Digital System Design Lab 3 Report

김지현 (20220302)

04/03/2023

1. 개요

이번 실습에서는 디코더와 멀티플렉서에 대해 알아볼 것이다. 첫번째로는 2 -to-4 active low 디코더를 이용해 4-to-16 active low 디코더를 구현할 것이다. 그리고 4 bit input 이 들어왔을 때 소수 판별기와 배수 검출기 (2, 3, 5, 7 , 11) 를 구현할 것이다. 마지막으로, 5 bit input 이 들어왔을 때 majority 를 나타내는 F 를 출력하는 회로를 구현할 것이다.

2. 이론적 배경

1) 디코더:

디코더란 multiple-input, multiple-output 이 존재하는 로직에서 binary 인 n 개의 input code 를 받아와 최대 2^n 개의 output code 를 출력하는 것이다. Decoder 는 종종 minterm generator 라고도 불리는데 각 output 출력값에 대해 minterm number 들을 OR gate 로 합해서 SOP 꼴로 만든 결과와 같기 때문이다. 이번 실습에서는 active low enable 디코더가 사용된다. Input 값으로는 I1, I0, 그리고 EN 이 존재한다. EN 이 1 일때는 I1 과 I0 의 값에 상관 없이 모든 출력값이 1 을 가지고 EN 이 0 을 가질 경우에는 출력값 중 3 개가 1 을 가지게 된다.

2) 디코더의 확장:

이번 실험에서는 5 개의 2-to-4 active low enable 디코더를 사용해서 4-to-16 active low decoder 를 만들게 된다. 하나의 2-to-4 디코더의 출력값이 다른 4 개의 디코더에서의 EN 으로 활용된다.

3) 소수 판별기/ 배수 판별기:

소수 판별기는 입력값이 소수 일 때, 결과가 1, 또는 참,을 가지게 된다. 소수는 1 과 자기 자신만을 약수로 가지는 수를 말한다. 이번 실험에서 소수 판별기에서는 4-bit input 이 들어오게 된다. 즉, 0010 이 입력 된다면 2 를 뜻하여 출력은 1, 또는 참,이 될 것이다.

배수 판별기는 4bit 입력이 2, 3, 5, 또는 7 의 배수인지를 나타낸다. 예를 들어서 0010 이 입력된다면 2 를 뜻하는 것이고 2 는 2 의 한 배수가 맞기 때문에 출력은 1, 또는 참, 이 될 것이다.

4) 멀티플렉서 (Multiplexer / MUX)

멀티플렉서는 여러개의 input 값들 중 하나를 골라 하나의 output 을 출력하도록 하는 회로이다. 입력값은 두 종류가 있는데 2^n개의 data input 과 n 개의 control input 이 존재한다.

5) Majority function

Majority function 은 홀수 개의 입력에 따라 1 이 더 많으면 1 을 출력하고 0 이 더 많으면 0 을 출력하는 함수이다. 이번 실험에서는 5 bit majority function 을 구현하는 것이기 때문에 5 개의 입력값을 비교해 각각의 다수를 출력할 것이다.

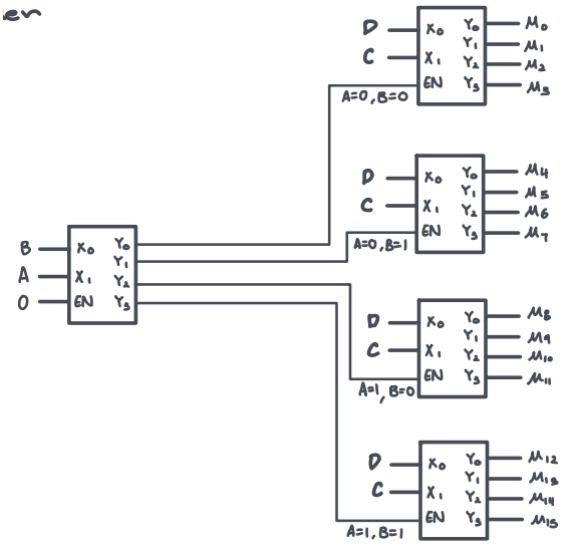
3. 실험 준비

1) 2-to-4 디코더로 4-to-16 디코더 구현:

왼쪽은 사진은 2-to-4 디코더를 이용했을 때의 진리표를 나타낸 것이다. 오른쪽 사진은 5 개의 2-to-4 디코더를 이용해 4-to-16 디코더를 구현했을 때의 회로도 모습이다.

