실험 2. 불 대수식의 단순화

20210774 김주은

1. 개요

이론시간에 배운 불 대수식을 단순화하는 것에 대해 직접 실습을 통해 구현하고자 하는 것이 이 실험의 목표이다. 불 대수식을 단순화하기 전의 식을 직접 구현해보고, 단순화한 후의 식을 다시 구현해보면서 이들 간의 차이를 살펴보며 비교한다. 비교 과정을 통해 단순화의 효과를 알아본다.

이 실험에서는 단순화하기 전의 식을 표현하는 과정에서 sop form으로 그려보고, 이를 직접 단순화하는 과정에서는 k-map 알고리즘을 사용하여 단순화한다.

2. 이론적 배경

먼저, 불 대수식의 표현방법이다. 불 대수식을 표현하기 위해서는 sop form으로 표현할 수 있다. SOP form이란, sum of products을 뜻하며 변수들을 and로 묶여 있는 것들이 or로 묶이는 방식이라고 생각할 수 있다.

예를 들어, abc + ad + be 는 sop form 이며, ab(d+c) + ce는 sop가 아니다. 만약 ab(d+c) + ce와 같이 sop form이 아닌 것들을 sop로 만들기 위해서는 전개하는 과정이 필요하다.

SOP form으로 만들기 위해서는 각 변수들의 조합이 1이 되는 것들을 선택하고 이 조합 안에서 각 변수들을 and로 묶은 후 각 조합들을 마지막에 or로 합친다고 볼 수 있다. SOP와 비슷한 것으로 POS가 존재하는데 이는 product of sums를 뜻하고, or로 묶여 있는 것들이 and로 묶이는 방식이라고 할 수 있다. 그리고 SOP form와 POS form 모두 항상 two level이라는 성질을 가지고 있다.

그 다음은 불 대수식을 단순화하는 과정이다. 이는 Simplification라고 한다. Simplification을 하는 이유는 여러 가지가 있다. 회로의 size를 작게 하거나, 비용을 줄이고, 회로를 빠르게 하고, 전력을 적게 소모하는 등이 존재한다. 그리고 simplification이 되었음을 비교하는 기준으로는 literals, gates, cascaded levels of gates 총 3개의 개수가 있고, 이들을 통해 간단화가 되었는지 여부를 비교할 수 있다. 하지만 손으로 직접 불 대수식을 단순화하게 되면 하는 사람마다 결과가 다를 수 있고, 이가 정확하게 가장 단순화한 것인지 알 수 없다는 문제가 존재한다. 그래서, simplification의 방법으로는 Karnaugh map method와 quine-McClusky method가 존재한다. 이 중에서 이번 실험 때 직접 사용하는 방법은 카노맵이다. 사실 quine-McClusky method의 경우 Karnaugh map과 거의 비슷한 방법이라고 할 수 있지만 이는 systematic하여 컴퓨터 프로그램에 최적화된 방법이라고 할 수 있다. 그래서 이 실험에서 쓰는 카노맵의 경우 minimal SOP를 나타내는 것이라고 할 수 있다. 먼저 각 변수의 truth value에 따른 결과값들을 K-map table의 형태로 그려주고, 인접한 1인 것들을 모아서 합치는 방식이다. 이 때, 인접한 1들을 합치는 경우 항상 2^n 개씩 묶을 수 있다. 또 주의하여야 할 점은 3변수 이상부터는 행과 열에 gray code로 작성해야 된다.

3. 실험 준비

1) A>B

입력		A1A0			
		00	01	11	10
B1B0	00	0	1	1	1
	01	0	0	1	1
	11	0	0	0	0
	10	0	0	1	0

expression: A1′A0B1′B0′ + A1A0B1′B0′ + A1A0B1′B0 + A1A0B1B0′ + A1A0′B1′B0′ + A1A0′B1′B0이고,

카노맵을 적용해보면,

입력		A1A0			
		00	01	11	10
B1B0	00	0	1	1	1
	01	0	0	1	1
	11	0	0	0	0
	10	0	0	1	0

이므로, A1B1′ + A0B1′B0′ + A1A0B0′ 이다.

