컴퓨터 공학 기초 실험2 보고서

실험제목: Traffic Light Controller with/without Left Turn Signals

실험일자: 2018년 09월 19일 (수)

제출일자: 2018년 10월 02일 (화)

학　　과: 컴퓨터공학과

담당교수: 공진흥 교수님

실습분반: 수요일 0, 1, 2

학　　번: 2015722025

성　　명: 정 용 훈

1. 제목 및 목적

A. 제목

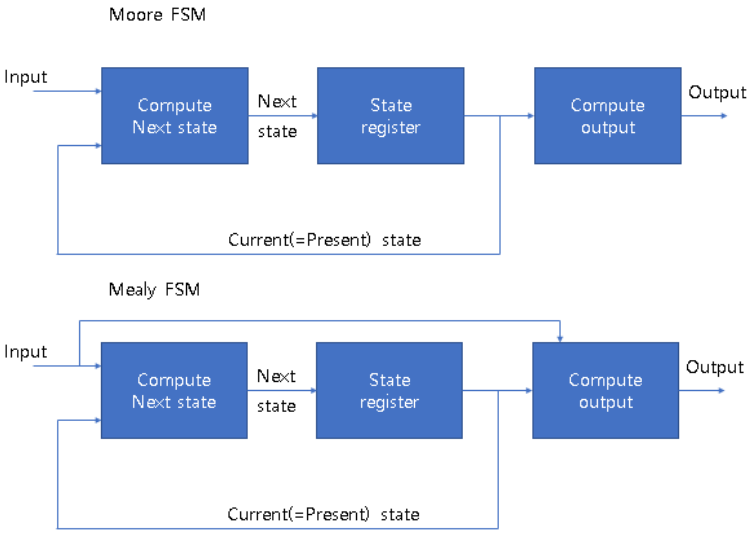
Traffic Light Controller with/without Left Turn Signals

B. 목적

1학기 디지털 논리회로에서 가장 중요하게 다루었던 FSM을 사용하여 Traffic Light Controller를 구현하는 과제입니다. Moore FSM의 동작을 다시 이해하며 이를 응용하여 자동차가 도로에 서있는 것을 인식하여 FSM에 따라 신호등의 불빛이 바뀌는 프로그램을 구현하며 Top module의 instance될 sub module들의 각 기능을 이해하며 구현합니다.

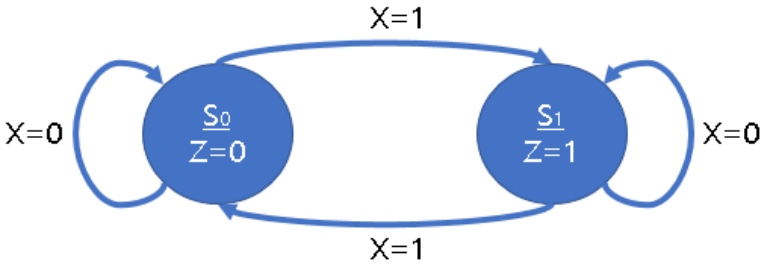
2. 원리(배경지식)

이번 과제는 앞서 목적에서 말한 것처럼 FSM을 응용하는 실험입니다. FSM이란 Finite State Machine을 약자로 함으로써 유한 상태 기계를 의미합니다. 동작으로는 조건이 있는 명령어에 따라 몇 개의 동작을 반복해서 수행하는 기계를 뜻합니다. 이런 FSM의 종류에는 Mealy FSM과 Moore FSM 두가지 종류로 나뉘게 되는데 두 FSM의 차이 상태에 따른 조건이 뿐만 아니라 input에 대한 조건이 output에 영향을 주는 유무에 차이가 있습니다. 아래 Mealy와 Moore의 FSM의 이미지를 확인하면 쉽게 이해할 수 있습니다.

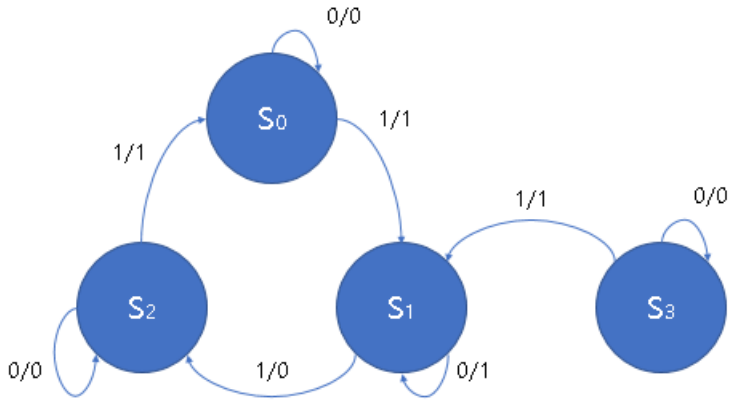


조금 더 이해를 돕기 위해선 위 FSM을 사용한 Diagram을 통해서 output의 변화를 보면 알기 쉽습니다.

Moore FSM



Mealy FSM



위와 같은 diagram을 보면 Moore FSM 각 상태에 output의 값이 결정되었지만 Mealy FSM은 output의 결정이 상태에도 있지만 input의 값에 따라 바뀔 수 있습니다. 이번 실습에서는 특정 상태에서 신호등의 불빛이 결정되는 Moore FSM을 사용한 신호등을 구현하는 것이 목적입니다. 해당 과제를 Verilog로 구현하기 위해서는 해당 input과 output에 대한 규칙을 카노맵을 통하여 불식을 도출하여 module을 구현할 수 있어야 합니다. 세부 설계사항에서 더 자세하게 다룰 카노맵의 사용법과 예시로는 아래와 같습니다.

