컴퓨터 공학 기초 실험2 보고서

실험제목: Traffic Light Controller with/without Left Turn Signals

실험일자: 2023년 10월 11일 (수)

제출일자: 2023년 10월 15일 (일)

학 과: 컴퓨터정보공학부

담당교수: 이혁준 교수님

실습분반: 수요일 0, 1, 2

학 번: 2022202075

성 명: 우나륜

1. 제목 및 목적

A. 제목

Traffic Light Controller with/without Left Turn Signals

B. 목적

Traffic Light Controller with/without Left Turn Signal을 Verilog로 구현하고 그 과정에서 finite state diagram과 encoding states를 설계하고 Finite State Machine을 이해하는 데 목적을 둔다.

2. 원리(배경지식)

1. FSM (Finite State Machine)

   Finite State Machine란 유한한 개수의 상태를 가질 수 있는 기계를 설계하는 데 사용되는 수학적 모델을 말한다. FSM에는 동작 방식에 따라 Moore machine과 Mealy machine로 구분되어진다.

2. Moore FSM

   Moore FSM이란 output의 값들이 현재 상태에 의해 결정되는 회로를 말한다. Moore Machine의 설계는 현재 상태에만 영향을 받아 결과값이 출력되므로 간단한 설계 방식이다.

3. Mealy FSM

   Mealy FSM은 output의 값들이 현재 상태와 입력에 의해 결정되는 회로를 말한다. Mealy Machine은 현재 상태와 입력되는 input들의 상태에 의해 output의 값들이 결정되기 때문에 직관적이지 않아 이해하기 어렵다는 단점이 있다. 하지만 Moore FSM보다 더 적은 state를 가지고 설계할 수 있다는 장점을 가진다.

4. FSM Design procedure

   다음은 FSM을 설계할 때 거쳐야 할 과정이다.

   1) Drawing the Finite State Diagram

      Finite State Diagram을 그리는 과정에서는 현재 문제에서 주어진 상태 (state)와 입력 (input), 출력 (output)을 결정하고 diagram을 설계한다.

   2) Encoding states

      Encoding state는 상태를 encoding하여 나타내는 방식이다. 이 방식에는 binary encoding과 one-hot encoding 두 가지 방법이 존재한다.

      2-1) Binary encoding

      Binary encoding은 2진수를 사용하며 하나의 비트가 2개의 상태를 나타낼 수

있다. 만약 7가지의 상태를 나타내야 한다면, 최소 4개의 비트를 사용해야 한다.

2-2) One-hot encoding

하나의 비트가 하나의 상태를 나타낸다. 따라서 위의 binary encoding보다 더 많은 수의 flip-flop을 요구한다. 만약 7가지의 상태를 나타내야 한다면, 7개의 비트가 필요하고 7개의 flip-flop을 사용한다. 하지만 상태를 변화시킬 때, 현재 상태에서 다음 상태로 변화시킬 두 개의 flip-flop만 변화하면 되기 때문에 상태 변화에 대한 비용이 적게 든다.

3) Coding the module header

4) Coding sequential circuits - state registers (Flip-Flops)

5) Coding combinational circuits

3. 설계 세부사항

1. Traffic Light Controller

1) Finite State Machine

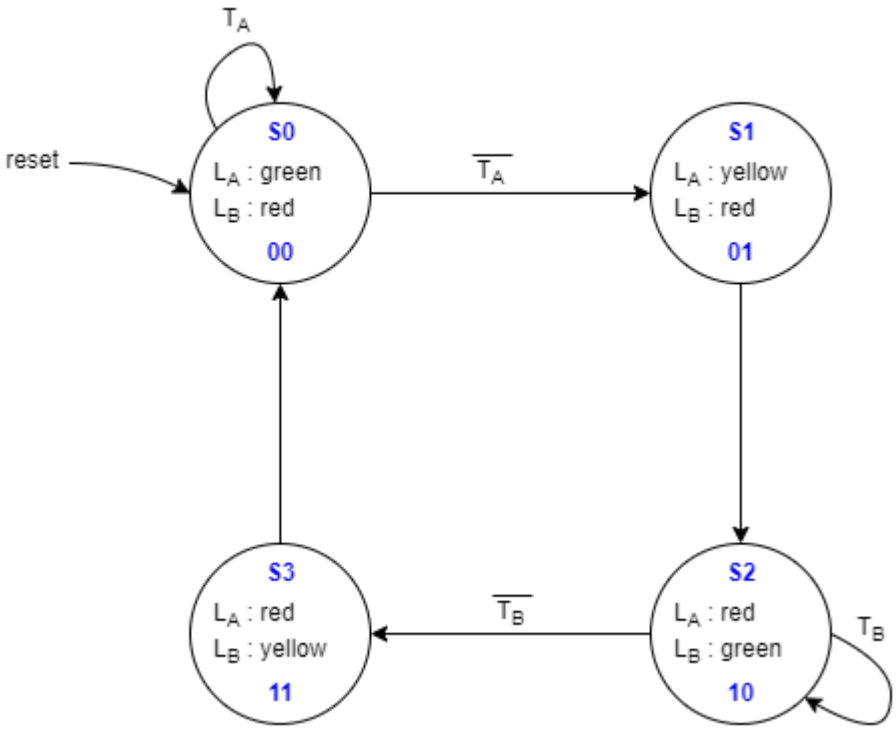


그림1. Traffic Light Controller FSM

위 그림은 Traffic Light Controller의 Finite State Machine을 나타낸 그림이다. Reset을 한 상태는 S0이며 T_A가 true일 때 동안 S0의 상태를 유지한다. 만약 T_A가 false가 되면 S1의 상태로 넘어가고 S2의 상태가 된다. 이때부터 T_A의 값에 상관없이 T_B의 값이 true일 동안 S2의 상태가 유지된다. 만약 T_B가 false가 되면 S3의 상태로 변화하고 S0의 상태로 다시 돌아간다.

