B38DB: Digital Design and Programming
Combinational Logic Design – Logic Gates

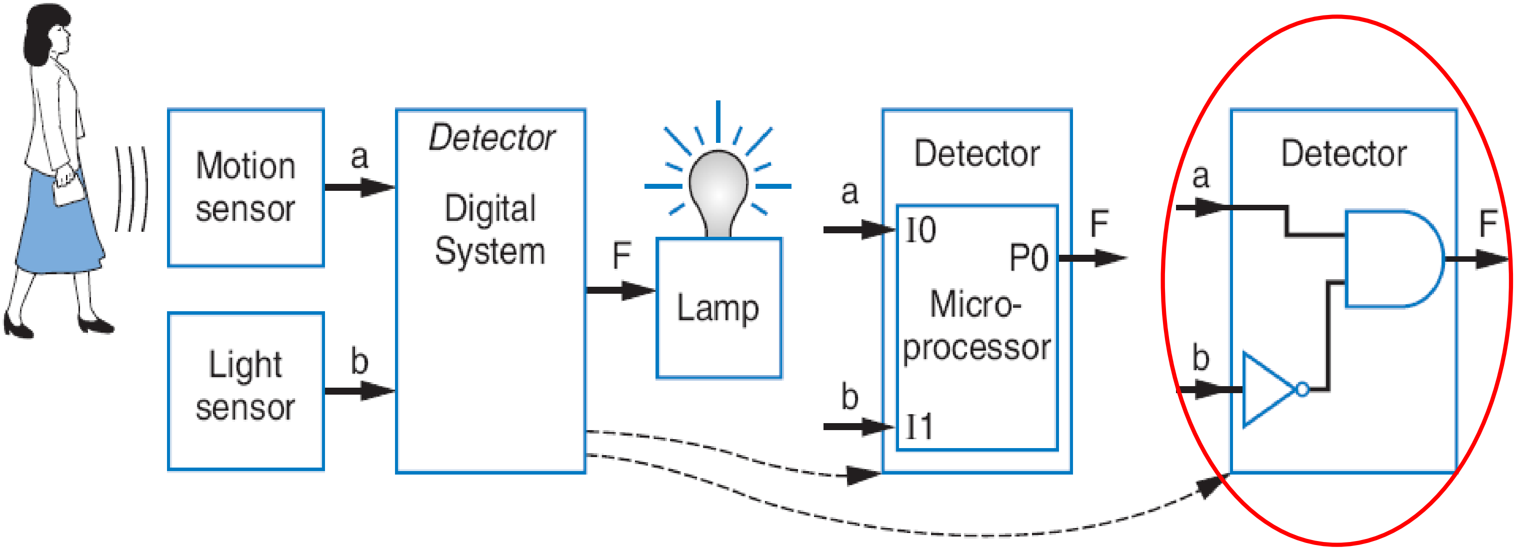
Mustafa Suphi Erden

Heriot-Watt University
School of Engineering & Physical Sciences
Electrical, Electronic and Computer Engineering
Room: EM 2.01
Phone: 0131-4514159
E-mail: m.s.erden@hw.ac.uk

