Computer Logic and Digital Circuit Design (PHYS306/COSC330): Unit 2

Jordan Hanson

October 12, 2021

Whittier College Department of Physics and Astronomy

Summary

Unit 2 Summary

Functions of Combinatorial Logic

Reading: 6-1 - 6-6 (Tuesday)

Reading: 6-7 - 6-11 (Thursday)

1. Half-Adders and Full-Adders
   - Example from study guide
   - Propagation delays
2. Comparators
   - The XNOR gate
   - Multi-bit comparators
   - Inequalities
3. Decoders/Encoders
   - Binary to decimal circuits
   - Decimal to binary circuits
4. Multiplexing and Demultiplexing

Half-Adders and Full-Adders, Ripple-Carry

Half-Adders and Full-Adders, Ripple-Carry

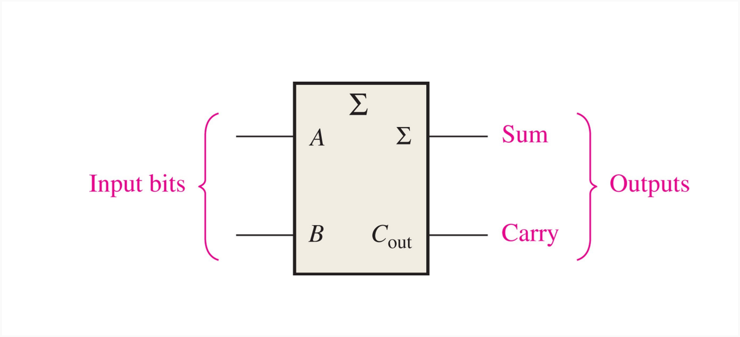


Figure 1: The desired inputs and outputs of the half-adder. There is no carry-input.

Half-Adders and Full-Adders, Ripple-Carry

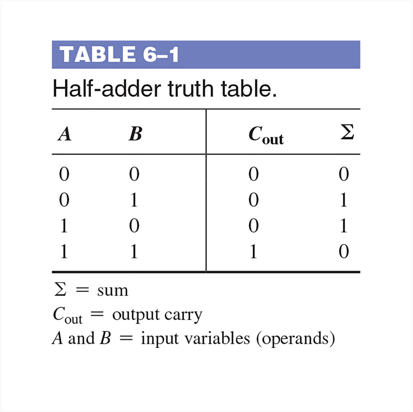


TABLE 6–1

Half-adder truth table.

A	B	C_{out}	Σ
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Σ = sum
C_{out} = output carry
A and B = input variables (operands)

Figure 2: The truth table of the half-adder for 2-bits. What gate action does this match?

Half-Adders and Full-Adders, Ripple-Carry

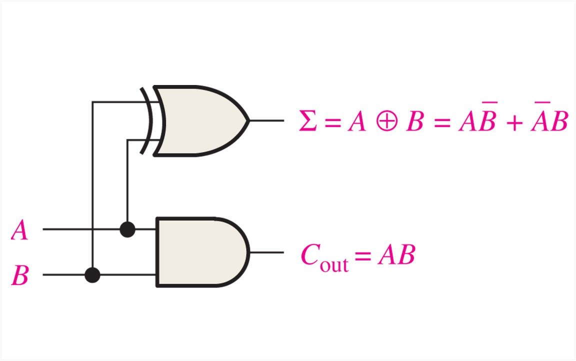


Figure 3: The logic function circuit diagram for the half-adder.

Half-Adders and Full-Adders, Ripple-Carry

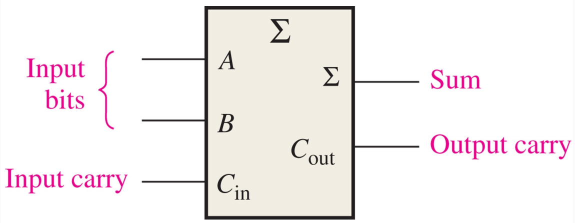


Figure 4: The desired inputs and outputs for the full-adder, with carry-input and carry-output.

