Lecture 16 In-Class Worksheet

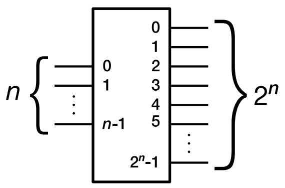
1 Learning Objectives

This worksheet is based on Lumetta course notes section 2.5 and Patt & Patel textbook section 3.3. After completing this lesson, you will know how to:

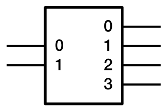
- [S16-1] Build a decoder with a set of arbitrary parameters.
- [S16-2] Build a multiplexer with a set of arbitrary parameters.
- [S16-3] Implement arbitrary functions using a decoder and **OR** gate.
- [S16-4] Implement arbitrary functions using a multiplexer.

2 Decoders

A *decoder* is a logic device that takes n bits of input and produces 2^n bits out output. The general form of a decoder is shown on the right. Exactly one of the output bits of the decoder is 1 and the others are all 0. Which bit is 1 is determined by the input: each input corresponds to a distinct output bit being one. A common scheme is to treat the n bits of input as a binary number that encodes which of the 2^n output bits is 1. For example, shown below on the left is a 2-to-4 decoder and its truth table. The diagram on the right shows its implementation using **AND** gates.

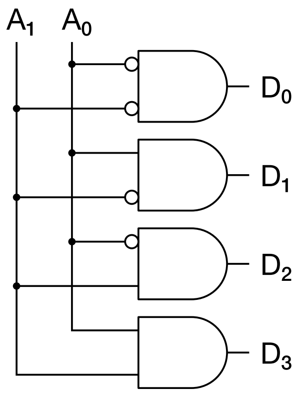


n-to-2^n decoder

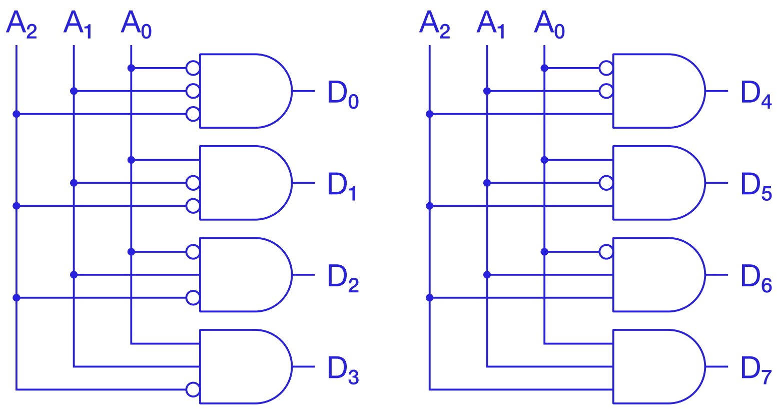


2-to-4 decoder

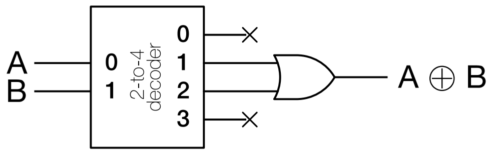
A_1	A_0	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



Q1. Implement a 3-to-8 decoder using **AND** gates an inverters.



The outputs of a n-to-2^n decoder are all of the minterms of a function on n variables. Recall that to construct the canonical SOP expression for a Boolean function, we **OR**-ed all of its minterm implicants (minterms corresponding to inputs where the function had value 1). We can thus implement an arbitrary function by **OR**-ing the appropriate subset of the outputs of a decoder. For example, to implement the **XOR** function on two variables, we can use a 2-to-4 decoder to produce the minterms on A and B, and then **OR** the appropriate minterms:

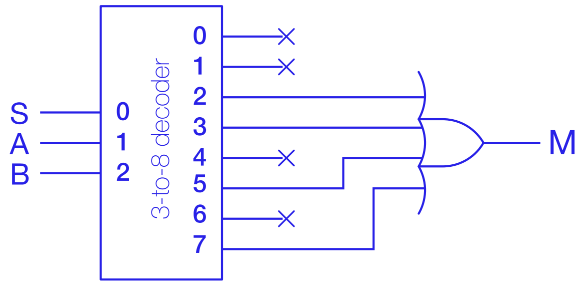


Q2. Implement the following function using a 3-to-8 decoder and an **OR** gate:

$$M(S,A,B) = \begin{cases} A & \text{if } S = 0, \\ B & \text{if } S = 1. \end{cases}$$

The truth table for M is shown on the left and final circuit on the right.

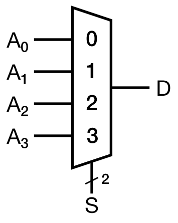
S	A	B	M
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1



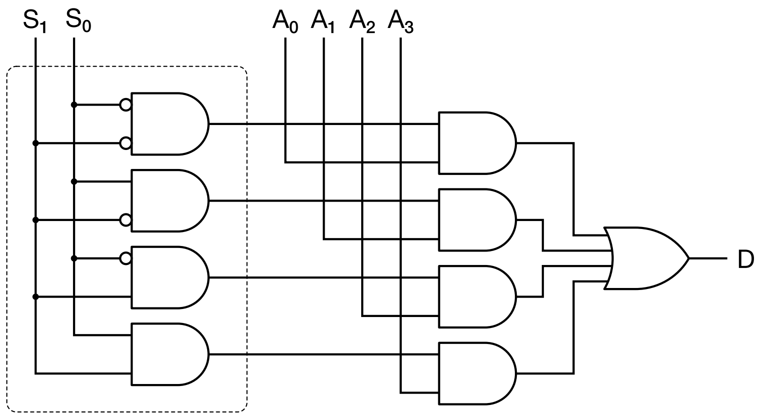
3 Multiplexers

A 2^n-to-1 multiplexer is a logic device that takes an n-bit *select* input (also called *selector*) and 2^n data inputs, and produces a single bit of output. The select input determines which of the 2^n possible inputs are sent to the output. In general, each of the 2^n inputs may be m-bits wide, in which case the multiplexer output is also m bits wide, giving a $m2^n$-to-m multiplexer. Such a multiplexer is equivalent to m 2^n-to-1 multiplexers with a common select input.

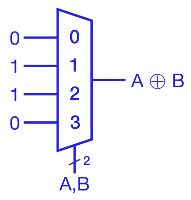
The diagram symbol, function summary, and implementation of a 4-to-1 multiplexer is shown below. The trapezoidal symbol shown below is the standard symbol for a multiplexer.



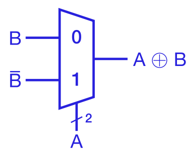
S_1	S_0	D
0	0	A_0
0	1	A_1
1	0	A_2
1	1	A_3



Q3. Implement a two-input **XOR** function using a 4-to-1 multiplexer.


0
1
1
0
0
1
2
3
A ⊕ B
2
A,B

Q4. Implement a two-input **XOR** function using a 2-to-1 multiplexer and inverters.


B
$\bar{B}$
0
1
$A \oplus B$
2
A