2) A=B

입력		A1A0			
		00	01	11	10
B1B0	00	1	0	0	0
	01	0	1	0	0
	11	0	0	1	0
	10	0	0	0	1

expression: A1′A0′B1′B0′ + A1′A0B1′B0 + A1A0B1B0 + A1A0′B1B0′이고,

카노맵을 적용시켜도 인접한 것들이 없어서 변함없이 A1′A0′B1′B0′ + A1′A0B1′B0 + A1A0B1B0 + A1A0′B1B0′ 이다.

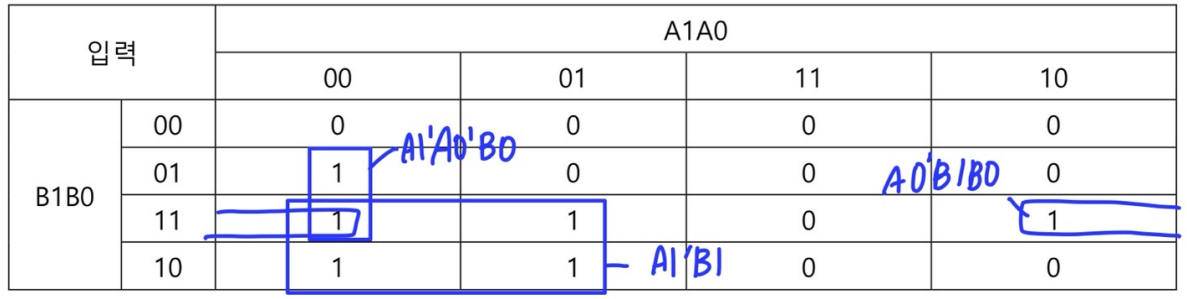
3) A<B

입력		A1A0			
		00	01	11	10
B1B0	00	0	0	0	0
	01	1	0	0	0
	11	1	1	0	1
	10	1	1	0	0

expression: A1′A0′B1′B0 + A1′A0′B1B0 + A1′A0′B1B0′ + A1′A0B1B0 + A1′A0B1B0′ + A1A0′B1B0이고,

카노맵을 적용시키면,

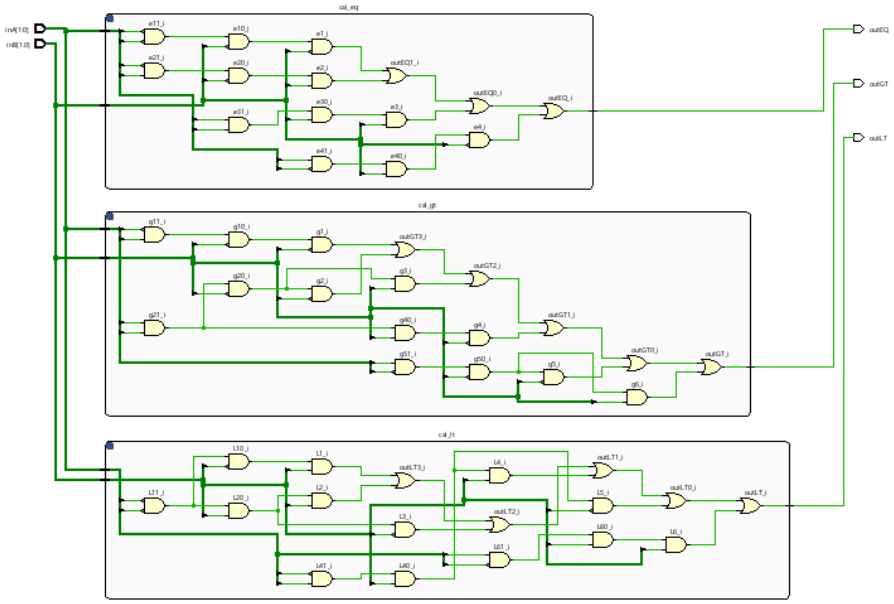
입력		A1A0			
		00	01	11	10
B1B0	00	0	0	0	0
	01	1	0	0	0
	11	1	1	0	1
	10	1	1	0	0



