2016312761 여혁수 HW0 Traffic Light Controller 보고서

1.Design Objective

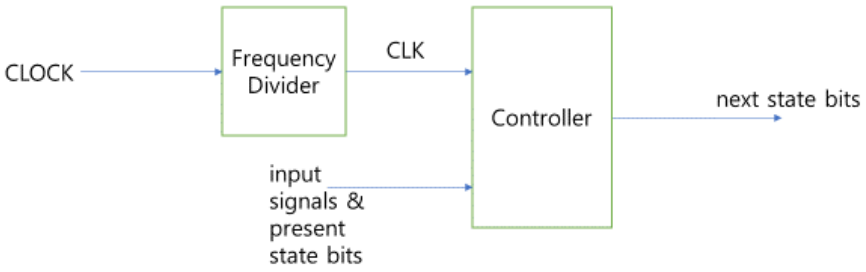
가상의 삼거리 도로에서 교통 신호를 상황에 맞게 변화시키는 논리 회로 시스템을 구현하는 것이 이번 디자인의 목표이다. 일정한 시간에 따라 교통 신호가 변하고, 어떤 쪽에서 교통량이 많아지거나 횡단보도 신호가 필요한 경우 같은 상황에서 교통 신호가 그에 맞게 변하는 디자인을 만든다. 또한 낮과 밤의 교통상황이 좀 다른 것으로 가정했다.

2.Design Specifications

우선 디자인할 회로 관련 플랫폼 프로그램이 필요한데, Logisim 프로그램을 사용하여 디자인할 것이다.

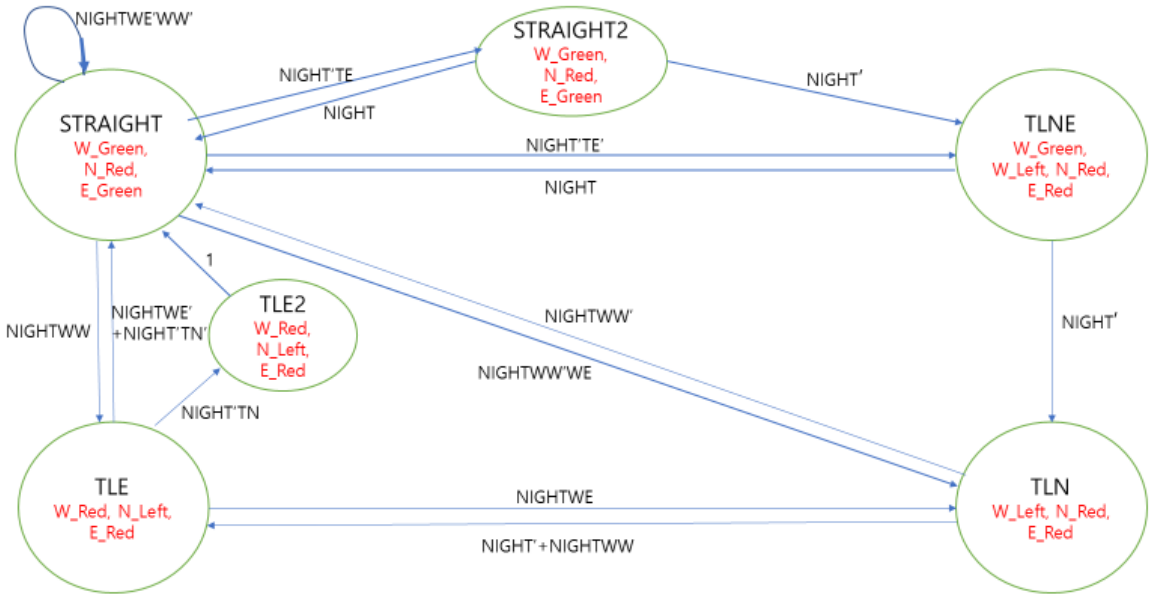
일정한 시간에 따라 교통 신호가 변하게 하기 위해 일정 주기가 되면 신호를 주는 CLOCK 인풋이 필요하다. Initial CLOCK 인풋은 64Hz 신호인 것으로 조건이 주어졌기 때문에 이를 낮추는 회로를 구현해야 한다. 이를 구현하기 위해 1비트 신호를 Rising Edge에서 패스시키는 D-FilpFlop을 여러 개 이어서 최종 아웃풋에 신호 1이 들어오는 주기를 늘리는 디자인을 계획했다.