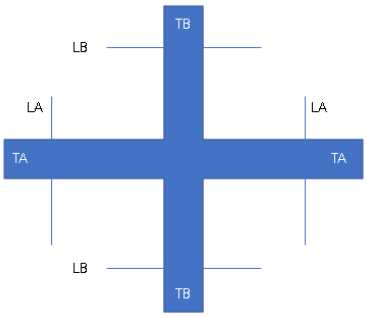
	00	01	11	10
00	0 0	0 1	0 3	0 2
01	0 4	1 5	1 7	0 6
11	0 12	1 13	1 15	0 14
10	1 8	0 9	0 11	1 10

X = A|B|D + BD
- A|B|D
- BD

다음 이미지는 input이 4개가 있을 때 형성될 수 있는 카노맵입니다. 표를 작성하게 되면 각각의 비트가 1씩 차이가 날 수 있도록 배열하며 1비트가 차이나는 output의 값이 1인 결과를 1, 2, 4 ,8... 의 규칙을 가지는 개수로 묶어서 다음과 같이 식을 도출할 수 있습니다. 즉 Traffic light를 만들기 위해서는 input에 따르는 상태에 대한 식과 그 상태에 따른 불빛의 output을 Encoding할 수 있어야 하며 그 output의 규칙을 카노맵에 따라 정리하여 식을 도출하고 도출된 식을 통하여 Verilog로 구현할 수 있어야합니다.

3. 설계 세부사항

A. Detail functions of Top modules



tl_cntr 동작 방식

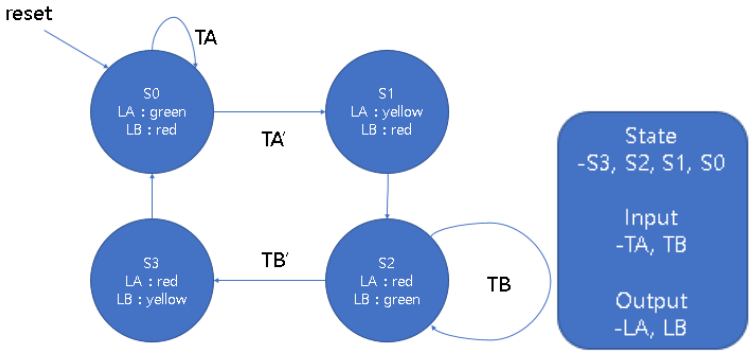
1. Traffic light는 교통이 도로에 서있는 차를 감지하여 차가 없을 때는 불이 초록색에서 노란색을 거쳐 빨간색으로 변한다.
2. 만약 light가 LA가 초록색 또는 노란색이라면 대응하는 light LB는 빨간색이다.
3. LB의 규칙 또한 1, 2번의 규칙과 동일하다.

tl_cntr_w_left 동작 방식

1. 도로에 차량이 없다면 초록불에서 노란색불을 거쳐 좌회전으로 변한다.
2. 좌회전하는 차량이 없다면 좌회전 신호에서 노란색불을 거쳐 빨간색불로 변한다.
3. 좌회전하는 차량이 없다고 가정해도 초록불에서 좌회전 신호는 반드시 거친다.
4. Light LA가 초록색, 노란색, 좌회전 신호일 경우 LB는 빨간색불을 유지한다.
5. LB의 경우도 마찬가지로 1, 2, 3, 4의 동작방식을 따르게 되어있다.

B. Details of tl_cntr

◆ State Diagram



Moore FSM을 응용한 형태로써 상태 S0, S1, S2, S3에 따라 output의 값이 결정된다. 상태에 따른 각각의 output은 원안에 그려져 있는 상태이며 S1의 상태와 S3의 상태는 특정한 input이 없어도 Clock에 의해 상태가 다음으로 바뀌게 된다.

◆ Encoded State Table

State	Encode
S0	00
S1	01
S2	10
S3	11

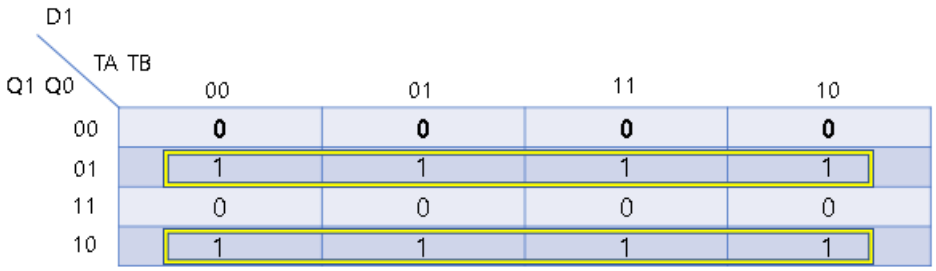
sLight	Encode
Green	00
Yellow	01
Red	10

위 표는 상태에 대한 Encode와 output이 될 불빛에 대한 Encode이다.

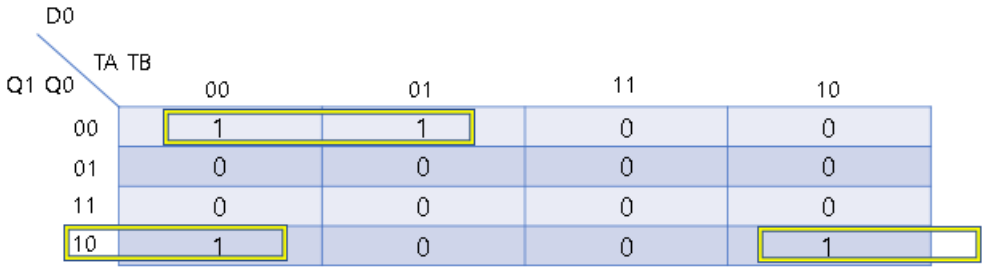
다음으로는 상태와 input에 대한 next state를 정리한 표이다.

