컴퓨터 공학 기초 실험2 보고서

실험제목: Traffic Light Controlloer with/without Left Turn signals

실험일자: 2021년 10월 12일 (수)

제출일자: 2021년 10월 25일 (화)

학 과:컴퓨터정보공학부

담당교수: 공진흥 교수님

실습분반: 수요일 0, 1, 2

학 번: 2019202021

성 명: 정 성 엽

1. 제목 및 목적

A. 제목

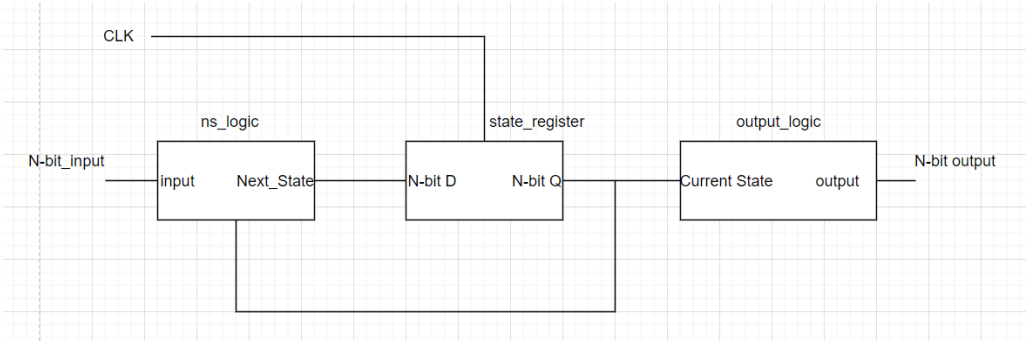
Traffic Light Controller with/without left Turn Signals

B. 목적

수업 시간에 배운 FSM의 원리를 이해하고 설계한다. 이때 Moore FSM을 적용하여 traffic light controller를 처음부터 직접 설계하고 이 설계에서 left Turn Signal을 들어왔을 때 좌회전 신호가 나타나는 FSM을 제작한다.

2. 원리(배경지식)

A. Finite State Machine(FSM)



Finite State Machine은 시스템의 상태 그리고 이동으로 나타낸다. FSM의 구조는 위의 그림과 같으며 input을 nextState를 register에 저장하는 ns_logic에 받고 상태값을 저장하는 register 그리고 최종 output 을 저장하는 output_logic으로 이루어져 있다. FSM 종류에는 Moore FSM과 MealyFSM이 있는데 이는 현재 상태에서 새로운 입력에 따라 달라진다면 Moore FSM 아니고 현재상태에 따라서 output이 달라진다면 MealyFSM이다. 이번 실험은 MooreFSM을 사용하여 Traffic Light Controller를 설계하고 구현한다.

B. Quine-mccluskey method

기존에 설계를 하기 위해서는 Karnaugh Map을 통해 논리식을 간소화 하는 방법을 사용하였지만, 방법은 더 복잡하지만 5개 이상의 변수를 가질 때 카르노맵보다 훨씬

효율적으로 작동시킬 수 있는 논리식을 찾기 수월하게 한다. 알고리즘화를 통해 프로그래밍이 가능하고 5개 이상의 변수를 가지고 있는 논리식을 간소화할 수 있다. 그리고 이번 실험에서는 next_state의 논리식을 간소화하여 설계할 때 사용하고 true minterms를 찾고 단계와 중복되는 것들은 거르고 변수들을 한데 묶어 최소한의 변수를 식별하고 prime implicant를 찾는 식별단계와 최소항을 찾는 선택 단계로 구분한다.

C. Traffic light controller

이번 실험은 신호등을 만드는 것으로 input Ta, Tb 는 각 도로의 차가 가야함과 아님을 나타낸다고 볼 수 있다. Ta가 set이 되었다면 Ta에 차가 가야함을 반대는 Tb 에 차가 가야함을 나타낸다. 이를 상상하여 FSM을 구성하여 diagram으로 쉽게 나타낼 수 있다. 만약 Ta, Tb 가 모두 set이 된다면 둘 다 이동하면 교차로에서 충돌하므로 이때는 이전 state를 유지할 수 있도록한다.

