실험 2. 불 대수식의 단순화

이름 : 오승준
학번 : 20210661

1. 개요

이번 실험은 불 대수식의 단순화 방법에 대해 이해하고, 단순화 전 후를 비교하여, 그 효과를 확인하는 실습이다. 이번 실습을 통하여, K-map 알고리즘을 이해하고, 와이어와 논리 게이트 개수를 확인하여 단순화 효과를 확인한다.

2. 이론적 배경

1) 불 대수식의 단순화

- 불 대수식의 복잡도는 식을 구현하는 과정에서 이용된 와이어의 수와, 논리 게이트의 수로 평가한다. 불 대수식을 단순화하게 되면 이 두 개의 수가 줄어든다. 또한, 회로가 단순화됨으로 소비 전력이 감소하며, 작동 속도 또한 증가하는 효과를 기대할 수 있다.

단순화 방법으로는 카노 맵, 퀸 매클러스키 알고리즘이 있다.

2) 2-Bit Magnitude Comparator

2-Bit Magnitude Comparator란 서로다른 2bit 수를 입력할 시 이의 대소관계에 대해 출력하는 회로이다. 입력값에 따라 A>B, A=B, A<B 중 맞는 곳에만 1이 출력되고 그 외에는 0이 출력되는 형식이다.

3. 실험 준비

1) 2-Bit Magnitude Comparator의 세 출력 각각에 대한 식을 단순화하지 않고 작성한다.

- A>B

Input		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

단순화 전 식

F=A1'A0B1'B0'+A1A0'B1'B0'+A1A0'B1'B0+A1A0B1'B0'+A1A0B1'B0+A1A0B1B0'

- A=B

Input		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

단순화 전 식

F=A1'A0'B1'B0'+A1'A0B1'B0+A1A0'B1B0'+A1A0B1B0

- A<B

Input		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

단순화 전 식
F=A1'A0'B1'B0+A1'A0'B1B0'+A1'A0'B1B0+A1'A0B1B0'+A1'A0B1B0+A1A0'B1B0

2) K-map을 통해 세 식을 단순화한다.

- A>B

Input		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

(2, 3, 6, 7), (1, 3), (3, 11)을 EPI로 볼 시,
F = A1B1' + A0B1'B0' + A1A0B0'

- A=B

Input		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

K-map 알고리즘을 통해 단순화를 할 수 없으므로,
F = A1'A0'B1'B0' + A1'A0B1'B0 + A1A0B1B0 + A1A0'B1B0'

- A<B

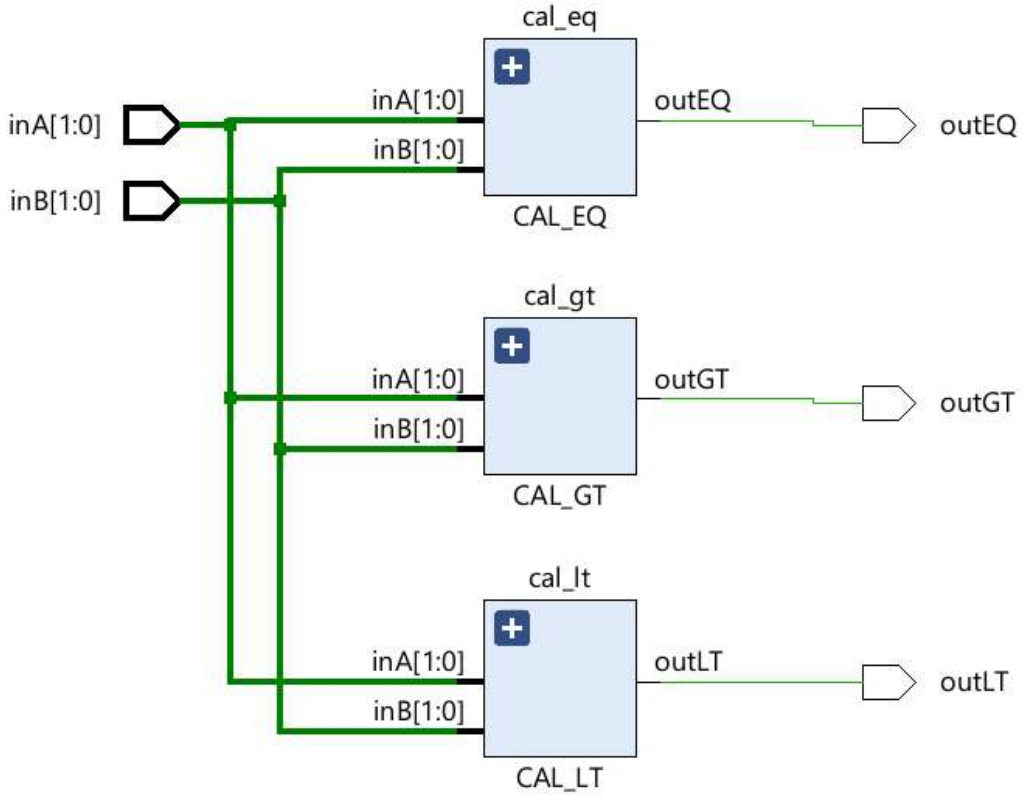
Input		A1A0			
		00	01	11	10
B1B0	00	0	0	0	0
	01	1	0	0	0
	11	1	1	0	1
	10	1	1	0	0