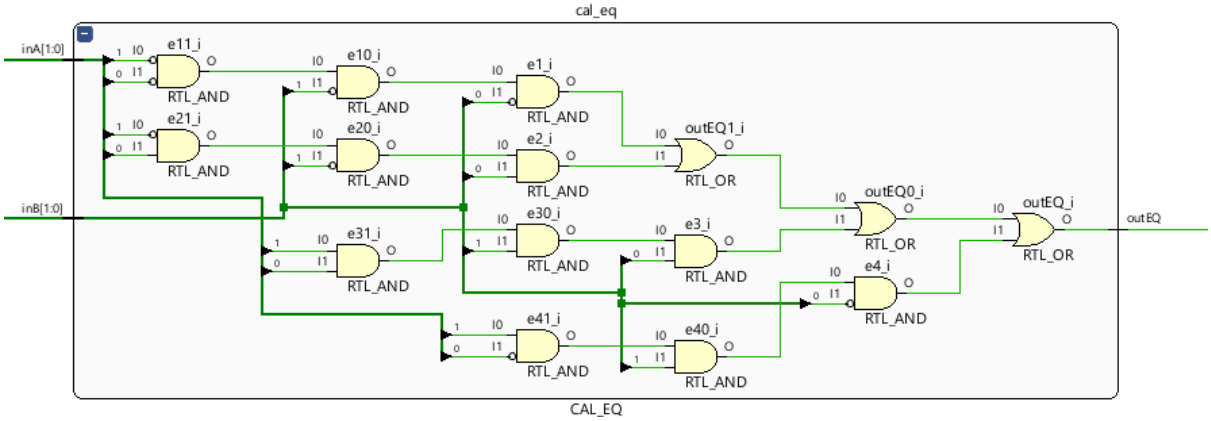
이므로, A1′B1+ A1′A0′B0 + A0′B1B0 이다.