Combinational Circuit

A digital circuit whose

outputs depend solely on the <u>present</u> combination of the inputs

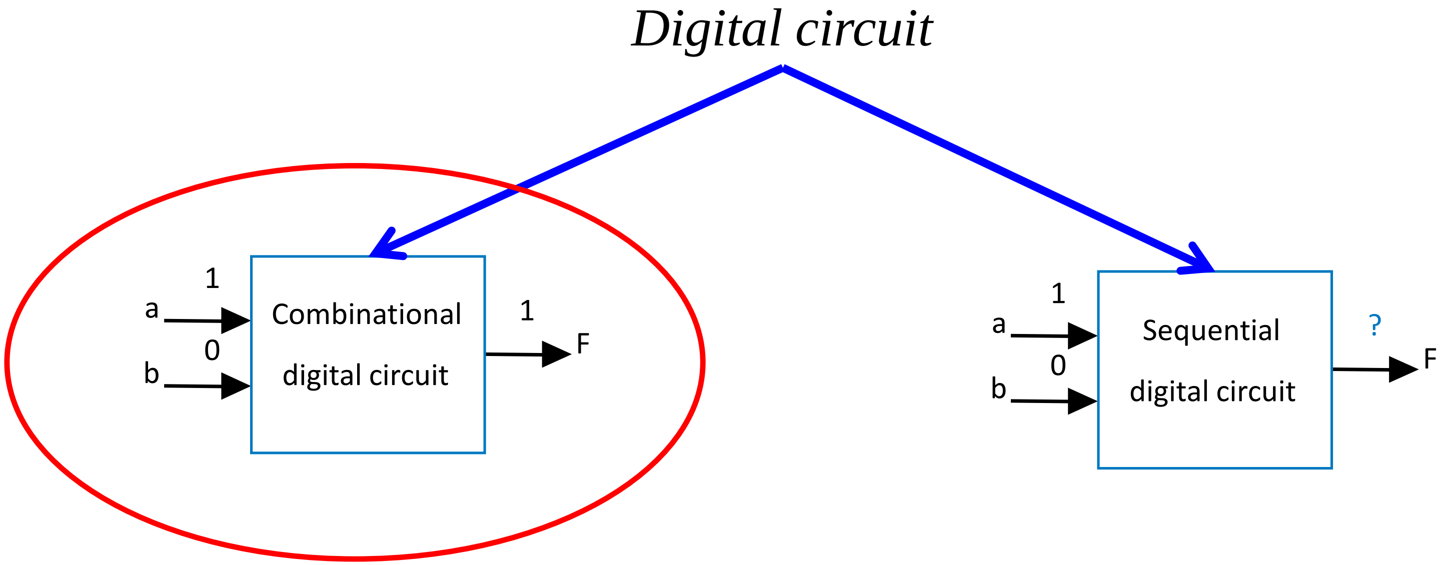


Combinational Circuit versus Sequential Circuit

<u>Sequential circuit</u>

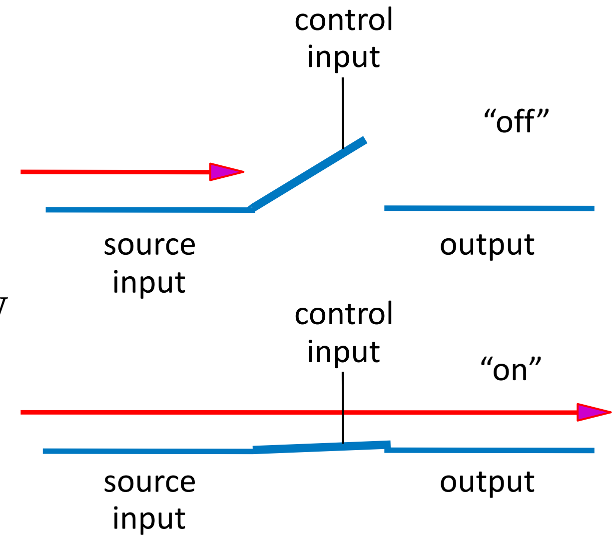
A digital circuit whose

outputs depend on the <u>present and previous</u> combination of the inputs

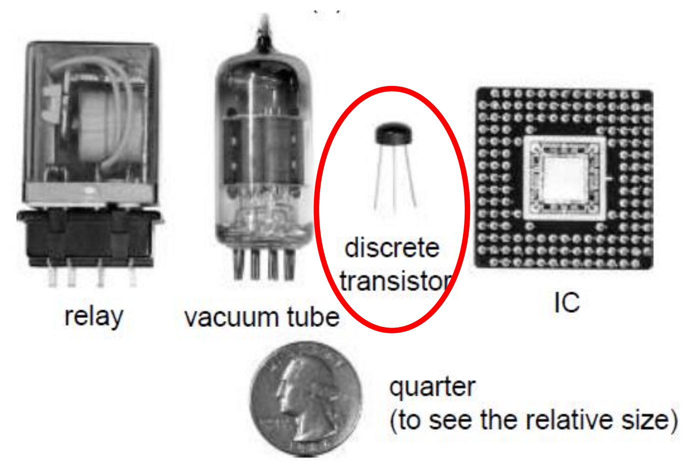


Switches

- A switch has three parts
  - Source input, and output
    - Current wants to flow from source input to output
  - Control input
    - Voltage that controls whether that current can flow