Half-Adders and Full-Adders, Ripple-Carry

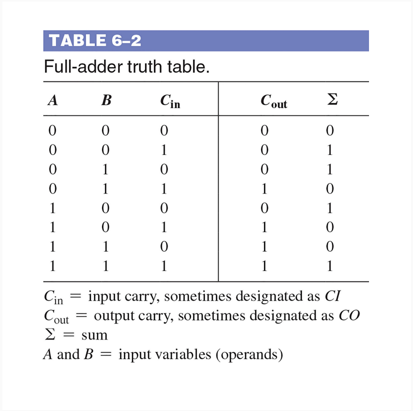


TABLE 6–2

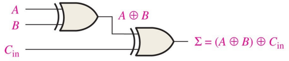
Full-adder truth table.

A	B	C_{in}	C_{out}	Σ
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

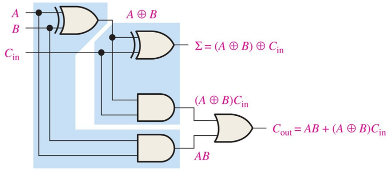
C_{in} = input carry, sometimes designated as *CI*
C_{out} = output carry, sometimes designated as *CO*
Σ = sum
A and *B* = input variables (operands)

Figure 5: The truth table for the full-adder is more complex due to the increased number of inputs.

Half-Adders and Full-Adders, Ripple-Carry



(a) Logic required to form the sum of three bits



A
B
C_{in}
$A \oplus B$
$\Sigma = (A \oplus B) \oplus C_{in}$
$(A \oplus B)C_{in}$
$C_{out} = AB + (A \oplus B)C_{in}$
AB

(b) Complete logic circuit for a full-adder (each half-adder is enclosed by a shaded area)

Figure 6: Circuit diagrams for the half-adder (left) and full-adder (right).

Half-Adders and Full-Adders, Ripple-Carry

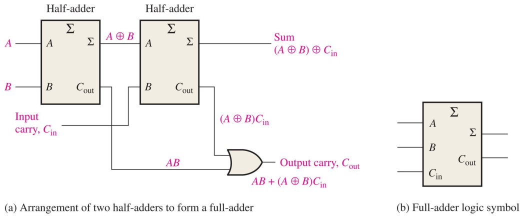


Figure 7: Two half-adders to form a full-adder.

Half-Adders and Full-Adders, Ripple-Carry

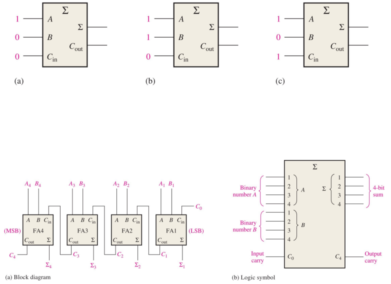


Figure 8: Four FA (full-adders) to add bits to the numbers being added.

Half-Adders and Full-Adders, Ripple-Carry

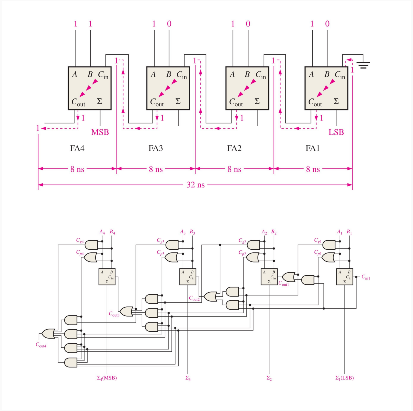


Figure 9: Propagation delays add serially in a full-adder with ripple carry topology.