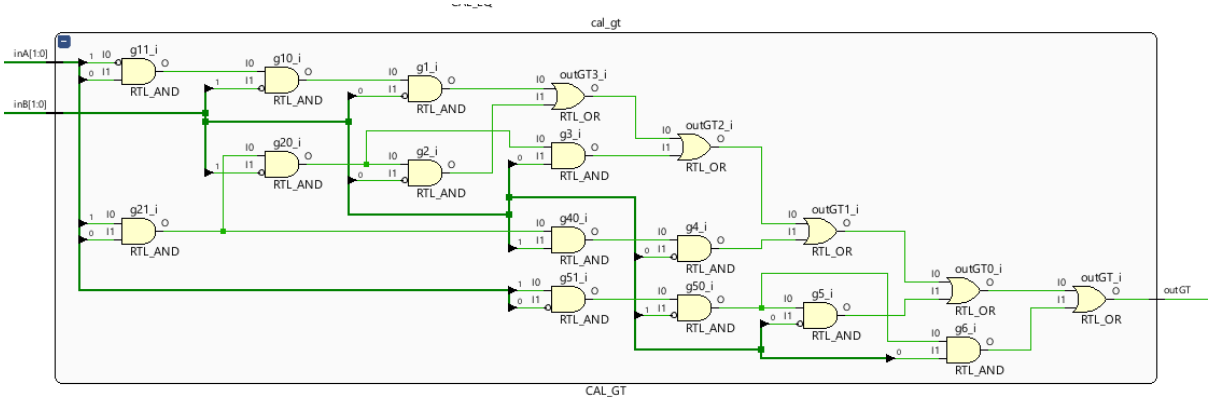
4. 결과

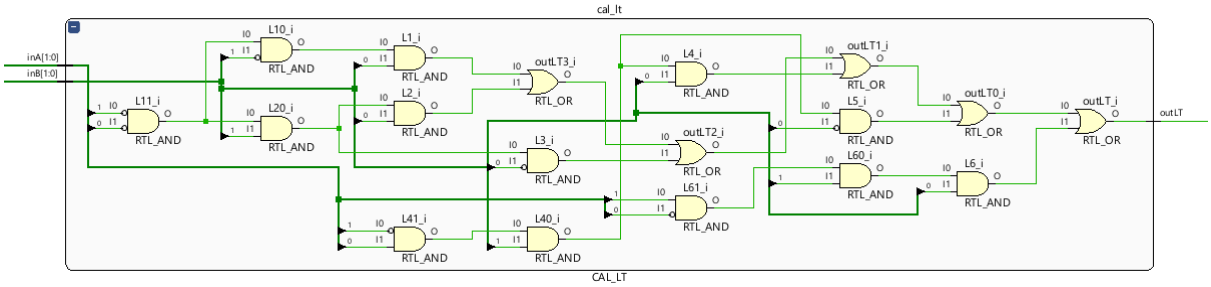
1) lab2_1.v의 회로도, 즉 단순화시키기 전 식의 회로도는 다음과 같았다.



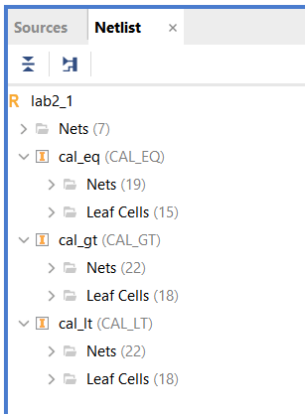
잘 보이지 않아 확대를 한 경우에는 이와 같았다.







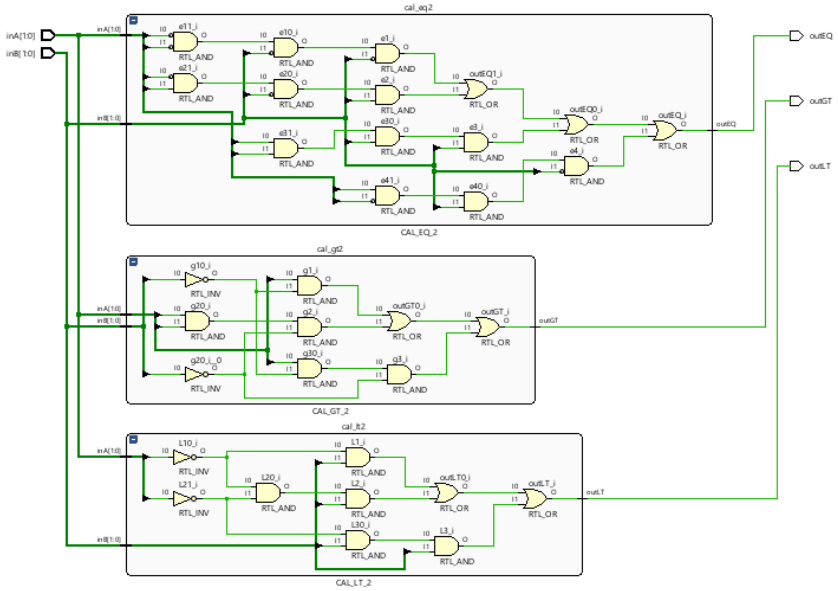
- 와이어 개수와 논리 게이트 개수는 다음과 같았다.



