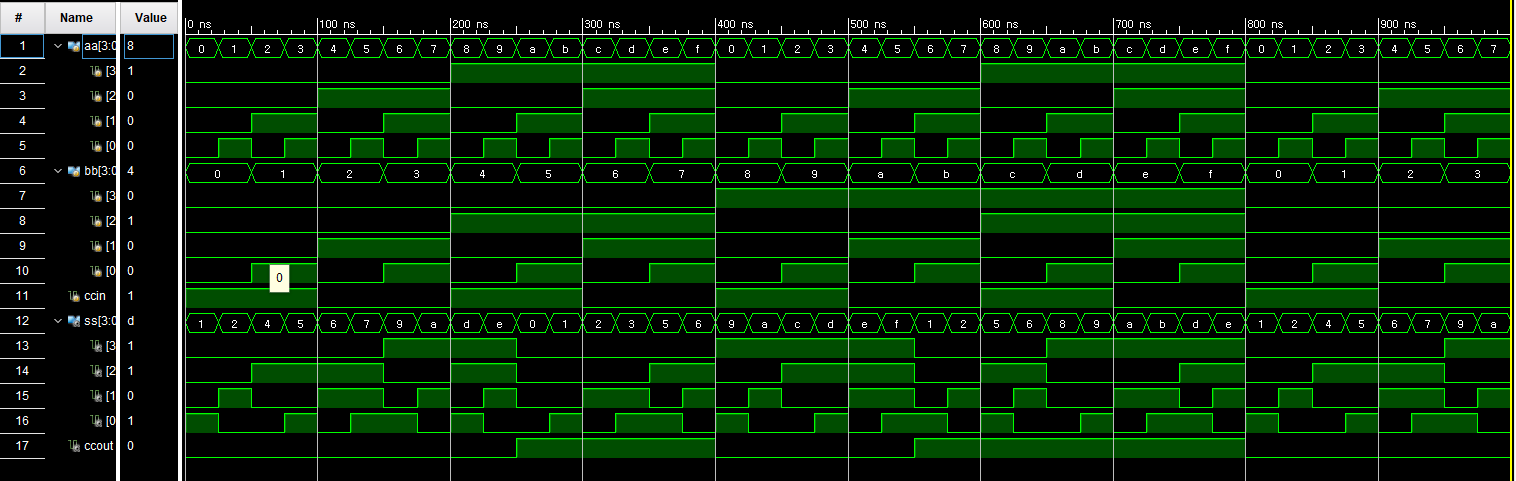
카르노맵으로 구한 논리식을 통해 다음과 같은 베릴로그 디자인 코드를 구현하였다.



4비트 2진수 a,b는 길이가 4인 배열로 구현하였으며, 1비트의 값이 배열의 각 원소에 저장된다. Wire[3:1] c는 연산과정에서 발생하는 carry값을 임시로 저장하기 위해 선언하였다. 낮은 자리수부터 덧셈을 수행하여 s[i+1], c[i+1]값을 구한다. 이때 c[i+1]의 값은 다음 연산의 carry값으로 사용되는 것은 해당 회로가 병렬 덧셈 연산을 하고 있음을 보여준다.

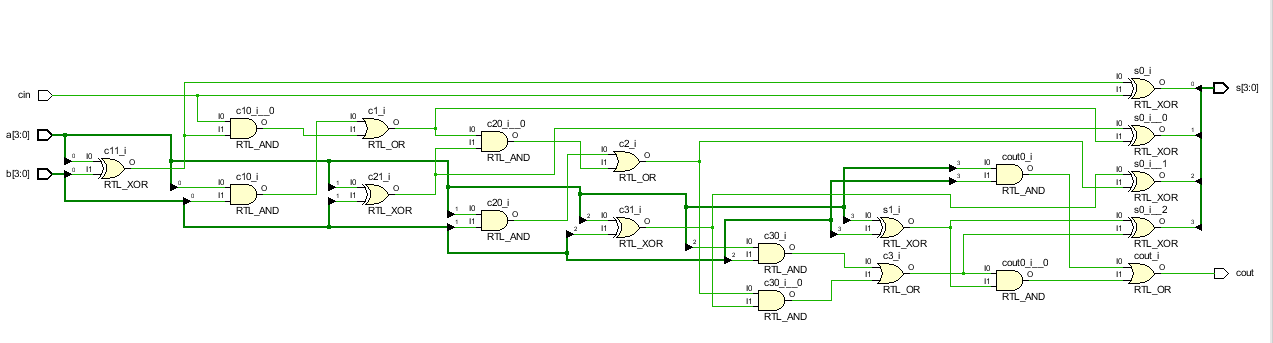
시뮬레이션 코드이다. 4비트를 저장한 배열 aa,bb에 0000을 할당하였으며, 클럭마다 1씩 증가하도록 하였다.