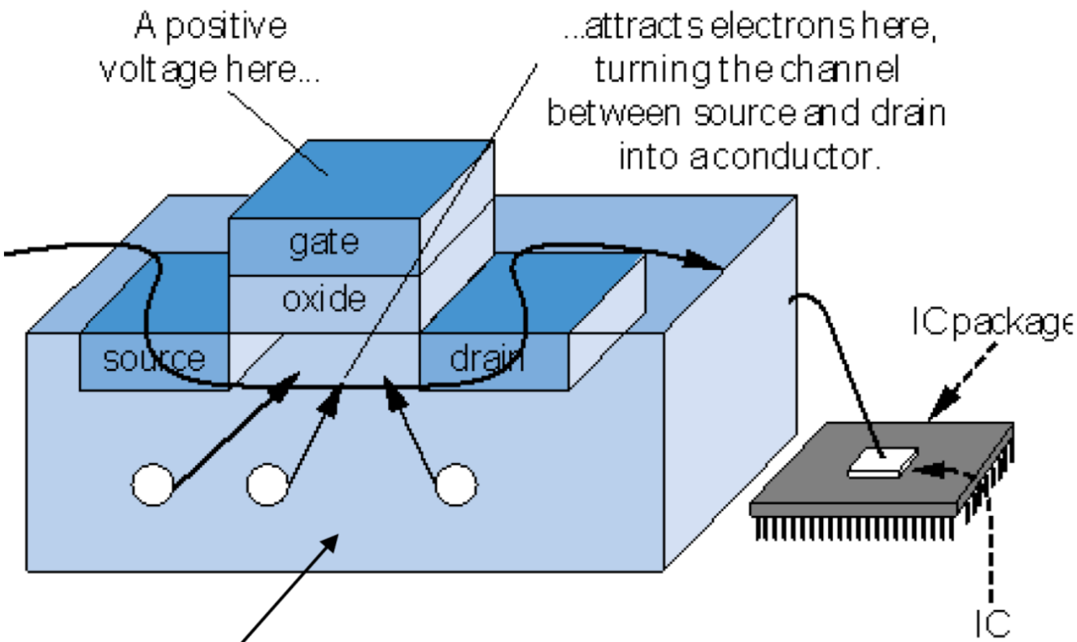
- The amazing shrinking switch
  - 1930s: Relays
  - 1940s: Vacuum tubes
  - 1950s: Discrete transistor
  - 1960s: Integrated circuits (ICs)
    - Initially just a few transistors on IC
    - Then tens, hundreds, thousands...