다음으로 삼거리 도로에서 교통 신호를 주기적으로 그리고 교통상황에 따라 알맞게 보내주기 위해 관련 알고리즘이 담긴 Controller회로가 필요하다. 회로를 만들기 위해 먼저 정상작동하는 디자인 파일[1-2]로 시뮬레이팅 해보고 그 인풋 신호들에 따라 현재 교통상황에서 다음 교통상황이 결정되는 State Diagram을 그린다. 그린 것을 여러 개의 Boolean function으로 변환하여 Espresso[3] 프로그램의 인풋으로 넣는다. 프로그램을 돌려 간소화된 논리를 얻어내고 그 결과를 pla2circ[4] 프로그램에 넣어서 circuit파일로 변환한다. 이렇게 Logisim에서 작동하는 회로를 디자인[5]할 것이다.



지금까지 말한 디자인을 block diagram으로 나타내면 위와 같다. Frequency Divider가 CLOCK 주기를 늘리는 회로이고, Controller가 다음 교통상황을 결정하는 state diagram을 회로화 한 것이다. 나올 수 있는 모든 교통상황을 숫자화해서 3bit이진수로 표현했는데 Present state bits와 Next state bits가 이 3bit수 중 하나인 것이다.

3.Controller State Diagram



STATE 설명)

STRAIGHT: 삼거리에서 좌측과 우측에서 직진이 가능하고 위쪽 차들은 정지하고 있는 상태이다.

TLNE: Turn Left to North and East의 줄임말로 좌측 차들이 좌회전 및 직진을 할 수 있고, 나머지 위쪽과 우측 차들은 우회전만 가능한 상태이다.

TLN: Turn Left to North의 줄임말로 좌측 차들이 좌회전만 가능하고 나머지 위쪽과 우측 차들은 우회전만 가능한 상태이다.

TLE: Turn Left to East의 줄임말로 위쪽 차들이 좌회전할 수 있고 좌측 차들은 정지해 있고, 우측 차들은 우회전만 가능한 상태이다.

Input/Output 설명)

Input: TN, TE, NIGHT, WW, WE

TN: 위쪽의 교통량이 많을 때를 나타내는 신호로 낮인 경우에만 유효한 신호이다. TLE state일 때 1이 들어오면 TLE state에 한 번 더 머무른다.

TE: 우측의 교통량이 많을 때를 나타내는 신호로 역시 낮에만 유효하다. STRAIGHT state일 때 1이 들어오면 STRAIGHT state에 한 번 더 머무른다.

NIGHT: 밤을 나타내는 신호이다. 낮인 경우에 평상시에 STRAIGHT->TLNE->TLN->TLE->STRAIGHT 순으로 state가 순환되지만 밤인 경우는 STRAIGHT state가 계속 반복된다.

WW: 좌측 횡단보도에 초록불을 주라는 신호로 TN, TE와 달리 밤에만 유효한 신호이다. 1이 들어오면 다음 state는 TLE가 된다.

WE: 우측 횡단보도에 초록불을 주는 신호로 역시 밤에만 유효하다. 1이 들어오면 다음 state는 TLN이 된다.

