Lab 3. 디코더와 멀티플렉서

Digital System Design (CSED273)

20220100 박기현

1. 개요

이번 실험은 Multiple-output 회로를 대표하는 디코더(Decoder)와 Multiple-input 회로를 대표하는 멀티플렉서(Multiplexer)의 기능을 이해하고, 이를 사용하여 verilog 로 회로를 구성하는 것이다. 세부적으로는 Active-low 디코더 확장 이해 및 구현, 특수 목적 디코더 구현, 멀티플렉서의 데이터 선택 기능을 활용한 Majority function 구현을 학습하는 것을 목적으로 한다.

2. 이론적 배경

1) Active-high / Active-low

Active-high 는 입력값이 High 일 때 동작하는 것을 의미하며, 반대로 Active-low 는 입력값이 Low 일 때 동작하는 것을 의미한다.

2) 디코더(Decoder)

디코더는 n 개의 이진 정보 입력에서 최대 2^n 개의 고유 출력으로 변환하는 다중 입력, 다중 출력 회로이다. n 개의 이진 입력과 2^n 개의 서로 다른 출력을 가지는 경우, 각 출력이 minterm 을 의미하기 때문에 minterm generator 라고도 부른다. 따라서 디코더로 임의의 함수를 구현할 수 있는데, 함수에 대한 최소항의 합이 있는 경우, 디코더와 OR 게이트를 통해 구현 가능하다.

실제 디코더 소자에는 추가적으로 EN(Enable input)이 존재한다. EN 은 디코더를 활성화하거나 비활성화하는 데 사용된다.

디코더의 입력과 출력의 특성을 표현할 때는 n-to-2^n 과 k-to-2^n 이라는 표현을 사용한다. 이는 직관적으로 해석할 수 있는데, n-to-2^n 은 n 개의 입력과 2^n 개의 출력을 가진다는 것을 의미하고, k-to-2^n 은 k 개의 입력과 2^n 개의 출력을 가진다는 것을 의미한다.

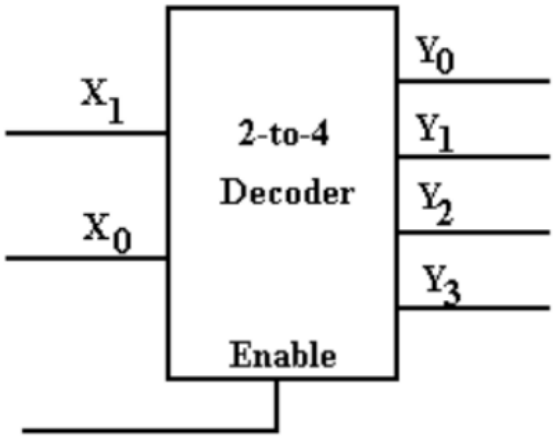


그림 1 2-to-4 디코더

3) 디코더 확장(Decoder Expansion)

EN 을 사용하여 디코더를 활성화하거나 비활성화할 수 있다는 점을 이용하여 여러 개의 디코더를 연결해 더 많은 수의 입력을 받을 수 있다. 이를 디코더 확장이라고 한다.

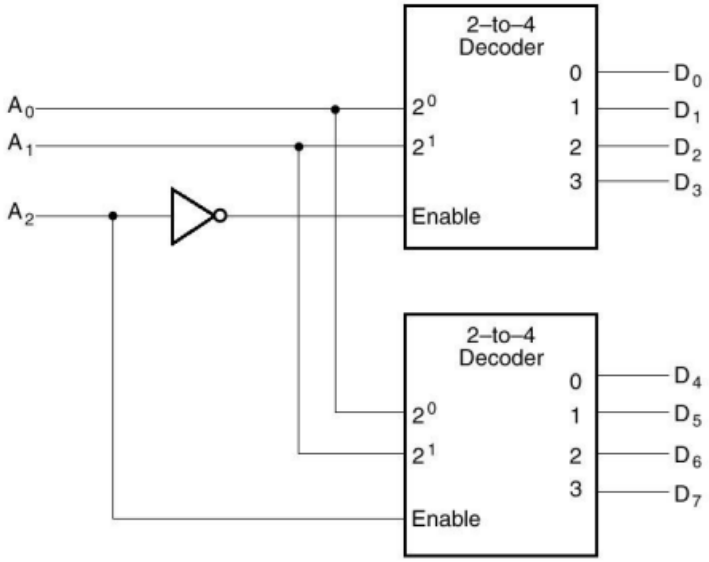


그림 2 2-to-4 디코더에서 3-to-8 디코더로의 확장

위 그림 2 는 2-to-4 디코더 2 개를 연결하여 3-to-8 디코더로 확장한 것이다.

