Review Outline

- Digital Circuits
  - Circuits that represent signals as discrete levels, rather than as a continuous range
- Digital Circuits include
  - Combinational Circuits
  - Sequential Circuits
- Bubble Pushing/DeMorgan Theorem

Combinational-Circuit Analysis

- *Combinational* circuits -- outputs depend only on current inputs (not on history), which means combinational circuit has no memory.
- Combinational analysis:
  - Truth table
  - K map
  - Optimized expressions/functions
  - simulation/test bench
    - Write functional description in HDL or build a circuit directly
    - Define test conditions/test vectors
    - Compare circuit output with functional description (or known-good realization)
    - Repeat for “random” test vectors

Switching Theory

- Since there are so many Boolean equations that produce the same truth table (and are therefore equivalent), there must be some way to convert from one form to another.

Postulate	$x + 0 = x$	$x \cdot 1 = x$
Postulate	$x + x' = 1$	$x \cdot x' = 0$
Theorem	$x + x = x$	$x \cdot x = x$
Theorem	$x + 1 = 1$	$x \cdot 0 = 0$
Theorem, involution	$(x')' = x$	
Postulate, commutative	$x + y = y + x$	$xy = yx$
Theorem, associative	$x + (y + z) = (x + y) + z$	$x(yz) = (xy)z$
Postulate, distributive	$x(y + z) = xy + xz$	$x + yz = (x + y)(x + z)$
Theorem, DeMorgan	$(x + y)' = x'y'$	$(xy)' = x' + y'$
Theorem, absorption	$x + xy = x$	$x(x + y) = x$

