Lab2. 불 대수식의 단순화

학번 : 20210273 이름 : 하태혁

1. 개요

Verilog로 불 대수식(Boolean expression)을 구현하는 과정에서 불 대수식을 단순화(Simplification)했을 때, 얻을 수 있는 이점에 대해 이해하고 Karnaugh map method를 사용하여 직접 불 대수식을 단순화해본다. 2-Bit Magnitude Comparator 알고리즘을 이해하고 Karnaugh Map을 그려서 식을 스스로 찾고 단순화하기 전과 후를 Verilog로 구현한다.

Schematic에서 구현한 모듈을 확인해 보면서 gate의 수와 wire의 수가 어떻게 변했는지 눈으로 확인해 본다. Netlist에서는 Nets과 Leaf Cells가 뜻하는 바를 이해하고 단순화하기 전과 후의 값 차이를 비교해 보는 것을 목표로 한다.

2. 이론적 배경

주어진 불 대수식 회로를 구현하기 전에 식을 단순화하는 과정을 거치게 되고 이 과정에서 이전에 배웠던 Boolean algebra의 공식들을 이용해 볼 수 있다. 단순화하게 되면 많은 이점을 얻을 수 있는데, 회로를 더 작게 만들거나 비용을 줄일 수 있고 전력 소모량을 감소시키거나 회로를 더 빠르게 만들 수 있다. 간단히 말하면 회로의 집적도(level of integration)를 높일 수 있다. 그러나 위 공식을 사용해 단순화하게 되면 주어진 식이 가장 단순하게 만들었는지를 확인하는 방법이 불명확하다. 주어진 불 대수식을 가장 단순한 식으로 만들어주는 방법에는 크게 두 가지가 있는데, Karnaugh map method와 Quine-McClusky method가 있다. Method를 사용하면 Literals의 수와 gates의 수를 가장 작게 만들고 Two-level gate networks로 회로를 구성할 수 있게 해준다.

3. 실험 준비

2-Bit Magnitude Comparator의 출력에 대한 불 대수식을 각각 작성하면 Karnaugh Map에서 각 요소를 인접 요소와 묶지 않고 그대로 표현한 식이 된다. 이후 단순화하는 과정에서는 인접 요소를 2^n 단위로 직사각형이 되도록 묶어서 단순화할 수 있다.

1) A>B일 때 Karnaugh Map

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

위 표에서 불 대수식을 단순화하지 않고 나타내면, A1'A0B1'B0' + A1A0B1'B0' + A1A0'B1'B0' +

A1A0B1′B0 + A1A0′B1′B0 + A1A0B1B0′ 이다.

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

단순화하기 위해 각 요소를 2^n 단위로 묶으면 다음과 같이 묶을 수 있고 식으로 나타내면 A1B1′ + A0B1′B0′ + A1A0B0′ 이다.

2) A=B일 때 Karnaugh Map

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

위 표에서 불 대수식을 단순화하지 않고 나타내면, A1′A0′B1′B0′ + A1′A0B1′B0 + A1A0B1B0 + A1A0′B1B0′이다. 위 표에서 각 요소는 2^n 단위로 묶을 수 없기 때문에 단순화할 수 없다. 즉, 단순화 전과 후의 식이 같다.

3) A<B일 때 Karnaugh Map

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

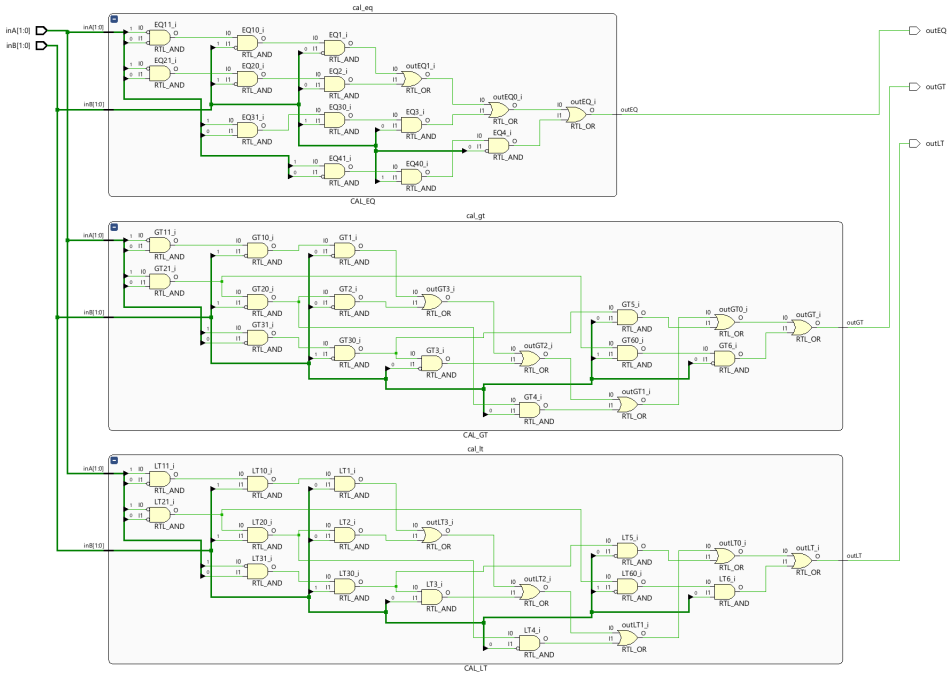
위 표에서 불 대수식을 단순화하지 않고 나타내면, A1A0′B1B0 + A1′A0′B1B0 + A1′A0B1B0 + A1′A0′B1B0′ + A1′A0B1B0′ + A1′A0′B1′B0 이다.