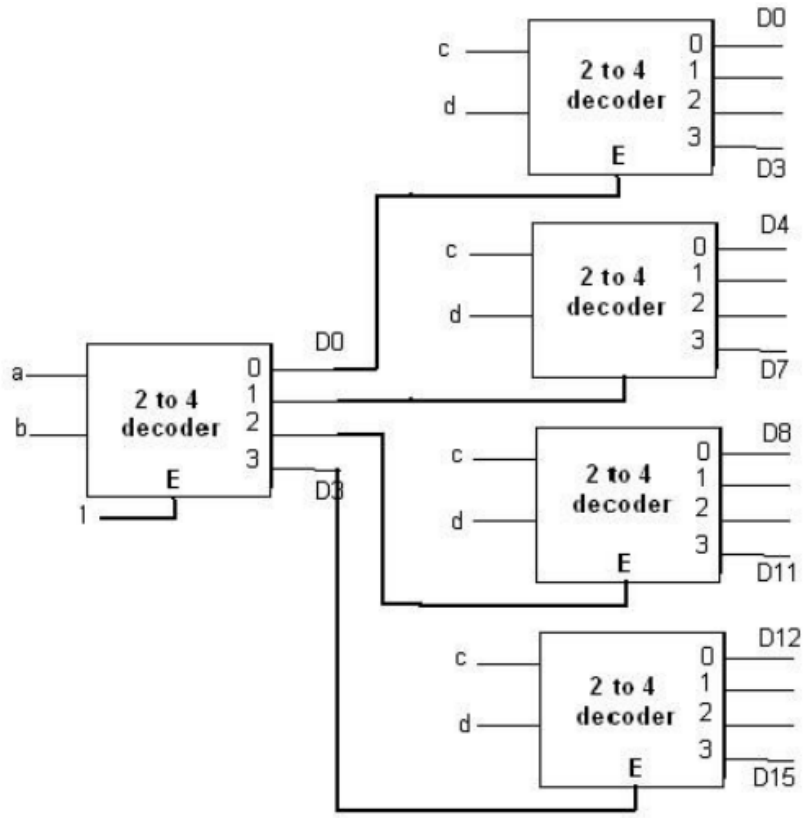


그림 3 2-to-4 디코더에서 4-to-16 디코더로의 확장

위 그림 3 은 2-to-4 디코더 5 개를 연결하여 4-to-16 디코더로 확장한 것이다.

이번 실험에서는 위의 그림처럼 2-to-4 디코더 여러 개를 연결하여 4-to-16 디코더를 구현하는 것이 목적이다. 이때 사용되는 2-to-4 디코더는 Active-low 디코더이다. Active-low 디코더는 NAND 게이트로 구성되어 있으며, maxterm 을 생성한다. 따라서 함수에 대한 최대항의 곱이 있는 경우, Active-low 디코더와 AND 게이트를 통해 쉽게 함수를 구현할 수 있다. NAND 게이트를 이용하면 최소항의 합으로도 표현 가능하다.

4) 특수 목적 디코더

- 소수 판별기(Prime Number Indicator)

소수 판별기는 주어진 입력이 소수일 때만 참을 출력하는 회로이다. 예를 들어 4-bit 소수 검출기가 있다고 할 때, $1101_{(2)}$을 입력하면 참, $1010_{(2)}$을 입력하면 거짓을 출력한다.

이번 실험에서 4-bit 소수 판별기를 구현하는 것을 목표로 한다.