Current state		Inputs		Next state	
Q1	Q0	TA	TB	D1	D0
0	0	0	X	0	1
0	0	1	X	0	0
0	1	X	X		0
1	0	X	0	1	1
1	0	X	1	1	0
1	1	X	X	0	0

다음으로는 식을 도출하기 위하여 각 input에 관련해 도출된 output을 바탕으로 카노맵을 사용하여 식을 도출합니다. 식은 다음과 같이 도출합니다.



D1 = Q1′Q0*Q1Q0′ = Q1⊕Q0

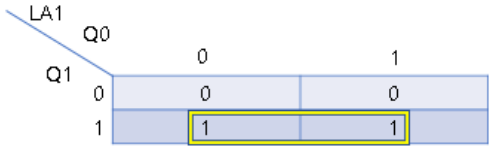


D0 = Q1′ * Q0′ * TA′ + Q1 * Q0′ * TB′

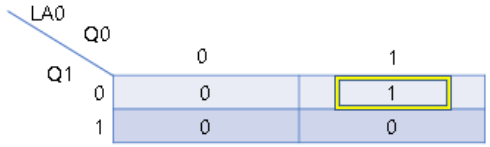
다음은 Current state에 대한 신호등 불빛에 대한 output 값을 정리한 표다.

Current state		**Outputs**			
Q1	Q0	LA1	LA0	LB1	LB0
0	0	0	0	1	0
0	1	0	1	1	0
1	0	1	0	0	0
1	1	1	0	0	1

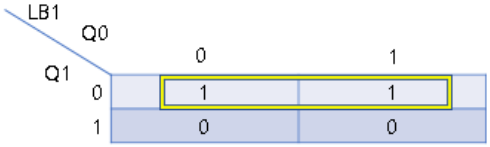
위와 같은 방법으로 input에서 도출한 output의 값을 토대로 불식을 카노맵을 이용하여 나타내면 아래와 같이 나타낼 수 있다.



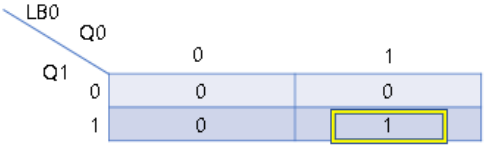
LA1 = Q1



LA0 = Q1′ * Q0

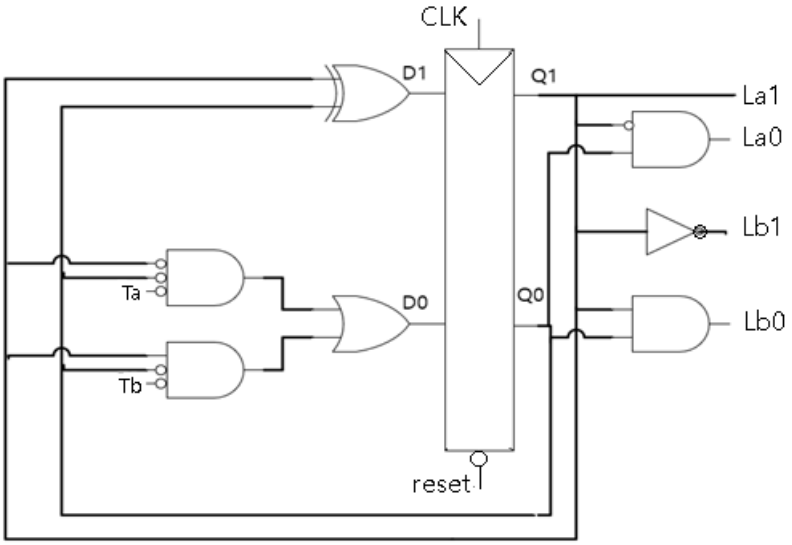


LB1 = Q1′



LB0 = Q1*Q0

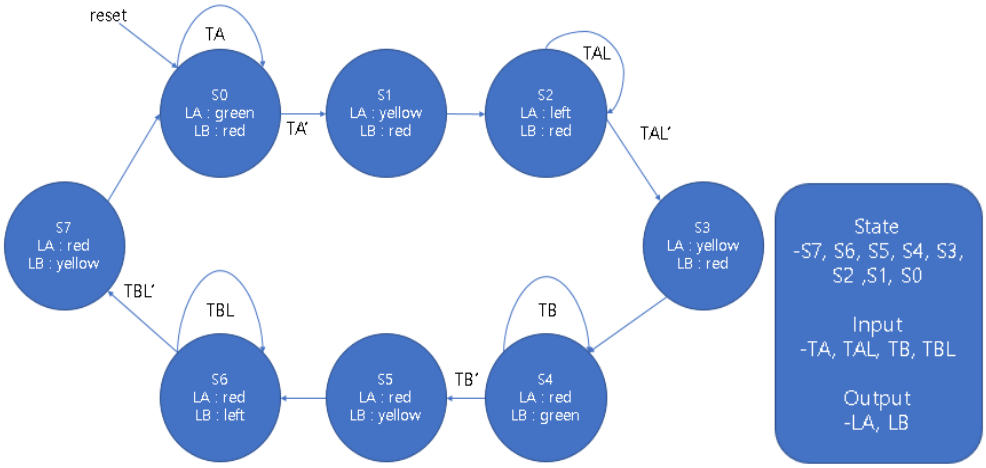
◆ FSM Schematic



Verilog Design		
Top module	**tl_cntr**	**ns_logic + _register2_r+o_logic**
Sub module	ns_logic	D1 = Q1⊕Q0 D0 = Q1'*Q0'*TA' + Q1*Q0'*TB'
Sub module	_register2_r	2-bits D flip-flop
Sub module	_dff_r	Asynchronous D flip-flop
Sub module	o_logic	LA1 = Q1, LA0 = Q1' * Q0 LB1 = Q1', LB0 = Q1 * Q0

