Lab3. 디코더와 멀티플렉서

학번 : 20210273 이름 : 하태혁

1. 개요

Multiple output 회로를 다룰 줄 알며 그 기능을 수행하는 디코더와 멀티플렉서를 이해하는 것을 목표로 한다. Lab3_1에서 사용한 Active low 디코더의 작동을 이해하고 더 많은 Input line과 output line을 가질 수 있도록 디코더의 확장을 구현한다. 특수한 기능을 실행하는 디코더를 구현하기 위해 진리표를 그리고 output을 연결해 구현한다. 멀티플렉서에 대해서도 특수한 기능을 구현하기 위해 진리표를 그리고 연결해 구현한다.

2. 이론적 배경

디코더는 n개의 2진 코드 입력에 대해 최대 2^n개의 출력이 나오게 하는 기능을 가진 회로이다. Enable input을 사용해 디코더를 켜거나 끌 수 있고 Enable input과 output에 not gate를 붙여서 디코더의 종류를 나누기도 한다. Activate high 디코더의 경우 minterm generator라고도 불리고 연결해서 쓰면 SOP로 사용할 수 있다. Activate low 디코더의 경우 AND-gate를 사용하면 POS를 구현할 수 있고 NAND gate를 사용하면 SOP를 구현할 수 있다.

멀티플렉서는 여러 입력 중 하나를 선택하여 선택된 입력을 출력하는 장치이다. n개의 control input에 대해 2^n개의 input이 존재하고 그중 하나의 input을 선택하여 출력하는 기능을 한다.

3. 실험 준비

1) lab3_1

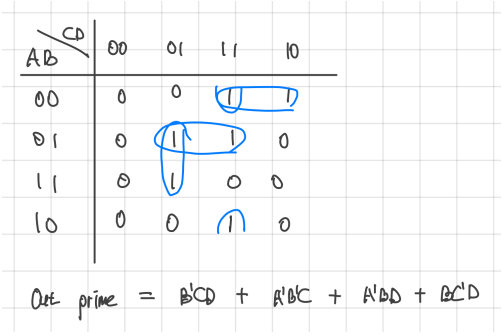
2-to-4 Active low enable, Active-low output, Active-high input 디코더로 동일한 조건의 4-to-16 디코더를 구현하기 위해서 우선 5개의 디코더가 필요하다. In[3], in[2]를 입력값으로 받고 네 개의 output line을 다른 4개의 디코더의 EN에 연결한다. 그리고 나머지 4개 디코더의 input은 in[1], in[0]으로 연결한다.

2) lab3_2

In[3] = A	In[2] = B	In[1] = C	In[0] = D	Out_prime	Out_mul[4]	Out_mul[3]	Out_mul[2]	Out_mul[1]	Out_mul[0]
0	0	0	0	0	1	1	1	1	1
0	0	0	1	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	1
0	0	1	1	1	0	0	0	1	0
0	1	0	0	0	0	0	0	0	1
0	1	0	1	1	0	0	1	0	0
0	1	1	0	0	0	0	0	1	1
0	1	1	1	1	0	1	0	0	0
1	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	1	0
1	0	1	0	0	0	0	1	0	1

1	0	1	1	1	1	0	0	0	0
1	1	0	0	0	0	0	0	1	1
1	1	0	1	1	0	0	0	0	0
1	1	1	0	0	0	1	0	0	1
1	1	1	1	0	0	0	1	1	0

Out_prime에 대한 K-map을 그리면 다음과 같다.



마찬가지로 out-mul에 대해서도 boolian expresion을 찾으면

Out_mul[4] = A'B'C'D' + AB'CD

Out_mul[3] = A'B'C'D' + A'BCD + ABCD'

Out_mul[2] = A'B'C'D' + AB'CD' + A'BC'D + ABCD

Out_mul[1] = A'B'C'D' + ABC'D' + AB'C'D + A'B'CD + A'BCD' + ABCD

Out_mul[0] = D'

3) lab3_3

In[4] = A	In[3] = B	In[2] = C	In[1] = D	In[0] = E	Out	
0	0	0	0	0	0	0
0	0	0	0	1	0	
0	0	0	1	0	0	
0	0	0	1	1	0	
0	0	1	0	0	0	DE
0	0	1	0	1	0	
0	0	1	1	0	0	
0	0	1	1	1	1	
0	1	0	0	0	0	DE
0	1	0	0	1	0	
0	1	0	1	0	0	
0	1	0	1	1	1	
0	1	1	0	0	0	D+E
0	1	1	0	1	1	