- 배수 검출기(Multiplier Indicator)

배수 검출기는 소수 판별기와 비슷한 원리로, 주어진 입력이 배수일 때만 참을 출력하는 회로이다. 배수 검출기는 입력에 따라서 모든 출력이 거짓일 수도 있고, 여러 출력이 동시에 참일 수도 있다.

이번 실험에서 2, 3, 5, 7, 11 의 배수 검출기를 구현하는 것을 목표로 한다.

4) 멀티플렉서(Multiplexer, MUX)

멀티플렉서는 선택 신호에 따라 여러 입력 신호 중 하나를 골라 출력하는 회로이다. 주로 2^n 개의 입력 신호를 n 개의 선택 신호로 선택하고, 그 하나를 출력한다. 이를 다르게 표현하면, 2^n 개의 데이터 입력, n 개의 컨트롤 입력, 1 개의 출력을 갖는다.

5) Majority / Minority function

홀수 개의 입력에 대해 다수를 차지하는 입력을 나타내는 함수를 Majority function, 소수를 차지하는 입력을 나타내는 함수를 Minority function 이라고 한다. 예를 들어, 세 개의 입력이 1, 1, 0 이라면 Majority function 의 결과는 1, Minority function 의 결과는 0 이다.

6) 멀티플렉서를 사용한 함수 표현

멀티플렉서는 임의의 함수를 구현하는 데 사용된다. 예를 들어 n=2 인 멀티플렉서는 S_0과 S_1로 I_0, I_1, I_2, I_3의 입력 중 하나를 선택한다. 식으로 나타내면 $O = \overline{S_0}\overline{S_1}I_0 + S_0\overline{S_1}I_1 + \overline{S_0}S_1I_2 + S_0S_1I_3$가 된다. 따라서 S_0, S_1, 그리고 $I_{0\sim3}$에 적절한 값을 대입하여 원하는 함수를 구현할 수 있다.

3. 실험 준비

1) 2-to-4 Active-low enable, Active-low output, Active-high input 디코더로 4-to-16 Active-low enable, Active-low output, Active-high input 디코더를 구현한다.

2-to-4 Active-low 디코더 5 개를 이용하여 4-to-16 Active-low 디코더를 구현한다. 이때, 1 개의 2-to-4 Active-low 디코더는 남은 4 개의 2-to-4 Active-low 디코더의 EN 으로 사용된다. 2 개의 입력은 EN 으로 사용되는 디코더의, 남은 2 개의 입력은 남은 4 개의 디코더의 입력으로 사용된다.

2) 4-bit 소수 판별기와 배수 검출기(2, 3, 5, 7, 11)의 진리표를 구하고, 식을 단순화한다.

- 4-bit 소수 판별기 진리표

A	B	C	D	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

단순화하기 전 식은 다음과 같다.

F = A'B'CD'+A'B'CD+A'BC'D+A'BCD+AB'CD+ABC'D

이를 k-map 으로 표현하여 단순화하면 다음과 같다.

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

위와 같이 총 네 번 묶어 단순화할 수 있다. 이를 단순화한 불 대수식으로 표현하면 다음과 같다.

F = A'B'C+BC'D+B'CD+A'CD

- 배수 검출기

- 2 배수

A	B	C	D	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

단순화하기 전 식은 다음과 같다.

F = A'B'CD'+A'BC'D'+A'BCD'+AB'C'D'+AB'CD'+ABC'D'+ABCD'

이를 k-map 으로 표현하여 단순화하면 다음과 같다.

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

위와 같이 총 세 번 묶어 단순화할 수 있다. 이를 단순화한 불 대수식으로 표현하면 다음과 같다.

F = AD'+BD'+CD'

- 3 배수

A	B	C	D	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

단순화하기 전 식은 다음과 같다.

F = A'B'CD+A'BCD'+AB'C'D+ABC'D'+ABCD

이를 k-map 으로 표현하여 단순화하면 다음과 같다.

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

위와 같이 총 다섯 번 묶어 단순화할 수 있다. 이를 단순화한 불 대수식으로 표현하면 다음과 같다.

F = A'B'CD+A'BCD'+ABC'D'+AB'C'D+ABCD

- 5 배수

A	B	C	D	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

단순화하기 전 식은 다음과 같다.

