실험 3. 디코더와 멀티플렉서

2022. 04. 01.
디지털 시스템 설계 (CSED273)

1. 개요

Multiple-output 회로를 대표하는 디코더와 Multiple-input 회로를 대표하는 멀티플렉서의 기능을 이해하고 이를 사용해 회로를 구성한다. 세부적인 학습 목표는 다음과 같다.

- Active-low 디코더 확장 이해 및 구현
- 특수 목적 디코더 구현
- 멀티플렉서의 데이터 선택 기능을 활용한 Majority function 구현

2. 이론적 배경

1) 디코더 (Decoder / De-Multiplexer)

디코더는 n개의 이진 입력을 받아 최대 2^n개의 고유 출력을 가지는 회로다. n개의 이진 입력과 2^n개의 서로 다른 출력을 가지는 경우, 각 출력이 곧 minterm이기 때문에 minterm generator라고도 한다. 실제 디코더 소자에는 n개의 입력뿐 아니라 EN, 혹은 Enable 입력이 추가로 존재한다. EN은 말 그대로 디코더를 켜거나 끄기 위해서 사용된다.

디코더의 입력과 출력 특성을 표현할 때는 n-to-2^n과 k-of-2^n라는 표현을 사용한다. n-to-2^n은 n개의 입력을 받아 2^n개의 출력을 가진다는 입력과 출력의 관계를 표현하고, k-of-2^n은 2^n개의 입력 중에서 k개의 입력이 동시에 참이 된다는 출력의 특성을 나타낸다.

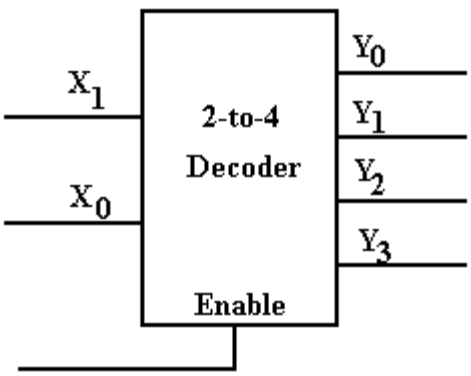

그림 1. 2-to-4 디코더

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

표 1. Active-high 디코더의 진리표

2) 디코더 확장

EN을 사용해 디코더를 켜거나 끌 수 있다는 점을 이용해 여러 개의 디코더를 연결하여 더 많은 수의 입력을 처리할 수 있다. 이를 디코더 확장(Decoder expansion)이라고 한다.

예를 들어, **그림 2**와 같이 2-to-4 디코더를 두 개 사용해 3-to-8 디코더를 구성하거나, **그림 3**과 같이 2-to-4 디코더를 다섯 개 사용해 4-to-16 디코더를 구성할 수 있다.