3. 설계 세부사항

A. Traffic Light Controller without left Turn Signal

1-1 Define state

신호 동작은 green->yellow->red->green 순으로 순환하므로 한쪽이 green이나 yellow이면 반대는 red이다.

S_1	S_0	State	La	Lb
0	0	S0	green	red
0	1	S1	yellow	red
1	0	S2	red	green
1	1	S3	red	yellow

1-2 Define inputs

각 상태 간의 연관 관계 그리고 다음 state에 대한 정보를 표로 나타낸다.

Current State	input		Next State
	Ta	Tb	
S0	0	x	S1
S0	1	x	S0
S1	x	x	S2
S2	x	0	S3

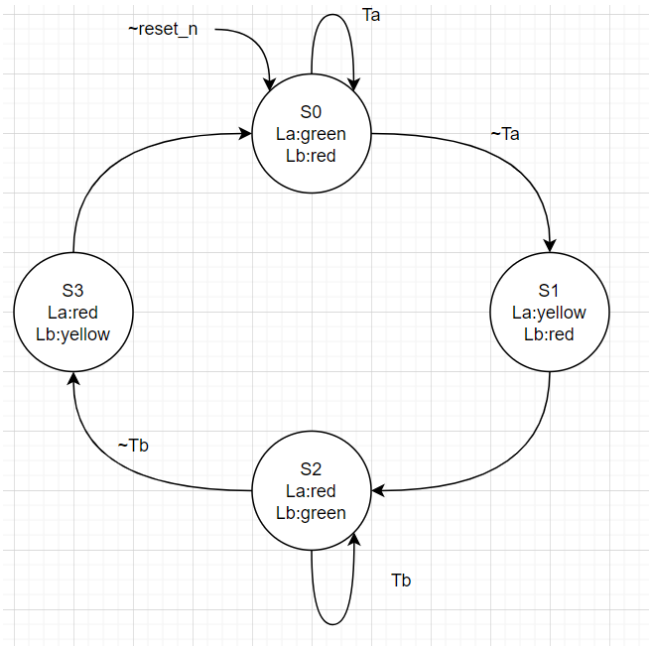
S2	x	1	S2
S3	x	x	S0

1-3 Define outputs

신호 색깔에 대한 출력을 정의한다.

Output	Encoding 2-bit La[1:0] or Lb[1:0]
Green	00
Yellow	01
Red	10

1-4 Draw the diagram



- Encoding states

1-5 Next state table

Moore FSM을 설계하기 위해 next state 에 관한 Boolean equation은 1-2의 표를 참고한다.

current state		inputs		next state	
Q1	Q0	Ta	Tb	D1	D2
0	0	0	x	0	1
0	0	1	x	0	0
0	1	x	x	1	0
1	0	x	0	1	1

1	0	x	1	1	0
1	1	x	x	0	0

D1은 Q1 또는 Q0가 set 되었을 때 set 되므로 $D1 = Q1 \oplus Q2$이고

D0는 앞서 설명한 quine mccluskey를 이용하면

$F(Q1, Q2, Ta, Tb) = F(a, b, c, d) = \Sigma m$으로 최소화할 수 있다.

a	b	c	d		decimal
0	0	0	0	0000	0
0	0	0	1	0001	1
1	0	1	0	1010	10
1	0	0	0	1000	8

	0000	0001	1000	1010
000	x	x		
-000	x		x	
10-0			x	x

$F = a'b'c + ab'd'$ 이므로

$D0 = \overline{Q1}\,\overline{Q0}\,\overline{Ta} + Q1\,Q0\,\overline{Tb}$

1-6 Output table

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

표에 따라 $La1 = Q1, La0 = \sim Q1\ Q0, Lb1 = \sim Q1, Lb2 = Q1\ Q0$로 나타낼 수 있다.