2-4 Truth table

E	X_1	X_0	Y_0	Y_1	Y_2	Y_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0



2) 소수 판별기와 배수 검출기:

밑의 사진은 소수 판별기와 배수 검출기를 나타낸 진리표 이다. 이 진리표를 이용해 각 출력에 따라 K-map 을 이용해 식을 단순화 했다.

#	I_3	I_2	I_1	I_0	Y_4 (11)	Y_3 (7)	Y_2 (5)	Y_1 (3)	Y_0 (2)	Prime
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0	1	1
3	0	0	1	1	0	0	0	1	0	1
4	0	1	0	0	0	0	0	0	1	0
5	0	1	0	1	0	0	1	0	0	1
6	0	1	1	0	0	0	0	1	1	0
7	0	1	1	1	0	1	0	0	0	1
8	1	0	0	0	0	0	0	0	1	0
9	1	0	0	1	0	0	0	1	0	0
10	1	0	1	0	0	0	1	0	1	0
11	1	0	1	1	1	0	0	0	0	1
12	1	1	0	0	0	0	0	1	1	0
13	1	1	0	1	0	0	0	0	0	1
14	1	1	1	0	0	1	0	0	1	0
15	1	1	1	1	0	0	1	1	0	0

$Y_0 = \Sigma M\,(2,4,6,8,10,12,14)$

$Y_1 = \Sigma m\,(3,6,9,12,15)$

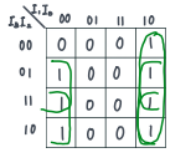
$Y_2 = \Sigma m\,(5,10,15)$

$Y_3 = \Sigma m\,(7,14)$

$Y_4 = \Sigma m\,(11)$

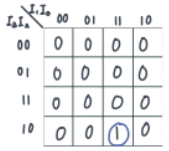
$Prime = \Sigma_1 m\,(2,3,5,7,11,13)$

$Y_0 : \Sigma M\,(2,4,6,8,10,12,14)$



→ $I_1I_0' + I_3I_0' + I_2I_0'$

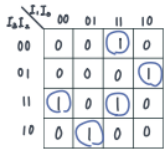
$Y_4 = \Sigma m\,(11)$



→ No further simplification available using k-map.

→ $I_3I_2'I_1I_0$

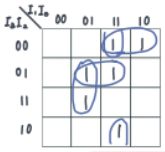
$Y_1 : \Sigma M\,(3,6,9,12,15)$



→ No further simplification available using k-map.

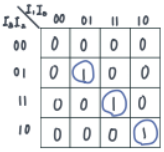
→ $I_3'I_2'I_1I_0 + I_3'I_2I_1I_0' + I_3I_2'I_1'I_0 + I_3I_2I_1'I_0 + I_3I_2I_1I_0$

$Prime = \Sigma m\,(2,3,5,7,11,13)$



→ $I_3'I_2'I_1 + I_3'I_2I_0 + I_2I_1'I_0 + I_2'I_1I_0$

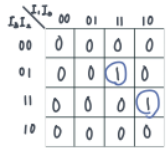
$Y_2 = \Sigma M\,(5,10,15)$



→ No further simplification available using k-map.