시뮬레이션 결과는 위 그림과 같다. 두 4비트 이진수 aa,bb를 더한 값이 ss에 저장되어 출력된다.

길이가 4인 배열에 0000부터 1111까지 저장될 수 있으며, 0+0000+0000부터 1+1111+1111까지 연산이 가능하다. 각각의 합의 0 + 0000, 1 + 1111으로 모든 덧셈의 경우의 수를 표현할 수 있다. 이는 두 이진수의 합이 ss의 표현범위를 넘어갈 경우 carry값인 cout값이 1로 출력되기 때문에 가능함을 알 수 있다. 예를 들어 1111(a) + 0111(b) + 0(cin) = 1(cout)0110(ss)으로 표현된다.



4bit binary parallel adder의 schematic이다.

2.

4bit binary parallel subtractor는 4비트로 이루어진 2진수 2개를 빼는 회로이다. 4비트 2진수 2개를 각각 a(a[3]a[2]a[1]a[0]), b(b[3]b[2]b[1]b[0])라고 할 때, a-b인 두 수의 차는 d(s[3]s[2]s[1]s[0]) 으로 표현할 수 있다. 이때 입력값에 borrow in값이 출력값에 borrow out값이 추가된다. 이는 각 비트의 뺄셈 결과 음수값이 나오므로 앞선 비트에서 값을 빌려왔음을 의미하는 변수이다. 아래 표는 i번째 비트의 뺄셈에 관한 진리표이다. A[i]와 b[i]는 각각 2진수 a,b의 i번째 비트 값이다. bo[i]는 이전 연산에서 빌려온 borrow값이다. i가 0일때에는 bin값이 borrow값으로 주어진다. d[i+1]은 두 이진수의 차를 저장한다. bo[i+1]는 a[i],b[i]의 차의 borrow값으로 다음 비트의 bin값으로 주어진다.

테이블이(가) 표시된 사진

자동 생성된 설명

위의 표에서 d[i+1]과 bo[i+1]의 값을 카르노맵으로 표현하면 다음과 같다.

텍스트이(가) 표시된 사진

자동 생성된 설명

D[i+1]=a[i]^b[i]^bo[i]

텍스트이(가) 표시된 사진

자동 생성된 설명

Bo[i+1]=(a[i]’&b[i])+(bo[i]&(a[i]^b[i])’)

카르노맵으로 구한 논리식을 통해 다음과 같은 베릴로그 디자인 코드를 구현하였다.

텍스트이(가) 표시된 사진

자동 생성된 설명

비트 2진수 a,b는 길이가 4인 배열로 구현하였으며, 1비트의 값이 배열의 각 원소에 저장된다. Wire[3:1] bo는 연산과정에서 발생하는 borrow값을 임시로 저장하기 위해 선언하였다. 낮은 자리수부터 뺄셈을 수행하여 d[i+1], bo[i+1]값을 구한다. 이때 bo[i+1]의 값은 다음 연산의 borrow값으로 사용되는 것은 해당 회로가 병렬 뺄셈 연산을 하고 있음을 보여준다.

시뮬레이션 코드이다. 4비트를 저장한 배열 aa,bb에 0000을 할당하였으며, 클럭마다 1씩 증가하도록 하였다.

테이블이(가) 표시된 사진

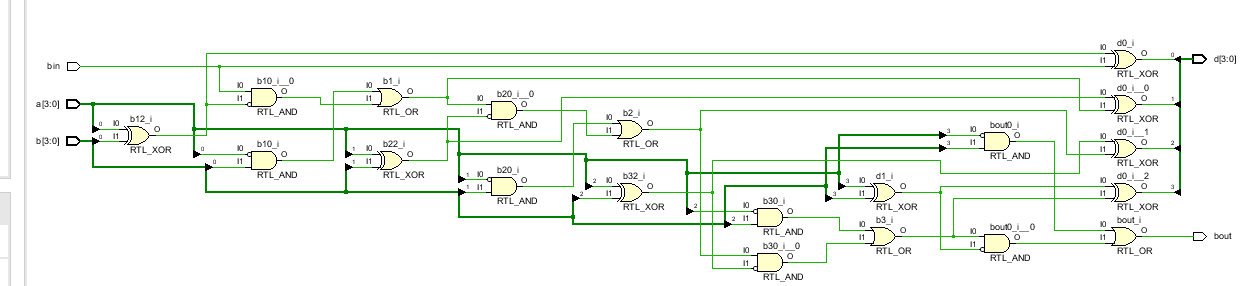
자동 생성된 설명

텍스트, 게임이(가) 표시된 사진

자동 생성된 설명

시뮬레이션 결과는 위 그림과 같다. 두 4비트 이진수 aa,bb를 뺀 값이 dd에 저장되어 출력된다.