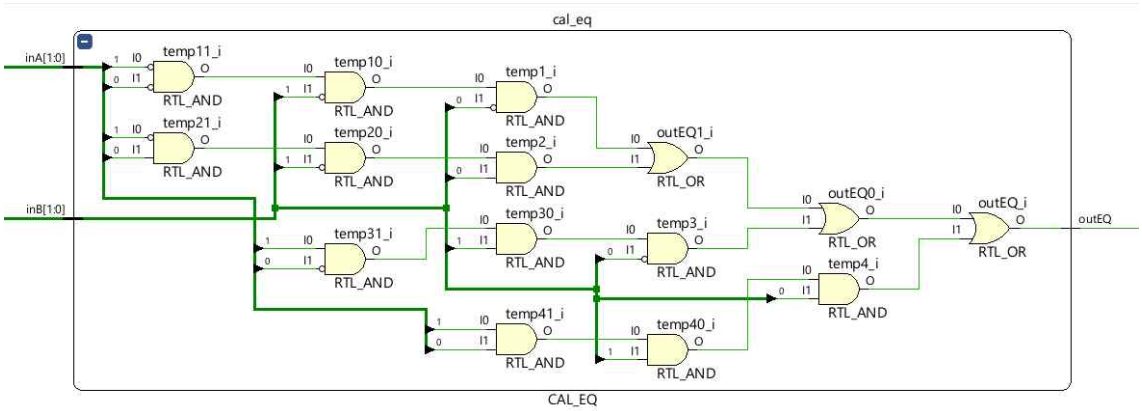
(4, 12), (8, 9, 12, 13), (13, 15)를 EPI로 볼 시,
F = A1'B1 + A0'B1B0 + A1'A0'B0