F = A'BC'D+AB'CD'+ABCD

이를 k-map 으로 표현하여 단순화하면 다음과 같다.

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

위와 같이 총 세 번 묶어 단순화할 수 있다. 이를 단순화한 불 대수식으로 표현하면 다음과 같다.

F = A'BC'D+ABCD+AB'CD'

- 7 배수

A	B	C	D	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

단순화하기 전 식은 다음과 같다.

F = A'BCD+ABCD'

이를 k-map 으로 표현하여 단순화하면 다음과 같다.

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

위와 같이 총 두 번 묶어 단순화할 수 있다. 이를 단순화한 불 대수식으로 표현하면 다음과 같다.

F = A'BCD+ABCD'

- 11 배수

A	B	C	D	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

단순화하기 전 식은 다음과 같다.

F = AB'CD

이를 k-map 으로 표현하여 단순화하면 다음과 같다.

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

위와 같이 총 한 번 묶어 단순화할 수 있다. 이를 단순화한 불 대수식으로 표현하면 다음과 같다.

F = AB'CD

3) 5-bit Majority Function 의 진리표를 구하고, 식을 SOP 꼴로 바꿔 8:1 멀티플렉서로 표현한다.

A	B	C	D	E	F
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
0	0	1	0	1	0
0	0	1	1	0	0
0	0	1	1	1	1
0	1	0	0	0	0
0	1	0	0	1	0
0	1	0	1	0	0
0	1	0	1	1	1
0	1	1	0	0	0
0	1	1	0	1	1
0	1	1	1	0	1
0	1	1	1	1	1
1	0	0	0	0	0
1	0	0	0	1	0
1	0	0	1	0	0
1	0	0	1	1	1
1	0	1	0	0	0
1	0	1	0	1	1
1	0	1	1	0	1
1	0	1	1	1	1
1	1	0	0	0	0
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	0	1	1
1	1	1	1	0	1
1	1	1	1	1	1

단순화하기 전 식은 다음과 같다.

F =
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

이를 k-map 으로 표현하여 단순화하면 다음과 같다.

AB \ CDE	000	001	011	010	110	111	101	100
00	0	0	0	0	0	1	0	0
01	0	0	1	0	1	1	1	0
11	0	1	1	1	1	1	1	1
10	0	0	1	0	1	1	1	0

위와 같이 총 두 번 묶어 단순화할 수 있다. 이를 단순화한 불 대수식으로 표현하면 다음과 같다.

F = CDE+BDE+BCE+BCD+ADE+ACE+ACD+ABE+ABD+ABC

Quine-McCluskey Method 를 활용하여 SOP 꼴로 단순화하여 나타내면 다음과 같다.

F(A, B, C, D, E) = ∑m(7, 11, 13, 14, 15, 19, 21, 22, 23, 25, 26, 27, 28, 29, 30, 31)

7	00111	V	(7, 15)	0-111	V	(7, 15, 23, 31)	--111	
11	01011	V	(7, 23)	-0111	V	(11, 15, 27, 31)	-1-11	
13	01101	V	(11, 15)	01-11	V	(13, 15, 29, 31)	-11-1	
14	01110	V	(11, 27)	-1011	V	(14, 15, 30, 31)	-111-	
19	10011	V	(13, 15)	011-1	V	(19, 23, 27, 31)	1--11	
21	10101	V	(13, 29)	-1101	V	(21, 23 29, 31)	1-1-1	
22	10110	V	(14, 15)	0111-	V	(22, 23, 30, 31)	1-11-	
25	11001	V	(14, 30)	-1110	V	(25, 27, 29, 31)	11--1	
26	11010	V	(19, 23)	10-11	V	(26, 27, 30, 31)	11-1-	
28	11100	V	(19, 27)	1-011	V	(28, 29, 30, 31)	111--	
			(21, 23)	101-1	V			
15	01111	V	(21, 29)	1-101	V			

23	10111	V	(22, 23)	1011-	V
27	11011	V	(22, 30)	1-110	V
29	11101	V	(25, 27)	110-1	V
30	11110	V	(25, 29)	11-01	V
			(26, 27)	1101-	V
31	11111		(26, 30)	11-10	V
			(28, 29)	1110-	V
			(28, 30)	111-0	V
			(15, 31)	-1111	V
			(23, 31)	1-111	V
			(27, 31)	11-11	V
			(29, 31)	111-1	V
			(30, 31)	1111-	V

