컴퓨터 공학 기초 실험2 보고서

실험제목: Traffic Light Controller

실험일자: 2020년 10월 9일 (금)

제출일자: 2020년 10월 18일 (일)

학 과: 컴퓨터공학과

담당교수: 이준환 교수님

실습분반: 금요일 5,6,7

학 번: 2019202009

성 명: 서여지

1. 제목 및 목적

A. 제목

Traffic Light Controller with/without Left Turn Signals

B. 목적

4거리의 Traffic Light Controller를 finite state machine으로 설계하고 Verilog로 작성한다. FSM의 의미와 구조를 이해하고 직접 설계할 수 있다. 설계한 FSM에 우회전 신호를 추가하며 몇 가지 state를 추가하고 새로운 FSM을 작성할 수 있다. quine-mccluskey 방법을 이용하여 boolean equation을 정리할 수 있다.

2. 원리(배경지식)

1. Finite state machine

Finite state machine은 한정된 개수의 state를 갖는 회로이다. FSM은 현재 회로의 state를 저장하는 register와 회로의 다음 상태를 계산하는 next state logic, 회로의 출력을 계산하는 output logic의 세 부분으로 나눌 수 있다. 회로의 next state logic에서 input과 current state에 따라 next state를 계산하며, 결국 회로는 몇 가지 state를 clock주기에 맞추어 반복하게된다. 각 state는 일정한 output을 가지고 있어 output logic에서 계산된 output또한 clock에 맞추어 변화한다. FSM은 output logic에 FSM의 input신호가 사용되는지 여부에 따라 Mealy machine과 Moore machine으로 나뉜다. 이번 실험에서 설계한 FSM은 모두 output logic에 FSM의 input신호가 사용되지 않는 Moore machine이다.

2. Qunie-McCluskey Minimization

K-Map을 이용한 방법은 input이 여러 개가 있는 경우 사용하기 어렵다는 한계가 있다. Qunie-McCluskey Minimization은 확장성이 좋은 대안이며, logic function minimization을 자동화할 수 있는 방법이다. Qunie-McCluskey 방법은 크게 두 단계로 나뉘어, implicant table을 구하는 과정과 PI table을 구하는 단계로 나눌 수 있다.

1. implicand table
   - A. 첫 번째 행에 true minterm과 don't care를 모두 나열한다. 이때 나열은 1의 개수에 따라 group으로 나누어 나열한다.
   - B. 이웃한 두 group에서 Hamming distance가 1인 것을 찾아 combining하고, 다음 행에 서로 다른 bit를 -로 대체하여 표기한다.
   - C. Hamming distance가 1인 것을 찾은 minterm과 찾지 못한 것을 따로 표기한다.

D. 모든 group의 minterm에 대해 과정을 반복하고, 더 이상 Hamming distance가 1인 것을 찾을 수 없을 때까지 combining을 반복한다.

2. PI table – minimum set

A. true minterm으로 column을 구성하고, prime impliment로 row를 구성한다.

B. PI가 포함하는 minterm에 표시한다.

C. 표시가 하나만 존재하는 column의 PI는 필수로 표시하고, 필수 PI에 포함되는 minturem을 모두 삭제한다.

D. 남은 column에 대해 모든 minterm을 포함하는 PI의 집합을 구성한다.

3. 설계 세부사항

1) Traffic Light Controller

1. 설계

A. Define states

이번 실습에서 구현하려는 신호등은 초록, 노랑, 빨강의 3가지 불빛을 가지고, 한 신호등의 불이 초록이나 노랑인 경우 다른 신호등은 반드시 빨간색이다. 따라서 총 4개의 state를 생성할 수 있다. 2bit를 이용하여 다음과 같이 encoding 하면 4개의 state를 모두 표현할 수 있다.

State	L_A	L_B	Q_1Q_0
S0	초록	빨강	0 0
S1	노랑	빨강	0 1
S2	빨강	초록	1 0
S3	빨강	노랑	1 1

B. Define inputs

문제에서 주어진 input은 Academic Ave의 sensor T_A와 Bravado Blvd의 sensor T_B이다. 두 input 모두 1 또는 0의 1bit 신호를 입력한다. 또한 Sequential circuit인 FSM이 동작하는 CLK와 reset signal이 input으로 필요하다.