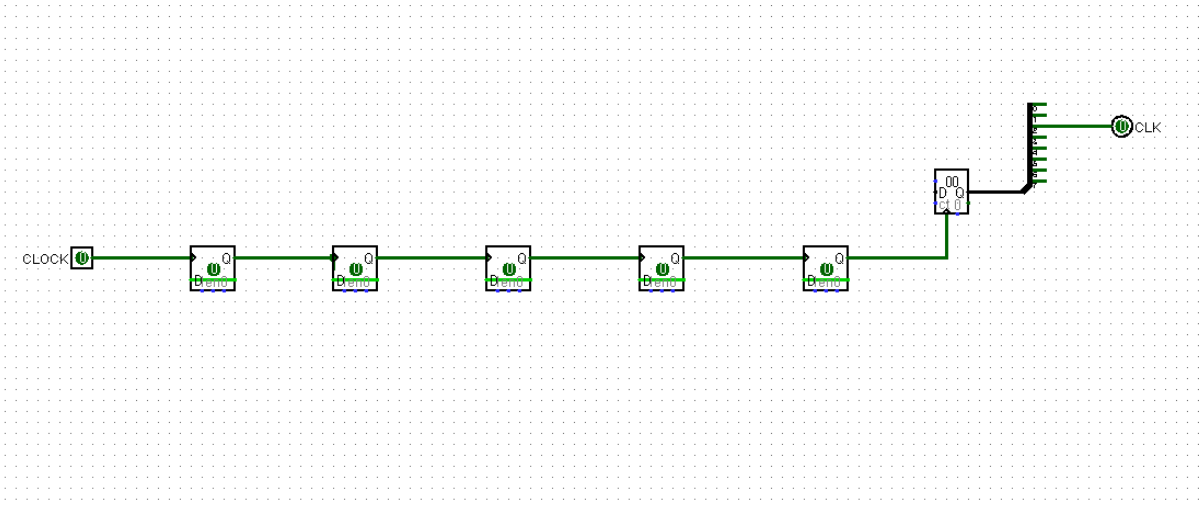
Output: W_Red, W_Green, W_Left, N_Red, N_Left, E_Green, E_Red

'_' 문자를 기준으로 오른쪽 문자는 신호등의 신호를 말하고, 왼쪽 문자는 그 신호가 어느 쪽 신호인지를 나타낸다. W_Red이면 좌측에 빨간불이 들어온 것이다.

4.Detailed Design

A. Frequency Divider

앞서 Design Spec. 에서 D-flipflop을 여러 개 이어서 구현한 것이라고 했다. 여기서 Decade Counter를 추가로 붙였다. D-flipflop만으로도 구현 가능할 것 같지만 Homework0 pdf[6]에 Counter도 언급되어서 써보기로 했다. 우선 D-flipflop의 동작에 대해서 알아야한다. D-flipflop은 CLOCK이 들어오면 Data를 Q(output)로 패스시키는 회로이다. 그러니까 Rising Edge로 설정하면 CLOCK이 0에서 1로 바뀔 때 Q로 패스하는 건데 이 때 보낼 Data를 Q'으로 주게 되면 한 번은 Q에 1이 보내질 것이고, 다음 번은 0이 보내질 것이다. 그 말은 CLOCK Rising Edge가 2번 나타나면 Q는 1번 1이 되었다가 0이 되는 것이다. 주기가 2배로 늘어난 것이 된다. 이 Q를 다시 다음 D-flipflop의 CLOCK으로 주면 이 D-flipflop의 Q는 주기가 4배로 늘어난 CLOCK이 될 것이다.[7]



Counter도 역시 CLOCK 인풋을 받는데 CLOCK이 Rising Edge일 때마다 1씩 counter를 증가시키는 회로이다. 4비트 counter로 설정하면 십진수로 15까지 수를 증가시킬 수 있는 것이다. Counter도 CLOCK 주기를 늘리려고 사용되었는데, 스플리터가 같이 있었기에 가능했다. 스플리터로 Counter의 8비트 아웃풋을 1비트씩 따로 나눠서 그중 오른쪽에서 3번째 비트만을 최종 아웃풋으로 설정했다. Counter는 CLOCK이 Rising Edge일 때마다 아웃풋을 0에서 1씩 증가시킬텐데 오른쪽에서 3번째 비트가 1이 되려면 4만큼 증가해야 할 것이다. 4만큼 또 증가하면 이진수로 1000이 되서 3번째 비트는 다시 0이 될 것이다. 그 다음 1이 들어오려면 이진수로 1100인 12, 그 다음은 10100인 20, 이렇게 8씩 증가해야 한다. 그렇기 때문에 CLOCK에서 8번의 Rising Edge가 최종 아웃풋에서 1번의 Rising Edge가 되는 것이다.