길이가 4인 배열에 0000부터 1111까지 저장될 수 있으며, 0000 – (1111 + 1) 부터 1111 – (0000 +0)까지 연산이 가능하다. 각각의 차는 1 + 0000, 0 + 1111으로 모든 뺄셈의 경우의 수를 표현할 수 있다. 이는 두 이진수의 차가 dd의 표현범위를 넘어갈 경우 borrow값인 bbout값이 1로 출력되기 때문에 가능함을 알 수 있다. 예를 들어 0100(aa) – 1010(bb) -0(bbin) =1010(dd)+1(bbout)으로 표현된다.



4bit binary parallel subtractor의 schematic이다.

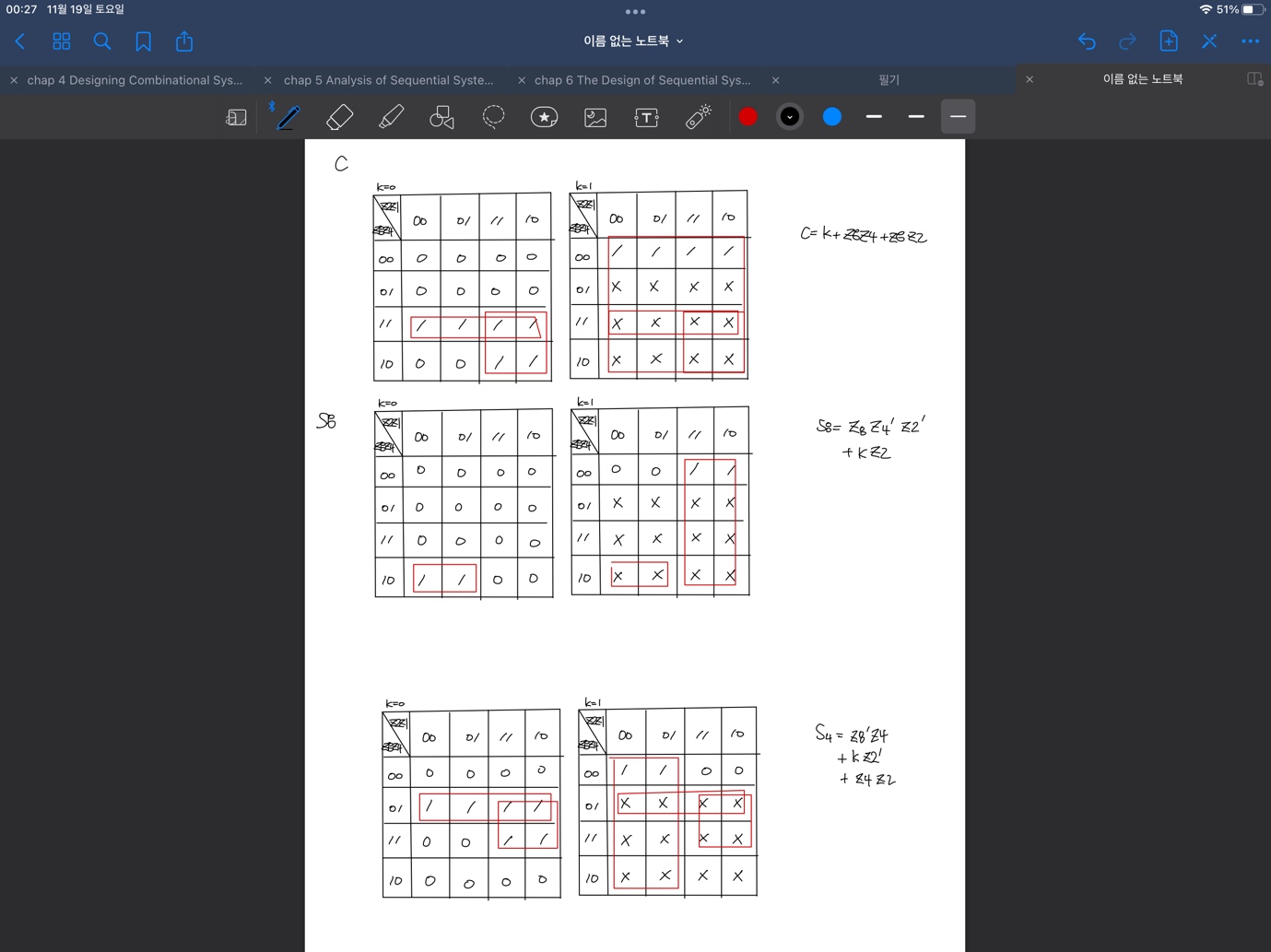
3.