C. Define output

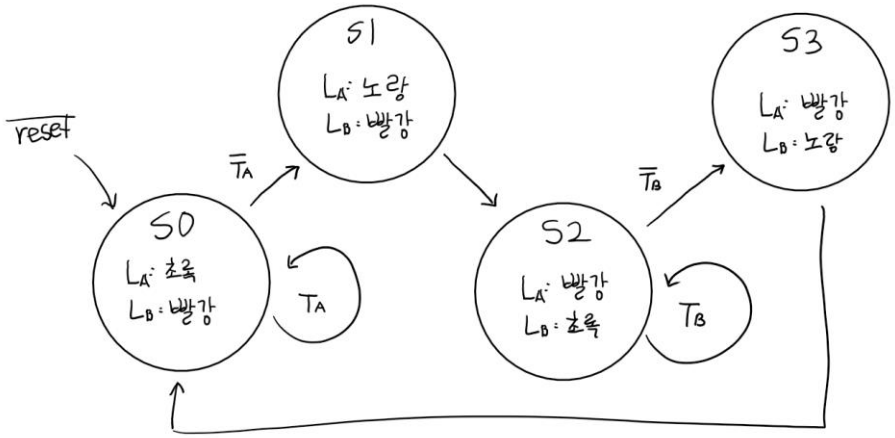
신호등의 output은 Academic Ave와 Bravado Blvd의 traffic light L_A와 L_B로

나타난다. 각각의 신호등은 초록, 노랑, 빨강을 나타낼 수 있으므로 3가지 이상을 표현할 수 있는 2bit를 이용하여 다음과 같이 encoding한다.

신호등 색	L_1L_0
초록	0 0
노랑	0 1
빨강	1 0

D. Draw the diagram

각 상태에서 새로 입력된 T_A, T_B값에 따라 다음과 같이 상태를 변화시킨다.



state transition table

Current state	Q_1	Q_0	T_A	T_B	Next state	D_1	D_0
S0	0	0	0	X	S1	0	1
S0	0	0	1	X	S0	0	0
S1	0	1	X	X	S2	1	0
S2	1	0	X	0	S3	1	1
S2	1	0	X	1	S2	1	0
S3	1	1	x	x	S0	0	0

$$D_1 = Q_1 \oplus Q_0$$
$$D_0 = \overline{Q_1 Q_0 T_A} + Q_1 \overline{Q_0 T_B}$$

output table

State	Q_1	Q_0	L_A	L_{A1}	L_{A0}	L_B	L_{B1}	L_{B0}
S0	0	0	초록	0	0	빨강	1	0
S1	0	1	노랑	0	1	빨강	1	0
S2	1	0	빨강	1	0	초록	0	0
S3	1	1	빨강	1	0	노랑	0	1

$$L_{A1} = Q_1$$
$$L_{A0} = \overline{Q_1}Q_0$$
$$L_{B1} = \overline{Q_1}$$
$$L_{B0} = Q_1Q_0$$

2. 작성한 모듈

A. tl_cntr

register와 next state logic, output logic을 모두 포함하는 top module이다. 각각의 역할을 하는 sub module을 instance하여 작성하였다. clk, reset_n, Ta, Tb를 input으로 받고, La, Lb를 출력한다.

B. ns_logc

current state와 sensor의 정보를 입력으로 받아 next state를 구하는 module이다. 앞에서 구한 Boolean equation을 바탕으로 gate module을 instance하여 구현하였다.

C. _register2_r_async

2 bit 정보를 저장할 수 있는 register이다. reset 기능은 asynchronous로 동작한다. clk, r과 함께 state를 입력으로 받아 저장한다. 2개의 _dff_r_async module을 instance하여 구현하였다.

D. _dff_r_async

1bit 정보를 저장할 수 있는 D flip-flop이다. reset 기능은 clk에 동기화되어있지 않다. D Flip-flop 실습에서 behavior implement한 _dff_rs_async module을 변형하여 사용하였다.

E. o_logic

입력받은 state에 따른 la, lb출력을 결정하는 output logic이다. 설계과정에서 작성한 Boolean equation을 assign문과 gate module을 이용하여 구현하였다.

2) Traffic Light Controller with Left Turn Signals

1. 설계

A. Define states

앞의 실습에서 구현한 신호등에 좌회전 신호가 더해졌으므로 새로운 state가 추가되어야 한다. 좌회전 신호의 경우에도 한 신호등의 불이 좌회전 인 경우 다른 신호등은 반드시 빨간색이어야 한다. 따라서 좌회전 신호와 노란불인 경우가 하나씩 추가된 것과 같으므로, 총 8개의 state를 생성할 수 있

다. 8가지 신호는 3bit로 encoding하여 나타낼 수 있다.

