Lab3. 디코더와 멀티플렉서

20200437 김채현

1. 개요

본 실험은 decoder와 multiplexer의 기능을 이해함으로써 Multiple-output 회로를 이해하고 구현한다. lab3_1로 기본적인 2-to-4 decoder 4개를 이용하여 4-to-16 decoder를 구현한다. 회로를 구상하고 simulation으로 결과를 확인한다. lab3_2로 특수 목적 decoder인 소수 판별기와 11, 7, 5, 3, 2의 배수 검출기를 구현한다. 각각 4-bit input이 들어오며 회로를 구상하고 simulation으로 결과를 확인한다. 마지막으로 multiplexer를 이용하여 5-bit Majority function을 구현하고 결과를 확인한다.

2. 이론적 배경

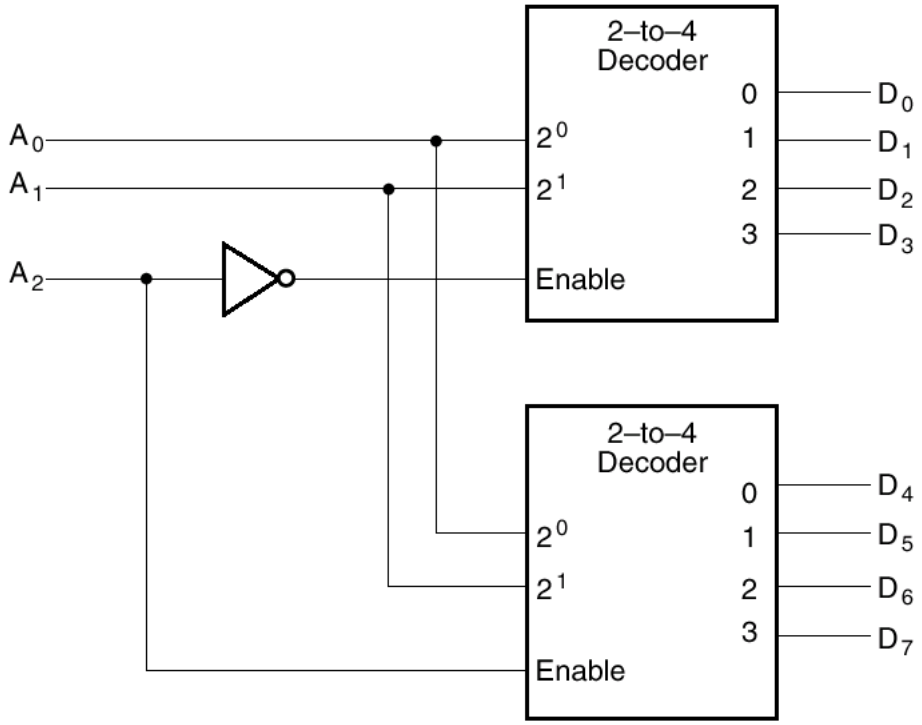
1) 디코더 (Decoder / De-Multiplexer)

디코더란 어떤 부호화된 형으로부터 다른 형으로 바꾸기 위한 회로와 장치를 의미한다. 디코더는 n개의 binary 입력을 받아 2^n개의 고유의 출력을 내는 회로이다. 각 출력이 곧 minterm이기 때문에 minterm generator라고도 불린다. 추가로 디코더에는 Enable 입력이 존재하는데 EN 값이 1일 때 디코더가 on 상태이고, 0일 때 디코더가 off 상태이다. K-to-2^n 디코더도 존재하는데, 이는 2^n 개의 출력 중에서 k개를 logically true로 출력한다는 의미이다. 디코더에는 Active-high decoder와 Active-low decoder가 존재하는데, Active-high란 회로에서 1을 high voltage로, 0을 low voltage로 나타내는 것으로 positive logic과 비슷하다. 반대로 Active-low란 회로에서 1을 low voltage로, 0을 high voltage로 나타내는 것으로 negative logic과 비슷하다.