주기를 얼마나 늘려야하냐면 64Hz의 CLOCK을 0.2~0.33Hz로 맞춰야한다. 그래서 D-flipflop 5개를 붙여 주기를 2^5배 늘리고, Counter 및 스플리터로 8배를 또 늘렸다. 이렇게 하면 64Hz가 0.25Hz가 된 셈이다.

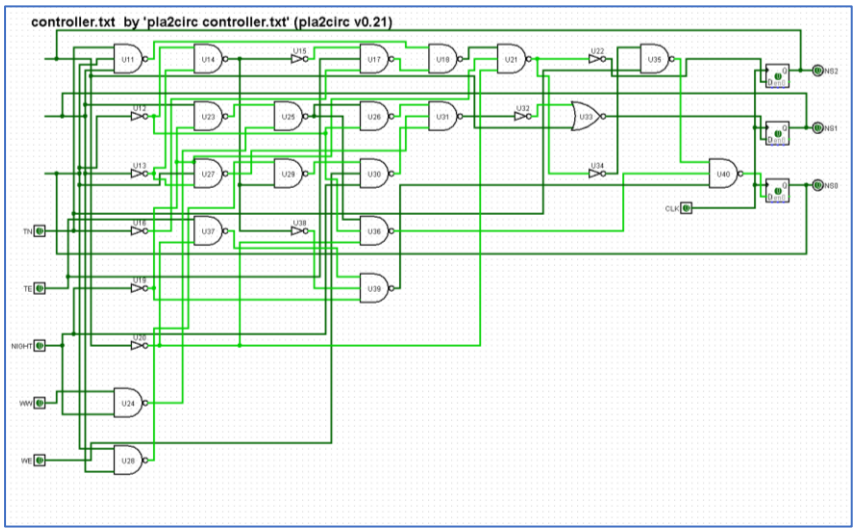
B. Walker Signal Modifier

앞서 나온 State Diagram의 인풋 신호 중 WW, WE는 사실 이 Walker Signal Modifier를 거쳐서 나온 신호이다. 이번 디자인의 조건 중 하나가 WW나 WE의 경우 어떤 타이밍에 신호를 주었든 다음 state로 갈 때까지 신호를 주고 있는 게 아니라 한 번 잠깐 주기만 하면 다음 state에서 그에 맞는 state로 가야하는 것이다. 그래서 WW, WE는 우리가 1을 주면 그 값을 저장해서 다음 state로 갈 때 사용할 수 있게 해주는 회로가 필요한 것이다. 그것이 Walker Signal Modifier이다.

기존 바탕이 되는 TrafficLight.circ의 회로를 그대로 사용하였는데, 여러 개의 D-flipflop으로 디자인되었다. 일단 CLOCK이 Rising Edge일 때 WW나 WE가 1이 들어오면 그 값이 D-flipflop의 Data여서 바로 Q로 패스시킨다. 이 Q가 modifier를 거친 WW와 WE의 값이다. 이렇게 하면 신호를 잠깐만 줘도 아웃풋은 계속 1을 유지할 것이다. 하지만 여기서 WW나 WE에 신호가 왔을 때 한 번 알맞은 state로 가고 다시 평상시 state로 돌아와야 하는데 계속 1을 유지한다면 조건이 계속 맞아 평상시 state로 돌아오지 않을 것이다. 그래서 WW나 WE 값을 리셋해주는 인풋이 있다. 이름은 RS_W, RS_E인데 한 번 알맞은 state로 갔을 때 이 값이 1이 되어서 WW와 WE가 연결되어 있는 D-flipflop에 Clear 핀으로 가게 된다. 그렇게 되면 아웃풋이 0으로 바뀌기 때문에 한 번 state가 바뀌면 그 state가 다음 state로 유지되는 일이 없어지게 된다.