C. Details of tl_cntr_w_left

◆ State Diagram



미리 설계된 tl_cntr과 같은 방법으로 설계된 Diagram입니다. 차이점으로는 left 신호가 추가된 특징이 있으며 left신호에서는 노란색불이 나오는 상태처럼 다음 상태로 자동으로 넘어가는 것이 아니라 특정한 input(왼쪽 방향 차량의 유무)에 의하여 다음 상태가 결정되는 모습을 볼 수 있습니다.

◆ Encoded state Table

State	Encode
S0	000
S1	001
S2	010
S3	011
S4	100
S5	101
S6	110
S7	111

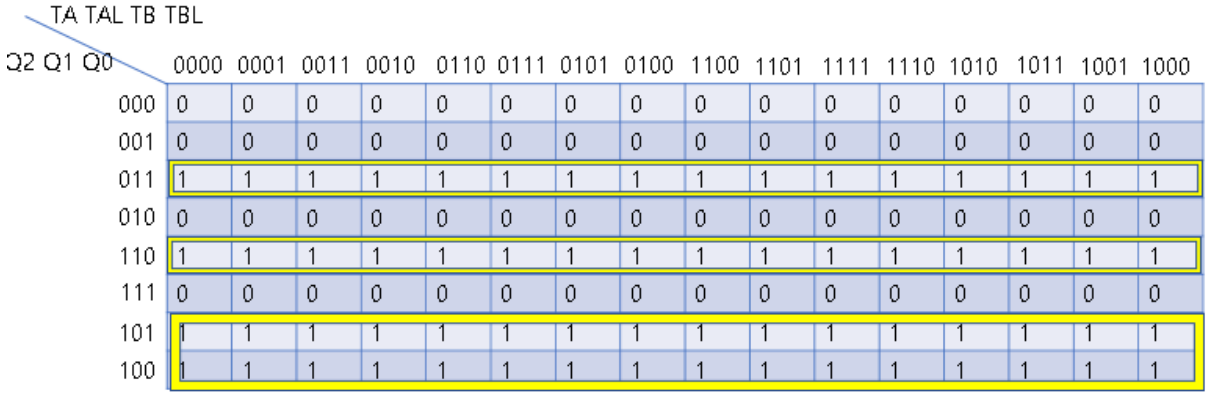
Light	Encode
Green	00
Yellow	01
Left	10
Red	11

위 표는 상태와 Light에 대한 Encode를 정리한 표입니다. 상태에 대한 비트가 하나 더 올라간 것을 확인 할 수 있습니다.

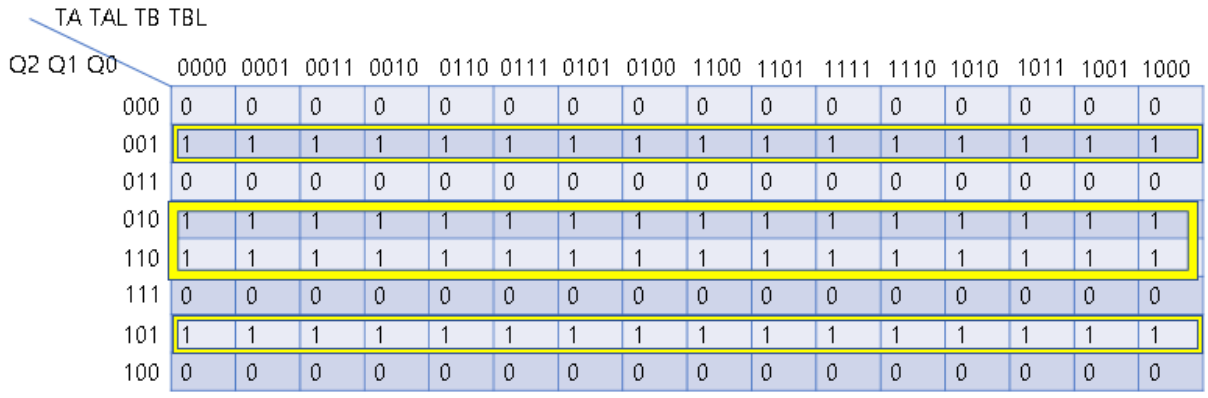
다음은 Input에 대한 상태의 output 결과를 Encode로 정리한 표입니다.