→ $I_3'I_2I_1'I_0 + I_3I_2'I_1I_0' + I_3I_2I_1I_0$

$Y_3 = \Sigma M\,(7,14)$

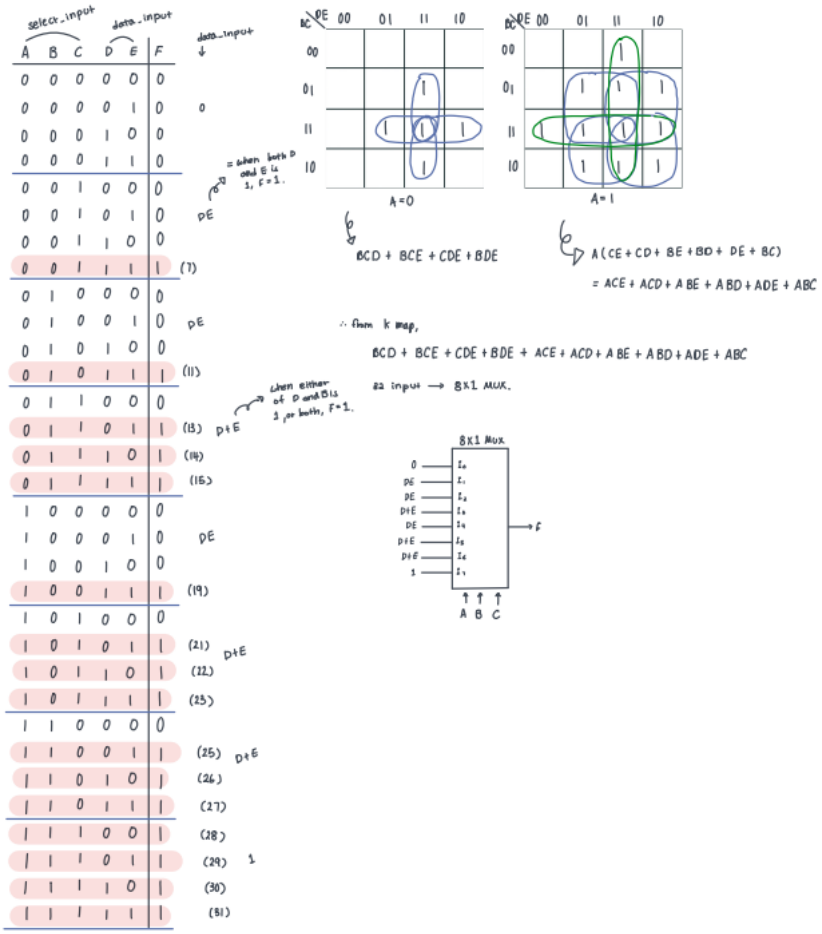


→ No further simplification available using k-map.

→ $I_3'I_2I_1I_0 + I_3I_2I_1I_0'$

3) 5 비트 majority function:

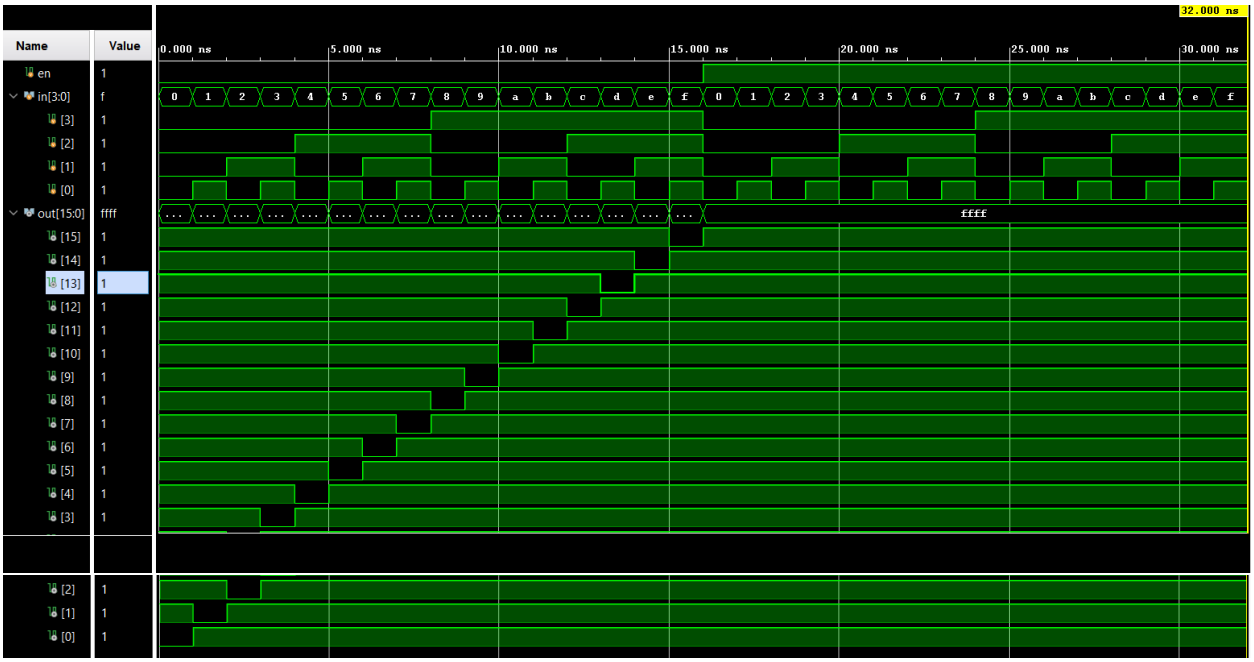
밑의 그림에서 왼쪽은 5 비트 majority function 의 진리표를 나타낸 것이고, 출력값이 1 인 것들의 합을 SOP 꼴로 만들었다. 5bit 가 입력 되면 총 경우의 수는 32 개가 되기 때문에 4 개씩 묶어서 8 개의 data input 으로 나타내고 3 개의 control input 을 가지는 8:1 멀티플렉서로 표현했다. 8:1 멀티플렉서는 그림에서 우측에 있는 K-map 밑에 표현되어있다.