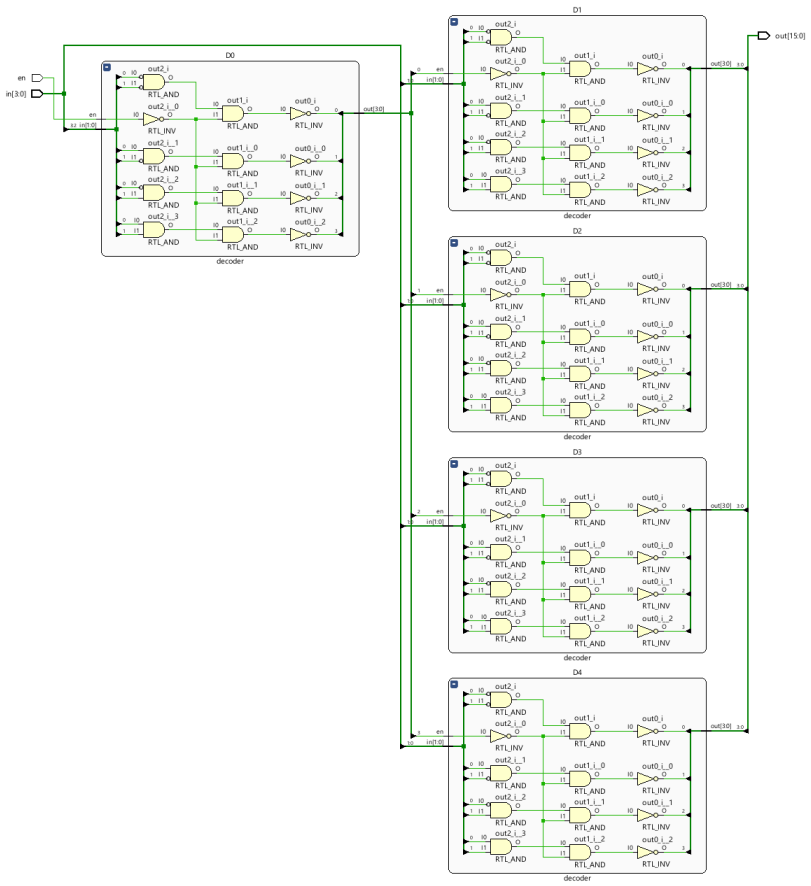
0	1	1	1	0	1	
0	1	1	1	1	1	
1	0	0	0	0	0	DE
1	0	0	0	1	0	
1	0	0	1	0	0	
1	0	0	1	1	1	
1	0	1	0	0	0	D+E
1	0	1	0	1	1	
1	0	1	1	0	1	
1	0	1	1	1	1	
1	1	0	0	0	0	D+E
1	1	0	0	1	1	
1	1	0	1	0	1	
1	1	0	1	1	1	
1	1	1	0	0	1	1
1	1	1	0	1	1	
1	1	1	1	0	1	
1	1	1	1	1	1	

Out = A′B′CDE + A′BC′DE + A′BCD′E + A′BCDE′ + A′BCDE + AB′C′DE + AB′CD′E + AB′CDE′ + AB′CDE + ABC′D′E + ABC′DE′ + ABC′DE + ABCD′E′ + ABCD′E + ABCDE′ + ABCDE

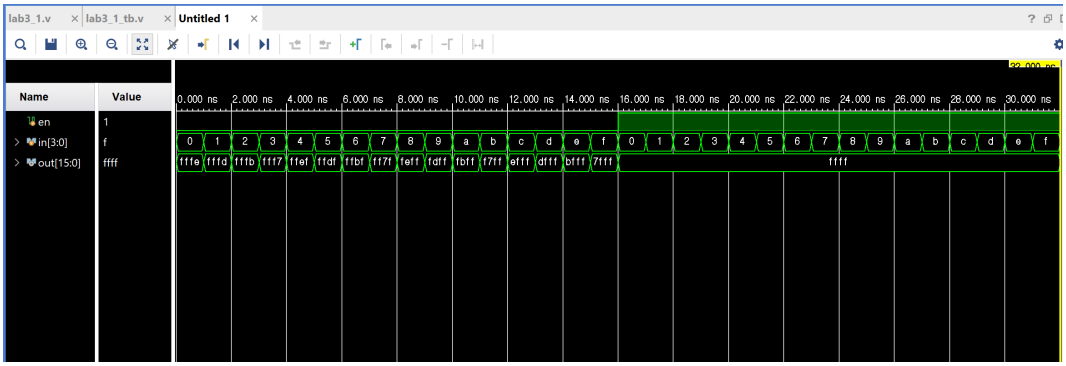
Out = A′B′CDE + A′BC′DE + A′BC(D+E) + AB′C′DE + AB′C(D+E) + ABC′(D+E) + ABC

4) 결과

1) lab3_1

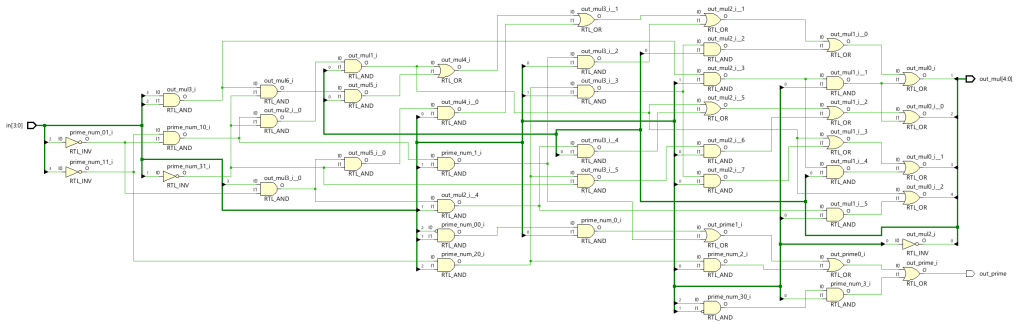


2-to-4 Active-low 디코더 모듈을 이용해 4-to-16 Active-low 디코더를 구현하면 다음과 같다. 디코더의 EN을 끄고 켜는 기능을 이용해 디코더를 확장했으며 모든 입출력은 Little-endian 형식으로 구현했다.