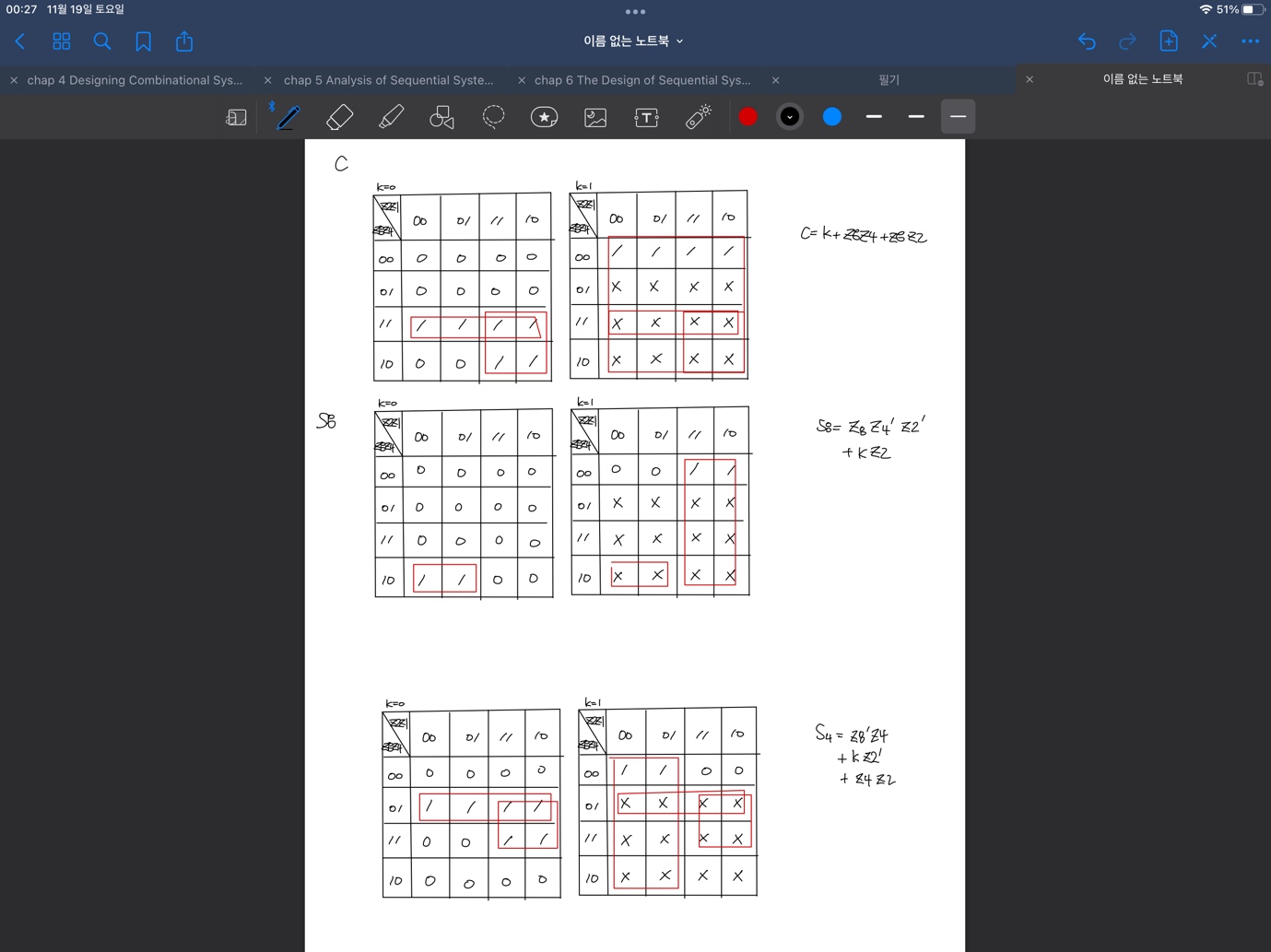
두 4비트 이진수의 합을 더하여 bcd 코드의 형태로 나타내는 회로이다. Bcd 코드는 한자리수 만을 나타낼 수 있기에 두 이진수의 합을 통해 2자리로 표현할지, 2자리로 표현한다면 어떻게 표현해야할지 결정해야 한다. 아래는 2진수 합에 따른 bcd결과를 나타낸 진리표이다. 이때 두 이진수의 합은 0부터 19까지이며, 이에 따라 10의 자리수를 나타내는 방법은 1비트 만을 사용하여 0또는 1로 표현할 수 있다.