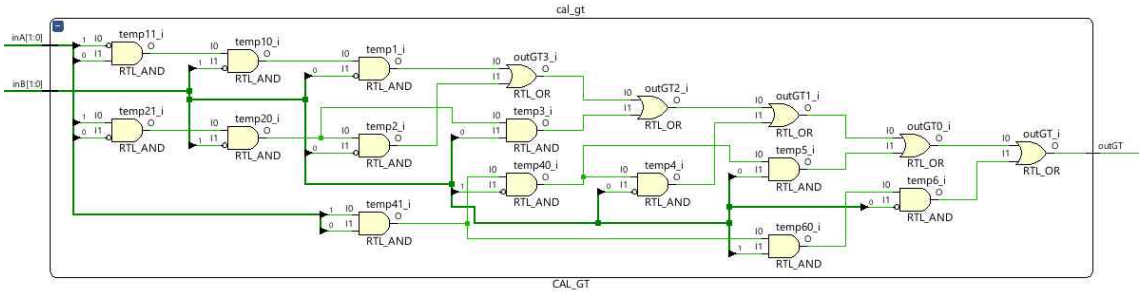
4. 실험 결과

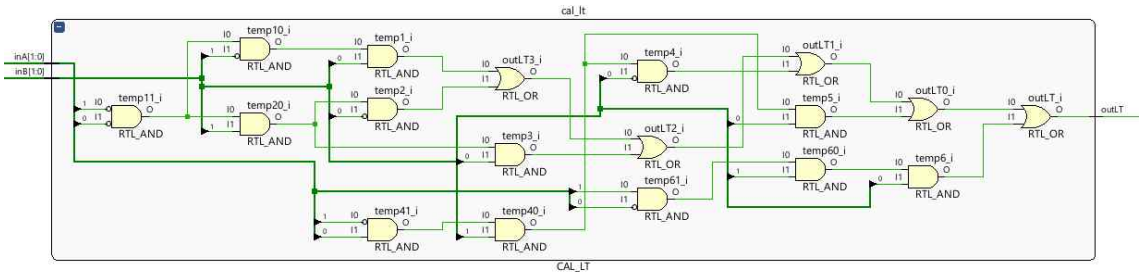
1)lab2_1

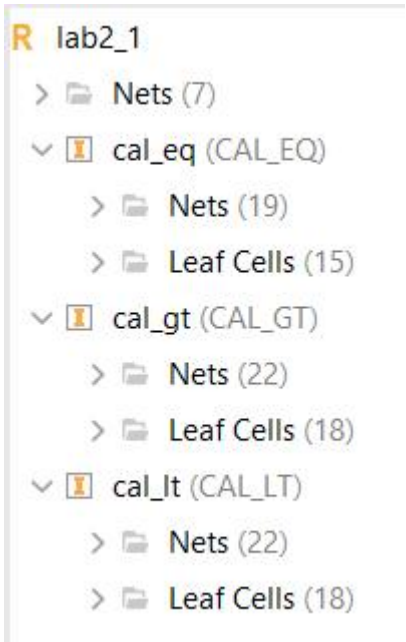
단순화 전 회로를 구성시 아래와 같다.