Current state		Inputs		Next states	
Q_1	Q_0	T_A	T_B	D_1	D_0
0	0	1	X	0	1
0	0	0	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

표1. FSM Encoded State Transition Table

위 표1은 current state과 inputs에 대한 next state를 나타낸 표이다. 이에 대한 next state의 Boolean equation을 나타내면 다음과 같다.

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

Current state		Outputs			
Q_1	Q_0	L_{A1}	L_{A0}	L_{B1}	L_{B0}
0	0	0	0	1	0
0	1	0	1	1	0
1	0	1	0	0	0
1	1	1	0	0	1

표2. FSM output table

위의 표2는 current state에 대한 output을 나타낸 표이다. 이에 대한 outputs의 Boolean equation을 나타내면 다음과 같다.

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

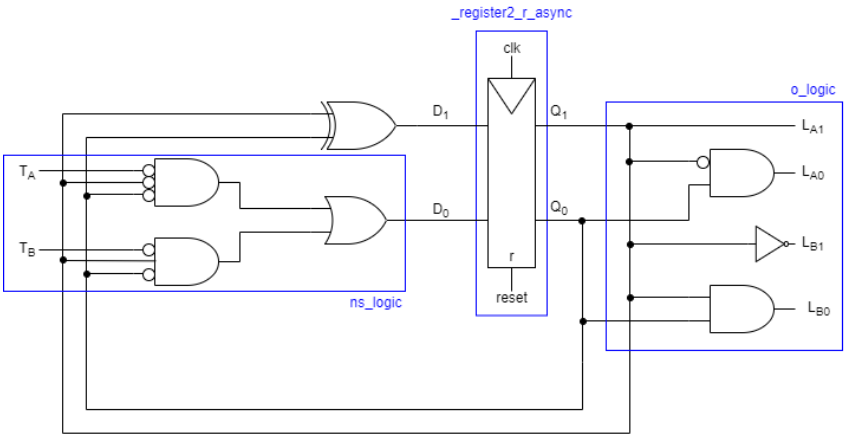


그림2. Diagram

위 그림2는 next state와 output table을 토대로 그린 diagram이다.

2) Encoding states

Output	Encoding
Green	00
Yellow	01
Red	10

표3. Encoding outputs

위의 표3은 신호등의 상태를 binary encoding을 사용하여 나타낸 것이다. 초록불 상태인 green은 00, 노란불 yellow는 01, 빨간불 red는 10이다.

State	Encoding
S0	00
S1	01
S2	10
S3	11

표4. Encoding states

위 표4는 상태 S0~S3까지를 binary encoding을 사용하여 나타낸 것이다.

3) Coding the module header

구분	이름	설명
Top module	tl_cntr	Traffic light controller의 top module
Sub module	ns_logic	Next state를 결정하는 combinational logic
Sub module	_register2_r_acync	현재 state 값을 저장하고 있는 register
Sub module	_dff_r_acync	Resettable D flip-flop with active low asynchronous reset
Sub module	o_logic	현재 state의 값으로 output 값을 결정하는 combinational logic

표5. Module configuration

4) Coding sequential circuits

Module name	구분	이름	비트 수	설명
_register_r_async	Input	clk	1-bit	Clock
		reset_n	1-bit	Active-low에서 동작하는 reset 신호로 state 초기화
		d	2-bit	Next state
	Output	q	2-bit	Current state

표6. I/O configuration of _register2_r_async

_register_r_async module은 _dff_r_async module 두 개를 가지고 있다. 각각의 _dff_r_async에 d과 q의 한 비트씩 전달하고 d에 따른 q의 값을 출력한다.

5) Coding combinational circuits

Module name	구분	이름	비트 수	설명
ns_logic	Input	Ta	1-bit	Traffic sensor A
		Tb	1-bit	Traffic sensor B
		q1	1-bit	Current state
		q2	1-bit	
	Output	d1	1-bit	Next state
		d0	1-bit	

표7. I/O configuration of ns_logic

ns_logic module은 현재 traffic light A와 B의 상태와 current state를 입력받아 next state를 계산하는 module이다.

Module name	구분	이름	비트 수	설명
o_logic	Input	q1	1-bit	Current state
		q0	1-bit	
	Output	La1	1-bit	신호등 값 출력 A
		La0	1-bit	
		Lb1	1-bit	신호등 값 출력 B
		Lb0	1-bit	

표8. I/O configuration of o_logic

o_logic은 current state를 입력받아 현재 traffic light A와 B의 신호등 상태 값을 출력한다.

2. Traffic Light Controller with Left Turn Signal

1) Finite State Machine

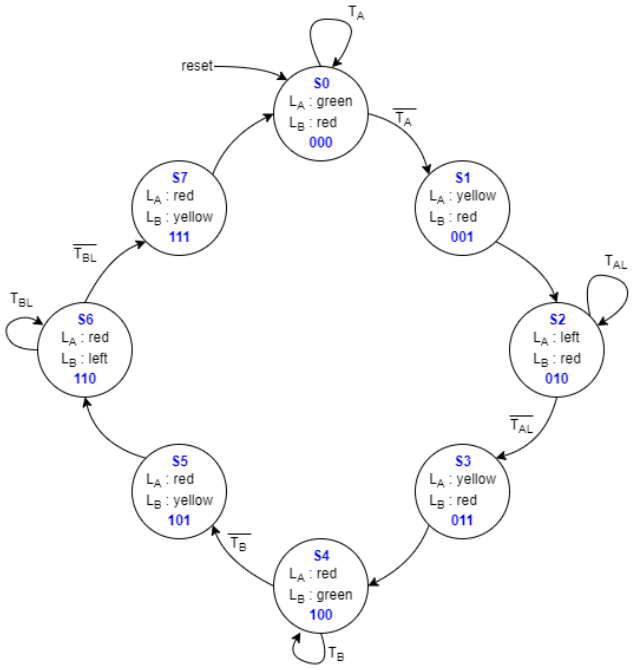


그림3. Traffic Light Controller with Left Turn Signal FSM

위 그림은 Traffic Light Controller with Left Turn Signal의 Finite State Machine을 나타낸 그림이다. Reset을 한 상태는 S0이며 T_A가 true일 때 동안 S0의 상태를 유지한다. 만약 T_A가 false가 되면 S1의 상태로 넘어가고 S2의 상태가 된다. 이때부터 T_A의 값에 상관없이 T_{AL}의 값이 true일 동안 S2의 상태가 유지된다. 만약 T_{AL}의 값이 false가 되면 S3의 상태로 넘어가고 S4의 상태가 유지된다. T_B의 값이 false가 되면 S5의 상태를 지나 S6의 상태가 된다. 그리고 T_{BL}의 값이 false가 되면 S7의 상태가 되고 처음 상태인 S0으로 돌아간다.

