Lab2. 불 대수식의 단순화

20200437 김채현

1. 개요

본 lab은 K-map을 이용해 불 대수식을 단순화하는 방법을 이해하고, 단순화 전과 후를 비교하여 그 효과를 확인하는 것을 목표로 한다. 먼저, 불 대수식을 단순화하기 전 회로를 그리고 와이어와 논리 게이트의 개수를 확인한다. 그 후 K-map을 이용하여 불 대수식을 단순화하고, 회로를 그려 와이어와 논리 게이트가 얼마나 줄었는지 비교한다.

2. 이론적 배경

1) Magnitude Comparator

Magnitude Comparator이란 입력으로 2개의 수를 이진 형태로 받고 논리적인 비교를 하여 '보다 작다', '보다 크다', '같다' 등의 조건을 정하는 하드웨어 전자 장치이다. 변수나 미지수를 비교하기 위해 만들어진 기기이다. 2-Bit Magnitude Comparator는 입력으로 들어오는 두 개의 수가 2-bit인 것이다. 둘의 대소 관계를 비교하여 알맞으면 1을, 아니면 0을 출력한다.

2) Simplification

Simplification을 하는 이유에는 소비 전력 절감, 작동 속도 상승, 회로의 소형화 등이 있다. 또한 회로가 실행될 때 발생하는 delay를 줄일 수 있다. 이 때문에 회로는 가급적이면 단순화된 형태로 사용하는 것이 좋다. 불 대수식의 Complexity는 그 식을 구현할 때 사용되는 와이어와 논리 게이트의 개수를 가지고 측정되며, 불 대수식이 복잡할수록 복잡도는 커진다. 단순화의 방법에는 크게 세 가지가 있다.

첫째, 불 대수식의 동치과정을 이용해 단순화하는 방법이 있다. 단점은 결과가 가장 단순한 결과인지 알 수 없다는 것이다. 두 번째로는 K-map(Karnaugh-Map)을 이용하는 방법이 있다. 이는 불 대수에서 확장된 논리 표현을 사람의 패턴인식에 의해 연관된 상호관계를 이용하여 줄이는 방법으로 본 실험에서 사용되는 단순화 기법이다. 마지막으로 Quine-McCluskey를 이용하는 방법이 있다. 방법은 K-map과 비슷하지만, 그림을 그려서 맞추는 K-map과 달리 표를 사용하기 때문에 컴퓨터에서 쉽게 돌릴 수 있다.

2-level Simplification을 하는 경우 K-map과 Quine-McCluskey를 이용하는 것이 가장 단순한 식인지 확인하기 좋은 방법이지만, Multi-level Simplification을 할 경우 이 방법만으로는 가장 단순한

식을 얻기 힘들 수도 있기 때문에 다시 불 대수식을 이용하여 마저 Simplification을 진행하는 경우도 있다.

3. 실험 준비

1) 단순화 전

(1) outGT(A>B)

input		A1A0			
		00	01	11	10
B1 B0	00	0	1	1	1
	01	0	0	1	1
	11	0	0	0	0
	10	0	0	1	0

표 1. 입력 A, B에 따른 출력 "A>B"

단순화 전 outGT의 식은 다음과 같다.

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

(2) outEQ(A=B)

input		A1A0			
		00	01	11	10
B1 B0	00	1	0	0	0
	01	0	1	0	0
	11	0	0	1	0
	10	0	0	0	1

표 2. 입력 A, B에 따른 출력 "A=B"

단순화 전 outEQ의 식은 다음과 같다.

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

(3) outLT(A<B)

input		A1A0			
		00	01	11	10
B1 B0	00	0	0	0	0
	01	1	0	0	0
	11	1	1	0	1
	10	1	1	0	0

표 3. 입력 A, B에 따른 출력 "A<B"

단순화 전 outLT의 식은 다음과 같다.

A1'A0'B1'B0+A1'A0'B1B0+A1'A0B1B0+A1A0'B1B0+A1'A0'B1B0'+A1'A0B1B0'

2) K-map을 이용한 단순화 후

(1) outGT(A>B)

input		A1A0			
		00	01	11	10
B1 B0	00	0	1	1	1
	01	0	0	1	1
	11	0	0	0	0
	10	0	0	1	0

표 4. 표 1 outGT(A>B)의 K-map

표 4을 따라 outGT의 식을 단순화하면 다음과 같다.

A1B1'+A0B1'B0'+A1A0B0'

(2) outEQ(A=B)

input		A1A0			
		00	01	11	10
B1 B0	00	1	0	0	0
	01	0	1	0	0
	11	0	0	1	0
	10	0	0	0	1

표 5. 표 2 outEQ(A=B)의 K-map

표 5을 따라 outEQ의 식을 나타내면 K-Map을 이용하여 더 이상 단순화하여 나타낼 수 없으므로 다음과 같다.