State	L_A	L_B	$Q_2Q_1Q_0$
S0	초록	빨강	0 0 0
S1	노랑	빨강	0 0 1
S2	좌회전	빨강	0 1 0
S3	노랑	빨강	0 1 1
S4	빨강	초록	1 0 0
S5	빨강	노랑	1 0 1
S6	빨강	좌회전	1 1 0
S7	빨강	노랑	1 1 1

B. Define inputs

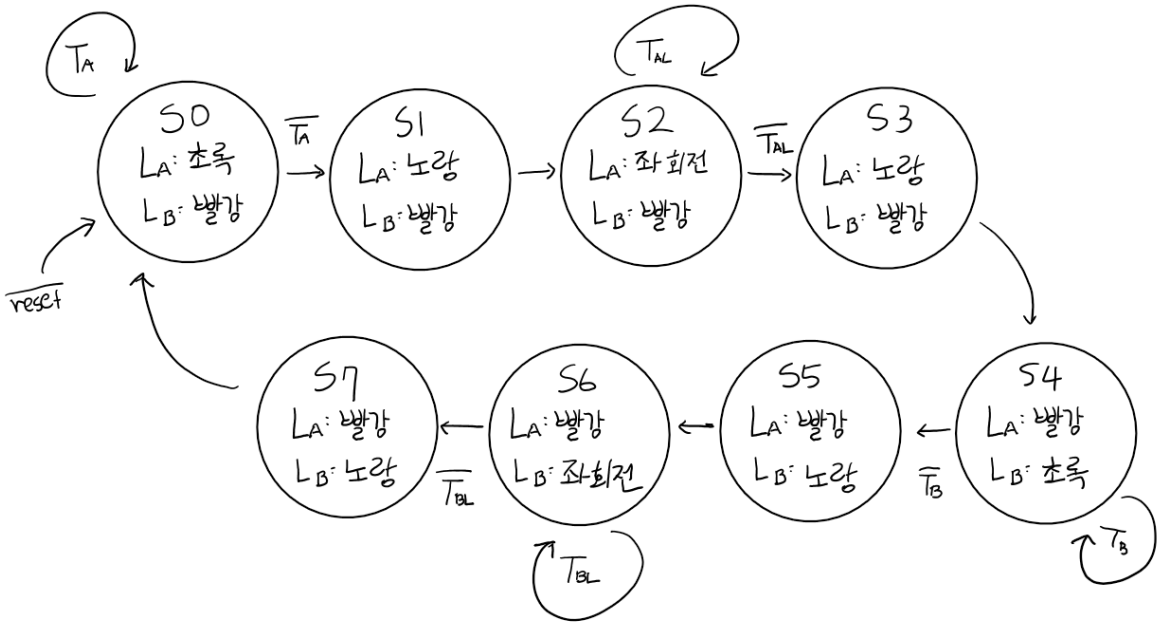
문제에서 주어진 input은 기존의 T_A, T_B와 더불어 Academic Ave의 좌회전 차량인 T_{AL}과 Bravado Blvd의 좌회전 차량인 T_{BL}이 추가되었다. 네 개의 input 모두 1 또는 0의 1bit 신호를 입력한다. 앞의 실험에서와 마찬가지로 FSM이 동작하는 CLK와 reset signal이 input으로 필요하다.

C. Define output

신호등의 출력은 앞의 초록, 노랑, 빨강 출력 외에 좌회전이 추가되어 모두 4가지이다. FSM의 output은 앞의 실험에서와 동일하게 2bit로 나타낼 수 있다.