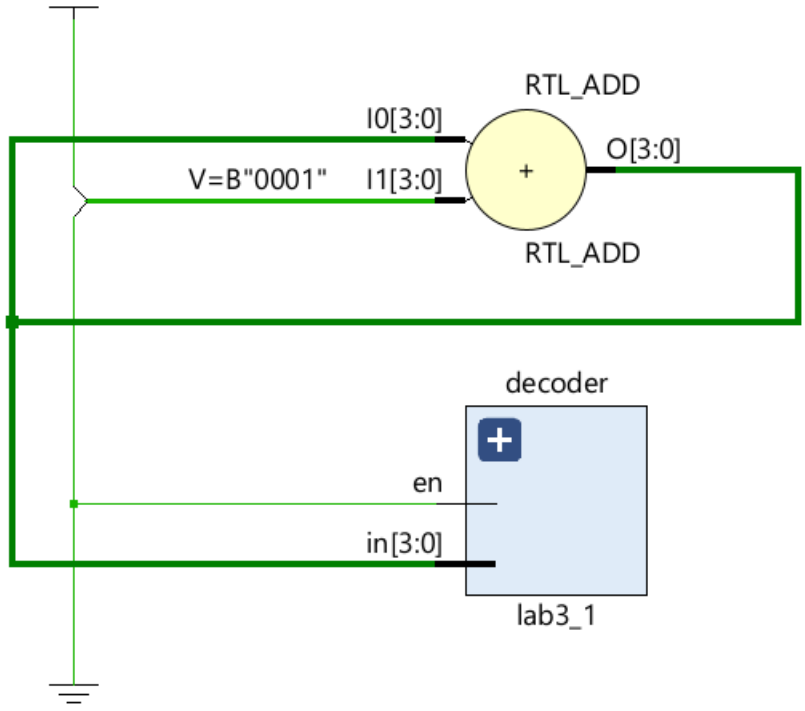
4. 결과

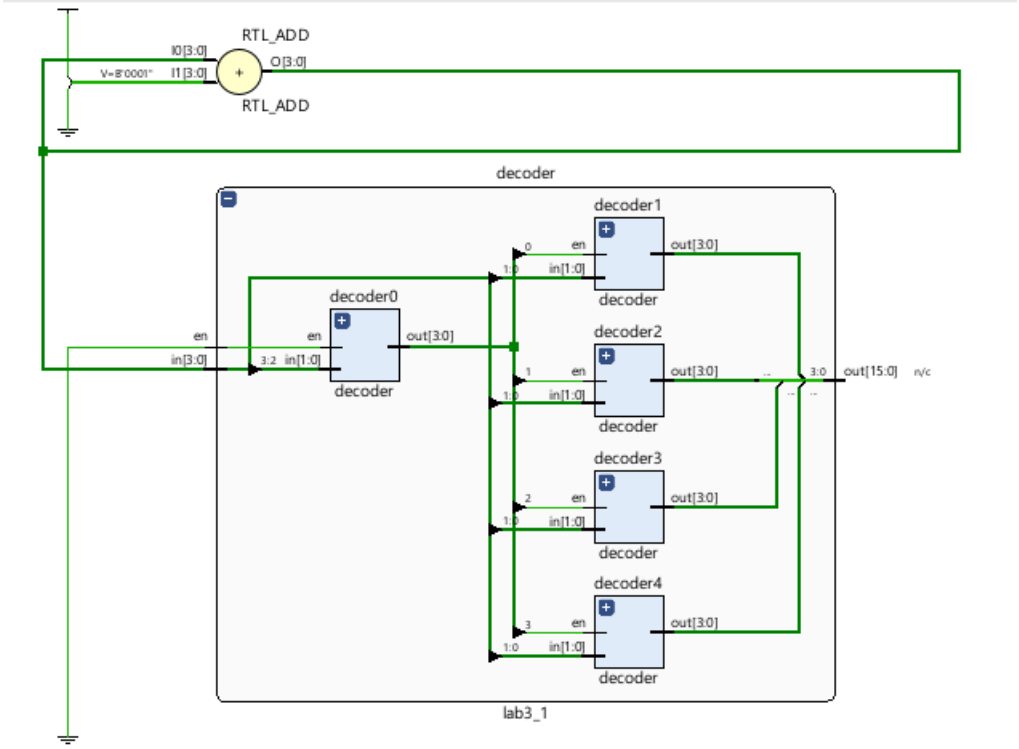
1) 4-to-16 decoder (lab3_1):

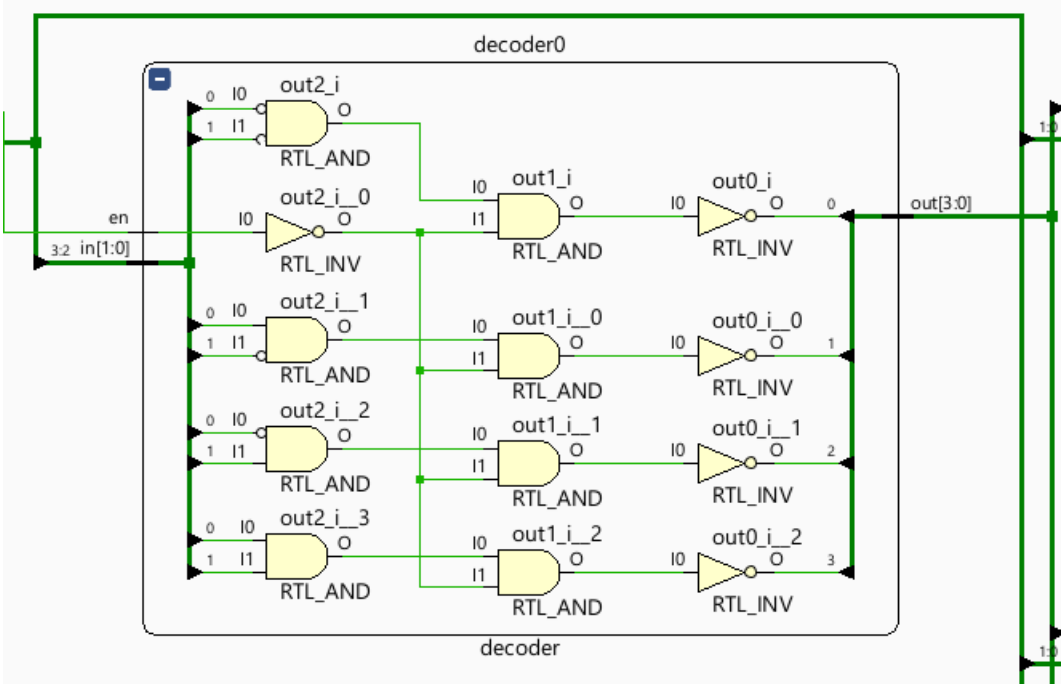
Lab 3_1 에서는 2-to-4 active low 디코더를 이용해 4-to-16 디코더를 구현했다. 실험 준비에서 첨부한 디코더의 회로도를 바탕으로 enable 과 output 은 active low, 그리고 active high 로 설정해서 구현하였다.