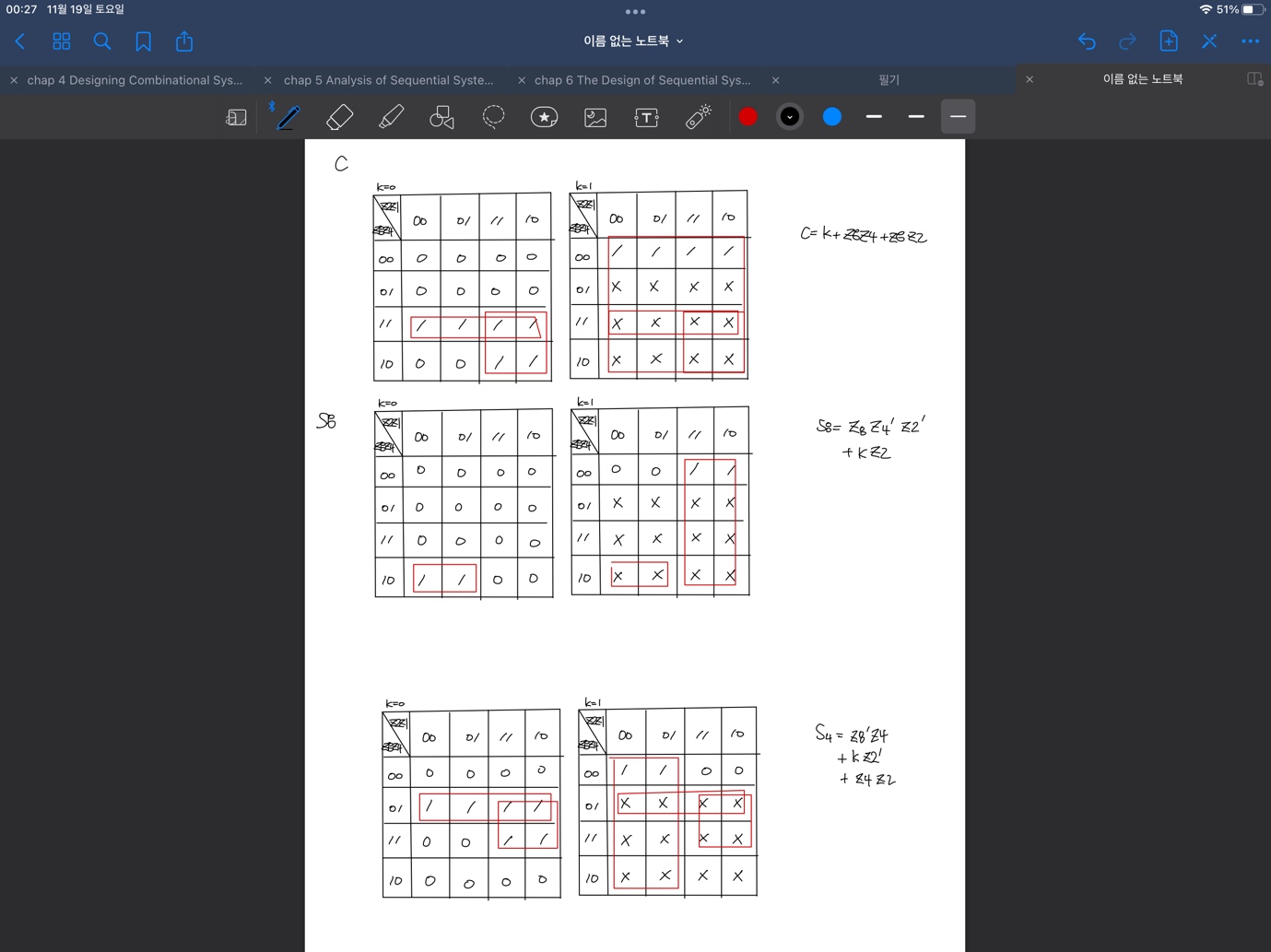
테이블이(가) 표시된 사진

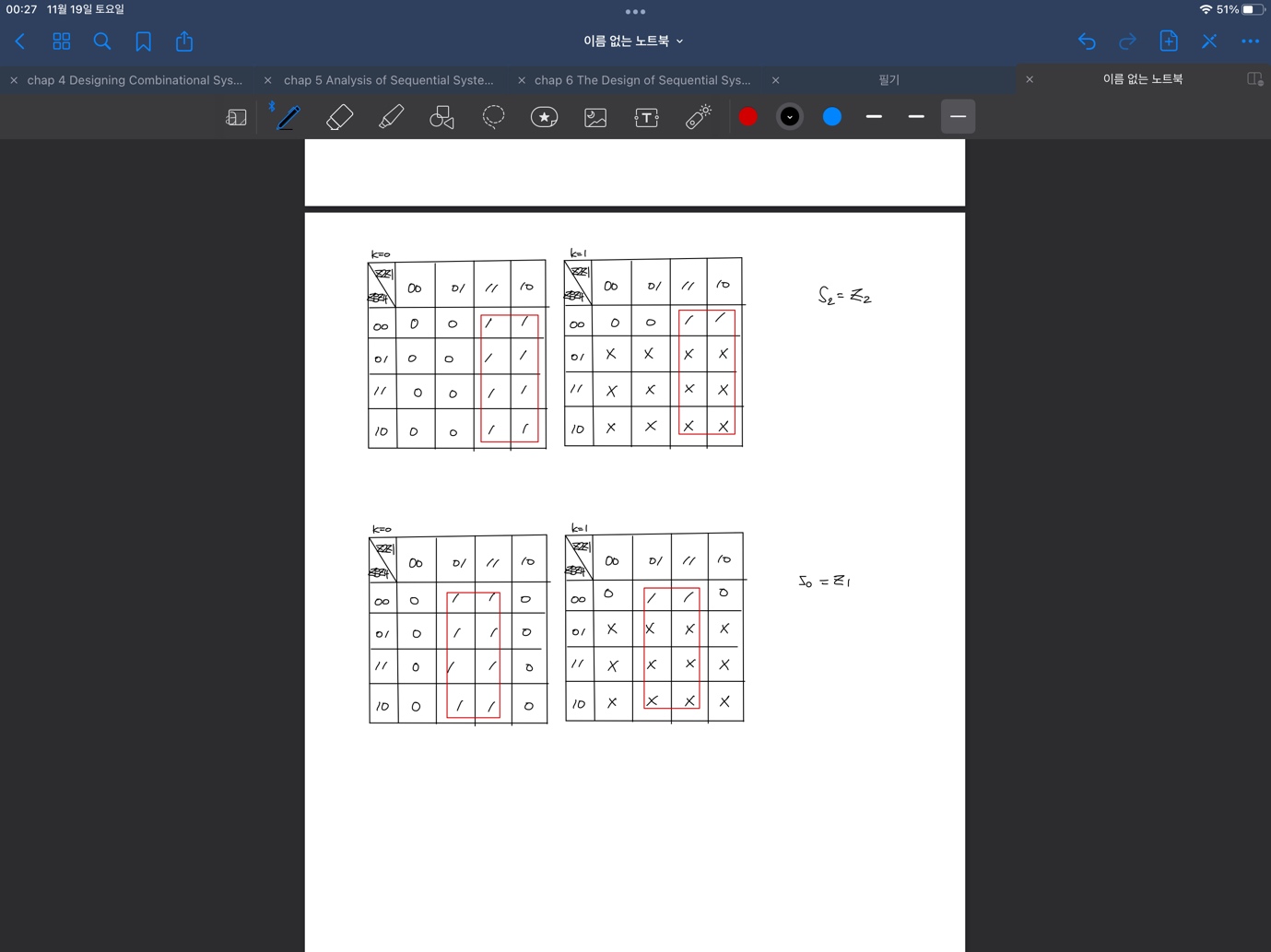