	7	11	13	14	15	19	21	22	23	25	26	27	28	29	30	31
(7, 15, 23, 31)	X				X				X							X
(11, 15, 27, 31)		X			X							X				X
(13, 15, 29, 31)			X		X									X		X
(14, 15, 30, 31)				X	X										X	X
(19, 23, 27, 31)						X			X			X				X
(21, 23, 29, 31)							X		X					X		X
(22, 23, 29, 31)								X	X					X		X
(25, 27, 29, 31)										X		X		X		X
(26, 27, 30, 31)											X	X			X	X
(28, 29, 30, 31)													X	X	X	X

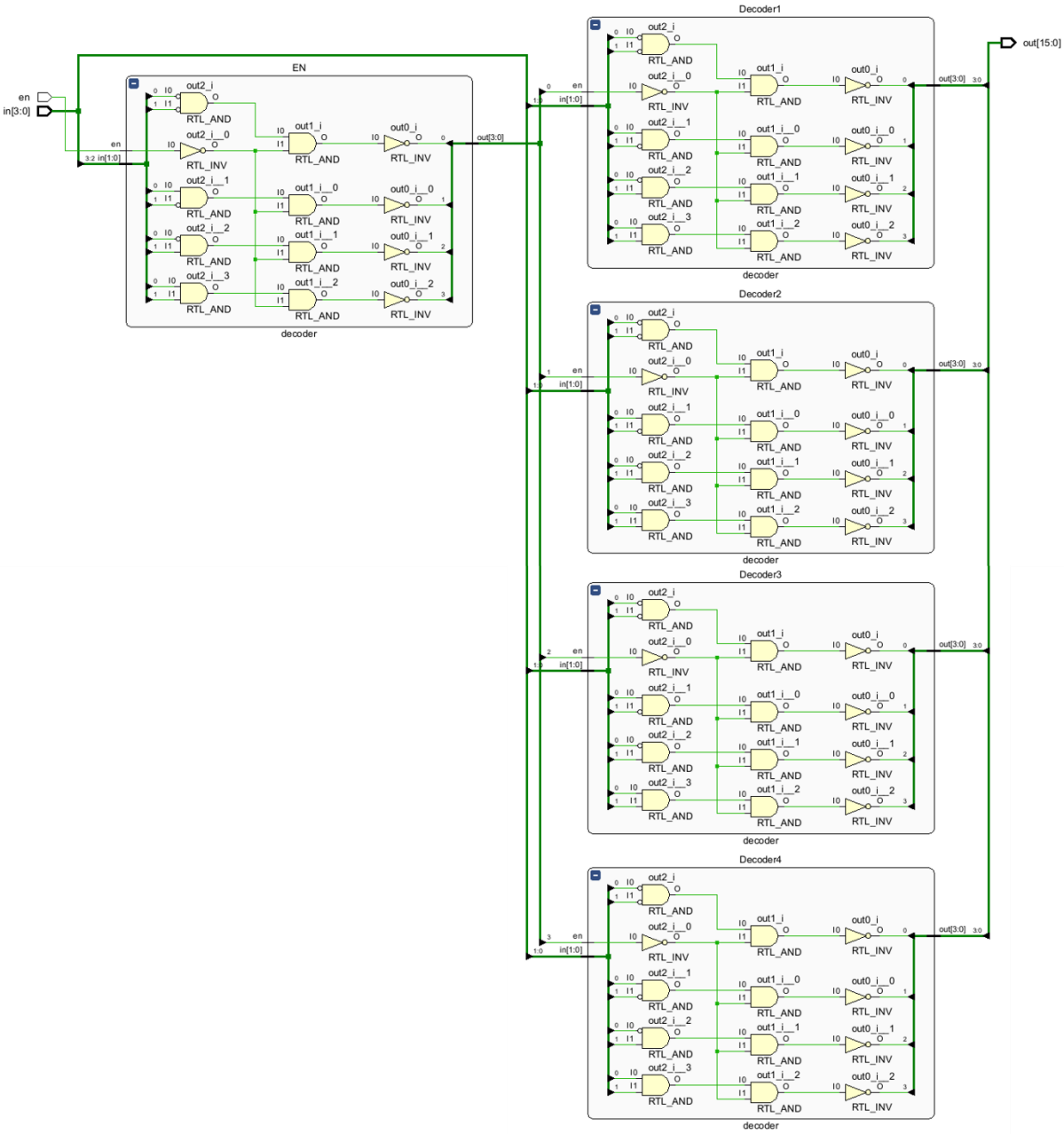
F = CDE+BDE+BCE+BCD+ADE+ACE+ACD+ABE+ABD+ABC