Input							Output		
Q2	Q1	Q0	TA	TAL	TB	TBL	D2	D1	D0
0	0	0	0	X	X	X	0	0	1
0	0	0	1	X	X	X	0	0	0
0	0	1	X	X	X	X	0	1	0
0	1	0	X	0	X	X	0	1	1
0	1	0	X	1	X	X	0	1	0
0	1	1	X	X	X	X	1	0	0
1	0	0	X	X	0	X	1	0	1
1	0	0	X	X	1	X	1	0	0
1	0	1	X	X	X	X	1	1	0
1	1	0	X	X	X	0	1	1	1
1	1	0	X	X	X	1	1	1	0
1	1	1	X	X	X	X	0	0	0

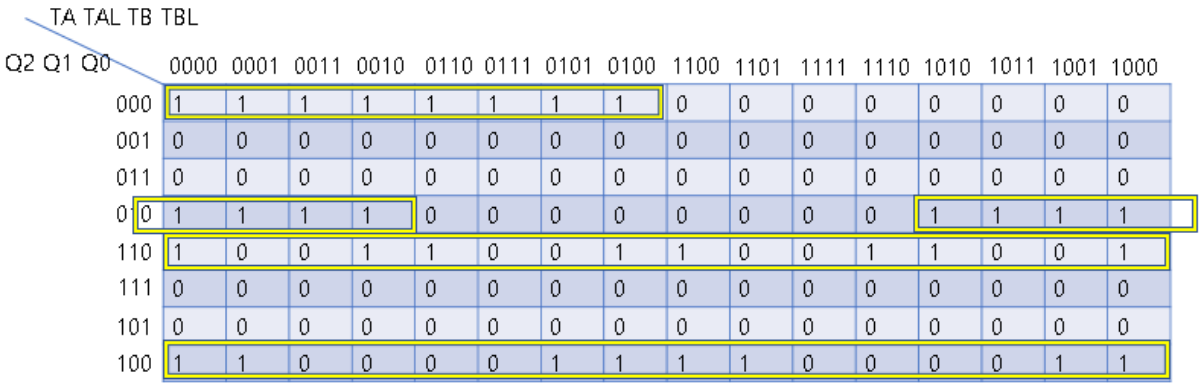
Input에 대한 결과값을 카노맵을 사용하여 나타내면 아래와 같습니다.



D2 = Q2′ * Q1 * Q0 + Q2 * Q1′ + Q2*Q1*Q0′



D1 = Q2′ * Q1′ * Q0 + Q1 * Q0′ + Q2*Q1′*Q0



TA TAL TB TBL / Q2 Q1 Q0	0000	0001	0011	0010	0110	0111	0101	0100	1100	1101	1111	1110	1010	1011	1001	1000
000	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
011	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
010	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1
110	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1
111	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
101	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
100	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1

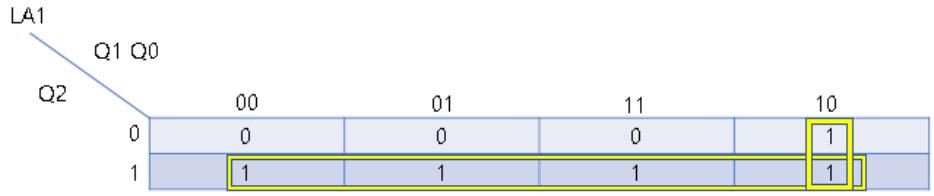
D0 = Q2′*Q1′*Q0′*TA′ + Q2′*Q1*Q0′*TAL′ + Q2*Q1′*Q0′*TB′ + Q2*Q1*Q0′*TBL′

5번째 8번째 표를 보면 1 bit씩 차이는 원소를 묶을 수 있는 카노맵의 특징을 사용하여 따로 카노맵을 사용하면 계산할 수 있다.

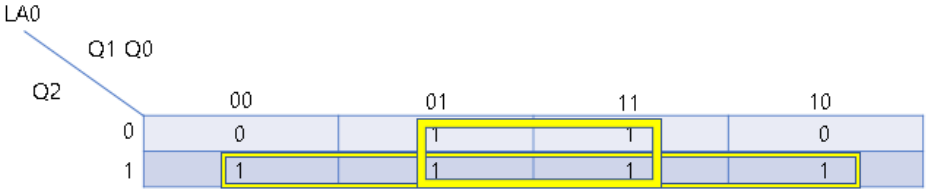
다음은 신호등의 light에 관련된 상태에 따른 Endcode값을 정리한 표입니다.

Current State			Outputs			
Q2	Q1	Q0	LA1	LA0	LB1	LB0
0	0	0	0	0	1	1
0	0	1	0	1	1	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1
1	0	0	1	1	0	0
1	0	1	1	1	0	1
1	1	0	1	1	1	0
1	1	1	1	1	0	1

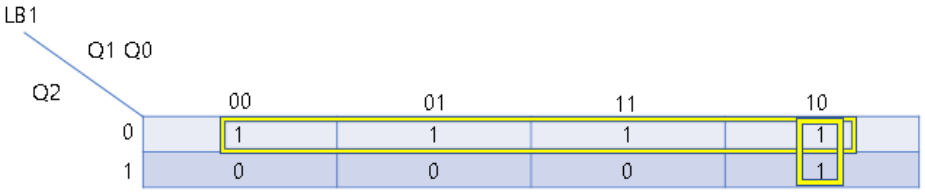
Outputs에 관련한 카노맵은 다음과 같이 나타낼 수 있다.