2) 디코더의 확장 (Decoder Expansion)

디코더는 EN값을 활용하여 자리 수가 더 큰 디코더로 확장할 수 있다. 예컨대 2-to-4 디코더를 이용하여 3-to-8 디코더를, 3-to-8 디코더를 이용하여 5-to-32 디코더를 만들 수 있다. 이를 디코더 확장(Decoder Expansion)이라고 부른다. 아래

그림은 2-to-4 디코더 3개를 이용하여 3-to-8 디코더로 확장한 것이다.

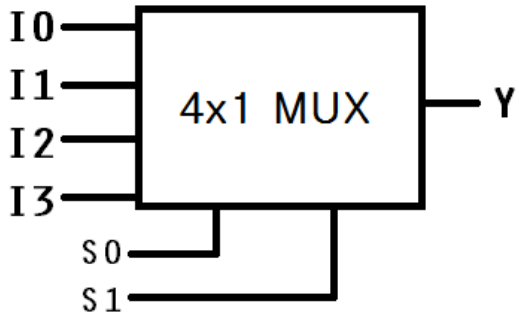


3) 특수 목적 디코더 (Special-Purposed Decoder)

디코더 중에는 특수한 목적을 가지고 작동하는 디코더가 존재하는데 이를 특수목적 디코더(Special-Purposed Decoder)라고 부른다. 특수 목적 디코더에는 7-Segment Decoder, Prime Number Indicator, Multiplier 등이 있다. 본 실험에서는 주어진 입력이 소수일 때 참을 출력하는 소수 판별기(Prime Number Indicator)와 주어진 입력이 배수인지를 출력하는 배수 검출기(Multiplier)를 구현하였다.

4) 멀티플렉서 (Multiplexer / MUX)

멀티플렉서는 여러 개의 input 중에 하나의 input만을 output으로 출력하는 회로이다. 일반적으로 멀티플렉서는 2^n개의 input, n개의 input address와 1개의 output을 가지며 n개의 input address가 가리키는 data input을 output으로 출력한다.



5) Majority / Minority function

Majority function이란 홀수 개의 input이 주어졌을 때, 0의 개수와 1의 개수를 비교하여 많은 쪽을 출력하는 함수이다. 이는 multiplexer를 통해 간단하게 구현될 수 있다.

6) 멀티플렉서를 사용한 함수 표현

위와 같은 n이 2인 멀티플렉서를 생각해보았을 때, S0과 S1을 이용해 I0, I1, I2, I3 중 하나의 입력을 선택한다. 예컨대, 위의 멀티플렉서를 함수로 표현하면 Y = S0'S1'I0 + S0S1'I1 + S0'S1I2 + S0S1I3이 된다. 이와 같이 입력에 적절한 값을 대입하면 원하는 함수를 쉽게 구현할 수 있다.

3. 실험 준비

1) 4-to16 Active-low 디코더

디코더 확장을 이용해 2-to-4 decoder를 이용하여 4-to-16 decoder를 구현할 수 있다. 그 방법은 처음 decoder로 나온 결과를 다른 decoder의 EN input으로 넣으면 된다. 이처럼 디코더 확장을 해주고 회로로 표현하기 위해 schematic을 해주면 다음과 같다. 주의해야할 점은 본 실험의 경우 Active-low decoder를 이용하기 때문에 이를 고려하여 실험을 진행해주어야 하는데, 대신 확장된 4-to-16 decoder도 Active-low decoder가 되게 구현해주어야 하므로 일반적인 decoder expansion 방법 같이 연결해주면 된다.

2) 4비트 소수 판별기 및 배수 검출기

(1) 4비트 소수 판별기

4비트 소수 판별기의 진리표를 작성하면 다음과 같다.

CD / AB	00	01	11	10
00	0	0	1	1
01	0	1	1	0
11	0	1	0	1
10	0	0	1	0

K-Map을 이용하여 단순화를 하면 F=A'CD+A'B'C+B'CD+BC'D이다.