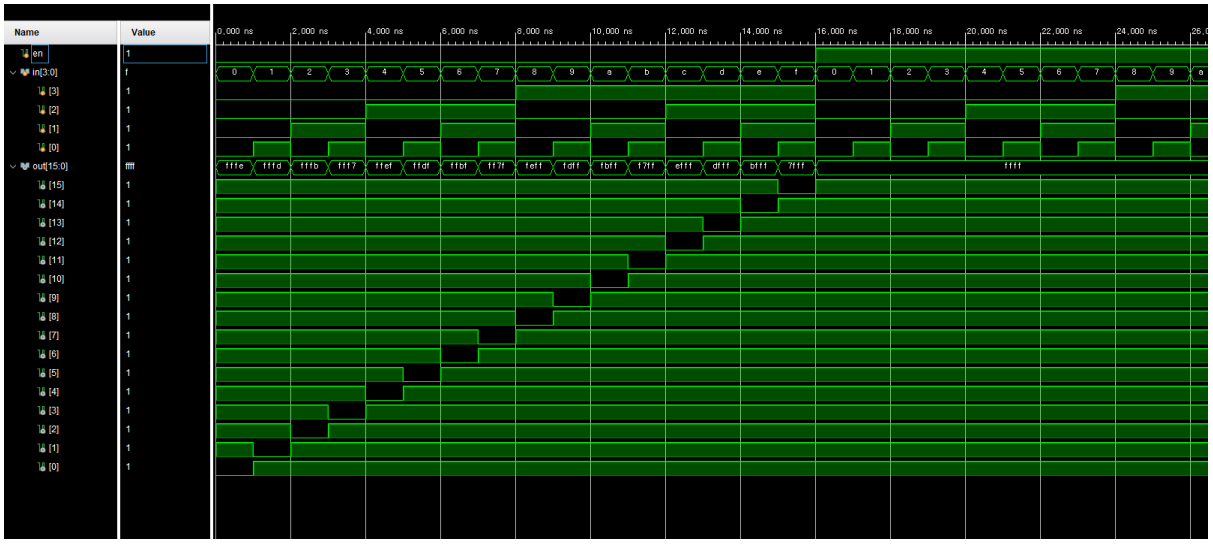
A	B	C	D	E	F	
0	0	0	0	0	0	0
0	0	0	0	1	0	
0	0	0	1	0	0	
0	0	0	1	1	0	
0	0	1	0	0	0	DE
0	0	1	0	1	0	
0	0	1	1	0	0	
0	0	1	1	1	1	
0	1	0	0	0	0	DE
0	1	0	0	1	0	
0	1	0	1	0	0	
0	1	0	1	1	1	
0	1	1	0	0	0	D+E
0	1	1	0	1	1	
0	1	1	1	0	1	
0	1	1	1	1	1	
1	0	0	0	0	0	DE
1	0	0	0	1	0	
1	0	0	1	0	0	
1	0	0	1	1	1	
1	0	1	0	0	0	D+E
1	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	

8:1 멀티플렉서로 표현하기 위해 D 와 E 입력을 이용하여 8 개의 데이터 입력을 구현하고, A, B, C 입력을 컨트롤 입력으로 활용한다.