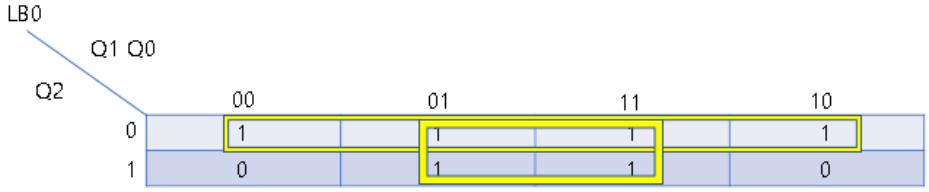
LA1 = Q1*Q0' + Q2



LA0 = Q0 + Q2

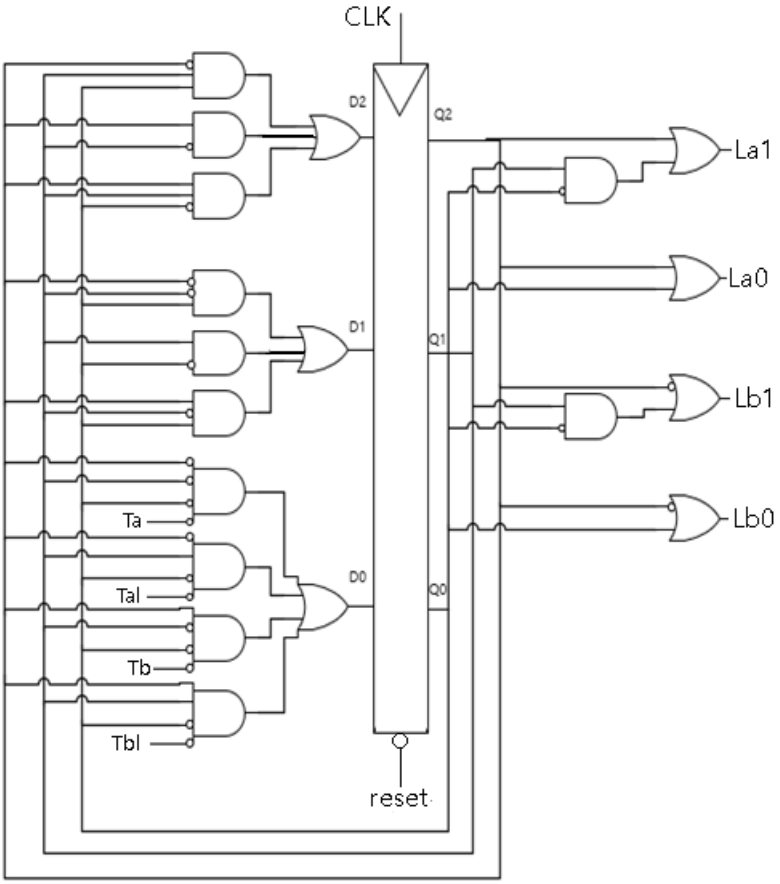


LB1 = Q1*Q0' + Q2'



LB0 = Q0 + Q2'

◆ **FSM Schematic**

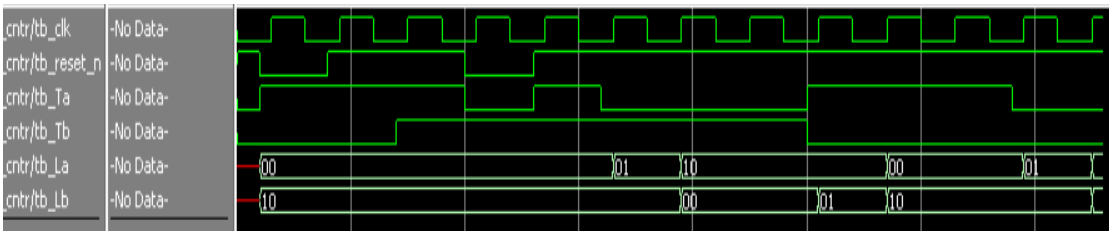


Verilog Design		
Top module	**tl_cntr_w_left**	**ns_logic + _register3_r + o_logic**
Sub module	ns_logic	D2 = Q2'*Q1*Q0+Q2*Q1'+Q2*Q1*Q0' D1=Q2'*Q1'*Q0+Q1*Q0'+Q2*Q1'*Q0 D0=Q2'*Q1'*Q0'*TA'+Q2'*Q1*Q0'*TAL'+ Q2*Q1'*Q0'*TB'+Q2*Q1*Q0'*TBL'
Sub module	_register3_r	3 bits D flip flop
Sub module	_dff_r	Asynchronous D flip fop
Sub moudle	o_logic	LA1 = Q1*Q0'+Q2 LA0 = Q0+Q2 LB1 = Q1*Q0'+Q2' LB0 = Q0+Q2'

4. 설계 검증 및 실험 결과

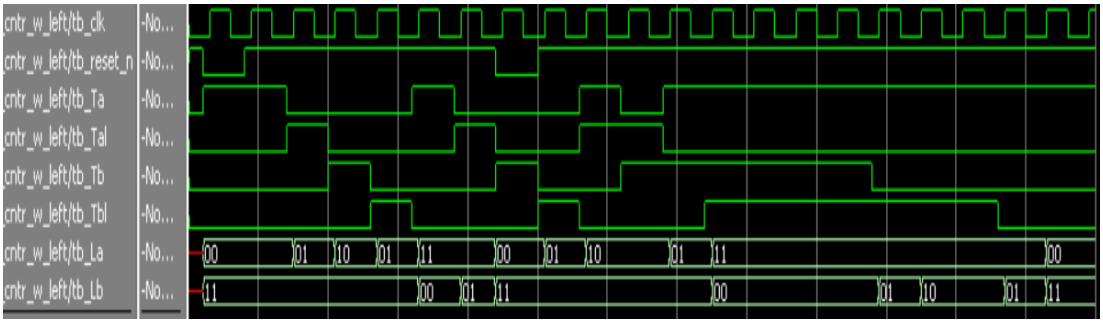
A. 시뮬레이션 결과

tl_cntr



Reset의 동작을 확인하기 위하여 중간에 한번 reset을 더 넣게 되었습니다. Ta와 Tb의 값은 wave form을 보면 변화를 간단히 볼 수 있습니다. 첫번째 reset부터 두번째 reset까지는 Ta의 값이 계속 1로 rising되어 있기 때문에 상태의 변화가 없습니다 Ta의 값이 0이되고 clk이 rising이 되지만 reset이 동작하므로 변화가 없는 것을 볼 수 있습니다. 그 이후로는 FSM의 규칙에 맞게 인풋과 상태에 따라 상태가 계속 변하는 것을 확인 할 수 있습니다.