(2) 배수 검출기

① 2배수 검출기

CD / AB	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	1	0	0	1
10	1	0	0	1

K-Map을 이용하여 단순화를 하면 F=D'이다.

② 3배수 검출기

CD / AB	00	01	11	10
00	0	0	1	0
01	0	0	0	1

11	1	0	1	0
10	0	1	0	0

K-Map을 이용하여 단순화를 하면 F=A′B′CD+A′BCD′+ABC′D′+ABCD+AB′C′D이다.

③ 5배수 검출기

CD AB	**00**	**01**	**11**	**10**
00	0	0	0	0
01	0	1	0	0
11	0	0	1	0
10	0	0	0	1

K-Map을 이용하여 단순화를 하면 F=A′BC′D+ABCD+AB′CD′이다.

④ 7배수 검출기

CD AB	**00**	**01**	**11**	**10**
00	0	0	0	0
01	0	0	1	0
11	0	0	0	1
10	0	0	0	0

K-Map을 이용하여 단순화를 하면 F=A′BCD+ABCD′이다.

⑤ 11배수 검출기

CD AB	**00**	**01**	**11**	**10**
00	0	0	0	0

01	0	0	0	0
11	0	0	0	0
10	0	0	1	0

K-Map을 이용하여 단순화를 하면 F=AB'CD이다.

3) 5비트 Majority function

다음은 5비트 Majority function의 진리표와 K-Map을 나타낸 것이다.

A	B	C	D	E	OUTPUT	F
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

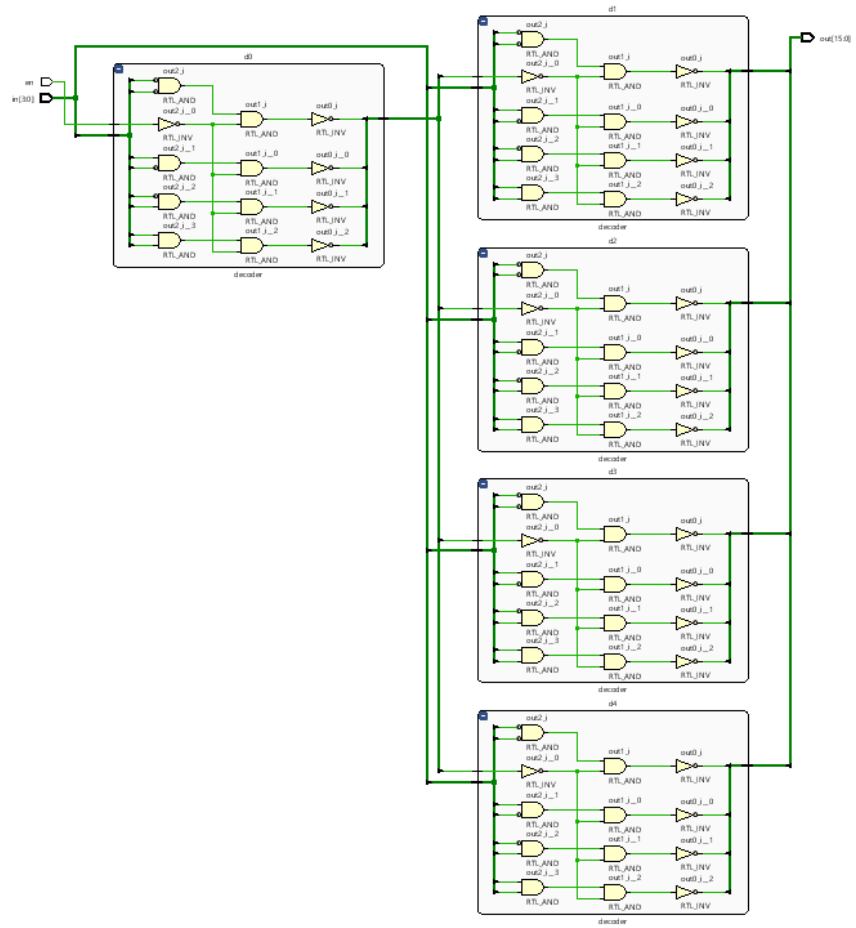
1	0	1	0	1	1	
1	0	1	1	0	1	
1	0	1	1	1	1	
1	1	0	0	0	0	D+E
1	1	0	0	1	1	
1	1	0	1	0	1	
1	1	0	1	1	1	
1	1	1	0	0	1	1
1	1	1	0	1	1	
1	1	1	1	0	1	
1	1	1	1	1	1	