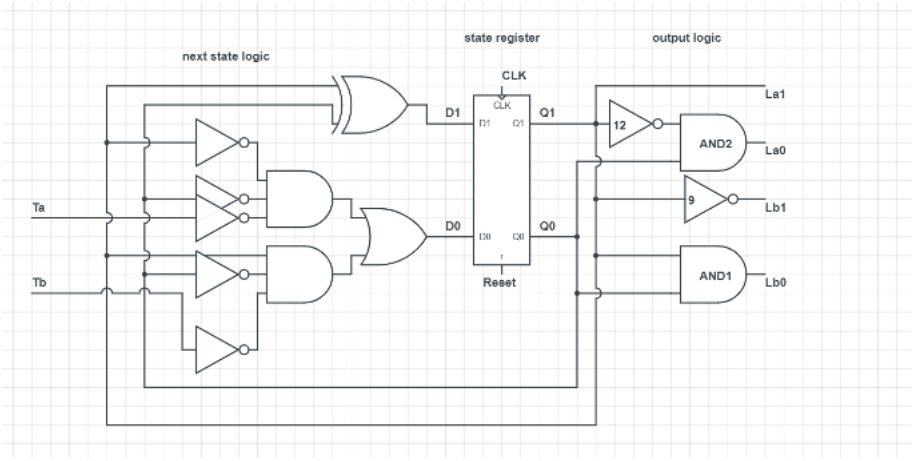
- Coding the module header

구분	이름	설명
Topmodule	tl_cntr	top module
Submodule	ns_logic	next state
	_register2_r	2-bit resettable register
	_dff_r	Resettable d flipflop
	o_logic	output logic (Moore Machine)

Top module	port	name	bandwidth	설명
tl_cntr	input	clk	1bit	clock
		reset_n	1 bit	active low reset
		Ta	1 bit	a Traffic
		Tb	1 bit	b Traffic

	output	La	2 bit	signal color
		Lb	2 bit	signal color

- Coding state registers(flipflops)-sequential circuits & Coding combinational circuit



B. Traffic Light Controller with left Turn Signal

1-1 Define state

기존 TLC에서 좌회전 신호를 추가하는 것으로 좌회전 신호가 없어도 초록불 뒤에는 좌회전 신호가 와야한다.

S_2	S_1	S_0	state	La	Lb
0	0	0	S0	green	red
0	0	1	S1	yellow	red
0	1	0	S2	left	red
0	1	1	S3	yellow	red
1	0	0	S4	red	green
1	0	1	S5	red	yellow
1	1	0	S6	red	left
1	1	1	S7	red	yellow

1-2 Define inputs

위 표를 참고하여 input 과 다음 상태를 표로 나타낸다

current State	input data				next state
	Ta	Tal	Tb	Tbl	
S0	0	X	X	X	S1
S0	1	X	X	X	S1
S1	X	X	X	X	S2

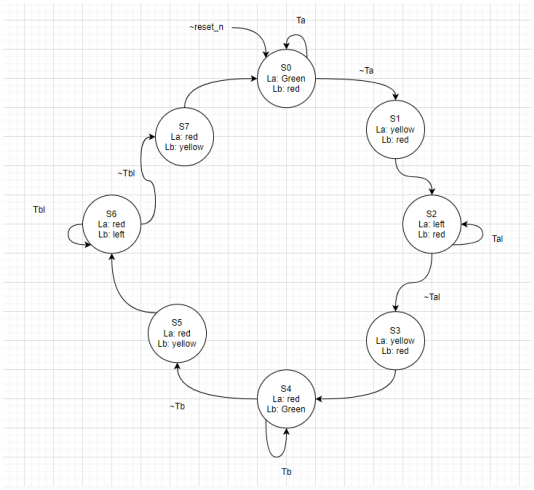
S2	X	0	X	X	S3
S2	X	1	X	X	S3
S3	X	X	X	X	S4
S4	X	X	0	X	S5
S4	X	X	1	X	SS4
S5	X	X	X	X	S6
S6	X	X	X	0	S7
S6	X	X	X	1	S6
S7	X	X	X	X	S1

1-3 Define outputs

남아있던 11에 LEFT 신호로 바꾼다.

OUTPUT	Encoding 2-bit La[1:0] Lb[1:0]
green	00
yellow	01
left	10
red	11

1-4 Draw the diagram



-Encoding state

1-5 State transition table

current state			inputs				next state		
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	1
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	x1	1	1	0
1	1	1	x	x	x	x	0	0	0

이번 경우에는 변수가 7개 이므로 프로그램을 사용해 구하였다. 간단히 표현하면 F(Q2,Q1,Q0,Ta,Tal,Tb,Tbl)=F(a,b,c,d,e,f,g)=Σm 이고 표가 너무 길어 모두 생략하였다.