Half-Adders and Full-Adders, Ripple-Carry

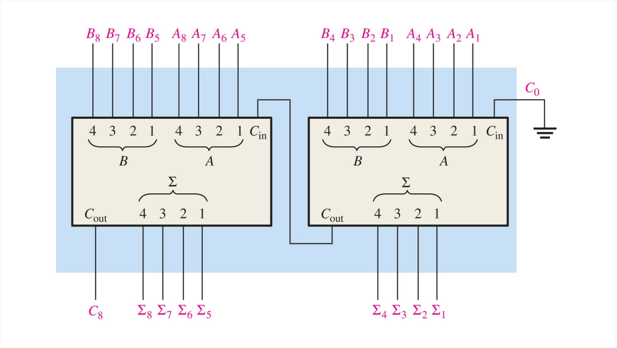

B_8 B_7 B_6 B_5 A_8 A_7 A_6 A_5
B_4 B_3 B_2 B_1 A_4 A_3 A_2 A_1
C_0
4 3 2 1 4 3 2 1 C_{in}
B A
Σ
C_{out} 4 3 2 1
4 3 2 1 4 3 2 1 C_{in}
B A
Σ
C_{out} 4 3 2 1
C_8
Σ_8 Σ_7 Σ_6 Σ_5
Σ_4 Σ_3 Σ_2 Σ_1

Figure 10: Two 4-bit FA connected to form an 8-bit FA, accounting for carries.

Comparators

Comparators

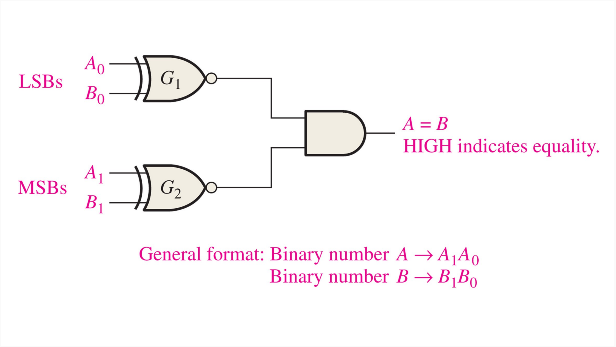
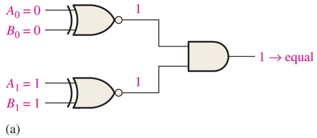

LSBs
A_0
B_0
G_1
MSBs
A_1
B_1
G_2
$A = B$
HIGH indicates equality.
General format: Binary number $A \rightarrow A_1A_0$
Binary number $B \rightarrow B_1B_0$

Figure 11: The basic idea behind the comparator. (a) Review the truth table for the XNOR-gate, which is the conjugate of the XOR gate. (b) What is the function of the AND gate?

Comparators



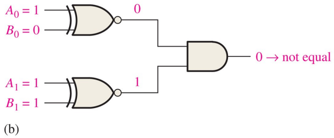
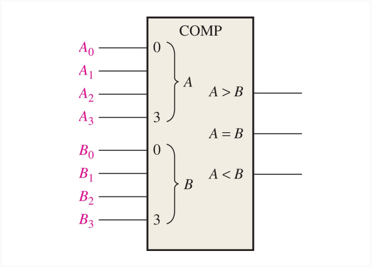


Figure 12: (a) Example of comparison of 2-bit binary numbers. (b) What is the truth table? (c) What is the logical representation of the function? (c) What is a logical representation for the inequality circuit?

Comparators



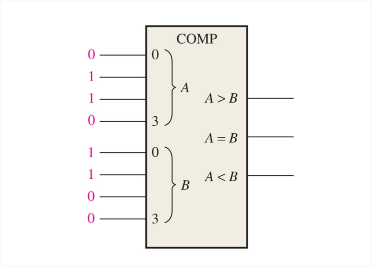


Figure 13: (Left) One *component* that is really 8 comparators linked to the same AND gate. (Right) What is the correct ouput? How to determine the inequality functions?