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

(3) outLT(A<B)

input		A1A0			
		00	01	11	10
B1 B0	00	0	0	0	0
	01	1	0	0	0
	11	1	1	0	1
	10	1	1	0	0

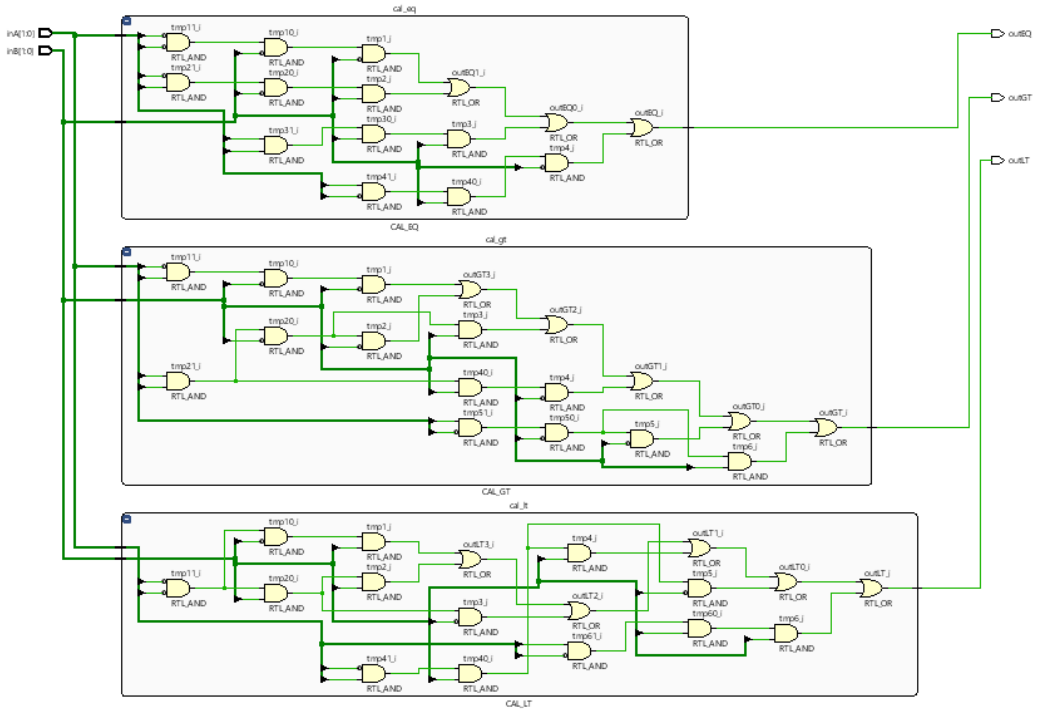
표 6. 표 3 outLT(A<B)의 K-map

표 6을 따라 outLT의 식을 단순화하여 나타내면 다음과 같다.