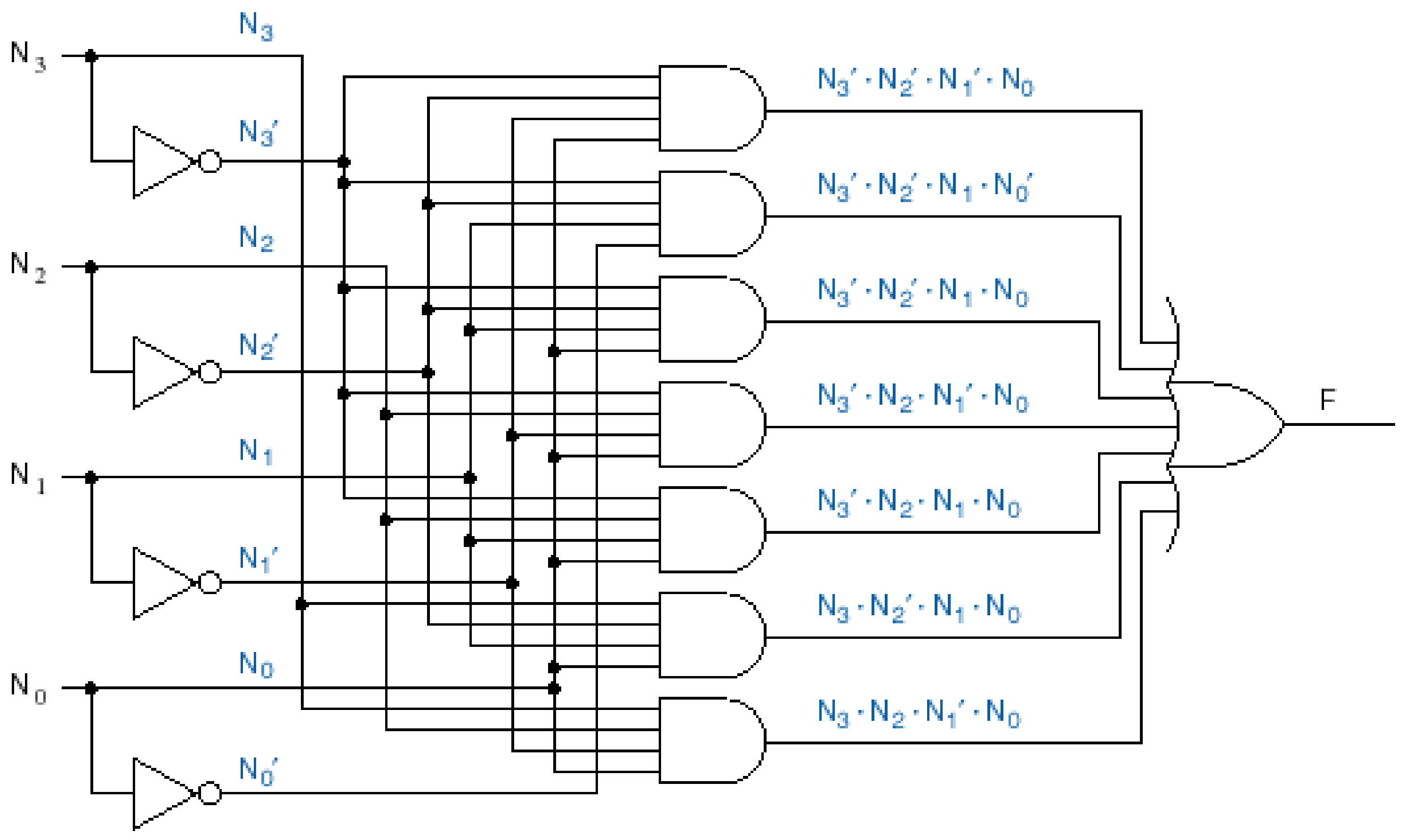
Brute-force design

- Truth table --> canonical sum (sum of minterms)
- Example: prime-number & 1 detector
  - 4-bit input, $N_3N_2N_1N_0$

$F = \Sigma_{N3N2N1N0}(1,2,3,5,7,11,13)$

row	N_3	N_2	N_1	N_0	F
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	0

$$
\begin{aligned}
F &= \Sigma_{N_3,N_2,N_1,N_0}(1,2,3,5,7,11,13) \\
&= N_3'\cdot N_2'\cdot N_1'\cdot N_0 + N_3'\cdot N_2'\cdot N_1\cdot N_0' + N_3'\cdot N_2'\cdot N_1\cdot N_0 + N_3'\cdot N_2\cdot N_1'\cdot N_0 \\
&\quad + N_3'\cdot N_2\cdot N_1\cdot N_0 + N_3\cdot N_2'\cdot N_1\cdot N_0 + N_3\cdot N_2\cdot N_1'\cdot N_0
\end{aligned}
$$



N_3
N_3
N_3'
N_2
N_2
N_2'
N_1
N_1
N_1'
N_0
N_0
N_0'
$N_3'\cdot N_2'\cdot N_1'\cdot N_0$
$N_3'\cdot N_2'\cdot N_1\cdot N_0'$
$N_3'\cdot N_2'\cdot N_1\cdot N_0$
$N_3'\cdot N_2\cdot N_1'\cdot N_0$
$N_3'\cdot N_2\cdot N_1\cdot N_0$
$N_3\cdot N_2'\cdot N_1\cdot N_0$
$N_3\cdot N_2\cdot N_1'\cdot N_0$
F