Current state			Inputs				Next states		
Q_2	Q_1	Q_0	T_A	T_{AL}	T_B	T_{BL}	D_2	D_1	D_0
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

표9. FSM Encoded State Transition Table

위 표9는 current state과 inputs에 대한 next state를 나타낸 표이다. 이에 대한 next state의 Boolean equation을 나타내면 다음과 같다.

$$D_2 = \overline{Q_2}Q_1Q_0 + Q_2\overline{Q_1Q_0T_B} + Q_2\overline{Q_1Q_0}\overline{T_B} + Q_2\overline{Q_1}Q_0 + Q_2Q_1\overline{Q_0}\overline{T_{BL}} + Q_2Q_1\overline{Q_0}T_{BL}$$
$$= Q_2\overline{Q_1Q_0}(\overline{T_B} + T_B) + Q_2Q_1\overline{Q_0}(\overline{T_{BL}} + T_{BL}) + \overline{Q_2}Q_1Q_0 + Q_2\overline{Q_1}Q_0$$
$$= Q_2\overline{Q_0}(\overline{Q_1} + Q_1) + \overline{Q_2}Q_1Q_0 + Q_2\overline{Q_1}Q_0$$
$$= Q_2\overline{Q_0} + \overline{Q_2}Q_1Q_0 + Q_2\overline{Q_1}Q_0$$
$$D_1 = \overline{Q_2}\overline{Q_1}Q_0 + \overline{Q_2}Q_1\overline{Q_0} + Q_2\overline{Q_1}Q_0 + Q_2Q_1\overline{Q_0}$$
$$= \overline{Q_1}Q_0(\overline{Q_2} + Q_2) + Q_1\overline{Q_0}(\overline{Q_2} + Q_2)$$
$$= Q_1 \oplus Q_2$$
$$D_0 = \overline{Q_2Q_1Q_0T_A} + \overline{Q_2Q_1Q_0T_{AL}} + Q_2\overline{Q_1Q_0}\overline{T_B} + Q_2Q_1\overline{Q_0}T_{BL}$$

Current state			Outputs			
Q_2	Q_1	Q_0	L_{A1}	L_{A0}	L_{B1}	L_{B0}
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

표10. FSM output table

위의 표10은 current state에 대한 output을 나타낸 표이다. 이에 대한 outputs의 Boolean equation을 나타내면 다음과 같다.

Q2 \ Q1Q0	00	01	11	10
0	0	0	0	1
1	1	1	1	1

표11. Karnaugh Map of Output Light L_{A1}

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

Q2 \ Q1Q0	00	01	11	10
0	0	1	1	0
1	1	1	1	1

표12. Karnaugh Map of Output Light L_{A0}

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

Q2 \ Q1Q0	00	01	11	10
0	1	1	1	1
1	0	0	0	1

표13. Karnaugh Map of Output Light L_{B1}

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

Q2 \ Q1Q0	00	01	11	10
0	1	1	1	1
1	0	1	1	0

표14. Karnaugh Map of Output Light L_{B0}

$L_{B0} = \overline{Q_2} + Q_2Q_0$

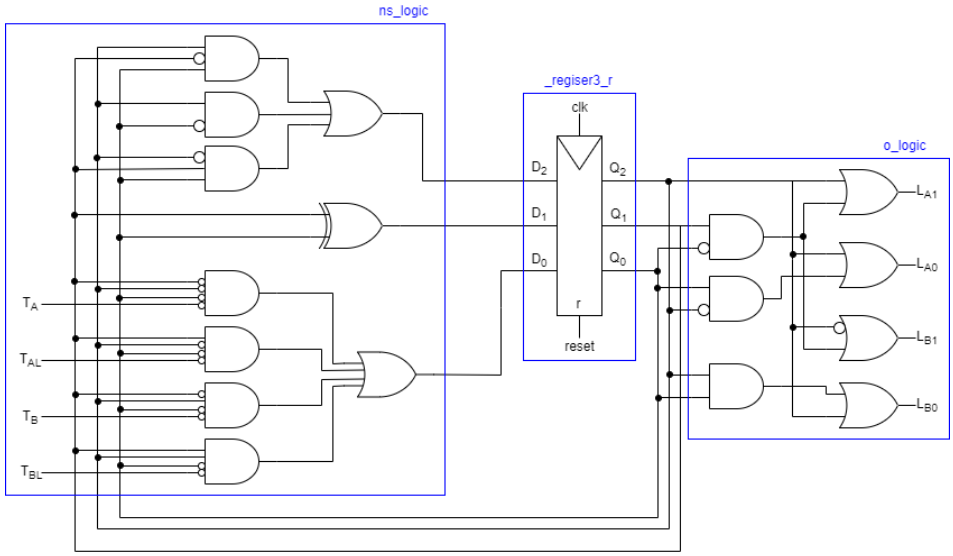


그림4. Diagram

위 그림4는 next state와 output table을 토대로 그린 diagram이다.

2) Encoding state

Output	encoding
Green	00
Yellow	01
Left	10
Red	11

표15. Encoding outputs

위의 표15은 신호등의 상태를 binary encoding을 사용하여 나타낸 것이다. 초록불

상태인 green은 00, 노란불 yellow는 01, 좌회전 신호는 10, 빨간불 red는 11이다.