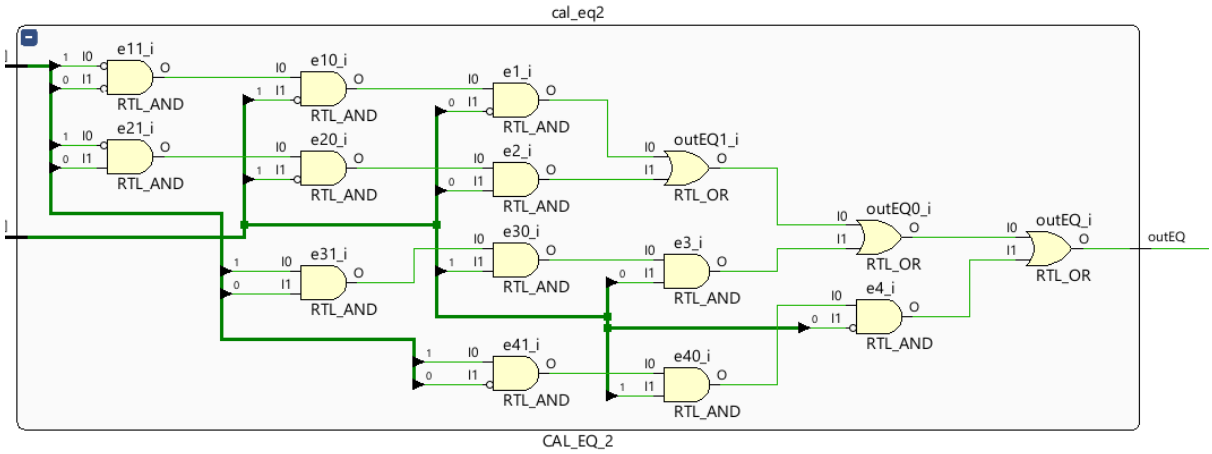
- cal_eq의 경우, 와이어는 19개, 논리 게이트는 15개가 나온다

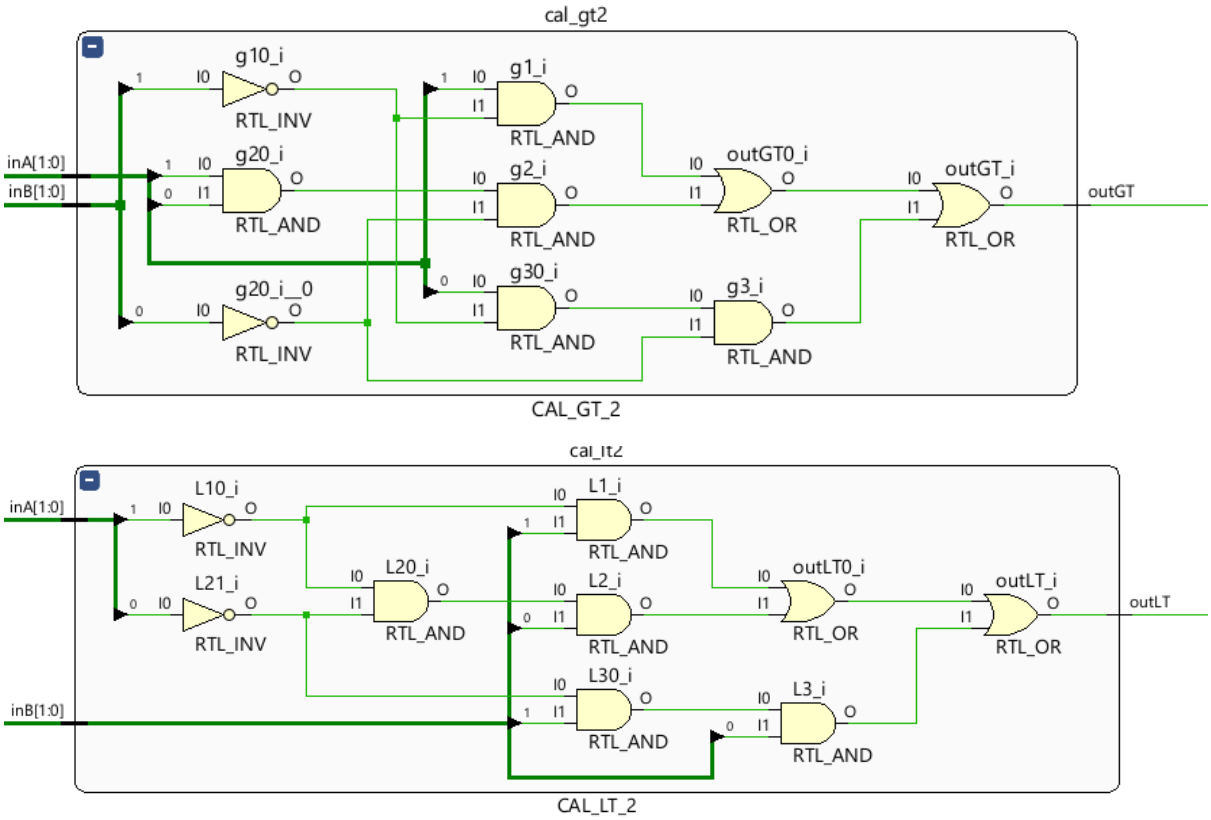
-cal_gt와 cal_lt는 와이어 22개, 논리 게이트는 18개가 나온다.