Comparators

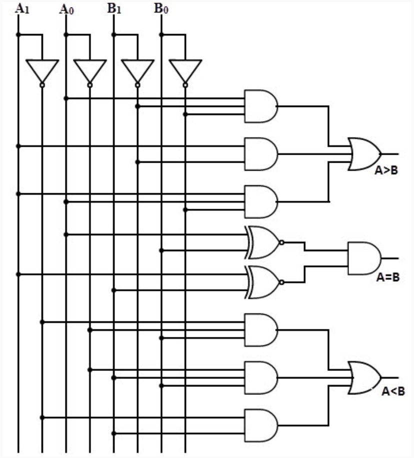


Figure 14: Try some simple cases: (a) A = 00 and B = 01, (b) A = 10 and B = 01.

Decoders

Decoders

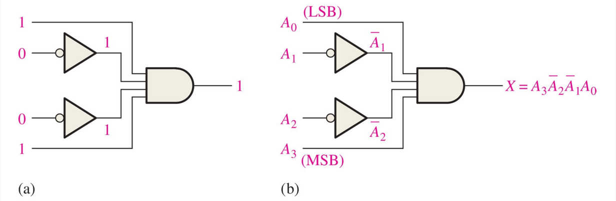


Figure 15: The binary decoder circuit for 9. This circuit is true if the binary number is 1001. (Pay attention to the order of MSB and LSB).

Decoders

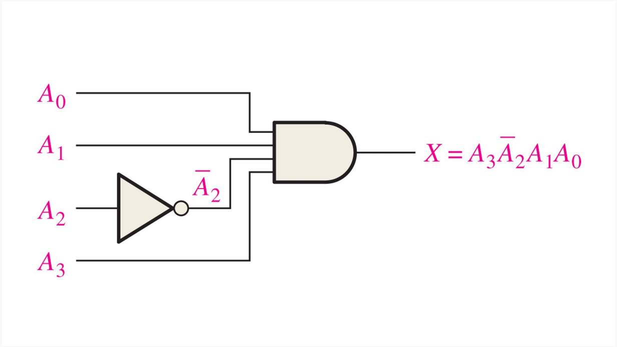


Figure 16: (a) Which binary number is being decoded here? (b) What would it take to decode all binary numbers of *n* bits?

Decoders

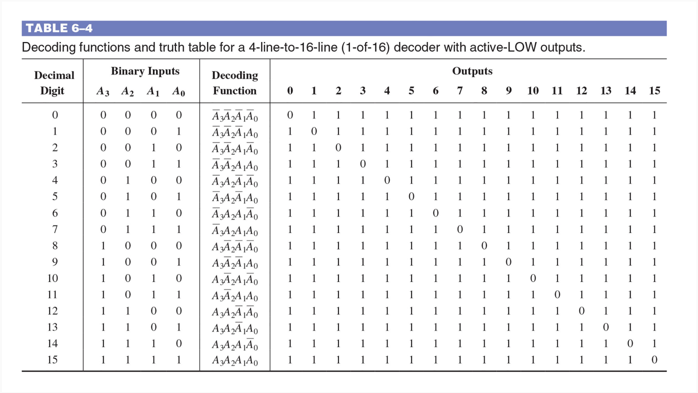


TABLE 6–4

Decoding functions and truth table for a 4-line-to-16-line (1-of-16) decoder with active-LOW outputs.

Decimal Digit	Binary Inputs				Decoding Function	Outputs															
	A_3	A_2	A_1	A_0		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	$\overline{A}_3\overline{A}_2\overline{A}_1\overline{A}_0$	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	0	0	1	$\overline{A}_3\overline{A}_2\overline{A}_1A_0$	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	0	0	1	0	$\overline{A}_3\overline{A}_2A_1\overline{A}_0$	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
3	0	0	1	1	$\overline{A}_3\overline{A}_2A_1A_0$	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
4	0	1	0	0	$\overline{A}_3A_2\overline{A}_1\overline{A}_0$	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
5	0	1	0	1	$\overline{A}_3A_2\overline{A}_1A_0$	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
6	0	1	1	0	$\overline{A}_3A_2A_1\overline{A}_0$	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
7	0	1	1	1	$\overline{A}_3A_2A_1A_0$	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
8	1	0	0	0	$A_3\overline{A}_2\overline{A}_1\overline{A}_0$	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
9	1	0	0	1	$A_3\overline{A}_2\overline{A}_1A_0$	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
10	1	0	1	0	$A_3\overline{A}_2A_1\overline{A}_0$	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
11	1	0	1	1	$A_3\overline{A}_2A_1A_0$	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
12	1	1	0	0	$A_3A_2\overline{A}_1\overline{A}_0$	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
13	1	1	0	1	$A_3A_2\overline{A}_1A_0$	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
14	1	1	1	0	$A_3A_2A_1\overline{A}_0$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
15	1	1	1	1	$A_3A_2A_1A_0$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