Algebraic simplification

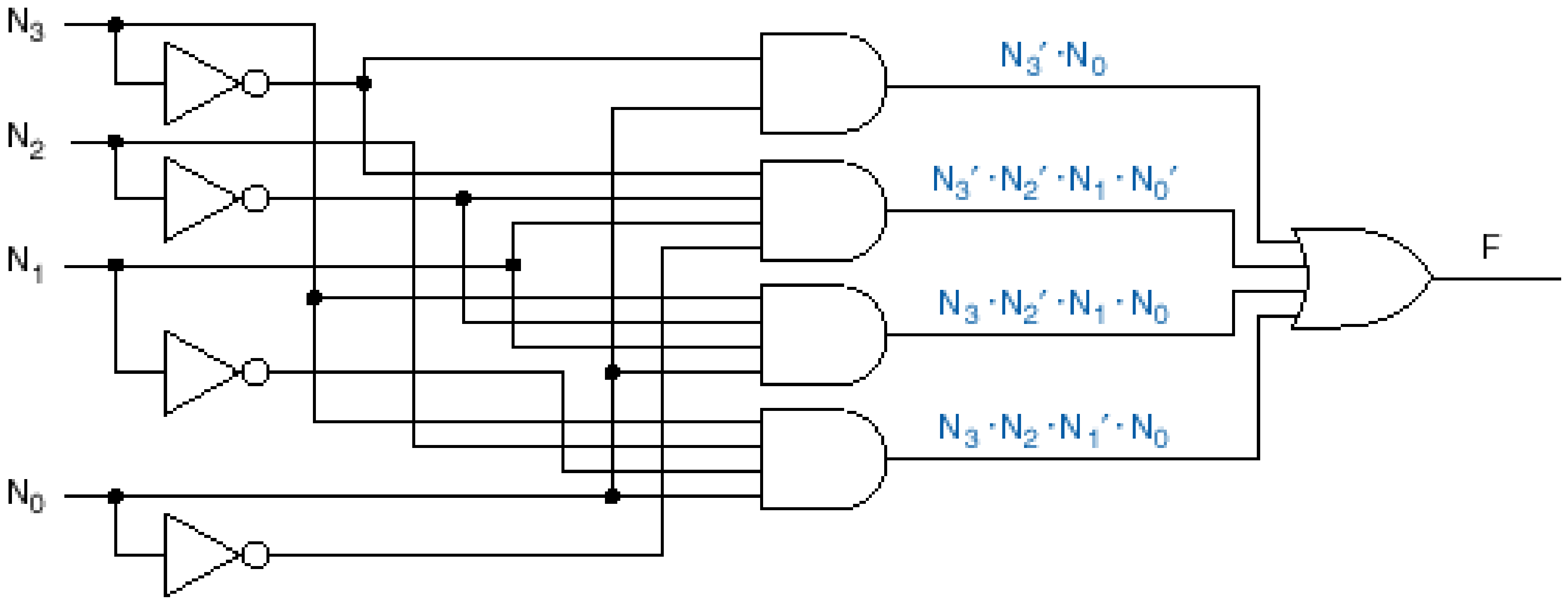
- Theorem T10

$$X \cdot Y + X \cdot Y' = X$$

$$\begin{aligned}
F &= \Sigma_{N_3,N_2,N_1,N_0}(1, 3, 5, 7, 2, 11, 13) \\
&= N_3' \cdot N_2' N_1' N_0 + N_3' \cdot N_2' \cdot N_1 \cdot N_0 + N_3' \cdot N_2 \cdot N_1' \cdot N_0 + N_3' \cdot N_2 \cdot N_1 \cdot N_0 + \ldots \\
&= (N_3' \cdot N_2' \cdot N_1' \cdot N_0 + N_3' \cdot N_2' \cdot N_1 \cdot N_0) + (\cdot N_3' \cdot N_2 \cdot N_1' \cdot N_0 + N_3' \cdot N_2 \cdot N_1 \cdot N_0) + \ldots \\
&= N_3' N_2' \cdot N_0 + N_3' \cdot N_2 \cdot N_0 + \ldots
\end{aligned}$$

- Reduce number of gates and gate inputs

Resulting circuit


N_3
N_2
N_1
N_0
$N_3' \cdot N_0$
$N_3' \cdot N_2' \cdot N_1 \cdot N_0'$
$N_3 \cdot N_2' \cdot N_1 \cdot N_0$
$N_3 \cdot N_2 \cdot N_1' \cdot N_0$
F

F=?

use K-map to optimize

Combinational-Circuit Design

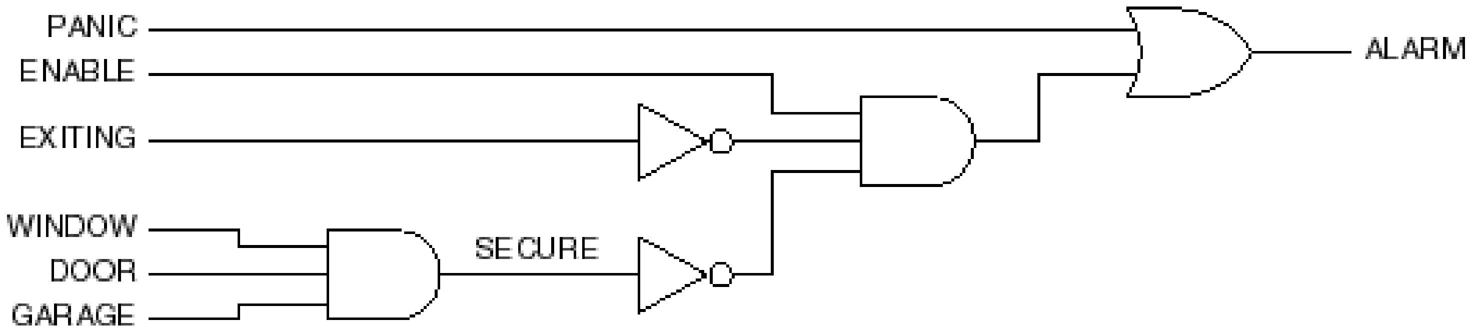
- Sometimes you can write an equation or equations directly using "logic" (the kind in your brain).
- Basic gates: **Inverter/AND/NAND/OR/NOR/XOR/XNOR**
- Ex: design an alarm system to meet the following specifications
  1. Whenever the panic signal is on, the alarm should be on
  2. When any of the window, door and garage open and the alarm system enabled and the existing condition is false, the alarm should be on
  - Note: on – '1', off – '0', open – '0', false – '0'