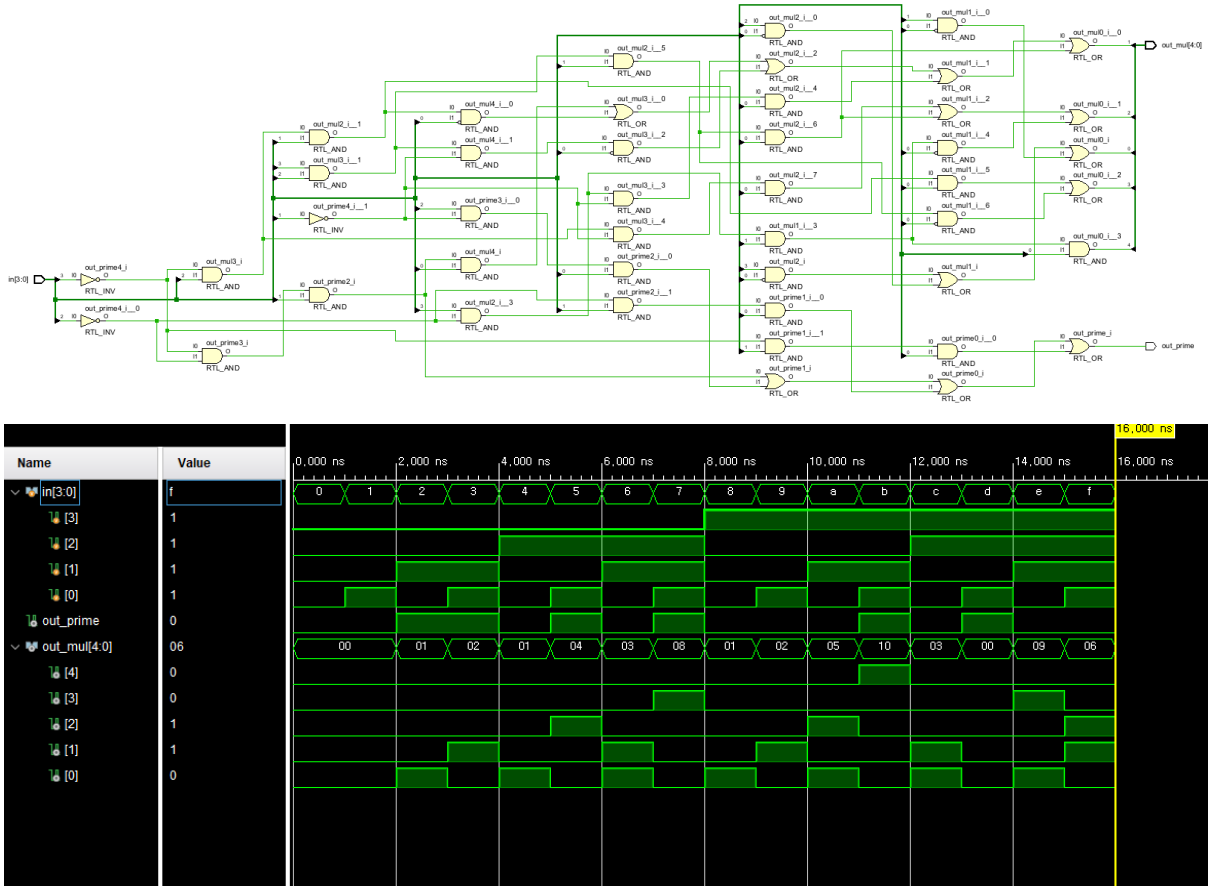
4. 결과

1) 4-to-16 디코더

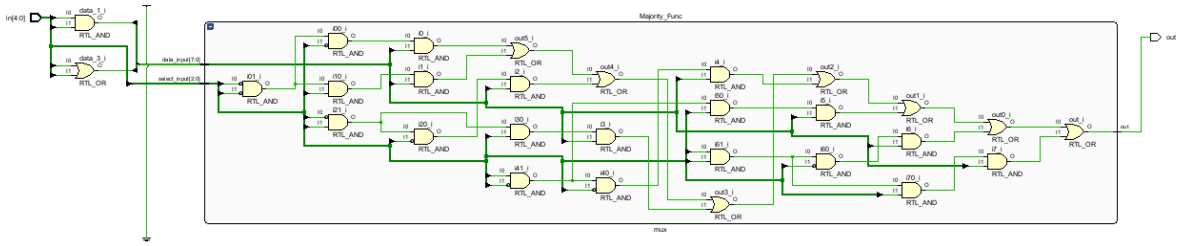


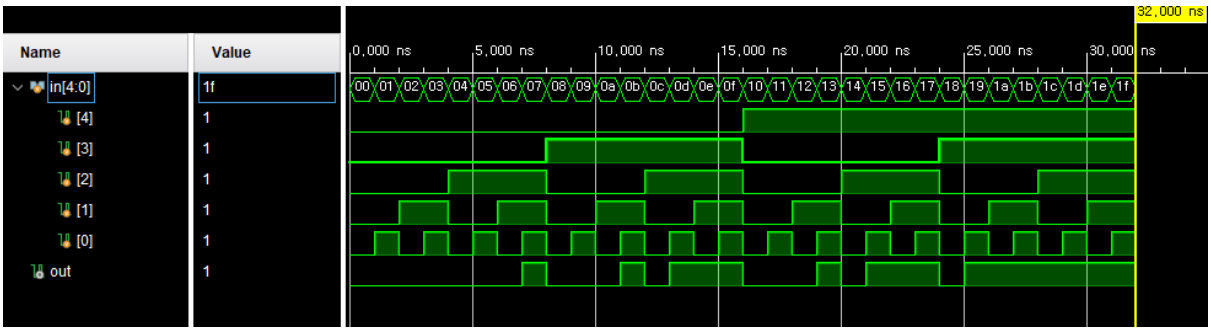


2) 4-bit 소수 판별기와 배수 검출기



3) 5-bit Majority Function





5. 논의

1) 느낀 점

이번 실험을 통해 디코더와 멀티플렉서의 개념, 디코더 확장 등을 이해할 수 있었다. 기존에는 단일 출력에 대한 회로에 대해서만 학습하였다면, 이번 실험을 통해 수업 시간에 배운 다중 출력 회로에 대해서도 복습할 수 있었고, 그 회로를 구현함에 있어 디코더와 멀티플렉서와 같은 회로를 공부할 수 있었다. 저번 실험에서 사용한 k-map 단순화를 사용하면서 한 번 더 나의 이해도를 점검할 수 있었고, 5 변수 함수에 대한 단순화에서 k-map 뿐만 아니라, Quine-McCluskey Method 를 활용하면서 다변수에서의 단순화에 대한 두 가지 방법을 모두 익힐 수 있었던 것 같다.

2) 어려웠던 점 및 해결 방법