신호등 색	L_1L_0
초록	0 0
노랑	0 1
좌회전	1 0
빨강	1 1

D. Draw the diagram



state transition table

Current state	Q_2	Q_1	Q_0	T_A	T_{AL}	T_B	T_{BL}	Next state	D_2	D_1	D_0
S0	0	0	0	0	X	X	X	S1	0	0	1
S0	0	0	0	1	X	X	X	S0	0	0	0
S1	0	0	1	X	X	X	X	S2	0	1	0
S2	0	1	0	X	0	X	X	S3	0	1	1
S2	0	1	0	X	1	X	X	S2	0	1	0
S3	0	1	1	X	X	X	X	S4	1	0	0
S4	1	0	0	X	X	0	X	S5	1	0	1
S4	1	0	0	X	X	1	X	S4	1	0	0
S5	1	0	1	X	X	X	X	S6	1	1	0
S6	1	1	0	X	X	X	0	S7	1	1	1
S6	1	1	0	X	X	X	1	S6	1	1	0
S7	1	1	1	X	X	X	X	S0	0	0	0

7개의 input에 대한 Boolean equation을 k-map을 이용하여 구하였다.

	D2	TA TAL TB TBL															
Q2 Q1 Q0		0000	0001	0011	0010	0110	0111	0101	0100	1100	1101	1111	1110	1010	1011	1001	1000
	000	0	0	0	0	0	0	0	0	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	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	010	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	110	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	111	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	101	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	100	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	D1	TA TAL TB TBL	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
Q2 Q1 Q0		0000	0001	0011	0010	0110	0111	0101	0100	1100	1101	1111	1110	1010	1011	1001	1000
	000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	001	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	011	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	010	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	110	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	111	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	101	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	D0	TA TAL TB TBL	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
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

$$D_2 = \overline{Q_2}Q_1Q_0 + Q_2Q_1\overline{Q_0} + Q_2\overline{Q_1}$$
$$D_1 = \overline{Q_1}Q_0 + Q_1\overline{Q_0}$$
$$D_0 = \overline{Q_2Q_1Q_0T_A} + \overline{Q_2}Q_1\overline{Q_0T_{AL}} + Q_2Q_1\overline{Q_0T_{BL}} + Q_2\overline{Q_1Q_0T_B}$$

output table

State	Q_2	Q_1	Q_0	L_A	L_{A1}	L_{A0}	L_B	L_{B1}	L_{B0}
S0	0	0	0	초록	0	0	빨강	1	1
S1	0	0	1	노랑	0	1	빨강	1	1
S2	0	1	0	좌회전	1	0	빨강	1	1
S3	0	1	1	노랑	0	1	빨강	1	1
S4	1	0	0	빨강	1	1	초록	0	0
S5	1	0	1	빨강	1	1	노랑	0	1
S6	1	1	0	빨강	1	1	좌회전	1	0
S7	1	1	1	빨강	1	1	노랑	0	1

L_{A1}

Q_2 \ Q_1Q_0	00	01	11	10
0	0	0	0	1
1	1	1	1	1

L_{A0}

Q_2 \ Q_1Q_0	00	01	11	10
0	0	1	1	0
1	1	1	1	1

$$L_{A1} = Q_2 + Q_1\overline{Q_0}$$
$$L_{A0} = Q_2 + Q_0$$

L_{B1}

Q_2 \ Q_1Q_0	00	01	11	10
0	1	1	1	1
1	0	0	0	1

L_{B0}

Q_2 \ Q_1Q_0	00	01	11	10
0	1	1	1	1
1	0	1	1	0

$$L_{B1} = \overline{Q_2} + Q_1\overline{Q_0}$$
$$L_{B0} = \overline{Q_2} + Q_0$$

2. 작성한 모듈

A. tl_cntr_w_left

register와 next state logic, output logic을 포함하는 top module이다. 해당하는 각 모듈을 instance하여 작성하였다. clk, reset_n, Ta, Tal, Tb, Tbl 입력을 받고, La, Lb 를 출력한다.

B. ns_logc

current state 와 Ta, Tal, Tb, Tbl을 이용하여 next state를 계산하는 moduel이다. 설계과정에서 구한 Boolean equation을 gate module을 instance하여 구현하였다.

C. _register3_r

3bit의 정보를 저장할 수 있는 register이다. tl_cntr에서 이용한 _register2_r_async module에 _dff_r_ module을 하나 더 추가하여 구현하였다.

D. _dff_r_

tl_cntr을 구현할 때 이용한 module과 동일하다.

E. o_logic

state를 이용하여 La와 Lb를 구하는 module이다. 설계과정에서 구한 Boolean equation을 이용하여 작성하였다.