제공된 테스트벤치를 이용해 시뮬레이션을 실행 한 파형을 캡쳐한 것이다. 실험준비에서 나타낸 것과 같이 EN 이 1 일때는 모든 출력값이 1 이고 그 외에는 active low output 에 맞게 오름차순으로 0 인 출력이 나타나는 것을 확인할 수 있다.



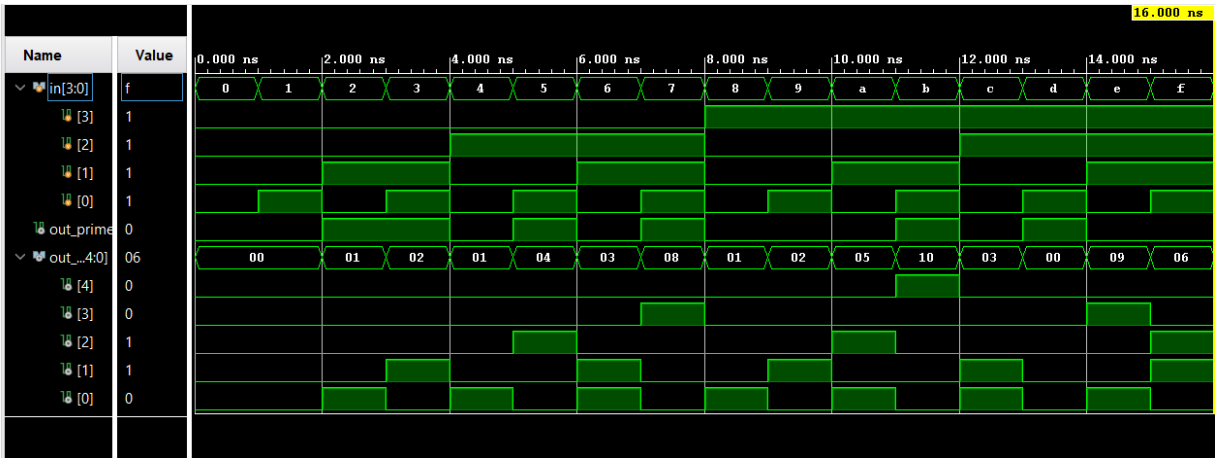




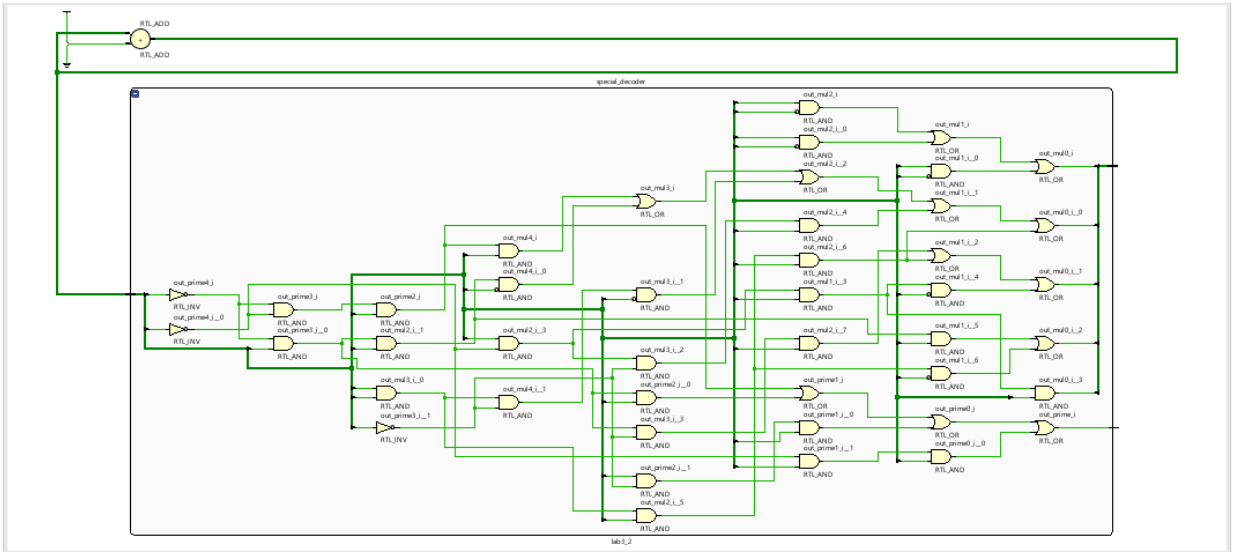
위의 사진들은 schematic 을 이용해 나타낸 회로도이다. 두번째 사진은 전체적인 4-to-16 디코더의 모습을 나타내고 세번째 사진은 각각의 2-to-4 디코더의 회로도를 나타낸 것이다.