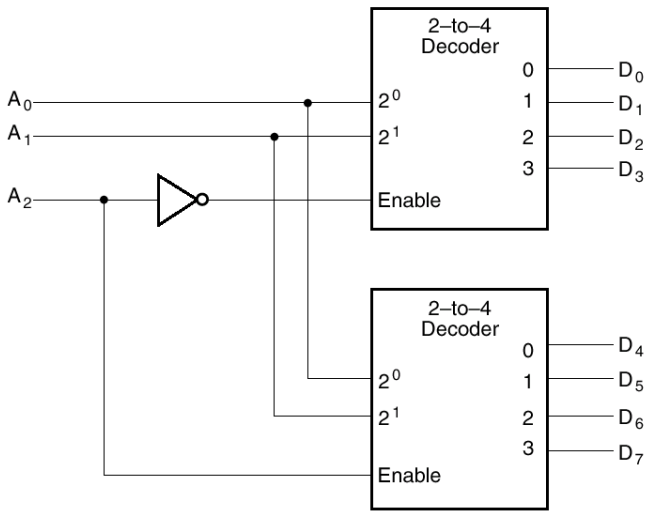

그림 2. 2-to-4 => 3-to-8 확장

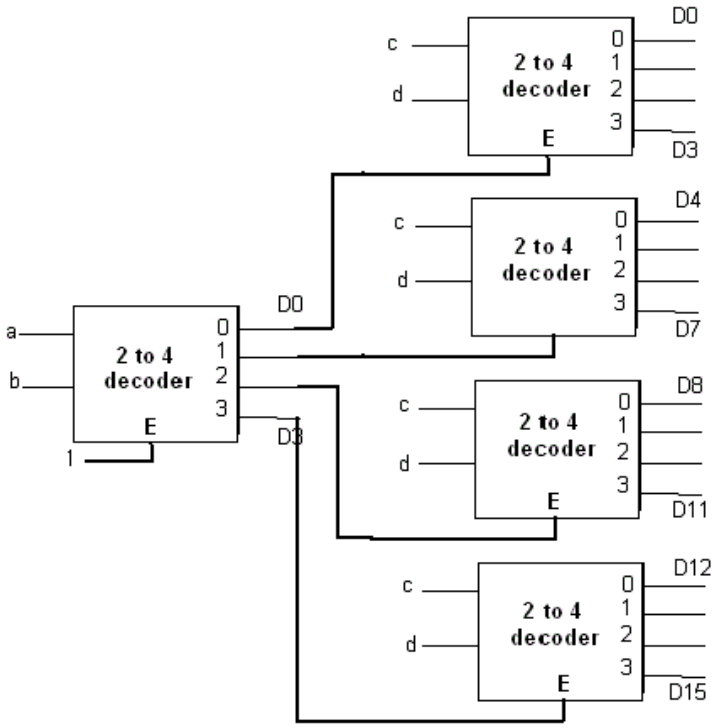

그림 3. 2-to-4 => 4-to-16 확장

3) 특수 목적 디코더

- 소수 판별기

주어진 입력이 소수일 때 참을 출력하는 회로다. 예를 들어 4-bit 소수 검출기에 $1101_{(2)}$을 입력하면 참, $1010_{(2)}$을 입력하면 거짓을 출력하게 된다.

- 배수 검출기

비슷한 원리로 예를 들어 입력이 각각 2, 3, 5, 7의 배수인지를 나타내는 디코더를 설계할 수도 있다. 이 경우 입력에 따라 단 하나의 출력도 참이 아닐 수도, 여러 출력이 동시에 참일 수도 있다.

4) 멀티플렉서 (Multiplexer; MUX)

멀티플렉서는 선택 신호에 따라 여러 입력 신호 중 하나를 골라 출력하는 회로다. 주로 2^n개의 입력 신호를 n개의 선택 신호로 선택하고, 그 하나를 출력한다.

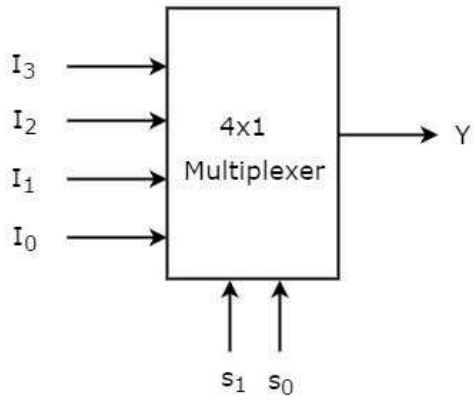

그림 4. 4:1 멀티플렉서

5) Majority / Minority function

홀수 개의 입력에 대해 다수 / 소수를 차지하는 입력을 나타내는 함수다. 예를 들어, 세 개의 입력이 1, 1, 0이라면 Majority function과 Minority function의 결과는 각각 1과 0이다.

6) 멀티플렉서를 사용한 함수 표현

예를 들어 $n=2$인 멀티플렉서는 S_0와 S_1로 I_0, I_1, I_2, I_3의 입력 중에서 하나를 선택한다. 이 식으로 나타내면 $O=\overline{S_0}\,\overline{S_1}\,I_0 + S_0\,\overline{S_1}\,I_1 + \overline{S_0}\,S_1\,I_2 + S_0\,S_1\,I_3$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) 4비트 소수 판별기와 배수 검출기(2, 3, 5, 7, 11)의 진리표를 구하고, 식을 단순화한다.
3) 5비트 Majority function의 진리표를 구하고, 식을 SOP꼴로 바꿔 8:1 멀티플렉서로 표현한다.

* 보고서 필수 내용
  - 실험 준비 과정

4. 실험

0) 공통 유의사항

- 주어진 코드에서 */* Add your code here */* 로 주석 처리된 부분만 수정하여 구현한다.
- 모든 입출력은 Little-endian 형식으로 표현한다.

* 보고서 필수 내용
  - Schematic 기능으로 생성한 회로도 캡처
  - 테스트벤치 시뮬레이션 파형 캡처

1) 4-to-16 Active-low 디코더 – lab3_1.v, lab3_1_tb.v

ㄱ. 주어진 2-to-4 Active-low 디코더 모듈을 사용해 4-to-16 Active-low 디코더를 구현한다.
ㄴ. Schematic 기능으로 회로를 확인한다.
ㄷ. 주어진 테스트벤치로 시뮬레이션을 실행해 정상 작동을 확인한다.

2) 4비트 소수 판별기 및 배수 검출기 – lab3_2.v, lab3_2_tb.v

ㄱ. 단순화한 소수 판별기와 배수 검출기를 구현한다.
  - MSB부터 LSB까지 순서대로 11, 7, 5, 3, 2의 배수인지를 출력한다.
ㄴ. Schematic 기능으로 회로를 확인한다.
ㄷ. 주어진 테스트벤치로 시뮬레이션을 실행해 정상 작동을 확인한다.

3) 5비트 Majority function – lab3_3.v, lab3_3_tb.v

ㄱ. SOP 꼴로 바꾼 Majority function을 주어진 멀티플렉서로 구현한다.
ㄴ. Schematic 기능으로 회로를 확인한다.
ㄷ. 주어진 테스트벤치로 시뮬레이션을 실행해 정상 작동을 확인한다.

5. 제출

{학번}_lab3.zip으로 다음 파일을 압축하여 PLMS로 제출한다.

- lab3_1.v *// 코드*
- lab3_2.v
- lab3_3.v
- lab3_report.pdf *// 보고서*