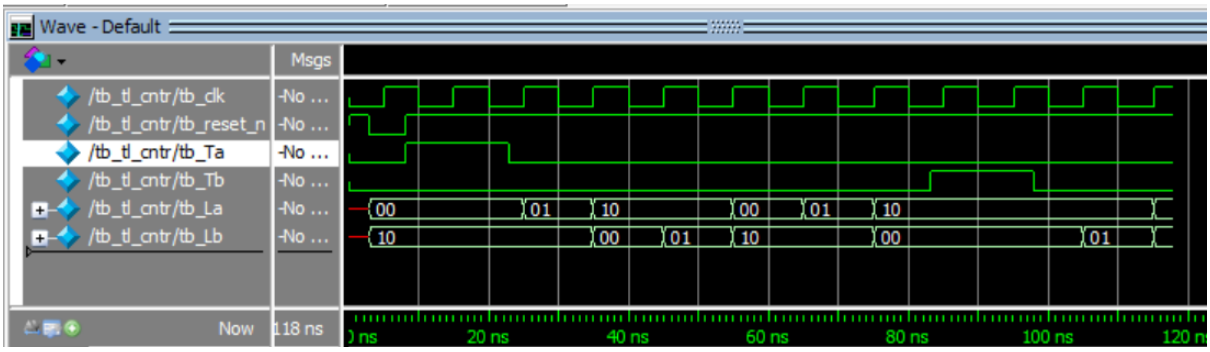
4. 설계 검증 및 실험 결과

A. 시뮬레이션 결과

1) tl_cntr

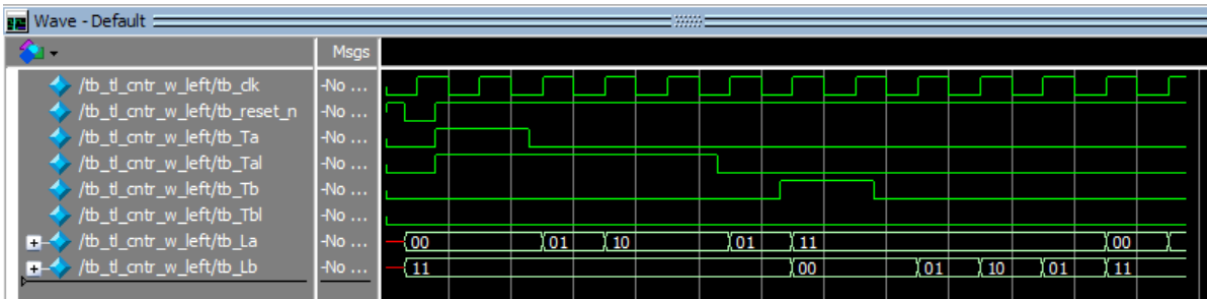
5ns 주기로 변화하는 clk를 이용하여 testbench를 작성하였다. reset은 active

low이므로 0일 때 동작한 것을 확인할 수 있다. clk의 주기에 따라 state가 s0부터 차례대로 순환하며 La, Lb가 적절히 출력되었다. Ta혹은 Tb신호가 있을 때 s0과 s2 state가 반복되며 여러 주기동안 나타났다.



2) tl_cntr_w_left

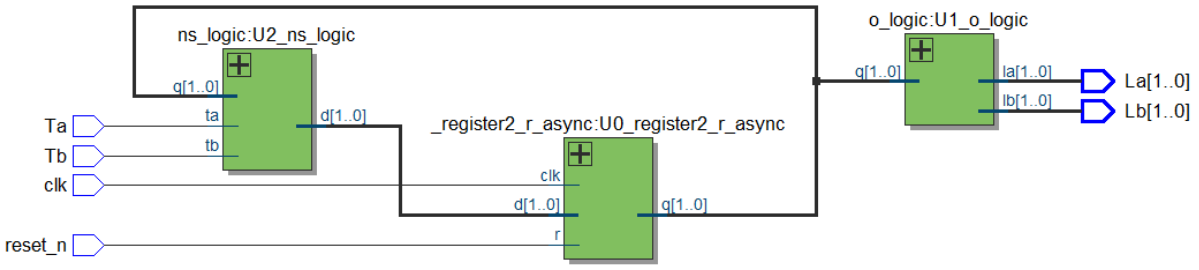
testbench의 testbench는 tl_cntr의 testbench에 Tal, Tbl을 추가하고, 내용을 변형하여 작성하였다. reset이 작동하여 S0으로 초기화 되었고, sensor의 신호가 존재하는 경우 해당 state가 반복되어 여러 주기동안 나타난 것을 확인할 수 있다. sensor의 신호가 0인 경우 clk의 주기마다 state가 정해진 순서대로 변한다.