Combinational-Circuit Design (cont)

- alarm system functions

$$\text{ALARM} = \text{PANIC} + \text{ENABLE} \cdot \text{EXITING}' \cdot \text{SECURE}'$$
$$\text{SECURE} = \text{WINDOW} \cdot \text{DOOR} \cdot \text{GARAGE}$$
$$\text{ALARM} = \text{PANIC} + \text{ENABLE} \cdot \text{EXITING}' \cdot (\text{WINDOW} \cdot \text{DOOR} \cdot \text{GARAGE})'$$

- Corresponding circuit



Karnaugh Maps (K-map)

- A K-map is a collection of squares
  - Each square represents a minterm
  - The collection of squares is a graphical representation of a Boolean function
  - Adjacent squares differ in the value of one variable
  - Alternative algebraic expressions for the same function are derived by recognizing patterns of squares
- The K-map can be viewed as
  - A reorganized version of the truth table
  - A topologically-warped Venn diagram as used to visualize sets in algebra of sets

Some Uses of K-Maps

- Provide a means for:
  - Finding optimum or near optimum
    - SOP and POS standard forms, and
    - two-level AND/OR and OR/AND circuit implementations

  for functions with small numbers of variables
  - Visualizing concepts related to manipulating Boolean expressions, and
  - Demonstrating concepts used by computer-aided design programs to simplify large circuits

Two Variable Maps

- **A 2-variable Karnaugh Map:**
  - Note that minterm **m0** and minterm **m1** are “adjacent” and differ in the value of the variable **y**
  - Similarly, minterm **m0** and minterm **m2** differ in the **x** variable.
  - Also, **m1** and **m3** differ in the **x** variable as well.
  - Finally, **m2** and **m3** differ in the value of the variable **y**

	y = 0	**y = 1**
x = 0	$m_0 = \bar{x}\,\bar{y}$	$m_1 = \bar{x}\,y$
x = 1	$m_2 = x\,\bar{y}$	$m_3 = x\,y$

K-Map and Truth Tables

- The K-Map is just a different form of the truth table.
- Example – Two variable function:
  - We choose a,b,c and d from the set {0,1} to implement a particular function, F(x,y).

Function Table

Input Values (x,y)	Function Value F(x,y)
0 0	a
0 1	b
1 0	c
1 1	d

K-Map

	y = 0	y = 1
x = 0	a	b
x = 1	c	d

K-Map Function Representation

- **Example: F(x,y) = x**

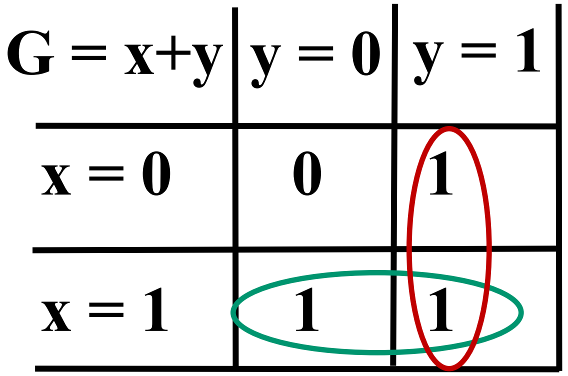
F = x	y = 0	y = 1
x = 0	0	0
x = 1	1	1

- **For function F(x,y), the two adjacent cells containing 1's can be combined using the Minimization Theorem:**

$$F(x, y) = x\,\overline{y} + x\,y = x$$

K-Map Function Representation

- **Example: G(x,y) = x + y**



G = x+y	y = 0	y = 1
x = 0	0	1
x = 1	1	1

- **For G(x,y), two pairs of adjacent cells containing 1's can be combined using the Minimization Theorem:**

$$G(x,y) = \left(x\,\bar{y} + x\,y\right) + \left(xy + \bar{x}\,y\right) = x + y$$

Duplicate xy

Bubble Pushing

- Start with network of AND / OR gates
- Convert to NAND / NOR + inverters
- Push bubbles around to simplify logic
  - Remember DeMorgan's Law