D0를 최소화하면 F=a'b'c'd'+a'bc'e'+ab'c'f'+abc'g'으로 표현할 수 있고

따라서 D0= $\overline{Q2}\,\overline{Q1}\,\overline{Q0}\,\overline{Ta} + \overline{Q2}\,Q1\,\overline{Q0}\,\overline{Tal} + Q2\,\overline{Q1}\,\overline{Q0}\,\overline{Tb} + Q2\,Q1\,\overline{Q0}\,\overline{Tbl}$이다.

D1을 최소화하면 F=b'c+bc'이고

$D1 = \overline{Q1}Q0 + Q1\overline{Q0}$이다.

D2를 최소화하면 F=a'bc+ab'+abc'이고

$D2 = \overline{Q2}\,Q1\,Q0 + Q2\overline{Q1} + Q2Q1\overline{Q0}$이다.

1-6 output table

Q2	Q1	Q0	La1	La0	Lb1	Lb0
0	0	0	1	1	0	1
0	0	1	0	0	1	1
0	1	0	0	1	1	1
0	1	1	1	0	1	1
1	0	0	0	1	1	1
1	0	1	1	1	0	0
1	1	0	1	1	0	1
1	1	1	1	1	1	0

이를 카르노 맵을 통해 구하면

$$La1 = Q1\overline{Q0} + Q2$$
$$La0 = Q0 + Q2$$
$$Lb1 = Q1\,\overline{Q0} + \overline{Q2}$$
$$Lb0 = Q0 + \overline{Q2}$$

이다.

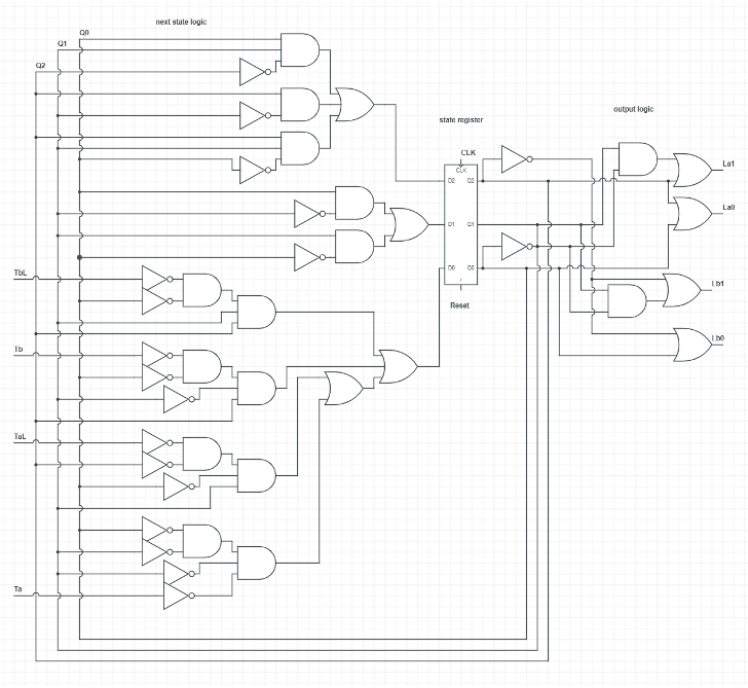
- Coding the module header

구분	이름	설명
Top module	tl_cntr_w_left	top module
Sub module	ns_logic	next state 결정

	_rigister3_r	3bit resettable register
	_dff_r	resettable D flipflop
	o_logic	output logic(moore machine)

top module	port	name	bandwidth	description
tl_cntr	input	clk	1bit	clock
		reset_n	1bit	active low reset
		Ta	1bit	a Traffic
		Tal	1bit	a left turn traffic
		Tb	1bit	b Traffic
		Tbl	1bit	b left turn traffic
	output	La	2bit	sinal color
		Lb	2bit	signal color