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

표16. Encoding states

위 표16은 상태 S0~S7까지를 binary encoding을 사용하여 나타낸 것이다.

3) Coding the module header

구분	이름	설명
Top module	tl_cntr_w_left	Traffic light controller의 top module
Sub module	ns_logic	Next state를 결정하는 combinational logic
Sub module	_register3_r	현재 state 값을 저장하고 있는 register
Sub module	_dff_r_acync	Resettable D flip-flop with active low asynchronous reset
Sub module	o_logic	현재 state의 값으로 output 값을 결정하는 combinational logic

표17. Module configuration

4) Coding sequential circuits

Module name	구분	이름	비트 수	설명
_register3_r	Input	clk	1-bit	Clock
		reset_n	1-bit	Active-low에서 동작하는 reset 신호로 state 초기화
		d	3-bit	Next state
	Output	q	3-bit	Current state

표18. I/O configuration of _register3_r

_register_r_async module은 _dff_r_async module 두 개를 가지고 있다. 각각의 _dff_r_async에 d과 q의 한 비트씩 전달하고 d에 따른 q의 값을 출력한다.

5) Coding combinational circuits

Module name	구분	이름	비트 수	설명
ns_logic	Input	Ta	1-bit	Traffic sensor A
		Tal	1-bit	Traffic sensor AL
		Tb	1-bit	Traffic sensor B
		Tbl	1-bit	Traffic sensor BL
		q2	1-bit	Current state
		q1	1-bit	
		q2	1-bit	
	Output	d2	1-bit	Next state
		d1	1-bit	
		d0	1-bit	

표19. I/O configuration of ns_logic

ns_logic module은 현재 traffic light A/AL와 B/BL의 상태와 current state를 입력받아 next state를 계산하는 module이다.

Module name	구분	이름	비트 수	설명
o_logic	Input	q2	1-bit	Current state
		q1	1-bit	
		q0	1-bit	
	Output	La1	1-bit	신호등 값 출력 A
		La0	1-bit	
		Lb1	1-bit	신호등 값 출력 B
		Lb0	1-bit	

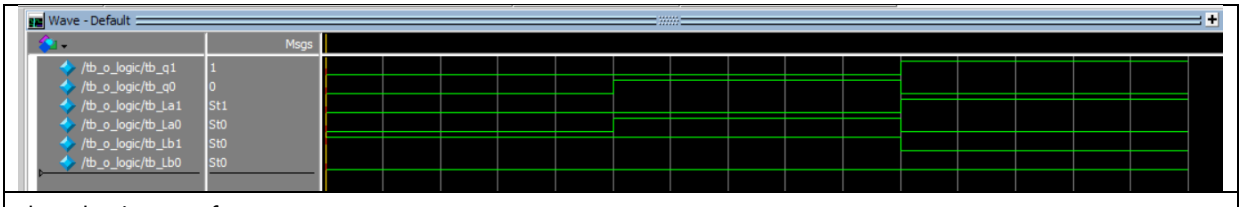
표20. I/O configuration of o_logic

o_logic은 current state를 입력받아 현재 traffic light A와 B의 신호등 상태 값을 출력한다.

4. 설계 검증 및 실험 결과

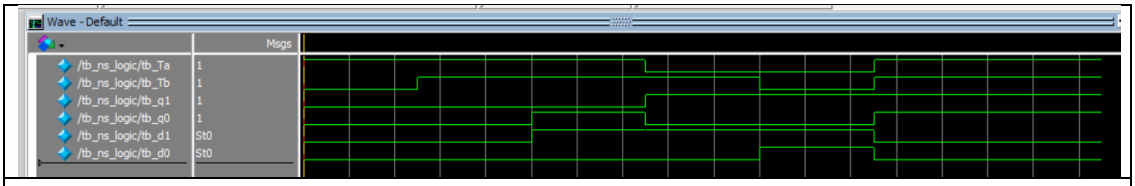
A. 시뮬레이션 결과

1. Traffic Light Controller



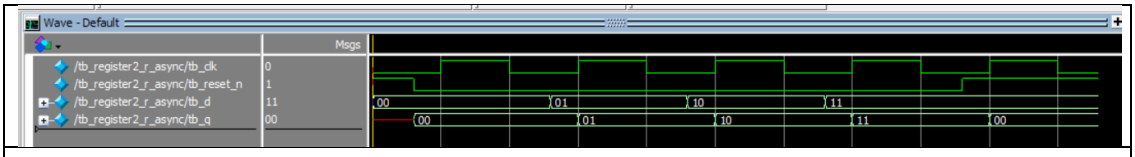
tb_o_logic waveform

tb_q1=0, tb_q0=0일 때 La는 green이므로 La1=0, La0=0의 값을 가지고 Lb는 red이므로 Lb1=1, Lb0=0가 된다. tb_q1=0, tb_q0=1일 때 La는 yellow이므로 La1=0, La0=1의 값을 가지고 Lb는 red이므로 Lb1=1, Lb0=0가 된다. tb_q1=1, tb_q0=0일 때 La는 red이므로 La1=1, La0=0의 값을 가지고 Lb는 green이므로 Lb1=0, Lb0=0가 된다. tb_q1=1, tb_q0=1일 때 La는 red이므로 La1=1, La0=0의 값을 가지고 Lb는 yellow이므로 Lb1=0, Lb0=1가 된다.



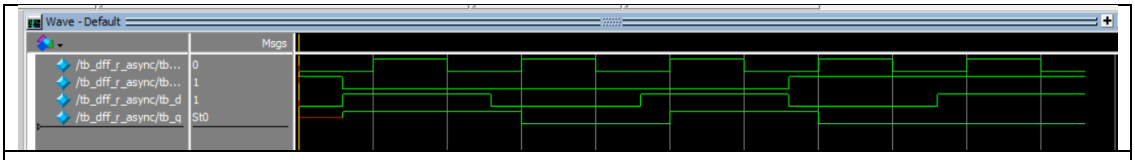
tb_ns_logic waveform

Ta, Tb, q1, q0의 값의 상태에 따라 다음 상태를 나타내는 d1과 d0가 변화하는 것을 확인할 수 있다.