tl_cntr_w_left

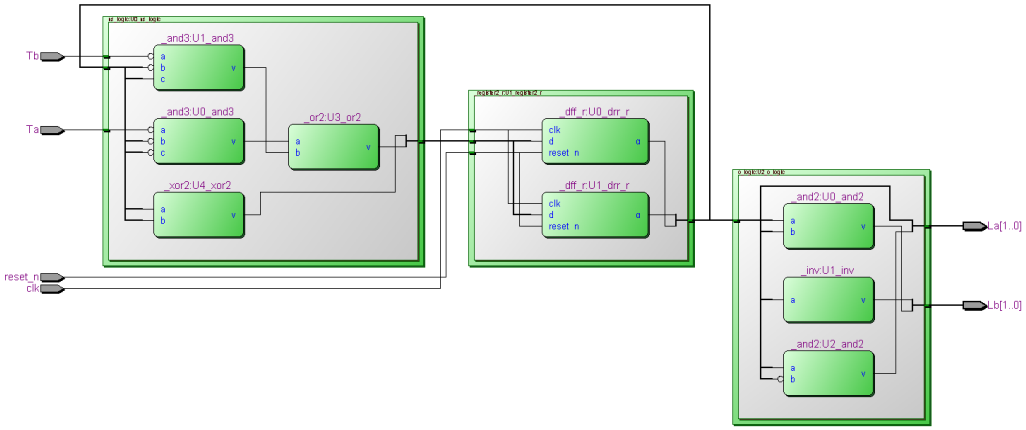


전 과정과 마찬가지로 reset의 동작을 확인하기 위하여 테스트벤치에 두번의 reset을 설정하여 결과를 확인했습니다. 첫번째 reset에서는 동작을 하고 있지만 Ta의 값이 1이기때문에 확실한 결과를 확인 할 수 없었습니다. 하지만 두번째 reset이 동작하는 부분을 보면 동작이 비동기식으로 동작하기 때문에 reset이 동작하자 마자 상태가 바로 처음으로 돌아가는 것을 확인 할 수 있었습니다. 그 후로는 La가 빨간색 불이 되기 전까지는 Lb의 값이 빨간색을 유지하는 것을 볼 수 있으며 그 반대의 경우도 마찬가지로 같습니다.

B. 합성(synthesis) 결과

1. tl_cntr

RTL Viewer



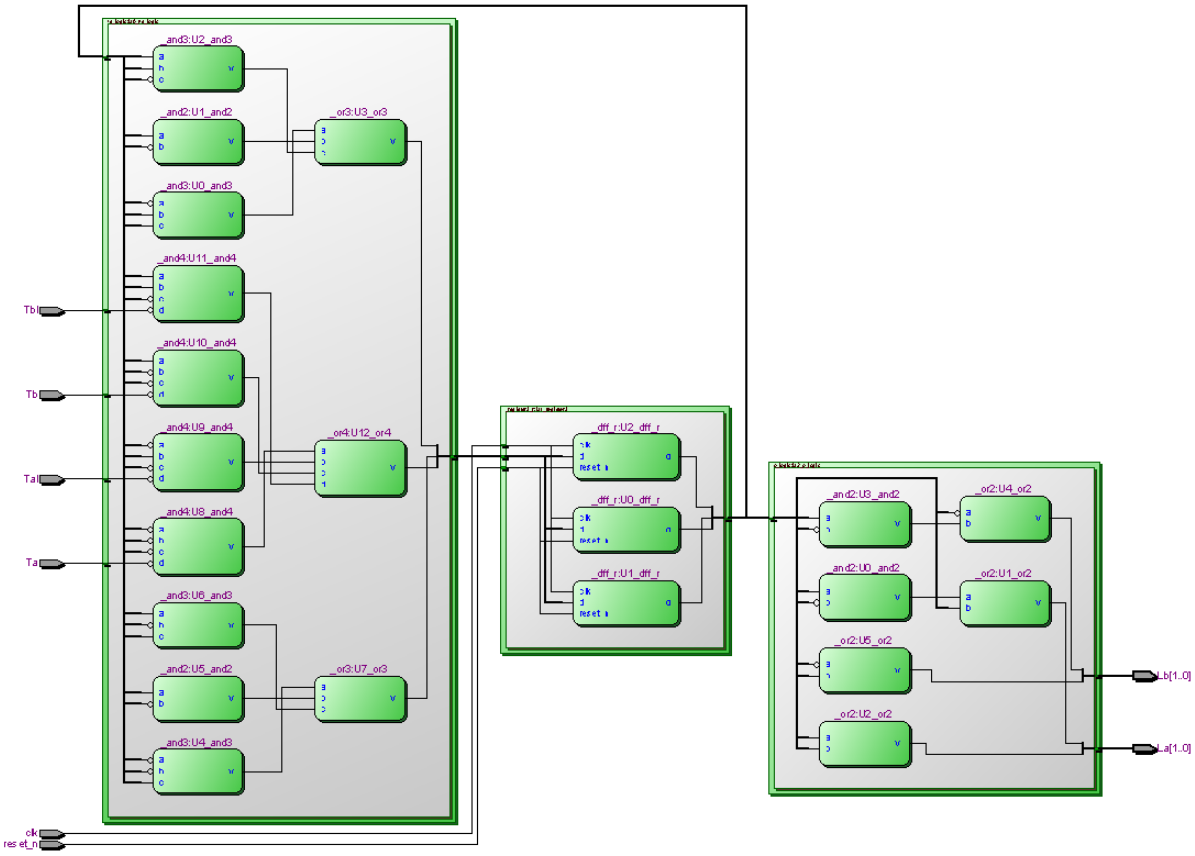
해당 RTL Viewer를 보게 되면 위에 게시된 FSM의 모습과 같은 것을 볼 수 있습니다. 가운데 있는 module이 register이며 clk이 rising edge가 되면 중간에서 값을 계속 이동시키는 역할을 합니다.