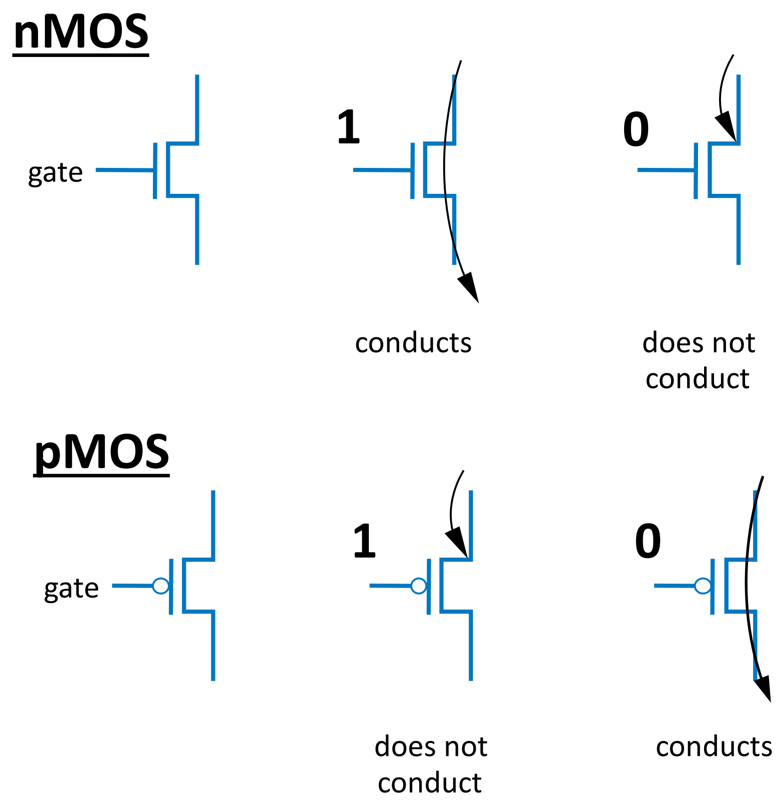
CMOS Transistor

- CMOS transistor
  - Basic switch in modern ICs
- Two types of CMOS transistor

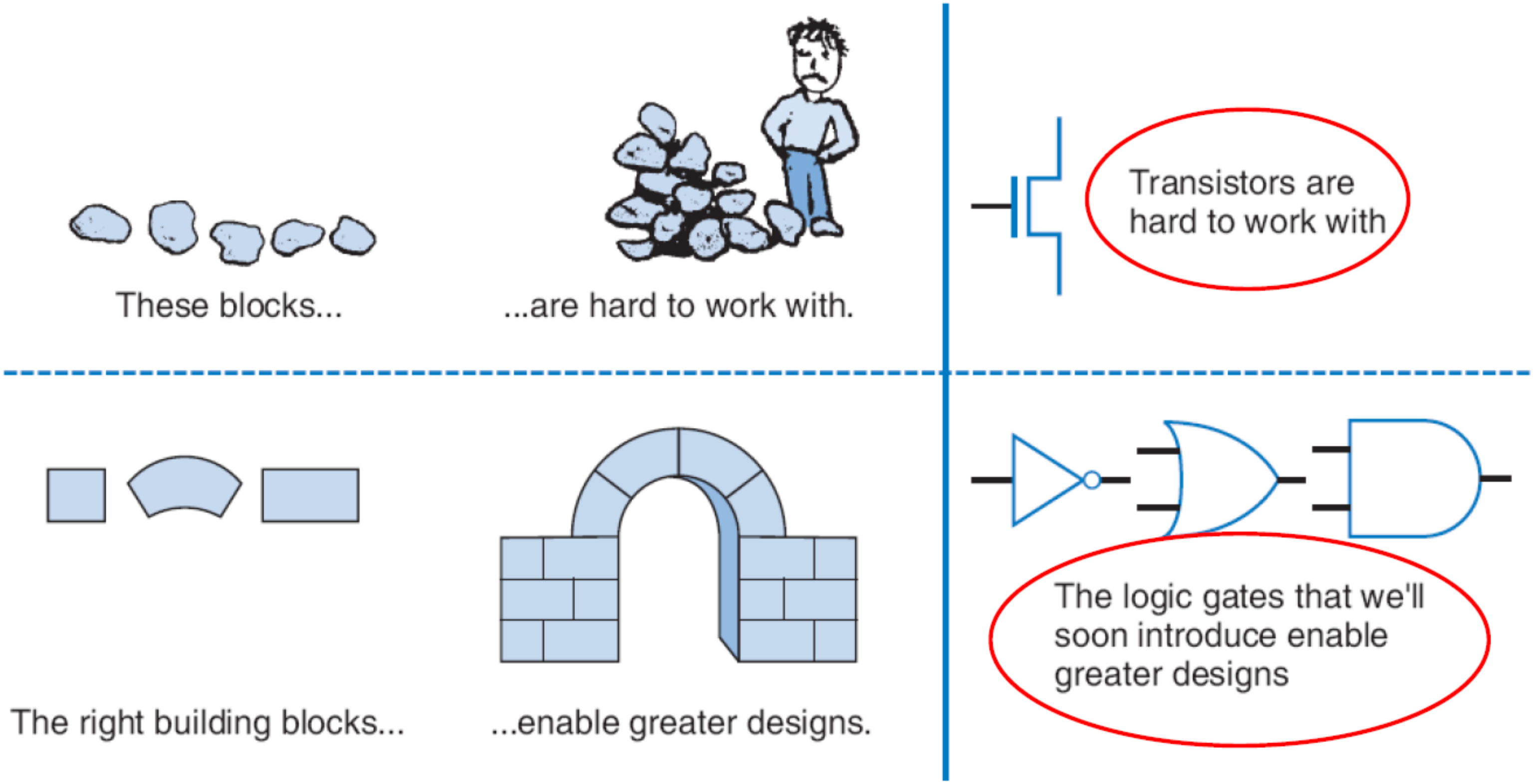
  nMOS and pMOS



Silicon – not quite a conductor or insulator:
Semiconductor

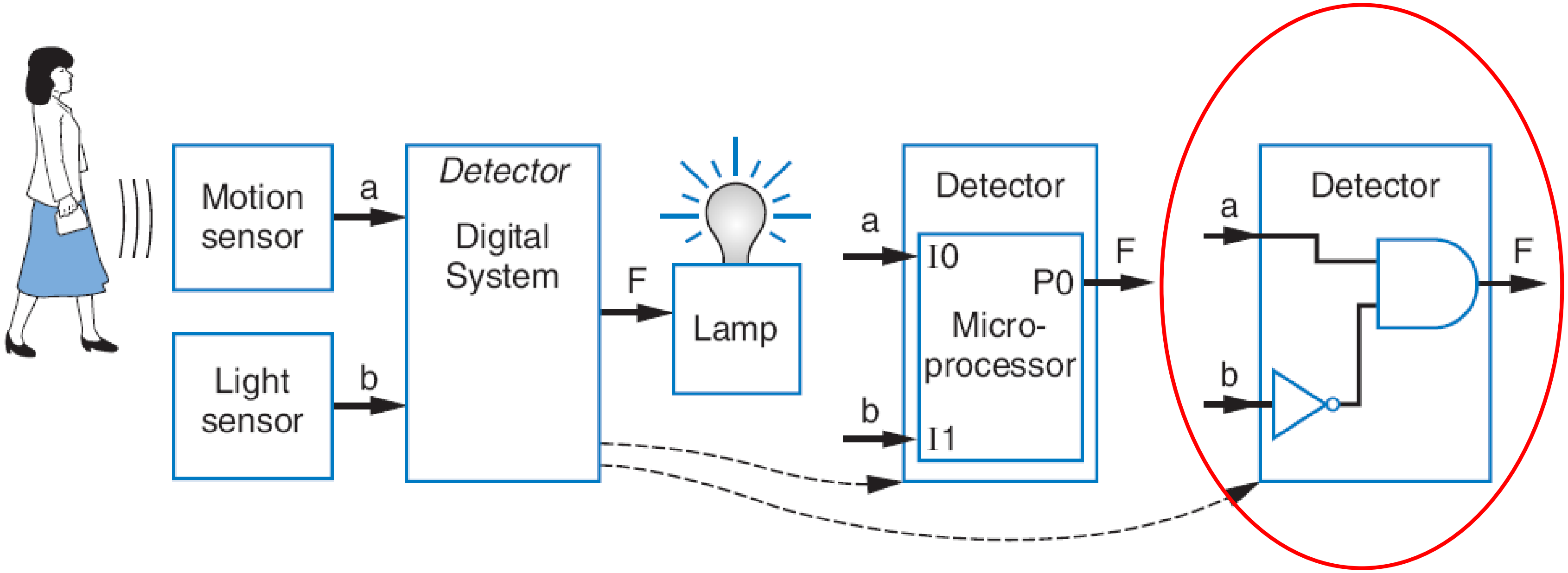


Boolean Logic Gates
Building Blocks for Digital Circuits
(Switches are Hard to Work With)



"Logic gates" are better digital circuit building blocks than switches (transistors).

Building Circuits Using Gates



- Turn on the lamp (F=1)

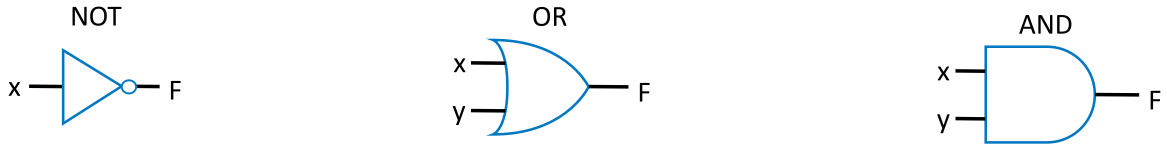
  if motion sensed (a=1) **AND** there is no light (b=0)

- F = a **AND** (**NOT** (b))

Relating Boolean Algebra to Digital Design

- Implement **Boolean operators** using transistors
  - Call those implementations ***logic gates***.