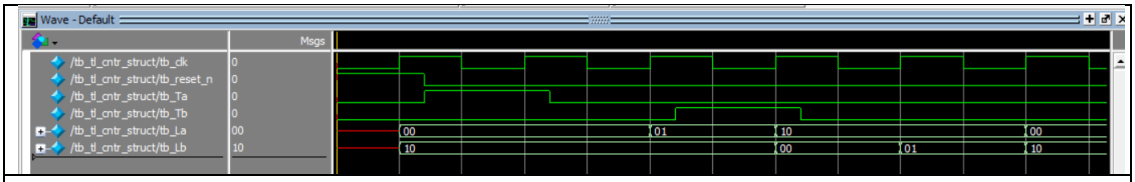
tb_register2_r_async waveform

Clock의 rising edge에서 tb_d의 값을 따라가는 것을 확인할 수 있다.



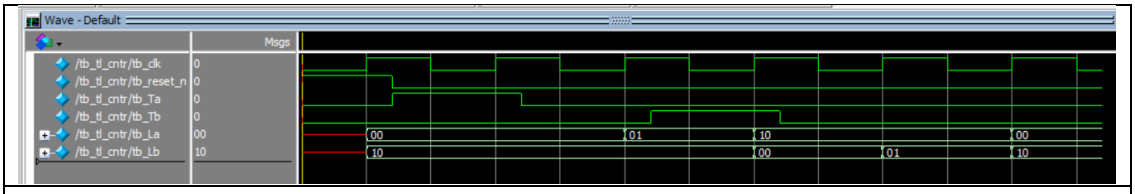
_dff_r_async waveform

Clock의 rising edge에서 tb_d의 값을 따라가며 reset이 걸리면 0의 값을 가지는 것을 확인할 수 있다.



tb_tl_cntr_struct waveform

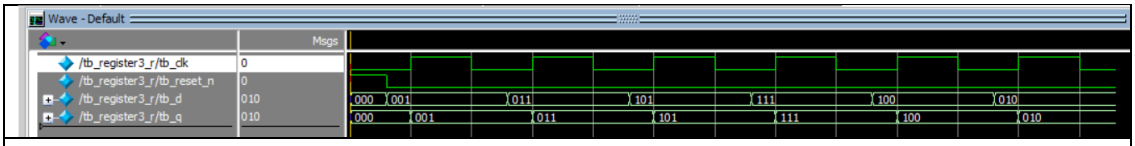
tl_cntr의 testnech 결과화면이다. Reset의 값이 active-low일 때 신호등이 동작하는 것을 확인할 수 있다. 초기에 Ta의 값이 1일 때, La의 값은 00으로 green이고 반대편의 신호인 Lb는 red가 되어야 하므로 10인 거을 확인할 수 있다. 다음으로 Ta의 값이 0이 되어 La의 값은 01로 yellow가 되고 Lb는 여전히 10이다. Tb의 값이 1이 되자 La의 값은 10로 red가 되고 Lb의 값은 00으로 green이 된다. Tb의 값이 0이 되면 Lb의 값은 01로 yellow가 되고 그때의 La는 여전히 10로 red가 유지되는 것을 확인할 수 있다.



tb_tl_cntr waveform

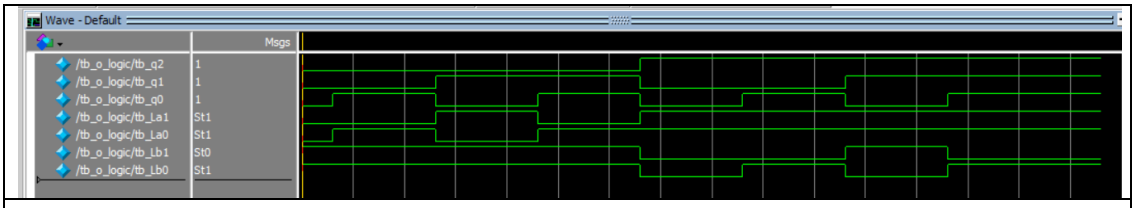
위의 waveform은 behavioral implementation의 testbench 결과 화면이다. structural implementation과 같은 testbench 값을 대입했으며, 결과적으로 같은 결과값을 보여주는 것을 확인할 수 있다.

2. Traffic Light Controller with Left



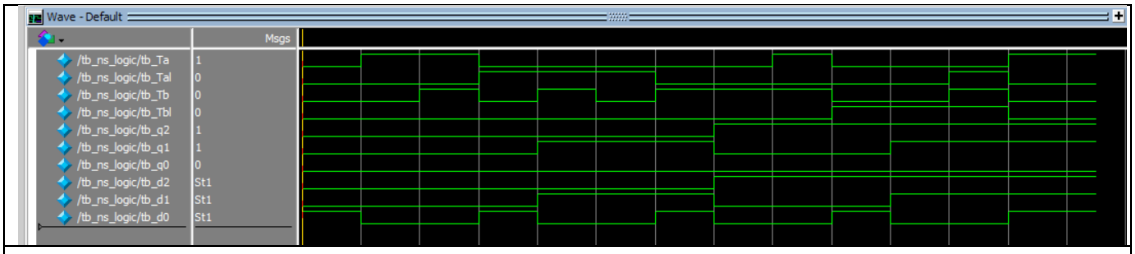
_register3_r waveform

Clock의 rising edge에서 tb_d의 값이 tb_q에 알맞게 할당되는 것을 확인할 수 있다.