Flow summary

Flow Summary	
Flow Status	Successful - Sat Sep 29 00:31:57 2018
Quartus II 64-Bit Version	13.0.0 Build 156 04/24/2013 SJ Web Edition
Revision Name	tl_cntr
Top-level Entity Name	tl_cntr
Family	Cyclone II
Device	EP2C70F896C6
Timing Models	Final
Total logic elements	1 / 68,416 (< 1 %)
Total combinational functions	1 / 68,416 (< 1 %)
Dedicated logic registers	1 / 68,416 (< 1 %)
Total registers	1
Total pins	8 / 622 (1 %)
Total virtual pins	0
Total memory bits	0 / 1,152,000 (0 %)
Embedded Multiplier 9-bit elements	0 / 300 (0 %)
Total PLLs	0 / 4 (0 %)

2. tl_cntr_w_left

RTL Viewer



RTL Viewer를 비교하였을 때 다소 조금 더 복잡해 보이지만 left가 추가되었을 뿐 같은 작용을 하는 것을 알 수 있습니다.

Flow summary

Flow Summary	
Flow Status	Successful - Sat Sep 29 00:23:32 2018
Quartus II 64-Bit Version	13.0.0 Build 156 04/24/2013 SJ Web Edition
Revision Name	tl_cntr_w_left
Top-level Entity Name	tl_cntr_w_left
Family	Cyclone II
Device	EP2C70F896C6
Timing Models	Final
Total logic elements	9 / 68,416 (< 1 %)
Total combinational functions	9 / 68,416 (< 1 %)
Dedicated logic registers	3 / 68,416 (< 1 %)
Total registers	3
Total pins	10 / 622 (2 %)
Total virtual pins	0
Total memory bits	0 / 1,152,000 (0 %)
Embedded Multiplier 9-bit elements	0 / 300 (0 %)
Total PLLs	0 / 4 (0 %)

5. 고찰 및 결론

A. 고찰

해당 과제를 하는데 있어 연결하는 과정이 많고 복잡하여 시간이 많이 걸렸지만 1학기때 배운 내용으로 이론적으로 이해가 안되고 어려웠던 점은 없었습니다. 문제로는 두번째 cntr_left를 구현하고 테스트벤치로 결과를 확인하였을 때 같은 값인 데도 불구하고 clk의 edged에서 반응하여 경계가 생기는 문제가 발생하였습니다. 해당 top module의 기능은 정상적으로 작동을 하지만 분명 sub module을 구현하는 중 약간의 문제가 있을 것이라고 판단하여 각 module을 확인해보았습니다. 확인결과 o_logic을 구성하는데 있어 PDF에서 불식을 간단하게 풀어놓은 식으로 하드웨어를 구현한 것이 아니라 그 전 단계인 조금 더 복잡한 단계의 식을 보고 하드웨어를 구성하게 되었습니다. 이를 조금 더 간편한 식으로 바꾸어 Verilog를 변경하였고 결과를 보니 edge에서 반응을 하던 모습이 사라졌습니다. 예상되는 이유로는 게이트의 수가 다르기 때문에 결과값 자체는 변하지 않지만 wave form에서의 차이가 있었다고 생각됩니다.

B. 결론

이번 실습을 통하여 평소에 평범하게 보던 신호등의 동작을 훨씬 정확하게 알 수 있었습니다. 특히 1학기때 다루지 않았던 left신호가 추가됨으로써 개념을 좀더 잡을 수 있는 기회가 되었습니다. 또한 실습한 FSM을 사용하여 신호등을 설계하는 것뿐 아니라 반복적으로 다양한 일을 하는 기계에 응용이 가능하다고 생각됩니다. 예를 들면 특정 음식을 만드는 기계를 설계한다고 가정하였을 때 음식을 만드는데 일정한 단계가 있고 검수를 통하여 단계가 부족하면 일정한 명령(input)을 통하여 해당 과정을 반복할 수 있으며 기계 동작 중 긴급 상황이 있으면 비상정지(비동기식을 통한 reset)를 통하여 기계를 멈출 수 있게 설계할 수 있다고 생각됩니다. 특히 비동기식을 사용한 방법은 신호등에 적용하는 것보다 예시를 든 기계에 더 적합하다고 생각됩니다.

6. 참고문헌

FSM에 대한 개념 / https://en.wikipedia.org/wiki/Finite-state_machine

Moore, Mealy FSM / https://blog.naver.com/tjdowl123/221189914369

김영민 / Finite State Machine / 광운대학교 / 2018