C. Controller

Controller 회로는 크게 2개의 모듈로 디자인되어 있다. 하나는 현재 state와 관련된 비트와 교통상황 변화에 관한 인풋 비트를 받아서 다음 state를 결정하는 Controller, 다른 하나는 그 다음 state에 대한 정보가 담긴 비트를 받아 실제 교통신호 아웃풋으로 변환해주는 Signaller이다.

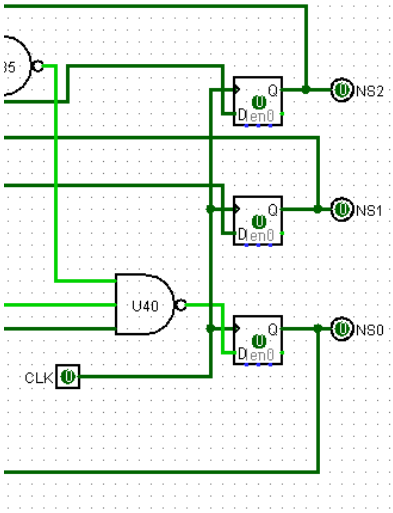


State는 총 6개로 정해서 현재 state나 다음 state를 나타내는 비트는 3비트로 0~6 사이의 숫자를 나타내고 있다. 전체 디자인에서는 위 사진과 같이 현재 state를 주는 인풋 핀이 따로 없는데 Controller의 아웃풋인 다음 state가 그 다음 CLOCK의 현재 state, 즉 인풋이 되니까 핀이 필요가 없다. 또한 이렇게 함으로서 현재 state를 보고 output이 정해지기 때문에 Moore machine이라고도 할 수 있게 된다.