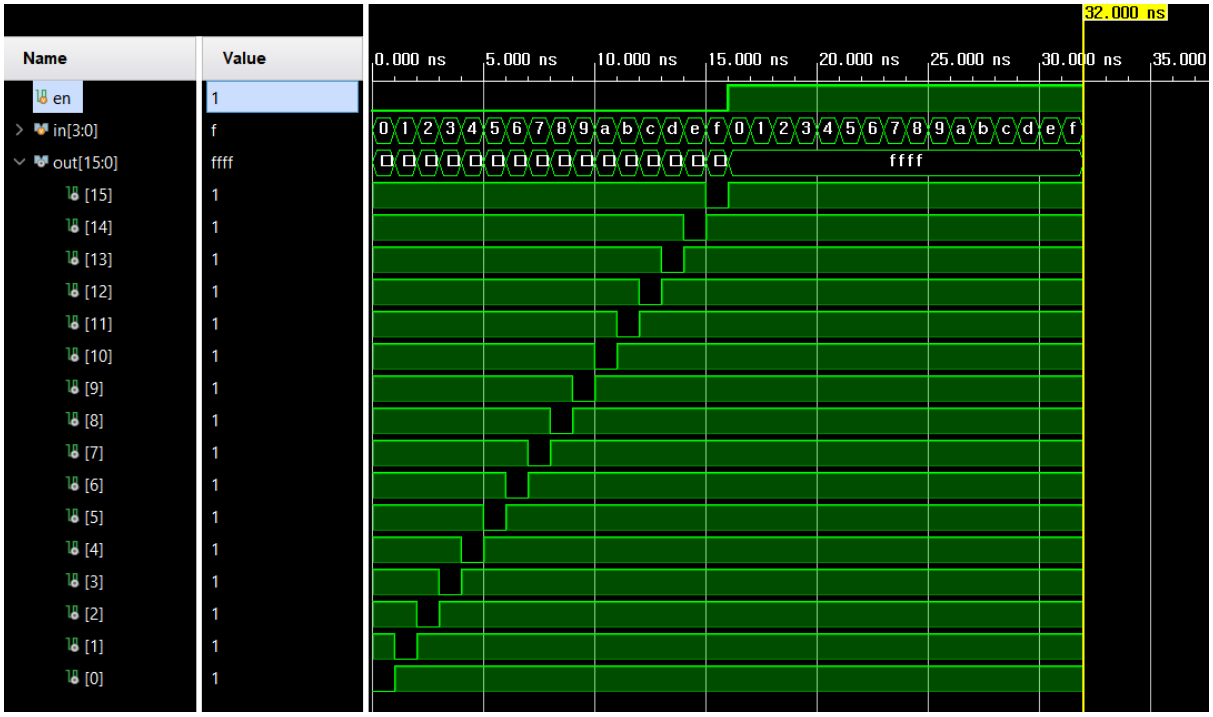
A, B, C에 따라 data input 8개가 정해지고 data input은 D, E를 통해 표현된다. K-Map을 바탕으로 MUX에는 Data input D, E와 이를 구분해야 할 Selection input으로 A, B, C가 들어간다.