2) 소수판별기/ 배수검출기

실험 준비에서 나타낸 것 과 같이 4 bit input 을 소수와 배수를 나타내는 진리표를 이용해 디코더로 회로를 구현했다.



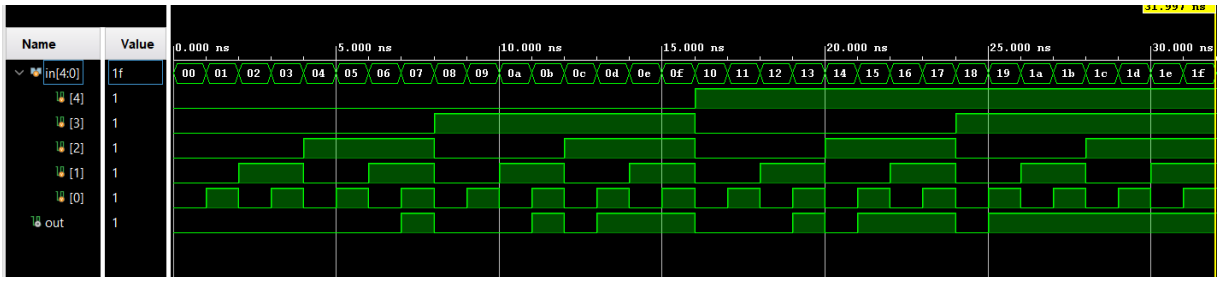
제공된 테스트 벤치를 이용해서 시뮬레이션을 돌린 후의 파형을 캡쳐한 것이다. 출력값 5~0 이 각각 11, 7, 5, 3, 2 의 배수일 때, 1 의 결과를 나타낸다는 것을 알 수 있다. 그리고 out_prime 은 소수 (2, 3, 4, 7, 11, 13) 일 때, 1 의 결과를 나타낸다는 것을 알 수 있다.



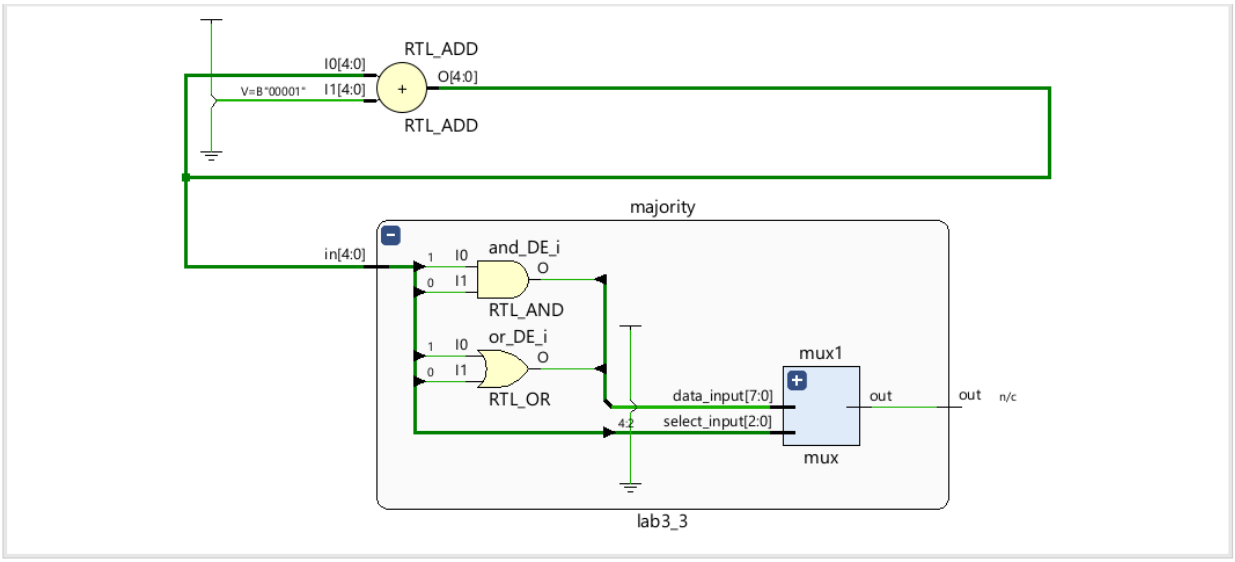
위는 schematic 을 이용해 회로도를 구현한 모습이다.

