

COM 251 Logic Design and Circuits

Combinational Logic Design: Examples

Prof. Dr. Halûk Gümüşkaya

haluk.gumuskaya@gediz.edu.tr

haluk@gumuskaya.com http://www.gumuskaya.com

Computer Engineering Department

GEDİZÜNİVERSİTESİ
izmir

Thursday, October 27, 2011

1

Acknowledgement

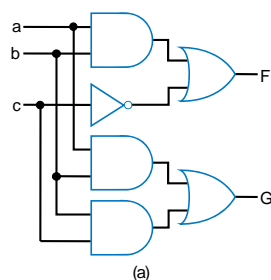
The slides have been based in-part upon original slides of a number of books including:

- *Digital Design with RTL Design, Verilog and VHDL*, 2nd ed., Frank Vahid, John Wiley, 2011.

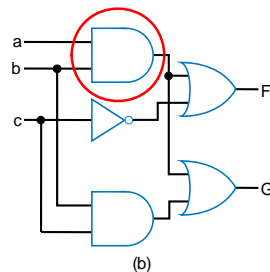
2

Multiple-Output Circuits

- Many circuits have more than one output
- Can give each a separate circuit, or can share gates
- Ex: $F = \underline{ab} + c'$, $G = \underline{ab} + bc$



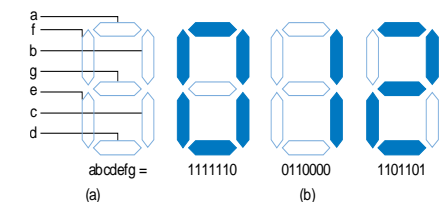
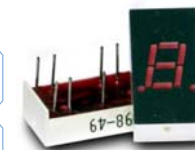
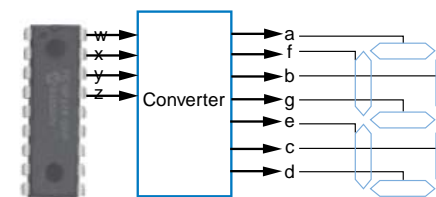
Option 1: Separate circuits



Option 2: Shared gates

3

Multiple-Output Example: BCD to 7-Segment Converter

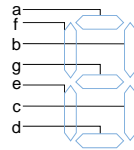


4

Multiple-Output Example: BCD to 7-Segment Converter

TABLE 2-4 4-bit binary number to seven-segment display truth table

w	x	y	z	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0



$$a = w'x'y'z' + w'x'yz' + w'xy'z + w'xyz + w'xy'z' + w'xyz' + wx'y'z' + wx'y'z$$

$$b = w'x'y'z' + w'x'y'z + w'x'yz' + w'xyz + w'xy'z' + w'xyz' + wx'y'z' + wx'y'z$$

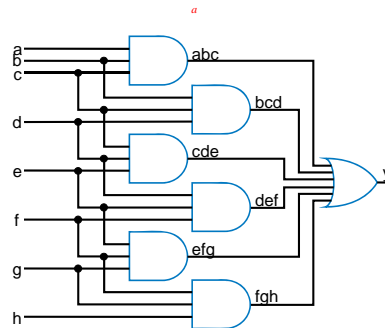
...

Combinational Logic Design Process

Step	Description
Step 1: Capture behavior	Capture the function Create a truth table or equations, <i>whichever is most natural for the given problem</i> , to describe the desired behavior of each output of the combinational logic.
Step 2: Convert to circuit	2A: Create equations This substep is only necessary if you captured the function using a truth table instead of equations. Create an equation for each output by ORing all the minterms for that output. Simplify the equations if desired. 2B: Implement as a gate-based circuit For each output, create a circuit corresponding to the output's equation. (Sharing gates among multiple outputs is OK optionally.)

Example: Three 1s Pattern Detector

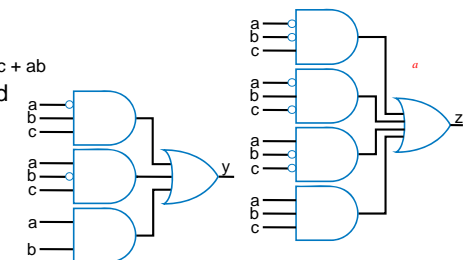
- Problem: Detect three consecutive 1s in 8-bit input: abcdefgh
 - 00011101 → 1
 - 10101011 → 0
 - 11110000 → 1
- Step 1: Capture the function**
 - Truth table or equation?
 - Truth table too big: 2⁸=256 rows
 - Equation: create terms for each possible case of three consecutive 1s
 - y = abc + bcd + cde + def + efg + fgh**
- Step 2a: Create equation** -- already done
- Step 2b: Implement** as a gate-based circuit