4. 결과

1) 4-to-16 Active-low 디코더

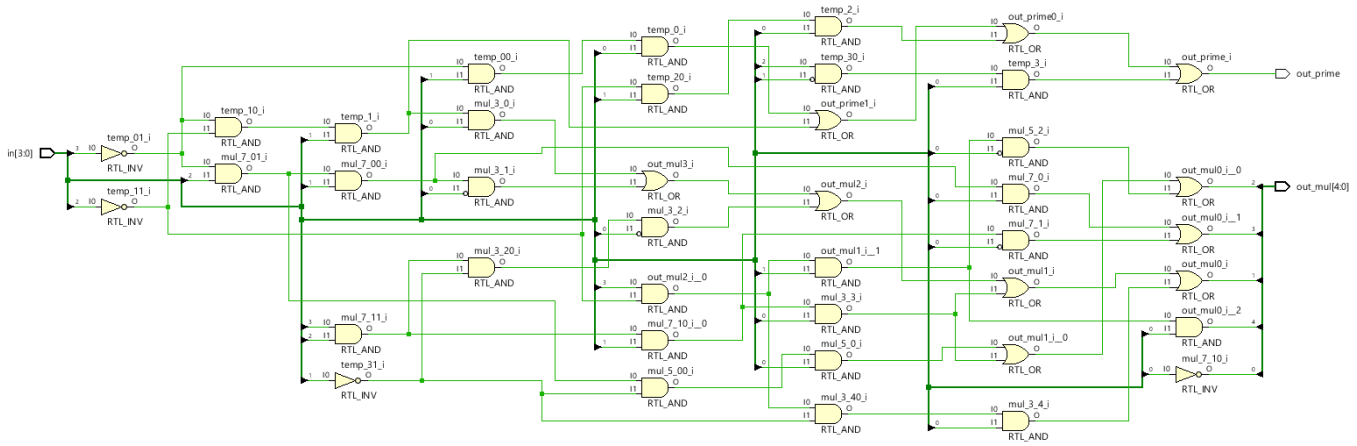
2-to-4 decoder 다섯 개를 이용하여 4-to-16 decoder를 구현하였다. schematic은 다음과 같고, testbench를 이용해 simulation을 하면 다음과 같이 잘 작동함을 볼 수 있다.

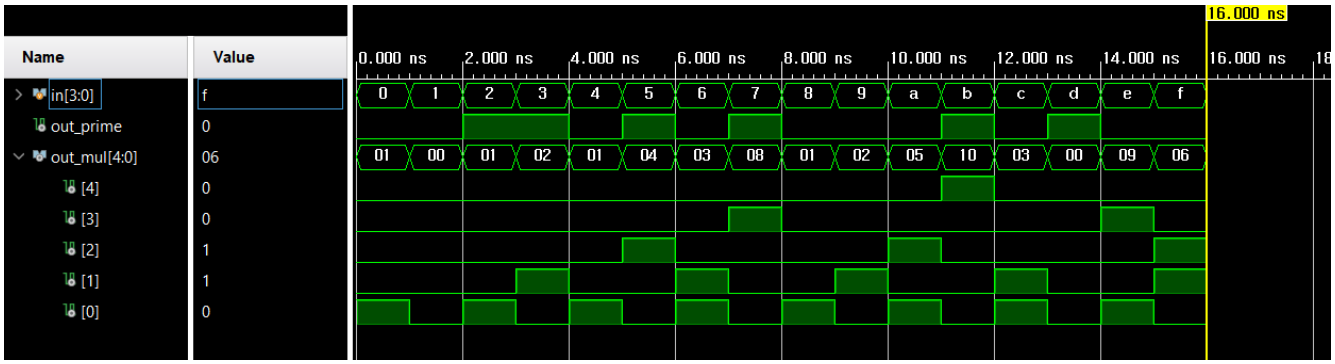



Name
Value
en
1
in[3:0]
f
out[15:0]
ffff
[15]
[14]
[13]
[12]
[11]
[10]
[9]
[8]
[7]
[6]
[5]
[4]
[3]
[2]
[1]
[0]
0 1 2 3 4 5 6 7 8 9 a b c d e f 0 1 2 3 4 5 6 7 8 9 a b c d e f
ffff
0.000 ns
5.000 ns
10.000 ns
15.000 ns
20.000 ns
25.000 ns
30.000 ns
35.000
32.000 ns