이때까지 AND, OR, NOT 게이트와 단순한 Boolean Algebra 연산을 통한 회로 구현을 하다가 디코더, 멀티플렉서와 같은 다중 출력 회로를 구현하고자 하니 처음에 이해하는 데에 있어 많이 헤맸던 것 같다. 특히 단순히 디코더를 구현하는 것이 아니라, 구현되어 있는 디코더로 다른 디코더를 만드는 디코더 확장 개념이 가장 어려웠다. 개념적으로도 어려웠기에 코드를 구현하는 데에 있어 어떻게 코드를 구성해야 하는지 알아내는 데 시간을 많이 쏟았다. 저번 시간에는 ~ 연산자만 사용하였는데, 이번에는 &, assign 연산자도 활용하면서 다른 modeling 방법에 대해서 하나씩 사용 방법을 익힐 수 있었다. module 개념이 객체지향프로그래밍에서의 클래스와 비슷한 개념이라는 것을 알고 나니 이해가 조금 수월했다. 배열을 매개변수로 하는 module 에 배열을

인수로 넘겨줄 때, 인덱스를 어떻게 구성해야 하는지 알기 어려웠는데, Little-endian 에 대해 알아보면서 MSB 와 LSB 순서를 스스로 정의한다는 것을 알게 되었다. 따라서 배열 선언은 Little-endian 으로 했기 때문에, 인수로 넘겨줄 때도 Little-endian 으로 구현하면 될 것이라 생각하여, 그에 따라 구현하였다. 그리고 배열을 넘겨줄 때는 중괄호({ })를 활용하여 넘겨준다는 것을 알게 되었다.