자동 생성된 설명

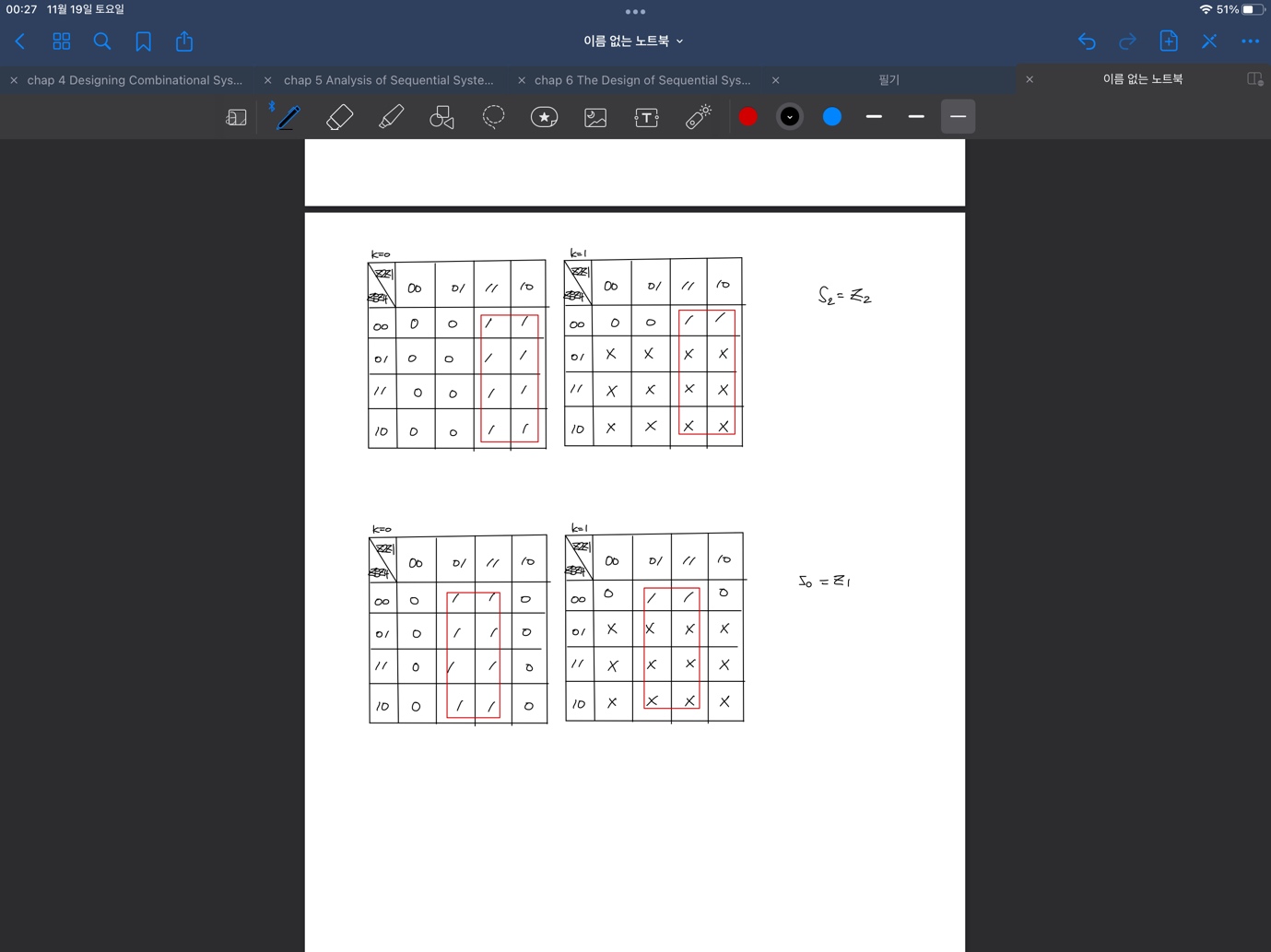
위의 표에서 Bcd의 값을 카르노맵으로 표현하면 다음과 같다.