Figure 17: The decoding table for ACTIVE LOW 4-bit binary. Terms from linear algebra: this matrix has 0 trace, and is symmetric (we can exchange rows and columns).

Encoders

Encoders

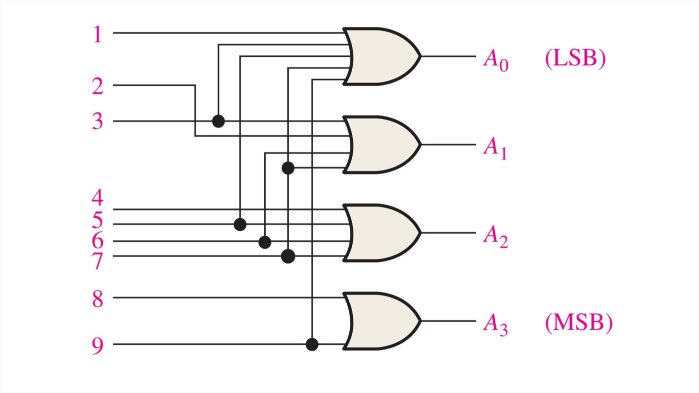


Figure 18: Recall the OR TT, and remember that this is a system in which there are forbidden or *don't care* states. Only one input line can be active at once.

Encoders

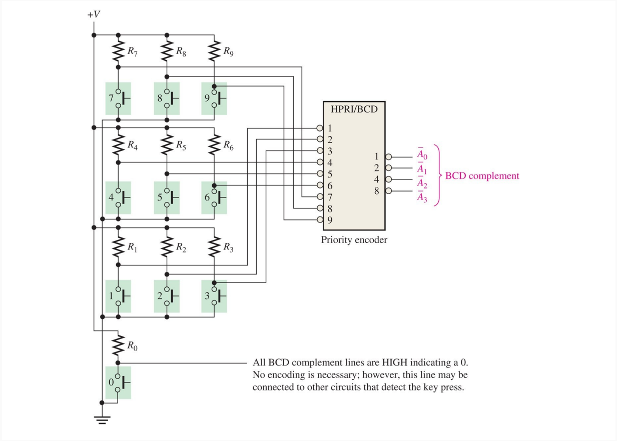


Figure 19: A keypad system. The binary encoder accepts only 9 digits because 0 is redundant. Note the use of *pull-up* resistors (recall LEDs operations).

Multiplexers

Multiplexers

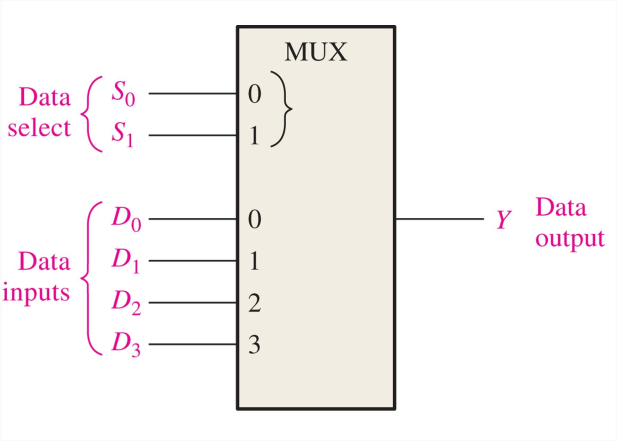


Figure 20: A general block diagram for a *multiplexer.* Parallel data input is successively rotated to the single output.