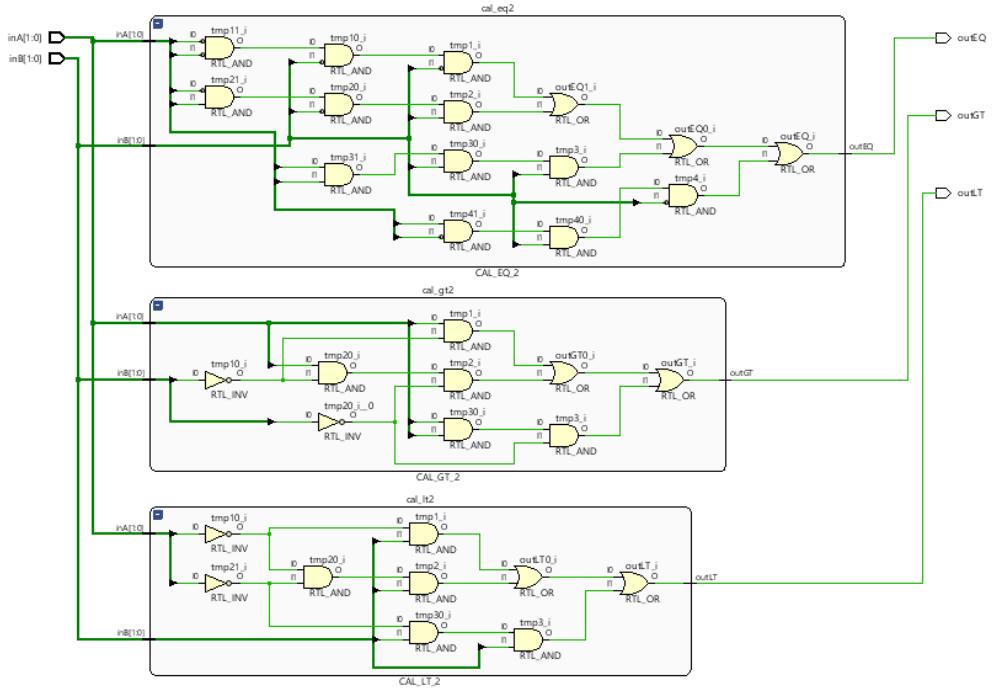
A1'B1+A1'A0'B0+A0'B1B0

4. 결과

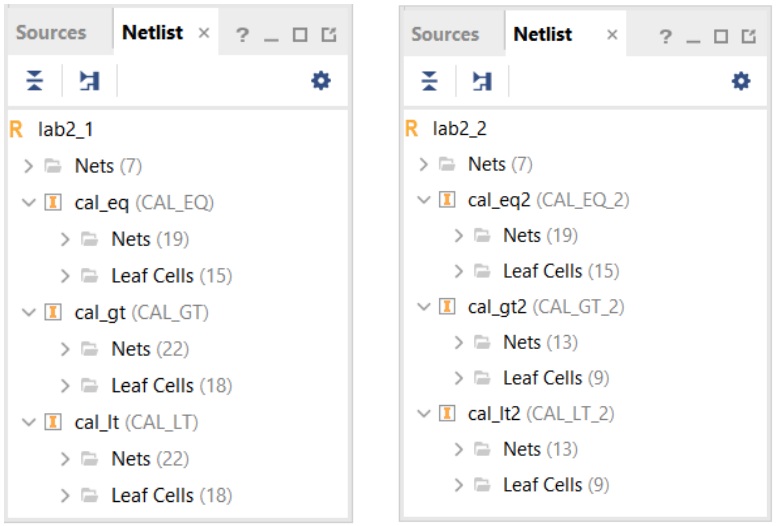
lab2_1의 schematic(단순화 전)은 다음과 같다.



lab2_2의 schematic(단순화 후)은 다음과 같다.



lab2_1과 lab2_2의 Netlist를 비교하면 다음과 같다.



Lab2_1에서 outGT의 와이어는 22개, 논리 게이트는 18개였다. outEQ의 와이어는 19개, 논리 게이트는 15개였다. outLT의 와이어는 22개, 논리 게이트는 18개였다. Lab2_2에서 outGT의 와이어는 13개, 논리 게이트는 9개였다. outEQ의 와이어는 19개, 논리 게이트는 15개였다. outLT의 와이어는 13개, 논리 게이트는 9개였다.

outGt(A>B)와 outLt(A<B)의 경우 단순화 전의 schematic에서는 22개의 와이어와 18개의 논리 게이트가 존재했지만, 단순화 후의 schematic에는 13개의 와이어와 9개의 논리 게이트가 존재하

였다. 와이어와 논리 게이트 모두 거의 절반만큼 그 개수가 줄어들게 되었으며, 이는 확연한 변화라고 할 수 있다. outEQ(A=B)의 경우 단순화 전의 schematic에서는 19개의 와이어와 15개의 논리 게이트가 존재했고, 단순화 후의 schematic에서도 19개의 와이어와 15개의 논리 게이트가 존재하여 그 개수가 이전보다 줄어들지 않았다. 나머지 두 식과 달리 simplification을 하여도 식의 변화가 없었기 때문인데, 이는 처음의 단순화 이전의 식 자체가 가장 단순화된 식인 최적의 결과라는 뜻이다.

5. 논의

K-map을 이용하여 불 대수식을 단순화하는 방법에 대해 익힐 수 있었다. 또한 실제로 단순화한 경우 와이어와 논리 게이트의 개수가 줄어드는 것을 schematic을 통해 시각적으로 확인할 수 있었다. 물론, 그렇지 않은 경우도 있었는데 outEQ의 경우가 그러했다. 초기의 식이 가장 단순화된 식이라면 K-map방법을 이용하여도 더 이상 단순화시킬 수 없다는 것을 알게 되었다. 이번 lab에서는 wire의 값 대입을 위해 assign 명령어 사용과 bitwise operator의 사용이 허용되어, gate-level-modeling을 하지 않고도 A'을 구현하였다. 다음에 기회가 있다면 단순화된 불 대수식이 소비 전력 절감, 작동 속도 상승, delay 감소 등의 효과가 있는지 알아볼 수 있는 실험을 하고 싶다.