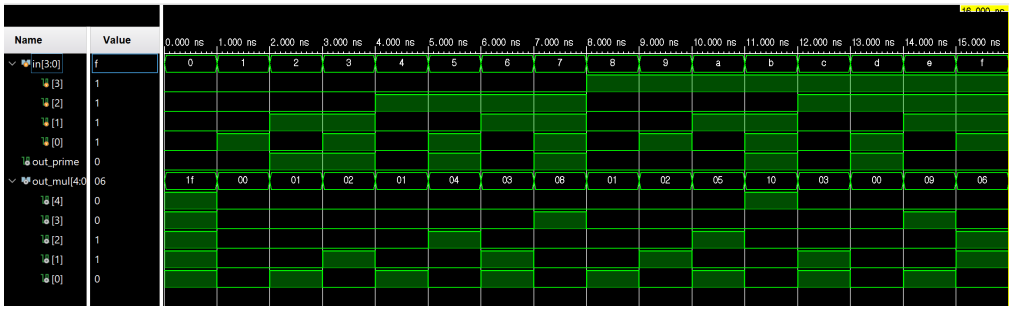


테스트벤치로 시뮬레이션을 실행한 디코더의 동작은 위 사진과 같다.

2) lab3_2

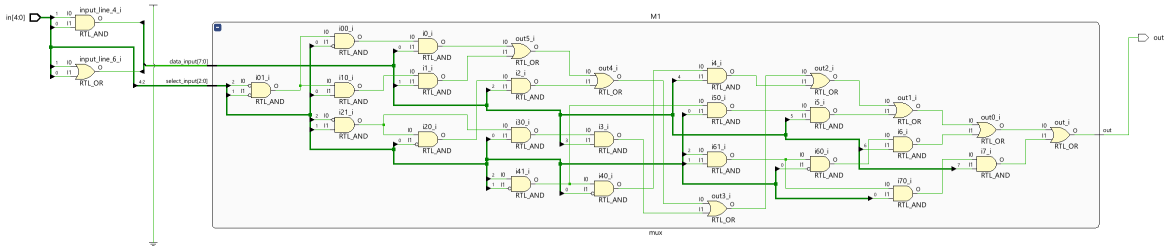


소수 판별기와 배수 검출기에 대한 Truth table을 그린 뒤, K-map을 그려 각각의 output에 대한 minimum boolian expresion을 찾고 assign keyword와 비교연산자를 사용하여 구현하였다.

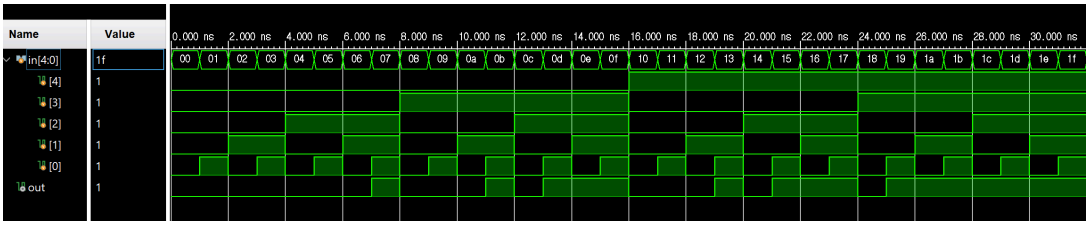


주어진 테스트벤치로 시뮬레이션을 돌렸을 때의 결과는 위와 같고 예상과 일치함을 알 수 있다.

3) lab3_3



5bit Majority function의 진리표를 구하고 8:1 mux를 사용해 구현해야 하므로 2개의 control input을 data input으로 사용하였다. In[1]과 in[0]를 data input으로 사용했을 때 필요한 논리연산자 등을 진리표에 나타냈고 assign 을 사용해 각 mux의 input line에 연결하여 구현했다.



결과는 다음과 같다. 진리표에 나타냈던 결과와 일치함을 알 수 있다.

5) 논의

디코더와 멀티플렉서의 기능을 이해하고 구현해 볼 수 있었다. 또한 기존에 실습했던 K-map 등의 개념도 꾸준히 사용하면서 문제를 해결했기 때문에, 다른 lab과제보다 오랜 시간이 걸렸지만, 베릴로그 문법이나 회로 구성을 더 이해할 수 있었다. Lab3_2에서는 진리표의 값만 보고 각 minterm을 wire에 assign하는 방식을 사용했다가 K-map으로 minimum boolian expresion을 찾은 뒤, wire에 assign하는 방식을 사용하였다.