Multiplexers

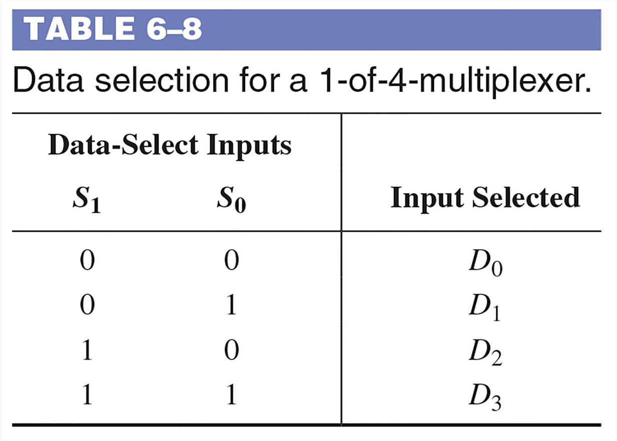


TABLE 6–8

Data selection for a 1-of-4-multiplexer.

Data-Select Inputs		
S_1	S_0	**Input Selected**
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

Figure 21: The desired TT for a 1-of-4 mux. The n-of-m jargon corresponds to inputs and outputs. We should also have *enable*, and one or more output lines.

Multiplexers

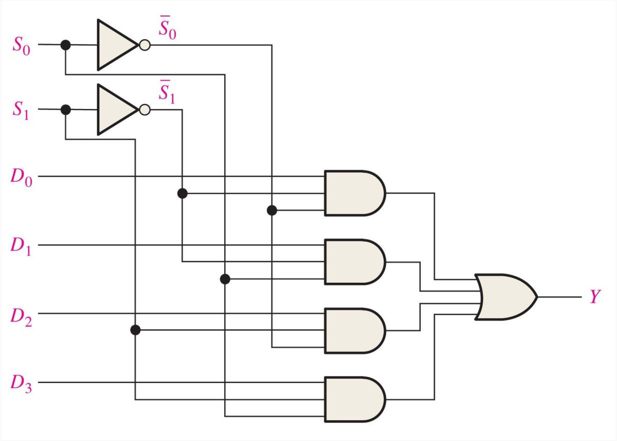


Figure 22: The AND-enable behavior is used in this 1-of-4 mux. Note the interconnections are the key to generating the right output logic.

Multiplexers

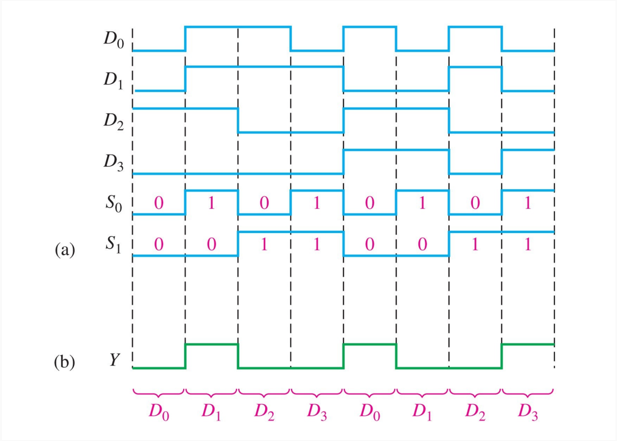


Figure 23: Exercise: verify the timing diagram output Y follows the 1-of-4 pattern. The data-select lines follow a 2-bit *counter* (coming soon).