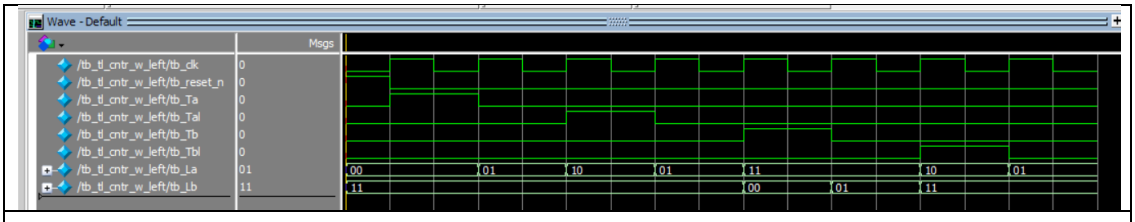
o_logic waveform

입력값인 현재 상태 q2, q1, q0에 따라 La1, La0, Lb1, Lb0에 신호등의 색을 나타내는 00, 01, 10, 11이 알맞게 나타나는 것을 확인할 수 있다.



ns_logic waveform

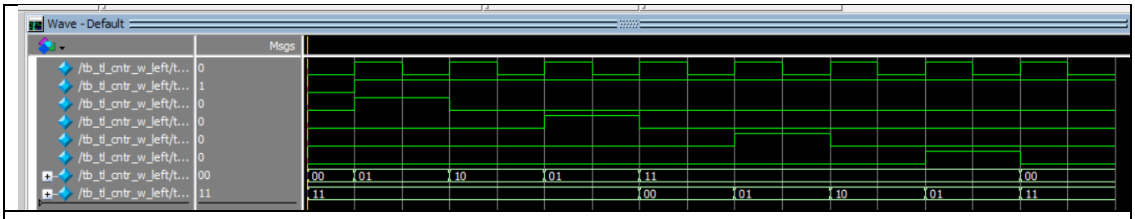
입력값 Ta, Tal, Tb, Tbl과 현재 상태 q2, q1, q0의 값에 따라 다음 신호등의 상태인 d2, d1, d0가 알맞게 출력되는 것을 확인할 수 있다.



tb_tl_cntr_w_left_struct waveform

위 그림은 tl_cntr_w_left의 testbench 결과화면이다. Ta의 값이 1일 때, La의 값은 00으로

green이고 Tb의 값은 11로 red이다. 이후에 Ta의 값이 0이 되자 La의 값은 01로 yellow로 변화하고 Tal의 값이 1이 되자 La의 값은 10으로 left를 나타내는 것을 확인할 수 있다. 이후에 Tal의 값이 0이 되면 La의 값은 01로 yellow가 된다. 그리고 Tb의 값이 1이 되자 La의 값은 11로 red가 되고 Lb의 값은 00으로 green이 된다. 이후 Tb=0, Tbl=1, Tbl=0의 결과는 Ta=0, Tal=1, Tal=0일 때와 동일하다.



tb_tl_cntr_w_left waveform

위의 waveform은 behavioral implementation의 testbench 결과 화면이다. structural implementation과 같은 testbench 값을 대입했으며, 결과적으로 같은 결과값을 보여주는 것을 확인할 수 있다.

B. 합성(synthesis) 결과

1. Traffic Light Controller

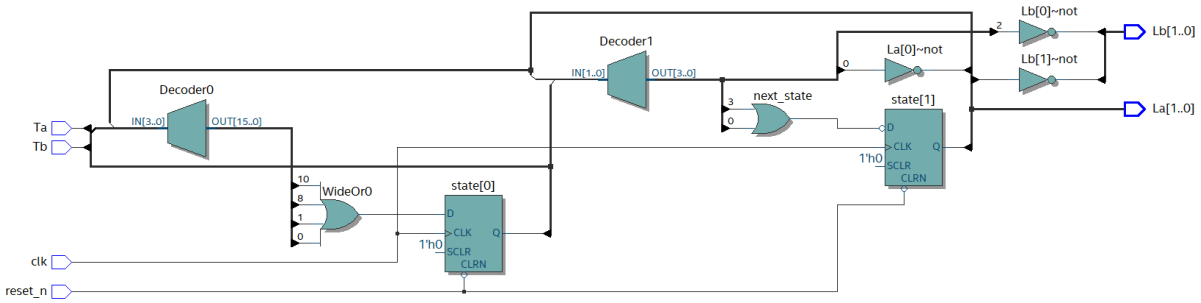
Flow Summary

<<Filter>>

Flow Status	Successful - Wed Oct 18 11:53:22 2023
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	tl_cntr
Top-level Entity Name	tl_cntr_struct
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	5 / 41,910 (< 1 %)
Total registers	2
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 %)

tl_cntr flow summary

tl_cntr module의 logic utilization은 5로 1% 미만이고 total register는 2개를 사용했으며 pins는 8으로 2%정도 사용하였다.