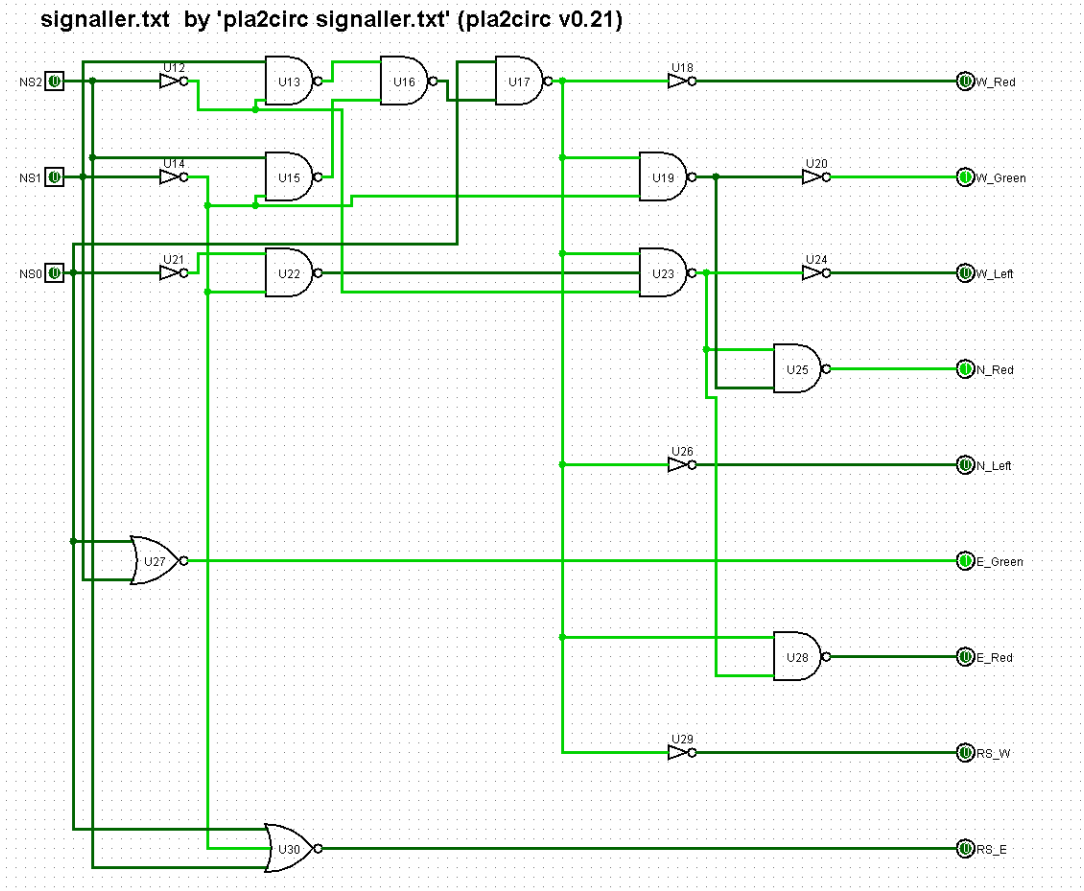
둘 중 우선 Controller는 앞서 나온 state diagram을 회로화 한 것이다. 각각의 state 변환 조건에 대해서 설명해보자면 우선 낮에는 별다른 인풋이 없으면 STRAIGHT->TLNE->TLN->TLE->STRAIGHT 순으로 state를 반복한다. 낮에 STRAIGHT state일 때 TE 인풋이 들어오면 STRAIGHT에 한 번 더 머무른다. 그리고 밤이 된 것이 아니라면 다시 TLNE부터 아까 말한 순서대로 반복한다. TN 인풋은 낮에 TLE state일 때 유효한데, TLE에 한 번 더 머무르게 되고, 그 다음 STRAIGHT부터 반복이 된다. 밤의 경우 별다른 인풋이 없으면 STRAIGHT state에 계속 머무른다. 하지만 WW 신호가 들어온 경우 TLE state로 가서 좌측 횡단보도로 건널 수 있게 해주고 그 다음 WE도 신호가 들어온 게 아니라면 다시 STRAIGHT로 돌아간다. WE도 신호가 들어온 경우 TLN state로 가서 우측 횡단보도로 건널 수 있게 해주고 WW에 신호가 들어오지 않으면 STRAIGHT로 돌아간다. WW에 신호가 들어온 경우 STRAIGHT로 가지 않고 TLE state로 간다. WW와 WE가 모두 들어오면 WW가 우선순위를 가져 TLE->TLN->STRAIGHT 순으로 state가 이동한다.

그리고 별다른 인풋이 없으면 낮이고 교통량이 많은 상황도 아니기 때문에 STRAIGHT의 다음state인 TLNE로 가게 되어있는데 Initial state를 STRAIGHT로 설정하기 위해 아웃풋을 D-flipflop에 저장하기로 했다. 아웃풋이 될 값을 D-flipflop의 Data로 주면 CLOCK이 Rising Edge일

때 비로소 결과를 내보내니까 Simulator가 CLOCK을 딱 시작했을 직후의 결과는 아무것도 건드리지 않은, 즉 모든 값이 0인 상태가 되므로 STRAIGHT state로 시작할 수 있다. 또한 이 D-flipflop의 Q인 아웃풋은 다시 다음 state를 구하기 위한 현재 state가 되므로 인풋 자리로 이동시킨다. 아래 사진에서 D-flipflop을 거친 값이 아웃풋이 되는 것을 볼 수 있다.



다음으로 Signaller는 다음 state를 알고 있는 상태에서 최종적으로 그에 맞는 교통 신호를 보내주는 회로이다. 예를 들어 STRAIGHT state로 가야한다면 W_Green, N_Red, E_Green만 1을 보내주는 방식이다.



D. Input/output files (if you use Espresso)

논리를 간소화하기 위해 Controller와 Signaller 각각에 Espresso 프로그램을 사용하였다. Input 텍스트의 각 논리식은 state diagram에서 하나의 상태변화와 같다.