Multiplexers

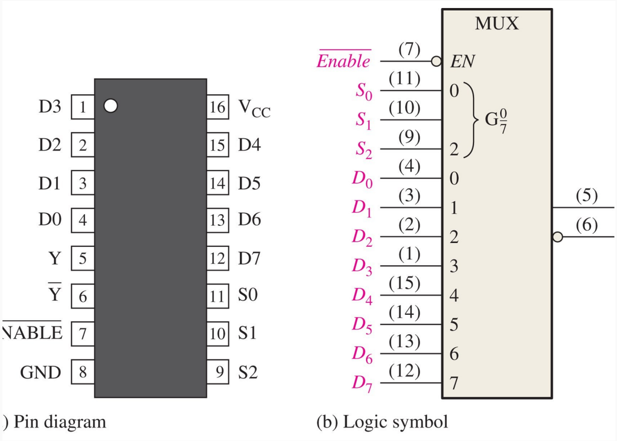


Figure 24: Active LOW enabled, 1-of-8 mux with complemented output.

Multiplexers

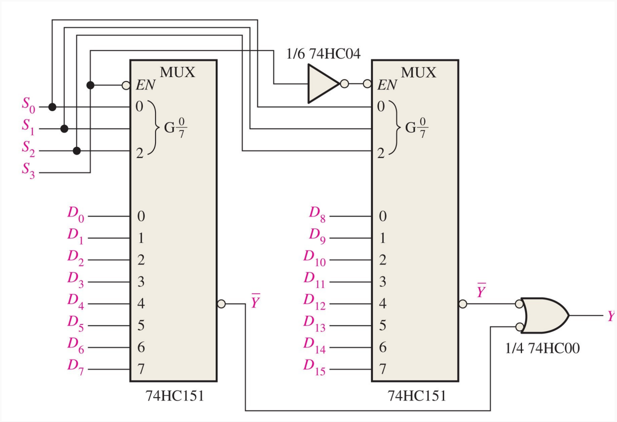


Figure 25: Active LOW enabled, 1-of-8 mux with complemented output, connected to another one. The overall system acts as a 1-of-16 with 4 data select lines.

Multiplexers

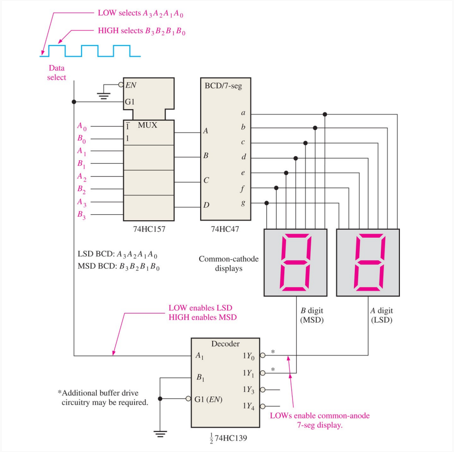
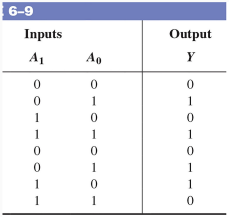


Figure 26: In this example, the mux is a *quad* 1-of-2, to help switch between binary numbers A and B. The BCD decoder converts the inputs, and the 2-to-4 decoder grounds either display.

Multiplexers



6–9

Inputs		Output
A_1	A_0	Y
0	0	0
0	1	1
1	0	0
1	1	1
0	0	0
0	1	1
1	0	1
1	1	0

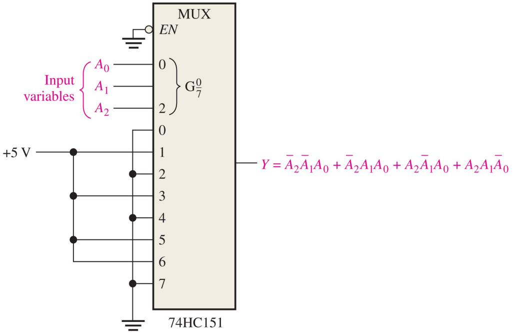

MUX
EN
Input variables
A_0
A_1
A_2
0
2
$G\frac{0}{7}$
+5 V
0
1
2
3
4
5
6
7
74HC151
$Y = \bar{A}_2\bar{A}_1A_0 + \bar{A}_2A_1A_0 + A_2\bar{A}_1A_0 + A_2A_1\bar{A}_0$

Figure 27: A mux can also be used to create any domain-N logic-function, provided we can connect HIGH and LOW to the desired input lines. Do you see the pattern?