카르노맵으로 구한 논리식을 통해 다음과 같은 베릴로그 디자인 코드를 구현하였다.

텍스트이(가) 표시된 사진

자동 생성된 설명

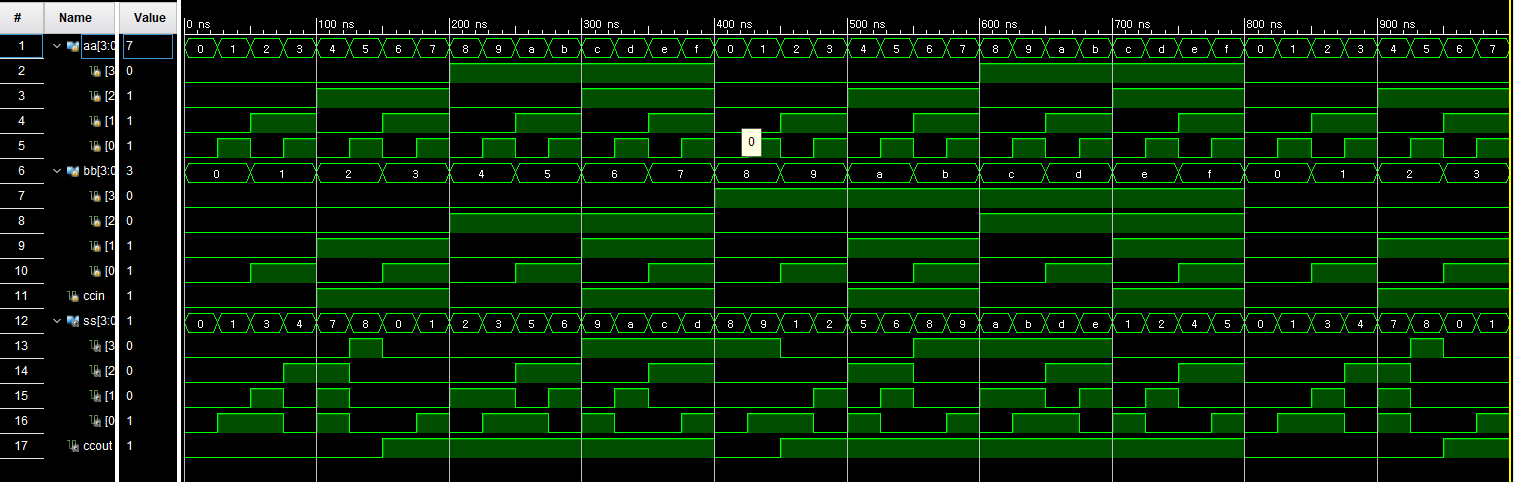


Bcd adder의 디자인 코드이다. 입력값으로 4비트 2진수 a,b를 배열의 형태로 선언하였으며, cin값도 입력값으로 선언하였다. 출력값은 1의 자리수를 4비트의 2진수로 나타낼 s배열과 10의 자리수의 존재를 나타낼 cout값을 선언하였다. Wire값은 bcd adder계산 과정 중 발생하는 수를 임시로 저장하기 위해 선언하였다. 우선 앞서 구현한 4bit binary parallel adder를 사용하여 입력으로 들어온 4비트 이진수 a와 b와 cin값을 더한다. 두 수의 덧셈값은 tmp 배열에 저장하며, 이때 carry 값은 k에 저장한다. 이후 bcd값으로 변환하기 위해 앞서 카르노맵으로 구한 논리식을 사용한다. C=k+Z8Z4+Z8Z2을 이용하여 cout=k|(tmp[3]&tmp[2])|(tmp[3]&tmp[1])의 값을 저장한다. 1의 자리수를 나타내는 4비트 이진수도 배열 s에 저장한다. 1의 자리수를 나타내는 값 역시 보정이 필요한데, 앞서 구한 cout값을 활용한다. 앞서 구한 cout값이 0이라면 두 이진수의 합이 1의 자리로 표현가능하므로, 그대로 출력한다. 그렇지 않고 cout값이 1이라면 두 이진수의 합이 1의 자리만으로 표현 불가능하다. 이때에는 tmp[2]와 tmp[3]값에 cout값을 더하여 연산한다.