Symbol



Truth table

NOT

x	F
0	1
1	0

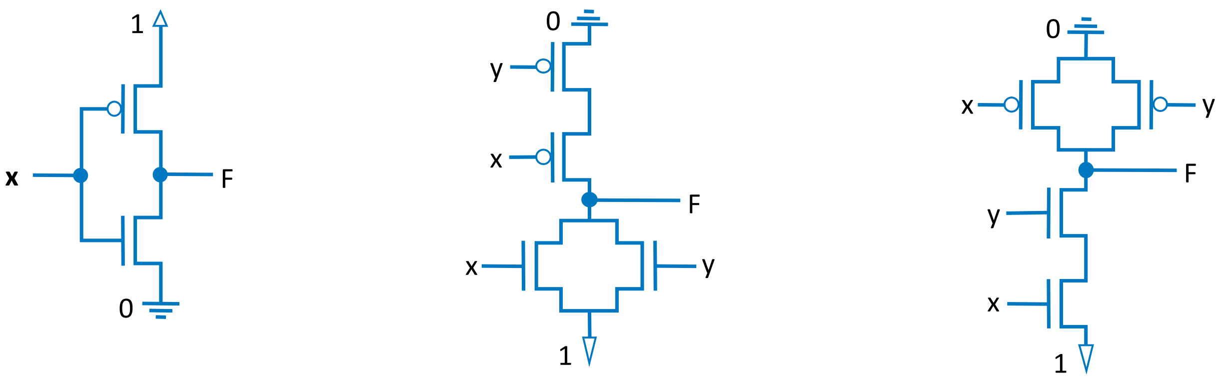
OR

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

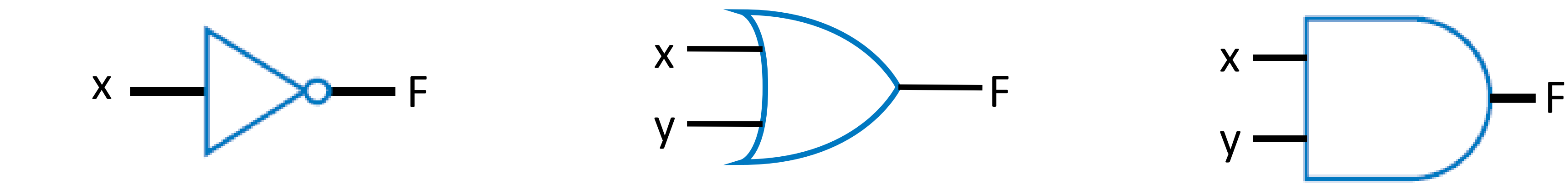
AND

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

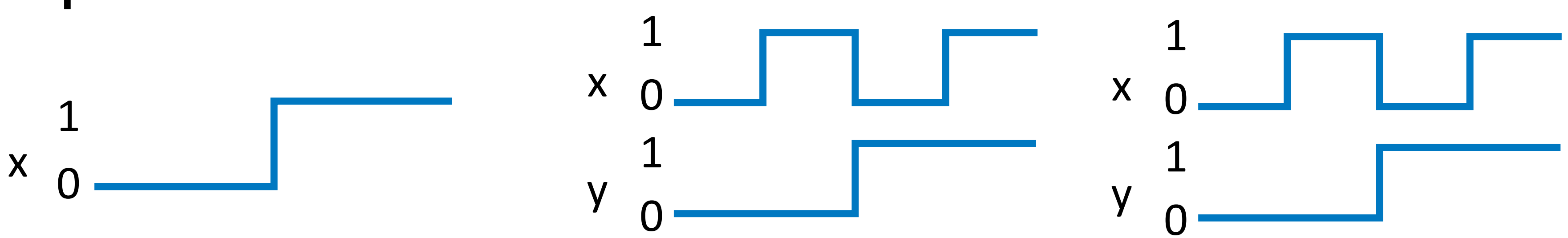
Transistor circuit



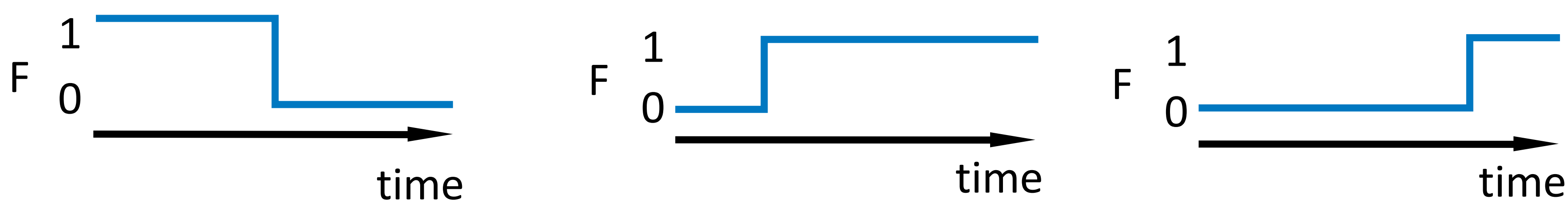
NOT/OR/AND Logic Gate Timing Diagrams



Input:

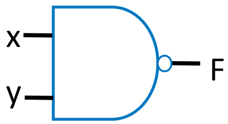


Output:



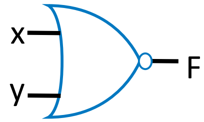
More Gates

NAND



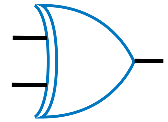
x	y	F
0	0	1
0	1	1
1	0	1
1	1	0

NOR



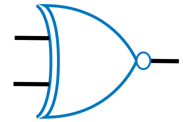
x	y	F
0	0	1
0	1	0
1	0	0
1	1	0

XOR (⊕)