2) 4비트 소수 판별기 및 배수 검출기

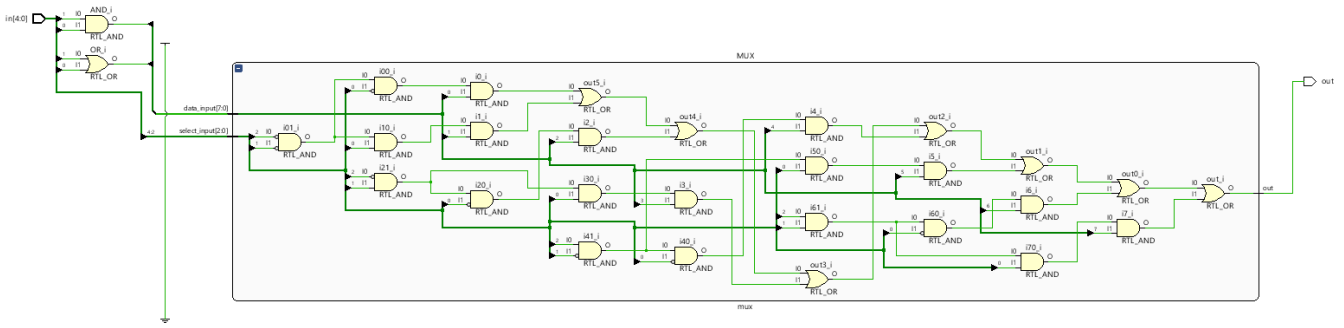
testbench를 통해 simulation을 하면 2, 3, 5, 7, 11, 13의 소수를 잘 판별하고, 2, 3, 5, 7, 11의 배수를 잘 판별함을 볼 수 있다. schematic은 다음과 같다.

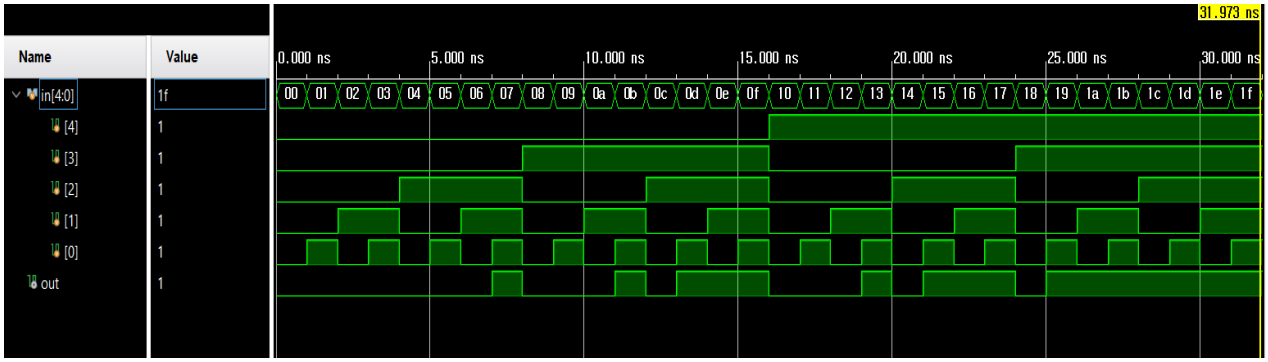




3) 5비트 Majority function

testbench를 통해 simulation을 하면 잘 작동함을 알 수 있다. schematic은 다음과 같다.





5. 논의

본 실험을 통해 수업시간에 배운 decoder와 multiplexer를 직접 구현해보며, 그 용도와 개념을 잘 이해할 수 있었다. decoder expansion을 직접 실험해보고, 특수 목적 decoder를 구현해보며 decoder의 필요성과 그 응용에 대해 배울 수 있었다. 또한 현재 컴퓨터구조라는 타 과목을 들으며 multiplexer를 CPU 구현에 사용하는 일이 많았는데, multiplexer의 중요성과 필요성을 잘 알기 때문에 실험에 더 집중할 수 있었다.