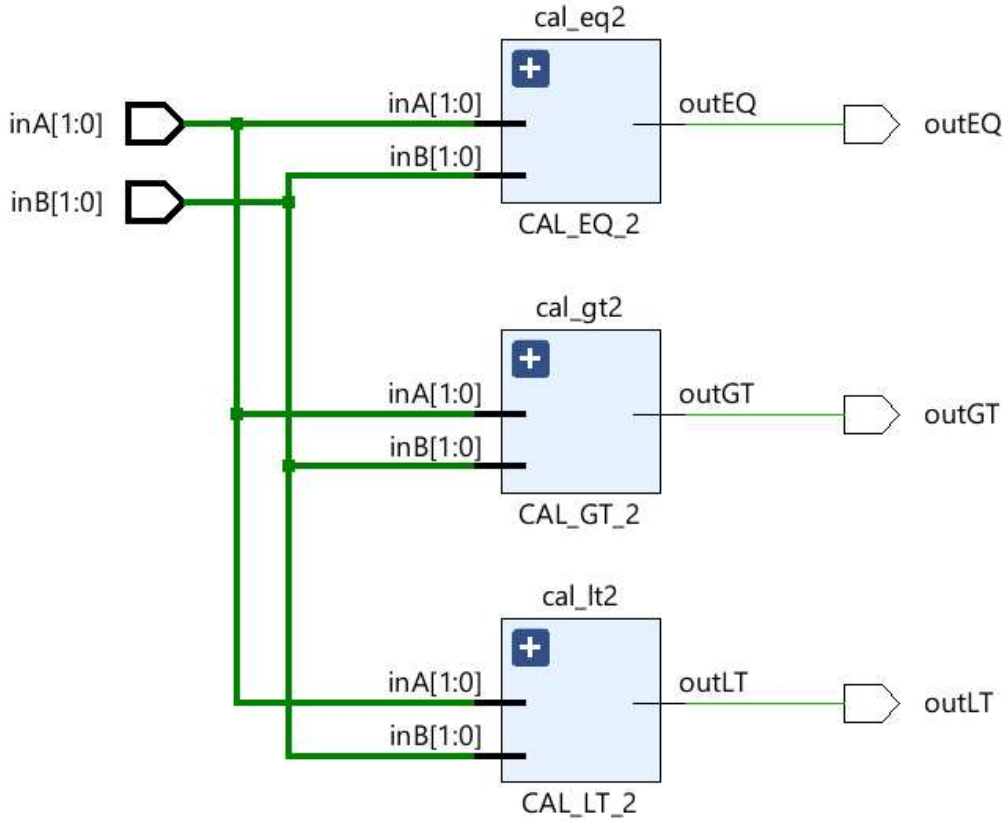


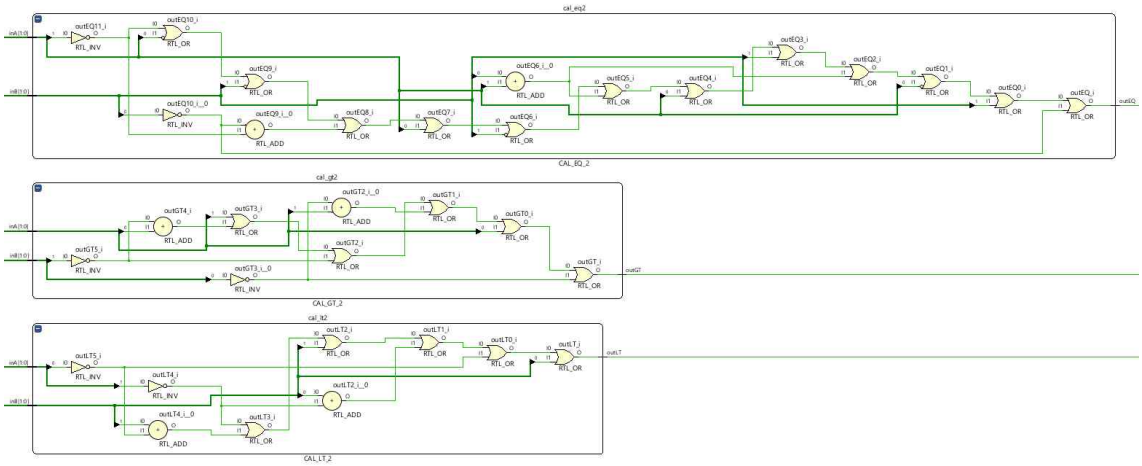


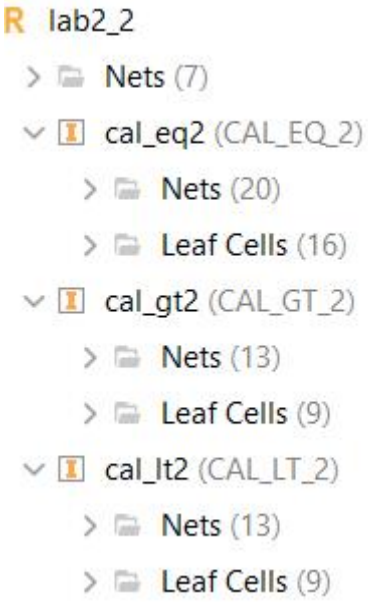


이때 각 Net, Leaf Cells의 수는 다음과 같다.
cal_eq : Nets : 19, Leaf Cells : 15
cal_gt : Nets : 22, Leaf Cells : 18
cal_lt : Nets : 22, Leaf Cells : 18

2) lab2_2
lab2_1의 회로를 단순화 한 후 구현시 다음과 같은 회로를 얻을 수 있다.


cal_eq2
inA[1:0]
inB[1:0]
inA[1:0]
inB[1:0]
outEQ
outEQ
CAL_EQ_2
cal_gt2
inA[1:0]
inB[1:0]
outGT
outGT
CAL_GT_2
cal_lt2
inA[1:0]
inB[1:0]
outLT
outLT
CAL_LT_2





이때의 각 Net와 Leaf Cells의 수는 다음과 같다.
cal_eq2 : Nets : 20, Leaf Cells : 16
cal_gt2 : Nets : 13, Leaf Cells : 9
cal_lt2 : Nets : 13, Leaf Cells : 9

실험 1과 비교 할 시, cal_eq2의 경우 Net와 Leaf Cell의 수가 1씩 증가 하였다. 그 이유는 cal_eq의 경우 K-map을 통해 단순화 할 수 없었기에 lab2_1과 lab2_2에서 동일한 식을 사용하였기 때문으로 보인다.
반면 cal_gt와 cal_lt의 경우 Net와 Leaf Cell의 수가 각각 9씩 감소한 것을 볼 수 있다. K-map을 통하여 단순화를 진행하였기 때문에 회로 구성에 이용된 Net와 Leaf Cell의 수가 감소한 것으로 보인다.

5. 논의
이번 실험을 통하여 K-map 알고리즘을 바탕으로 논리식을 단순화하는 방법에 대해서 익히고, Verliog 문법 중 Behavioral modeling 방식으로 코드를 작성하는 법을 익힐 수 있었다. K-map 방식이 항상 단순화가 가능한 것은 아님을 cal_eq의 케이스에서 확인 할 수 있었다. 코드가 Gate Level modeling으로 작성하는데 있어 복잡함이 있었으나 Lab2_2를 작성하는 과정에서 Behavioral Modeling 방식으로 코드를 짜는 법을 익혀 코드를 간결하게 작성할 수 있었다.