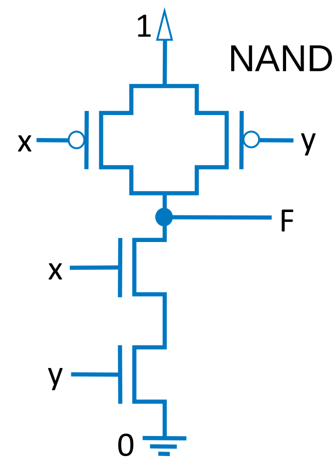
x	y	F
0	0	0
0	1	1
1	0	1
1	1	0

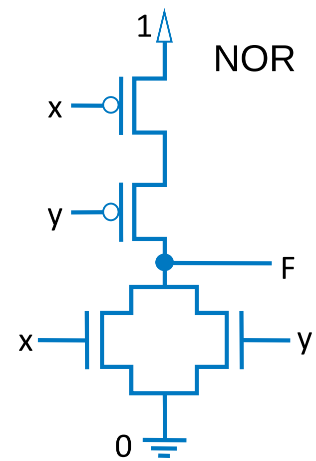
XNOR (⊗)



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

- NAND: Opposite of AND (“NOT AND”)
- NOR: Opposite of OR (“NOT OR”)
- XOR: Exactly 1 input is 1, for 2-input XOR.
  (For more inputs – odd number of 1s)
- XNOR: Opposite of XOR (“NOT XOR”)

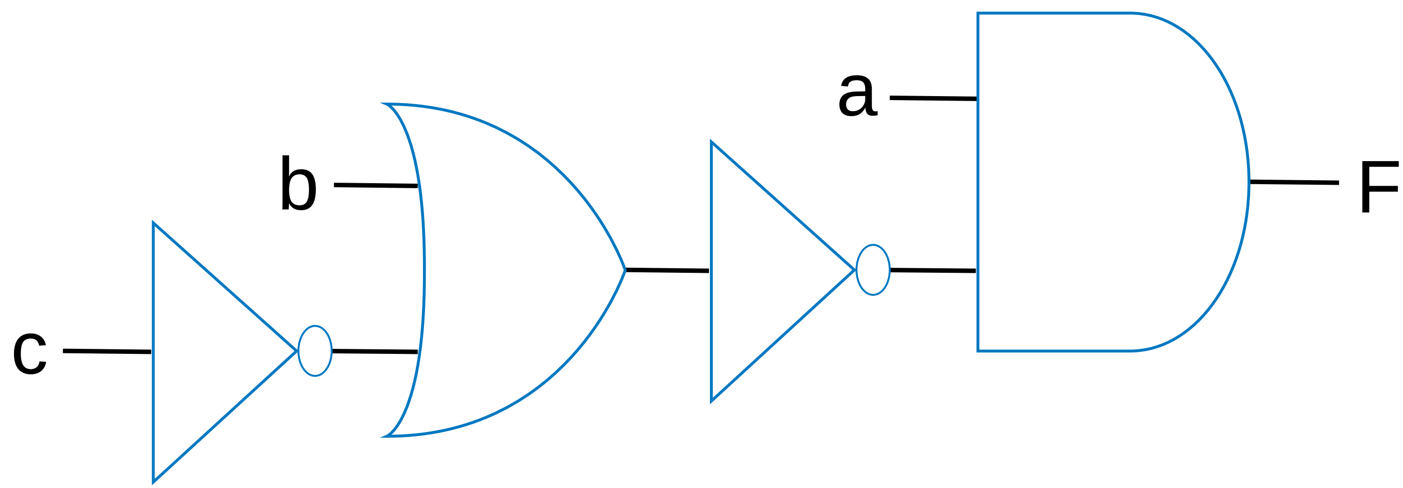




Example: Converting a Boolean Equation into a Circuit of Logic Gates

- Convert the following equation to logic gates:

F = a AND NOT(b OR NOT(c))

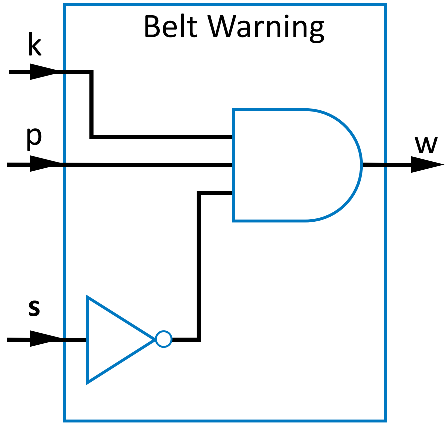


Example: Seat Belt Warning Light System
Converting to Boolean Equations

- Design circuit for warning light
- Sensors
  - s=1: seat belt fastened
  - k=1: key inserted
  - p=1: person in seat
- Capture Boolean equation
  - person in seat, and seat belt not fastened, and key inserted