2) lab2_2.v의 회로도, 즉 k-map 적용한 식의 회로도는 다음과 같았다.

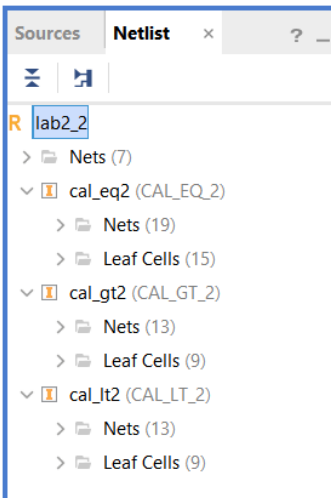


3개를 모두 확대하면,





와 같다. 다음으로, 와이어 개수와 논리 게이트 개수를 확인해보았다.



- cal_eq의 경우, 와이어는 19개, 논리 게이트는 15개로 이전과 같았다.

-cal_gt와 cal_lt는 와이어 13개, 논리 게이트는 9개가 나오며, 단순화시키기 전보다 개수가 줄었음을 확인할 수 있었다.

5. 논의

처음에는 어떻게 불 대수식을 SOP으로 잘 표현했지만 이를 베릴로그로 구현해야 할 때는 처음엔 막막했다. 그리고 SOP가 and와 or로 이루어져 있음을 다시 고려해보면서 먼저 SOP를 이루는 각각의 products들을 새로 만든 wire들을 output으로 잡아 and로 다 구현하려고 마음을 먹었다. 그리고 각각 products들을 구성하는 것들 중 complement들이 있었고 이들을 다 not으로 구현한 후 and로 연결하고, and의 output이었던 wire들을 or로 구현하는 계획을 세웠다. 하지만 complement가 너무 많아서 다 not으로 구현하기엔 힘들고 보기에 복잡해 보였다. 그래서 각각 앞에 ~을 붙여 표현하면서 훨씬 더 편리하고 보기 쉽게 코드를 짤 수 있었다. 사실 "~" 말고도 "&"와 "|"을 써도 된다는 말에 이런 것들도 써보려 했지만 저번 랩을 통해 and와 or으로 구현하는 것이 익숙해져서 ~만 not 대신에 사용하였다. 그렇지만 이 하나의 차이가 많은 편리함을 느끼게 해주었고 앞으로도 코드를 짬에 있어서 더 간단하고 편리한 코드를 짜기 위해 노력하여야 겠다는 다짐을 하였다. 그리고 이번 실습을 통해 회로도에서의 게이트 개수나 와이어 개수의 줄어듬을 확인하고, 이론 시간에 배웠던 simplification의 장점을 더더욱 체감할 수 있었다.