Control.txt (input)

```
.i 8
.o 3
.ilb S2 S1 S0 TN TE NIGHT WW WE
.ob NS2 NS1 NS0
000--1-- 000
000010-- 100
000--101 010
000--11- 011
000-00-- 001
001--0-- 010
001--1-- 000
010--10- 000
010--11- 011
010--0-- 011
011--1-1 010
011--1-0 000
0110-0-- 000
0111-0-- 101
100--1-- 000
100--0-- 001
101----- 000
.e
```

Controller.txt (output.txt)

```
.i 8
.o 3
.ilb S2 S1 S0 TN TE NIGHT WW WE
.ob NS2 NS1 NS0
.p 9
-00-00-- 001
000010-- 100
000--1-1 010
0111-0-- 101
011--1-1 010
0-0--11- 011
100--0-- 001
001--0-- 010
010--0-- 011
.e
```

S2, S1, S0이 현재 state에 대한 비트이고, NS2, NS1, NS0이 다음 state에 대한 비트이다.

Signal.txt (input)

```
.i 3
.o 9
.ilb NS2 NS1, NS0
.ob W_Red W_Green W_Left N_Red N_Left E_Green E_Red RS_W RS_E
000 010101000
001 011100100
010 001100101
011 100010110
100 010101000
101 100010110
.e
```

Signaller.txt (output)

```
.i 3
.o 9
.ilb NS2 NS1, NS0
.ob W_Red W_Green W_Left N_Red N_Left E_Green E_Red RS_W RS_E
.p 5
-00 010101000
010 001100101
101 100010110
011 100010110
001 011100100
.e
```

NS2, NS1, NS0이 다시 인풋이 되고 교통 신호로 변환된다.

5. Conclusions

Frequency Divider를 통해 CLOCK 신호의 주기를 조작할 수 있었고, state diagram과 여러 변환 프로그램을 통해 나의 논리가 구현된 Controller 회로를 만들 수 있었다. 그래서 최종적으로 교통 신호 컨트롤러 구현이라는 목표를 이룰 수 있었다. 처음엔 CLOCK을 어떻게 조작해야할지 감이 안와서 정말 답답했지만 hw0에 대한 영상[8]도 도움이 되었고 Dflipflop을 보고 깨닫게 되면서 많은 것들이 잘 풀렸습니다. 생각보다 재미있었습니다. 감사합니다!

6. References (Reference Rules in IEEE or Monash univ.[9-10])

[1] TrafficLight.circ, 민형복. Accessed: Sept.19, 2020 Available: http://class.icc.skku.ac.kr/~min/di/

[2] ice3024n.jar, 민형복. Accessed: Sept. 19, 2020 Available: http://class.icc.skku.ac.kr/~min/di/

[3] 민형복, "ESPRESSO, 논리 간소화 프로그램," 디지털 시스템. http://class.icc.skku.ac.kr/~min/di/ (Accessed Sept. 15, 2020)

[4] pla2circ v0.21, 민형복. Accessed: Sept. 20, 2020 Available: http://class.icc.skku.ac.kr/~min/di/

[5] 민형복, "Logisim 설치 및 사용 방법," 디지털 시스템. http://class.icc.skku.ac.kr/~min/di/ (Accessed Sept. 15, 2020)

[6] 민형복, "Homework0.pdf," [Online]. Available: http://class.icc.skku.ac.kr/~min/di/ . [Accessed Sept. 15, 2020].

[7] ElectronicsTutorials, "Frequency Divisions," [Online]. Available: https://www.electronics-tutorials.ws/counter/count_1.html . [Accessed Sept. 20, 2020].

[8] 민형복. ds_hw0.mp4, Topic: "Homework #0"

[9] Monash University, "About the IEEE Style: In-Text Citations and Reference List" [Online]. Available: http://nli.vnunet.com/news/1116995 . [Accessed Sept. 23, 2020].

[10] IEEEDataport, "How to Cite References: IEEE Documentation Style" [Online]. Available: https://ieee-dataport.org/sites/default/files/analysis/27/IEEE%20Citation%20Guidelines.pdf . [Accessed Sept. 23, 2020].