w = p **AND** (**NOT** (s)) **AND** k

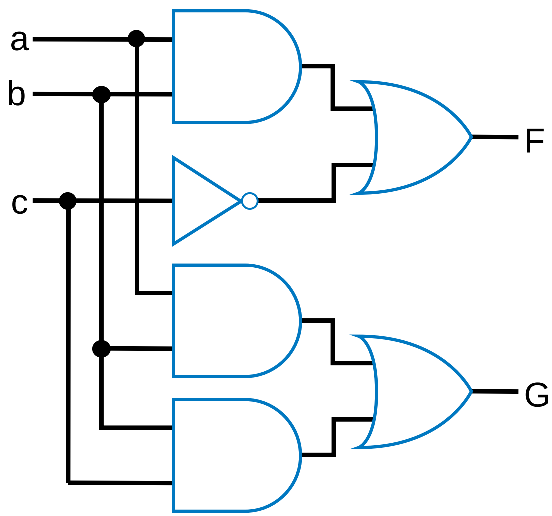
- Convert equation to circuit



- Notice
  - Boolean algebra enables easy capture as equation and conversion to circuit

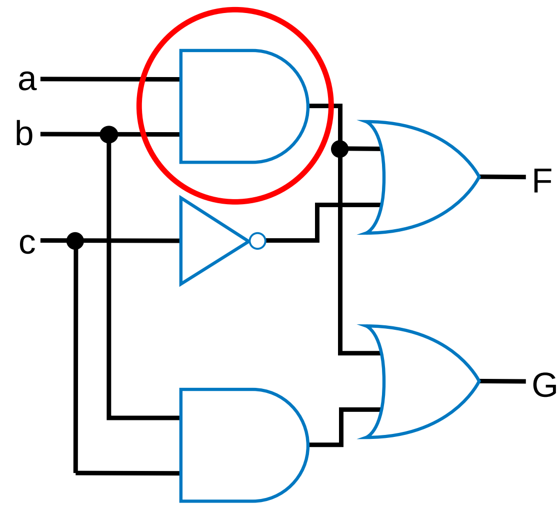
Multiple-Output Circuits

- Many circuits have more than one output
- They can have a separate circuit, or they can share gates
- Ex: F = ab + c', G = ab + bc



(a)

Option 1: Separate circuits



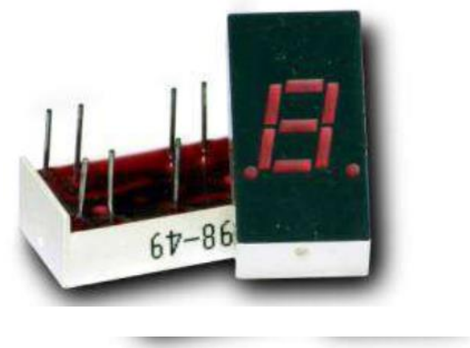
(b)

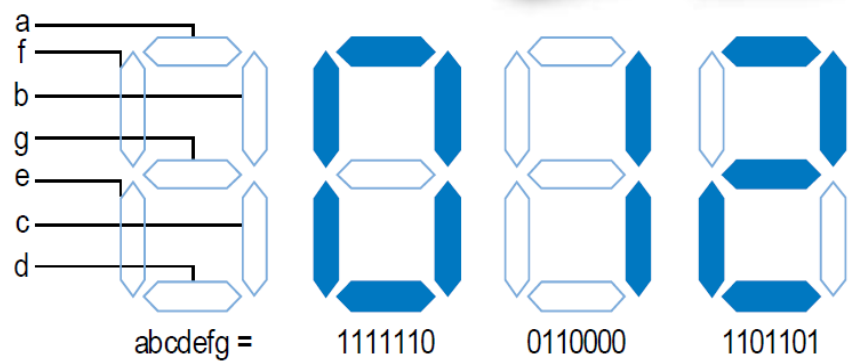
Option 2: Shared gates

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) (b)

a = w’x’y’z’ + w’x’yz’ + w’x’yz + w’xy’z +
w’xyz’ + w’xyz + wx’y’z’ + wx’y’z

b = w’x’y’z’ + w’x’y’z + w’x’yz’ + w’x’yz +
w’xy’z’ + w’xyz + wx’y’z’ + wx’y’z