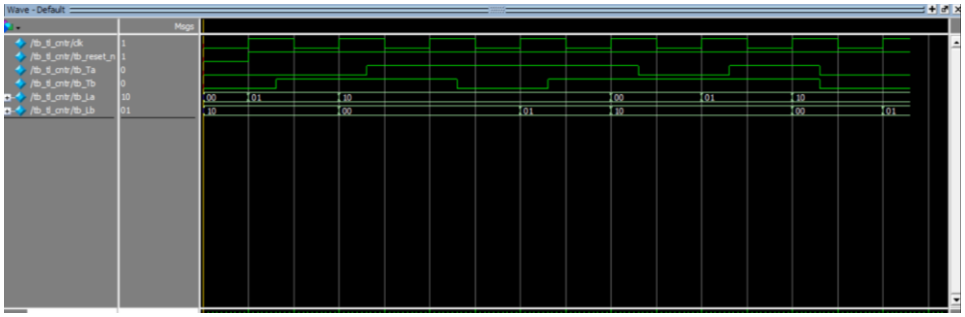
- Coding state register(flipflops)-sequential circuit & Coding combinational circuits



4. 설계 검증 및 실험 결과

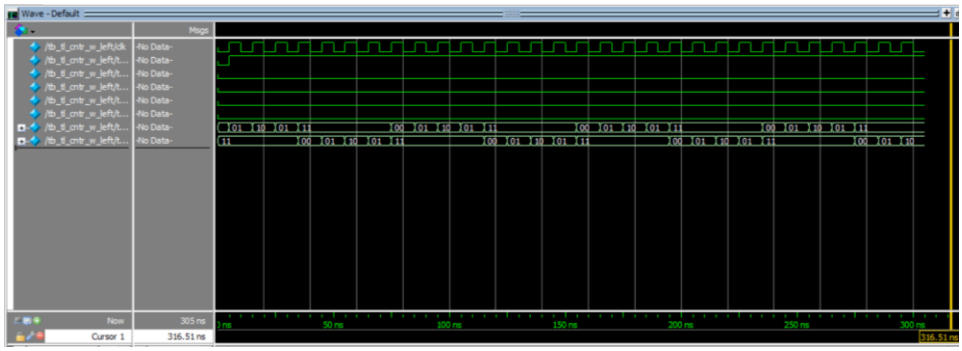
A. 시뮬레이션 결과

i. Traffic light controller



Ta, Tb의 입력에 따라 다이어그램의 state로 움직이고 각 signal color가 변화하는 것을 waveform을 통해 확인할 수 있다.

ii. Traffic light controller with left turn signals

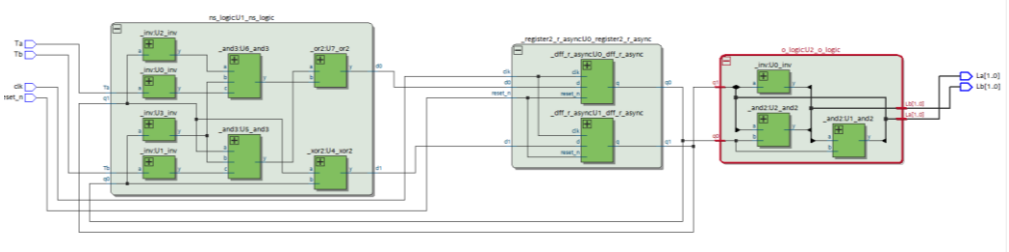


Ta, Tb가 모두 0일 때 설계한 FSM 에 따라 green -> yellow->left->red->green 을 반복하여 순환한다.

B. 합성(synthesis) 결과

i. Traffic light controller

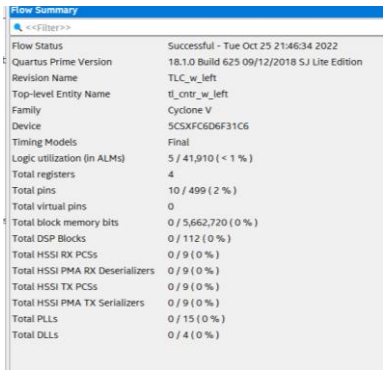
Flow Summary	
Flow Status	Successful - Tue Oct 25 21:42:16 2022
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	tlc
Top-level Entity Name	tl_cntr
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	3 / 41,910 (< 1 %)
Total registers	3
Total pins	8 / 499 (2 %)
Total virtual pins	0
Total block memory bits	0 / 5,662,720 (0 %)
Total DSP Blocks	0 / 112 (0 %)
Total HSSI RX PCSs	0 / 9 (0 %)
Total HSSI PMA RX Deserializers	0 / 9 (0 %)
Total HSSI TX PCSs	0 / 9 (0 %)
Total HSSI PMA TX Serializers	0 / 9 (0 %)
Total PLLs	0 / 15 (0 %)
Total DLLs	0 / 4 (0 %)