Demultiplexers

Demultiplexers

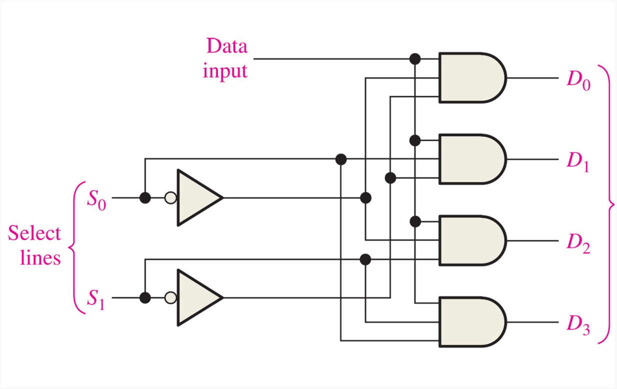


Figure 28: The reverse operation to a multiplexer is a demultiplexer, or demux. Note the AND-enable is still there on the input line, but we need 4 ANDs: 1 for each possible input data line on the mux.

Demultiplexers

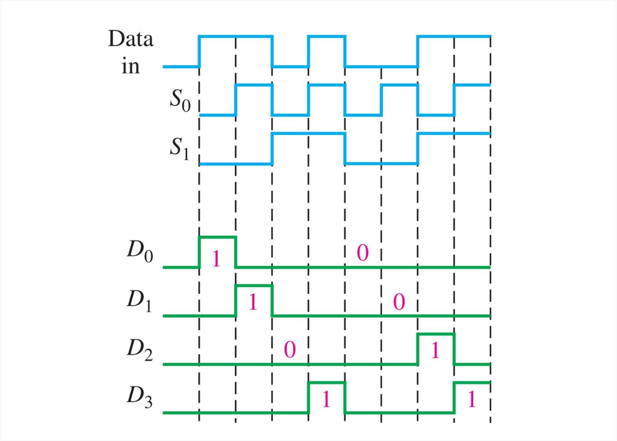


Figure 29: One data line in and four data lines out. *Note the cadence:* we cannot know what D_1 is until we know what D_0 is...

Demultiplexers

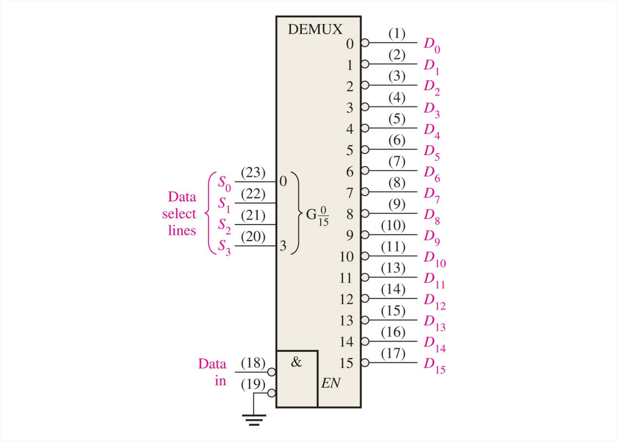


Figure 30: A decoder is very similar to a demux, but it has to have the data input lines at the bottom left.

Conclusion

Unit 2 Summary

Functions of Combinatorial Logic

Reading: 6-1 - 6-6 (Tuesday)

Reading: 6-7 - 6-11 (Thursday)

1. Half-Adders and Full-Adders
    - Example from study guide
    - Propagation delays
2. Comparators
    - The XNOR gate
    - Multi-bit comparators
    - Inequalities
3. Decoders/Encoders
    - Binary to decimal circuits
    - Decimal to binary circuits
4. Multiplexing and Demultiplexing