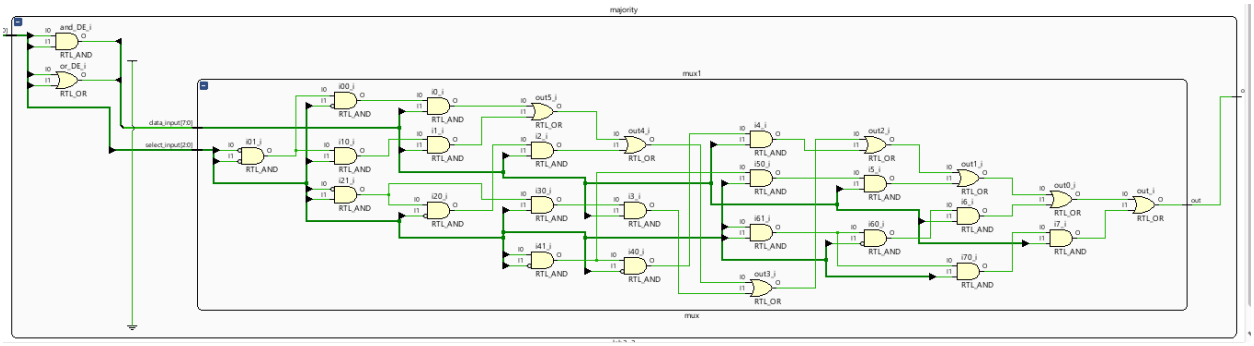
3) 5-bit majority function

실험 준비에서 언급한 SOP 식과 8:1 mux 를 이용해 majority function 을 구현했다 .



위는 제공된 테스트벤치를 이용해 시뮬레이션을 실행시킨 후의 파형을 캡쳐한 것이다. 파형에서 1 의 출력 값을 갖는 input 들을 확인해보면 실험 준비에서 나타낸 진리표와 같다는 것을 알 수 있다.





위는 schematic 기능을 이용해 그려낸 회로도이다.

5. 논의

이번 실험을 통해 decoder 와 MUX 의 기능에 대해 상기할 수 있었다. Decoder 를 사용할 때 input 값을 넣는 순서와 output 값의 순서를 정확히 알지 못해 실험 준비에서 시간이 비교적 오래 걸렸지만 이런 시행착오를 겪으며 decoder 의 작동을 이해할 수 있었다.

이번 실험 이전에는 모두 4 bit 보다 작은 input 값을 가지거나 5 bit 이상의 input 값을 가지면 K-map 이 아니라 Q-M 방법을 사용했었어서 5bit input 을 K-map 으로 구현해 낼 때 3 차원으로 확인해야 한다는 점이 어려웠다. 다행히 A=0 와 A=1 사이의 PI 가 존재하지 않았지만 만약 존재할 경우를 대비해 3 차원에서의 K-map 계산 방식을 알아두는 것도 좋을 것 같다.