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

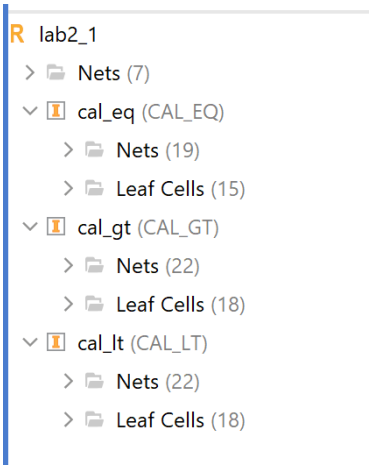
단순화하기 위해 각 요소를 2^n 단위로 묶으면 다음과 같이 묶을 수 있고 식으로 나타내면 A1'B1 + A0'B1B0 + A1'A0'B0 이다.

4. 결과

1) lab2_1

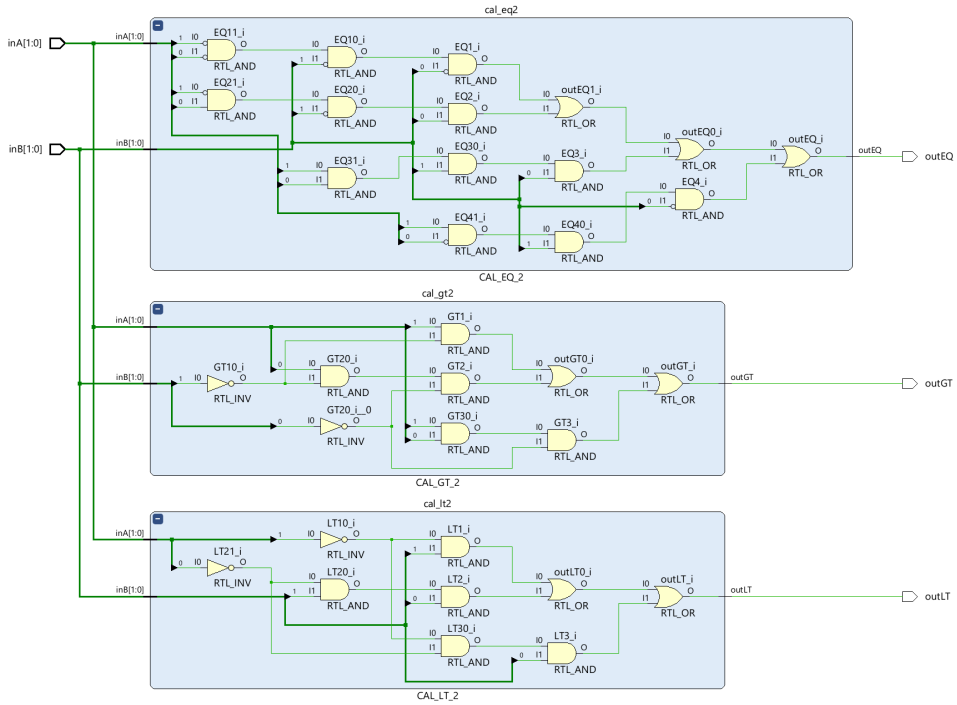


단순화 하기 이전에 각 모듈에 대한 식은 '3. 실험준비'에서 구한 식과 같다. 불 대수식을 바탕으로 알고리즘에 필요한 각 모듈을 구현하였다. SOP방식으로 구현되었으며 assign keyword를 이용해 Behavioral modeling으로 작성하였다.

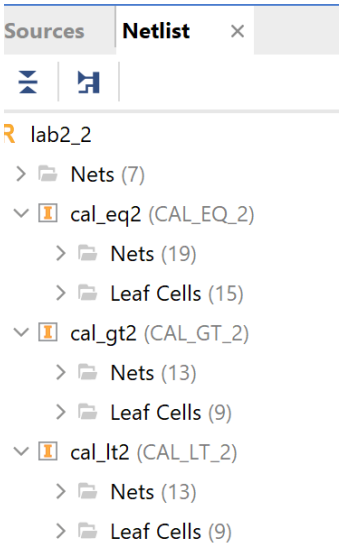


각 모듈에서 사용되는 Nets과 Leaf Cells의 수는 위 사진에 나온 바와 같다.

2) lab2_2



각 모듈에 대한 불 대수식을 단순화한 식은 '3. 실험준비'에서 Karnaugh Map으로 구한 식과 같다. 불 대수식을 바탕으로 알고리즘에 필요한 각 모듈을 구현하였다. SOP방식으로 구현되었으며 assign keyword를 이용해 Behavioral modeling으로 작성하였다.



각 모듈에서 사용되는 Nets과 Leaf Cells의 수는 위 사진에 나온 바와 같다. Lab2_1와 비교했을 때, cal_gt2와 cal_lt2의 gate 개수는 Nets(22)에서 Nets(13)으로 줄었고 wire의 개수도 Leaf Cells(18)에서 Leaf Cells(9)로 줄었다. cal_eq2의 수는 변하지 않았는데, 이는 eq 불 대수식은 단순화할 수 없기 때문이다.

5. 논의

Karnaugh Map을 사용하는 방식과 불 대수식을 단순화하여 모듈을 구성하는 방법을 실습해 볼 수 있었다. Netlist의 Nets와 Leaf Cells의 개수를 확인해보며 wire와 gate 수가 줄어든 것도 확인해 볼 수 있었다. 단순화의 장점 중, 회로의 크기를 줄여서 집접도를 올릴 수 있음을 확인할 수 있었다. 조금 더 확장해보고 싶은 점이 있다면, 단순화하여 속도가 빨라진 것도 확인할 수 있는 방법이 있다면 확인해 볼 수 있을 것 같다.