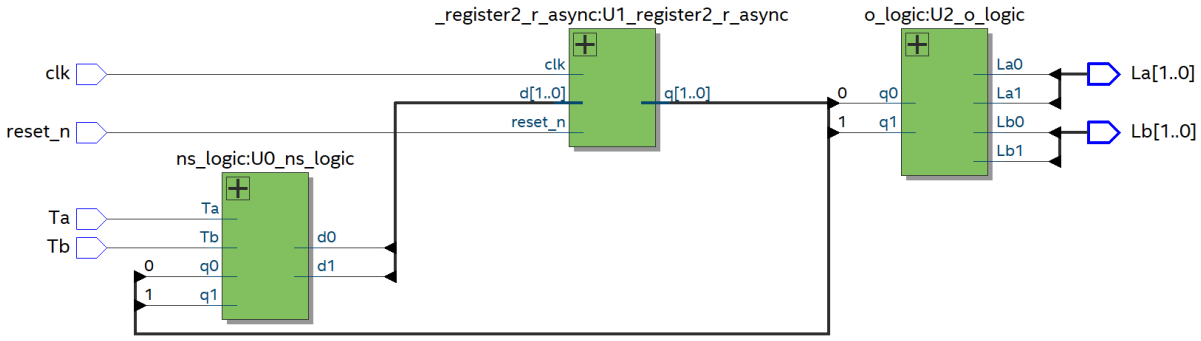
tl_cntr RTL viewer

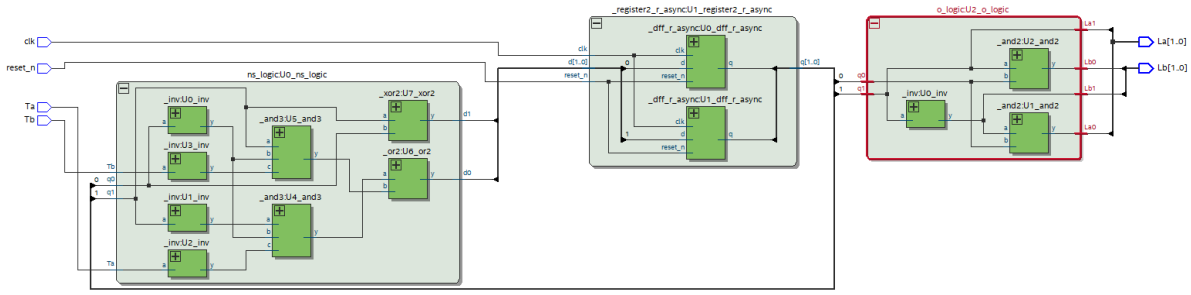
RTL viewer를 통해 tl_cntr module 내에서 decoder와 XOR, NOT, D flip-flop의 기능들이 수행되는 것을 확인할 수 있다.

Flow Summary	
<<Filter>>	
Flow Status	Successful - Wed Oct 18 11:53:22 2023
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	tl_cntr
Top-level Entity Name	tl_cntr_struct
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	5 / 41,910 (< 1 %)
Total registers	2
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 %)

tl_cntr_struct flow summary

tl_cntr_struct module의 logic utilization은 5로 1% 미만이고 total register는 2개를 사용했으며 pins는 8로 2%정도 사용하였다.





tl_cntr_struct RTL viewer

RTL viewer를 통해 ns_logic, o_logic, _register2_r_async module이 연결되어 동작하고 있다는 것을 확인할 수 있다. ns_logic과 o_logic module 내에는 next state인 d0와 d1 또는 outputs La와 Lb를 계산하기 위해 AND gates, OR gates, inverter가 사용된 것을 알 수 있다.

2. Traffic Light Controller with Left

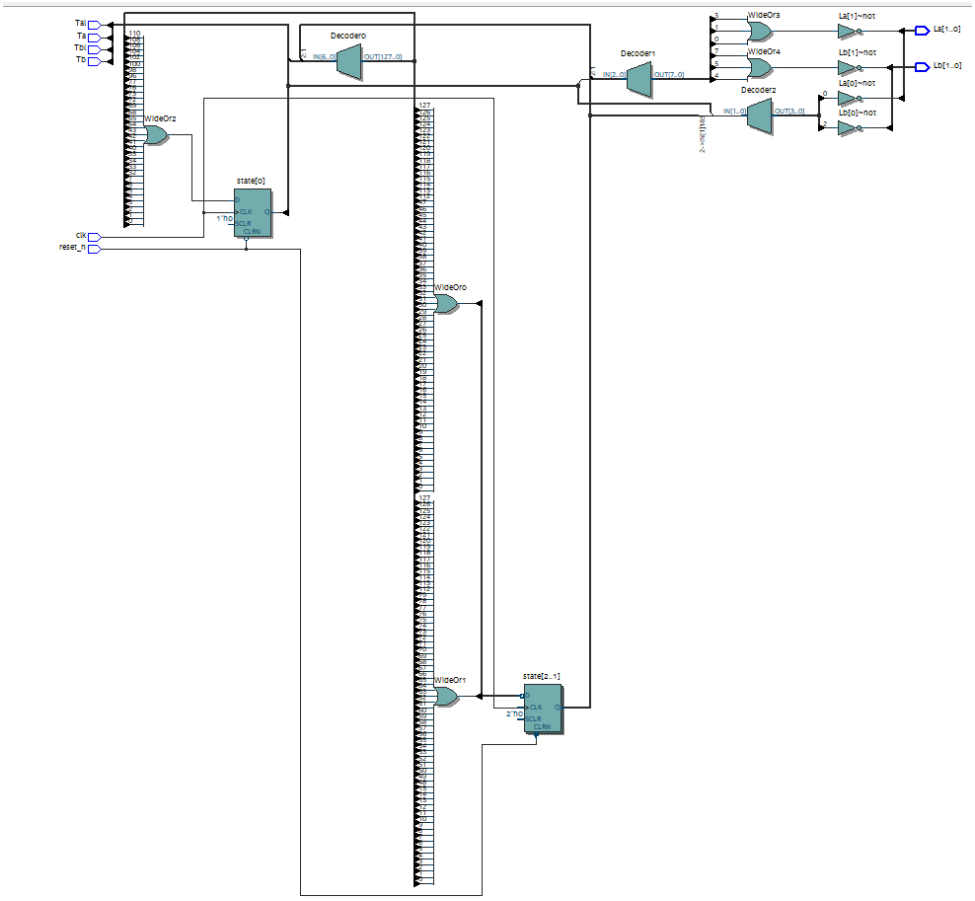
Flow Summary

<<Filter>>

Flow Status	Successful - Wed Oct 18 13:13:58 2023
Quartus Prime Version	18.1.0 Build 625 09/12/2018 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 %)

tl_cntr_w_left flow summary

tl_cntr_w_left module의 logic utilization은 5로 1% 미만이고 total register는 4개를 사용했으며 pins는 10으로 2%정도 사용하였다.



tl_cntr_w_left RTL viewer

RTL viewer를 통해 tl_cntr_w_left module 내에서 decoder와 XOR, NOT, D flip-flop의 기능들이 수행되는 것을 확인할 수 있다.