B. 합성(synthesis) 결과

1. tl_ctnr

tl_ctnr의 RTL viewer는 다음과 같이 나타났다. next state logic에 input과 current state가, output logic에는 current state만 이용된 Moore machine의 형태로 나타났다.



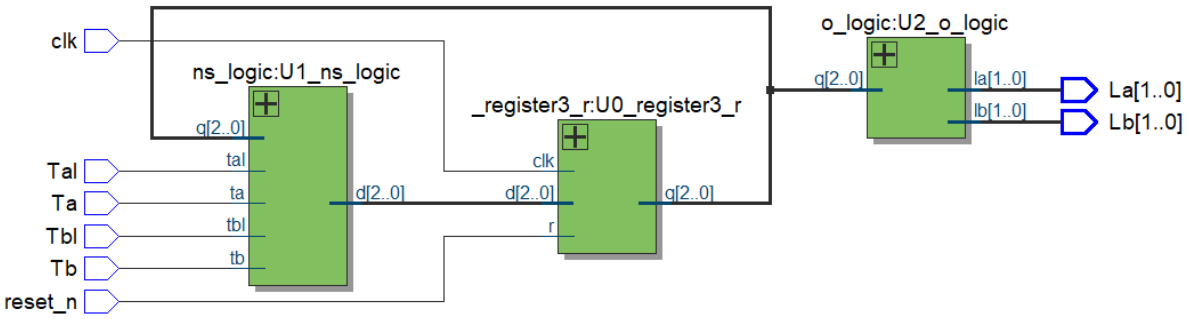
flow summary는 다음과 같다. ta, tb, clk, reset_n입력에 각각 하나의 pin을 이용하고, La, Lb 출력에 각각 2개의 pin을 이용하여 총 8개의 pin이 사용된

것을 확인할 수 있다.

Flow Summary	
Flow Status	Successful - Sat Oct 17 19:50:38 2020
Quartus Prime Version	15.1.0 Build 185 10/21/2015 SJ Lite Edition
Revision Name	tl_cntr
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 %)

2. tl_cntr_w_left

RTL viewer는 tb_cntr과 동일하게 Moore machine의 형태로 나타났다. Tal, Tbl 신호가 추가된 것을 회로에서 확인할 수 있다.



flow summary는 다음과 같다. Ta, Tal, Tb, Tbl, clk, reset_n입력에 각각 하나의 pin이 사용되었고, La, Lb 출력이 총 4개의 pin을 차지하여 총 10개의 pin이 사용된 것을 확인할 수 있다.

Flow Summary	
Flow Status	Successful - Sun Oct 18 01:12:51 2020
Quartus Prime Version	15.1.0 Build 185 10/21/2015 SJ Lite Edition
Revision Name	tl_cntr_w_left
Top-level Entity Name	tl_cntr_w_left
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	5 / 41,910 (< 1 %)
Total registers	4
Total pins	10 / 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 %)

5. 고찰 및 결론

A. 고찰

Qunie-McCluskey 방법을 이용하여 Boolean equation을 구하는 과정은 자동화에 특화되어 있고, 직접 구하는 것에는 적합하지 않아서 이번 실험에서는 사용하지 못했다. 7개의 input을 k-map을 이용하여 D0의 식을 구하는 과정에서 equation을 잘못 구하여 module 제작에 어려움을 겪었다. D1의 경우 설계에서 사용한 형태의 k-map으로는 최적화를 시킬 수 없어서 따로 계산하는 부분이 필요했다.

B. 결론

1학기 디지털논리회로시간에 중요하게 배운 FSM을 Verilog를 이용하여 직접 설계하는 실험이었다. 전체적인 회로 설계에 어려움은 없었으나, 위에서 언급한 Qunie-McCluskey 방법을 익히고, 시도하는 것에 오랜 시간이 걸렸다. 파이썬과 같은 프로그래밍 언어를 이용하여 문제를 해결하는 프로그램을 작성하는 것도 좋을 것 같다.

6. 참고문헌

이준환교수님, 디지털논리회로2 강의자료, 광운대학교 컴퓨터정보공학과,2020

이준환교수님, 컴퓨터공학기초실험2 강의자료, 광운대학교 컴퓨터정보공학과,2020

David Money Harris 외1인, Digital Design and Computer Architecture, Elesvier