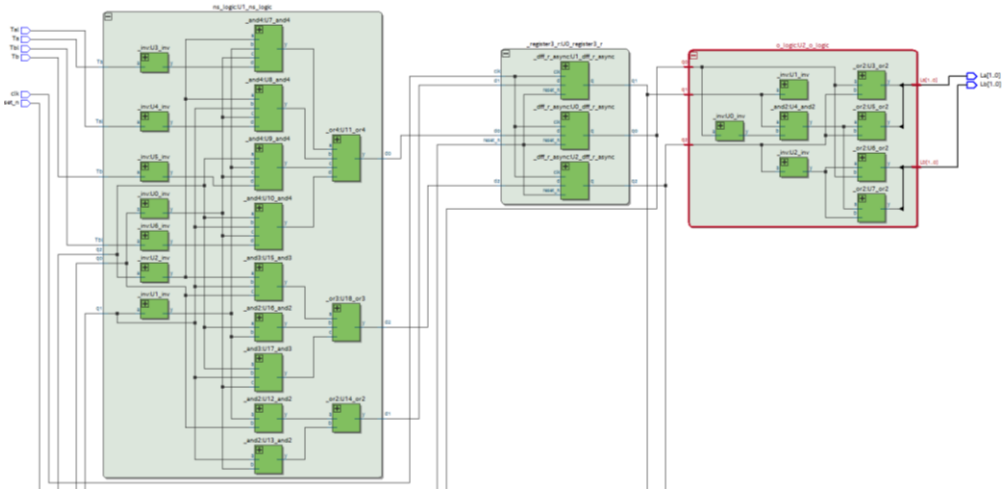


Moore Machine이 ns_logic 단, register 단, o_logic 단으로 구성됨을 확인할 수

있다.

ii. Traffic light controller with left turn signals





Moore Machine이 ns_logic 단, register 단, o_logic 단으로 구성됨을 확인할 수 있다. 이전과 다르게 input이 2개 더 추가되었고 register단 이 2bit가 아닌 3bit를 다루고 있다.

5. 고찰 및 결론

A. 고찰

이번 실험은 Traffic light controller와 이것에 left turn 신호를 추가하여 Verilog로 FSM을 설계하고 제작하는 것이었다. 이전 컴퓨터공학기초실험1에서 만들었던 FSM을 Verilog로 구현하는 거은 상당히 어려웠다.

Verilog를 통해 FSM을 구현할 때 ns_logic 단, register 단, o_logic 단으로 나누어서 구현하고 이를 top module인 tl_cntr에 instrance 하여 FSM의 전반적 구조를 module 단위로 쪼개 구현하였고 FSM을 Verilog로 구현하는 것은 오래 안 걸렸지만 이를 설계하는 것은 input과 output을 따져 카르노맵 또는 quine mccluskey 방법으로 만드는 것은 상당히 복잡하였고 그래서 특히 7개 변수인 경우 프로그램을 활용하여 논리식을 찾았다.

B. 결론

지금까지는 정해진 논리식에 따라 Verilog를 구현하는 것을 익혔다면 이번엔 FSM 설계를 위해 각 state, input , 등을 확인하고 logic을 구하여 논리식으로 먼저 설계하는 것을 배웠고 특히 FSM에서는 Diagram을 그려 전반적인 구조를 파악하는 것이 많은 도움이 되었다. 이를 통해 필요한 register의 개수를 정하고 Next state logic과 output logic을 구현하여 verilog를 완성한다.

이전에 브레드보드를 통해 만들어본 Traffic light controller라서 left turn 없이는 금방 알 수 있었지만 변수 하나만 더 따져도 다른 곳에서 따져서 가져와야함을 이번 실험을 통해 알 수 있었다.

6. 참고문헌

공영호 교수님/디지털논리회로2/ 광운대학교 컴퓨터정보공학부/2022

공진흥 교수님/컴퓨터공학기초실험2/광운대학교 컴퓨터정보공학부/2022