Example: Number of 1s Counter

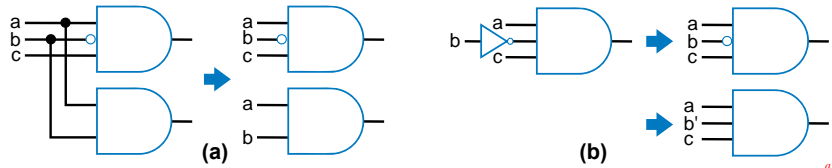
- Problem: Output in binary on two outputs yz the # of 1s on three inputs
 - 010 → 01
 - 101 → 10
 - 000 → 00
- Step 1: Capture the function**
 - Truth table or equation?
 - Truth table is straightforward
- Step 2a: Create equations**
 - y = a'bc + ab'c + abc' + abc
 - z = a'b'c + a'bc' + ab'c' + abc
 - Optional: Let's simplify y:
 - y = a'bc + ab'c + ab(c' + c) = a'bc + ab'c + ab
- Step 2b: Implement** as a gate-based circuit

Inputs			(# of 1s)	Outputs	
a	b	c		y	z
0	0	0	(0)	0	0
0	0	1	(1)	0	1
0	1	0	(1)	0	1
0	1	1	(2)	1	0
1	0	0	(1)	0	1
1	0	1	(2)	1	0
1	1	0	(2)	1	0
1	1	1	(3)	1	1



Simplifying Notations

- Used in previous circuit



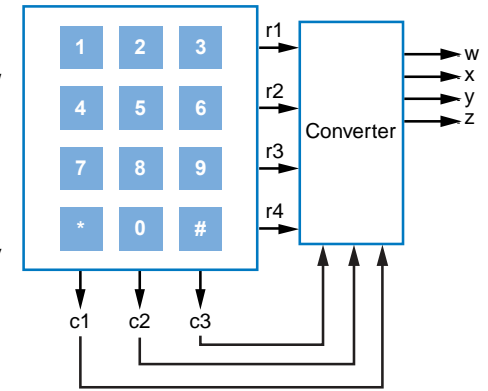
List inputs multiple times
→ Less wiring in drawing

- Draw **inversion bubble** rather than inverter, or
- List input as **complemented**.

9

Example: Keypad Converter

- Keypad has 7 outputs
 - One per row
 - One per column
- Key press sets one row and one column output to 1
 - Press "5" → r2=1, c2=1
- Goal: Convert keypad outputs into 4-bit binary number
 - 0-9 → 0000 to 1001
 - * → 1010, # → 1011
 - nothing pressed: 1111



10

Example: Keypad Converter

- Step 1: Capture behavior
 - Truth table too big (2^7 rows); equations not clear either
 - Informal table can help

TABLE 2.7 Informal table for the 12-button keypad to 4-bit code converter.

Button	Signals	4-bit code outputs			
		w	x	y	z
1	r1 c1	0	0	0	1
2	r1 c2	0	0	1	0
3	r1 c3	0	0	1	1
4	r2 c1	0	1	0	0
5	r2 c2	0	1	0	1
6	r2 c3	0	1	1	0
7	r3 c1	0	1	1	1
8	r3 c2	1	0	0	0
9	r3 c3	1	0	0	1
*	r4 c1	1	0	1	0
0	r4 c2	0	0	0	0
#	r4 c3	1	0	1	1
(none)		1	1	1	1

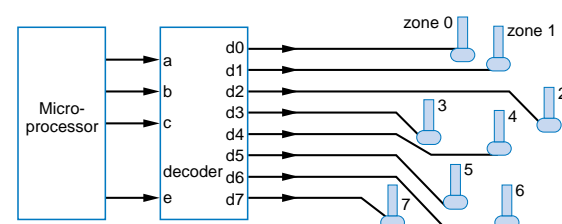
Step 2b: Implement as circuit (note sharable gates) ...

$$\begin{aligned}
 w &= r3c2 + r3c3 + r4c1 + r4c3 + r1'r2'r3'r4'c1'c2'c3' \\
 x &= r2c1 + r2c2 + r2c3 + r3c1 + r1'r2'r3'r4'c1'c2'c3' \\
 y &= r1c2 + r1c3 + r2c3 + r3c1 + r4c1 + r4c3 + r1'r2'r3'r4'c1'c2'c3' \\
 z &= r1c1 + r1c3 + r2c2 + r3c1 + r3c3 + r4c3 + r1'r2'r3'r4'c1'c2'c3'
 \end{aligned}$$

11

Example: Sprinkler Controller

- Microprocessor outputs which zone to water (e.g., cba=110 means zone 6) and enables watering (e=1)
- Decoder should set appropriate valve to 1



Step 1: Capture behavior

$$\begin{aligned}
 d0 &= a'b'c'e \\
 d1 &= a'b'ce \\
 d2 &= a'bc'e \\
 d3 &= a'bce \\
 d4 &= ab'c'e \\
 d5 &= ab'ce \\
 d6 &= abc'e \\
 d7 &= abce
 \end{aligned}$$

Equations seem like a natural fit

12

Example: Sprinkler Controller

- Step 2b: Implement as circuit