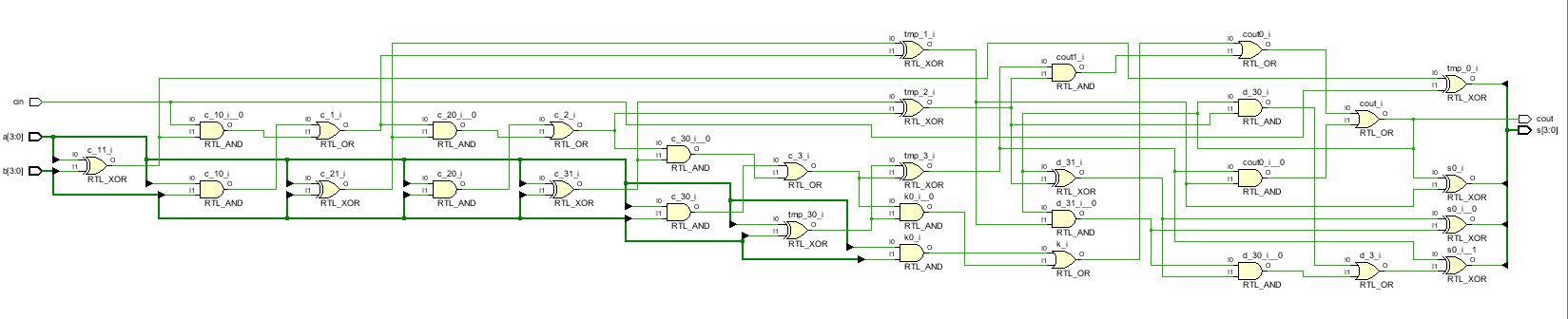
테이블이(가) 표시된 사진

자동 생성된 설명

시뮬레이션 코드이다. 4비트를 저장한 배열 aa,bb에 0000을 할당하였으며, 클럭마다 1씩 증가하도록 하였다.



Bcd adder의 시뮬레이션 결과이다. 2진수 aa,bb의 합을 bcd의 형태로 출력하였다. 이때 1의 자리수는 ss를 통해, 10의 자리수는 ccout을 통해 나타냈다. Ss가 a,b,c,d,e,f인 경우는 10 이상의 경우 이므로, ccout의 값을 1로 출력해 10의 자리수의 1을 나타내도록 했다. 또한 합이 10 이상인 경우 1의 자리는 한자리수(0000~1001)로만 표현하였다. 위 bcd 연산의 경우 합의 최대값이 19이므로 20이 넘는 값은 정상적으로 표현하지 못하고 있다. 이는 빨간색 X 부분으로 예외 처리하였다.



bcd adder의 schematic이다.

4.

4 bit parallel adder와 subtractor, bcd adder를 구현하였다. 배열로 들어온 입력에 대해 이전 실험에서 구현한 full adder와 full subtractor의 연산을 활용하여 4비트 연산을 병렬적으로 수행한다. 앞선 결과의 carry값과 borrow값이 다음 비트 연산에 영향을 주는 것을 확인할 수 있다. Bcd adder는 2개의 이진수 값을 더하여 bcd값으로 변환하는 연산을 수행한다. 이때 더한 값이 두자리로 표현될 때 두개의 코드로 변환하는 논리식이 필요했다.

5.

위 실험에서는 4bit adder와 4bit subtractor를 따로 구현하였다. 하지만 adder와 subtractor를 sign변수를 통해 하나의 코드로 구현 가능하다. 이는 2의 보수의 성질을 이용하며, 다음과 같이 디자인 코드를 구현할 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

B값에 sign값을 xor연산한다. 이로인해 b값은 덧셈일때 sign값이 0이므로, xor의 성질에 의해 b값이 그대로 입력된다. 반면 b값을 빼야할 경우 sign값은 1이고, 1의 보수가 입력된다. 이후 sign값 1이 cin값으로 더해져 2의 보수가 된다. 이를 시뮬레이션 코드로 구현하면 다음과 같다.

테이블이(가) 표시된 사진

자동 생성된 설명

시뮬레이션 결과는 아래 그림과 같다. Sign값이 0이면 덧셈이, sign값이 1이면 뺄셈이 이루어지는 것을 알 수 있다.