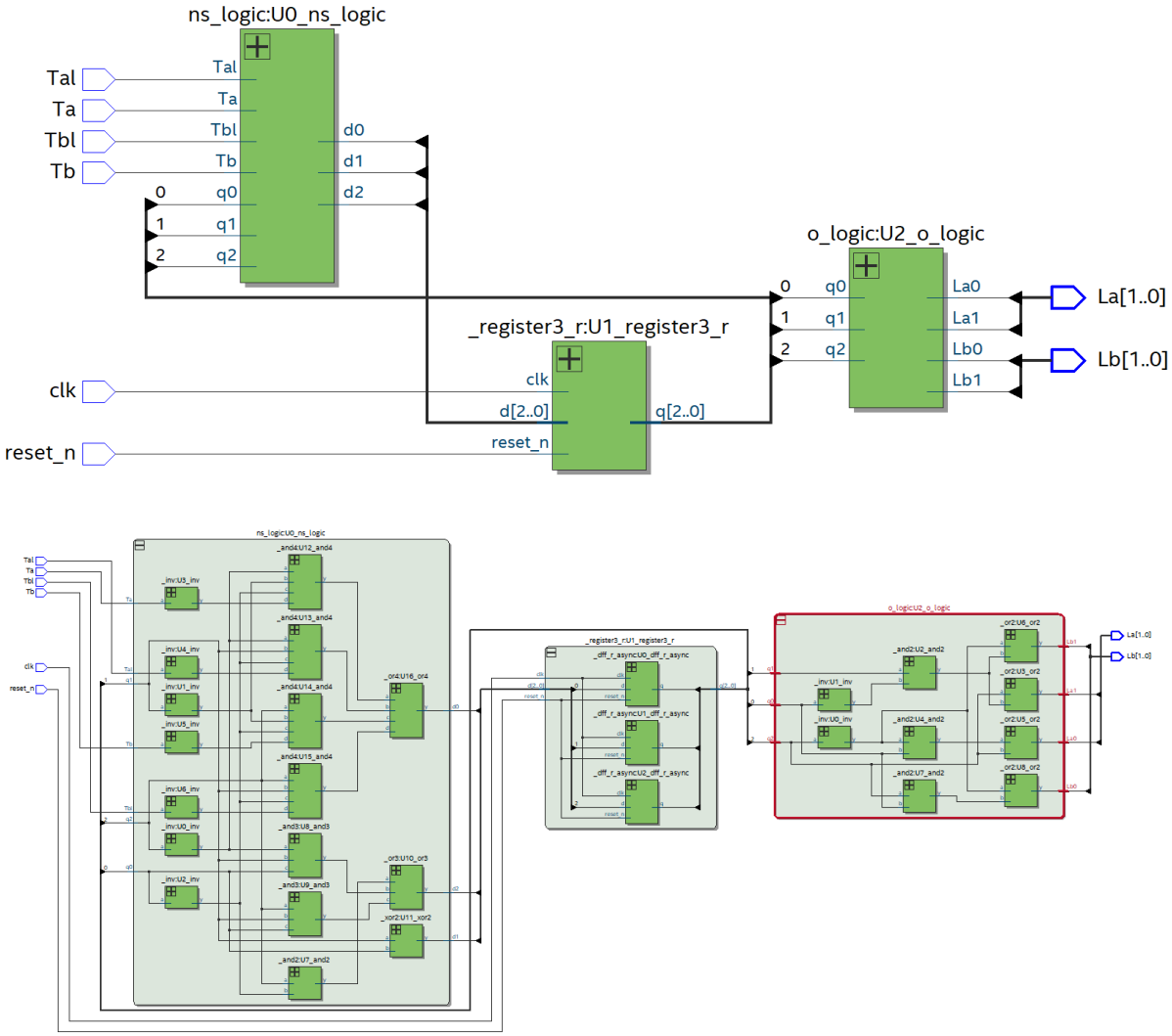
Flow Summary

<<Filter>>

Flow Status	Successful - Wed Oct 18 12:19:20 2023
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	tl_cntr_w_left
Top-level Entity Name	tl_cntr_w_left_struct
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 %)

tl_cntr_w_left_struct flow summary

tl_cntr_w_left_struct module의 logic utilization은 5로 1% 미만이고 total register는 5개를 사용했으며 pins는 10으로 2%정도 사용하였다.



tl_cntr_w_left_struct RTL viewer

RTL viewer를 통해 tl_cntr_w_left_struct module 내에서는 ns_logic, _register3_r, o_logic module을 하위 모듈로 가지고 있으며 서로 연결되어 있는 것을 확인할 수 있다.

5. 고찰 및 결론

A. 고찰

1. Testbench의 결과값이 원하는 원하는 기대값이랑 다르게 출력되었다. 이는 컴파일 도중에 발생한 오류가 아닌, 입력값에 해당하는 변수의 이름을 잘못 작성하여 발생한 오류였다.
2. Left Turn Light를 구현할 때 next states와 outputs의 값을 구하는데 많은 시간이 걸렸다. 각각의 Boolean equation을 구하기 위해 표를 그리고 계산을 하는 과정에서 식이 맞는지 확인하기 위해 많은 시간을 들였다.

B. 결론

1. FSM을 표과 diagram으로 그릴 나타낼 땐 쉽다고 생각하였었는데, 막상 Verilog로 코드를 짜려고 하니 많이 어려웠다. 특히 testbench에서 unknown값 x가 나타나 이를 해결하는데 많은 어려움을 겪었다.
2. 이번에 구현한 outputs들은 binary encoding을 사용하여 구현하였다. 하지만 이보다 더 많은 outputs의 값들, 즉 U턴 신호, 우회전 신호, X표 신호 등이 주어진다면 one-hot encoding을 사용하여 보다 알아보기 쉽게 만드는 방식으로 응용할 수 있을 것 같다.

6. 참고문헌

[1] FSM / https://catslikefish.tistory.com/entry/Finite-State-MachineFSM

[2] 유지현 / Factoring Finite State Machine & Timing Constraints / 광운대학교 / 2022년

[3] 이준환 / Overview of Verilog – Finite State Machine Design / 광운대학교 / 2023년

[4] 이혁준 / 컴퓨터공학기초실험2 Week #6 (Lab #7) / 2023년