Digital Logic Circuits

- **Digital circuits make up all computers and computer systems. The operation of digital circuits is based on Boolean algebra, the mathematics of binary numbers.**
- **Boolean algebra is very simple, having only three basic functions, AND, OR, and NOT.**
- **These basic functions can be combined in many ways to provide all the functions required in the central processor of a digital computer.**
- **Digital circuits operate by performing Boolean operations on binary numbers (more about binary numbers in EE 2310).**

First Boolean Function: NOT

- **NOT is the simplest logical function: 1 input and 1 output.**
- **NOT is defined as follows: "The output f of NOT, given an input a, is the complement or opposite of the input." Or :** $f = \bar{a}$
- **Since NOT can have only a 0 or 1 input, the output of NOT is the reverse, or complement, of the input.**
  - **If the input of NOT is 1, the output is 0.**
  - **If the input of NOT is 0, the output is 1.**
- **The NOT function is called inversion, and the digital circuit which inverts is an inverter. The electronic circuit symbol for NOT is:**

The Truth Table

- **The inverter input/output relationship, with one input and output, is easy to show.**
- **For complex functions, an I/O table is helpful.**
- **We call this a truth table, since it indicates the 1 ("true") outputs, although it normally shows outputs for all input combinations.**
- **We will demonstrate some Boolean functions using truth tables.**

Boolean Function Truth Table

Input x	Input y	Output f
0	0	0
0	1	1
1	0	1
1	1	0

Note: This 2-input truth table shows the output f for all possible combinations of the binary inputs x and y.

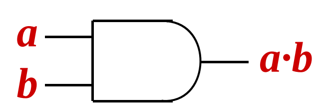
Second Boolean Function: AND

- **AND has <u>two or more inputs</u>.**
- **The truth table for a two-input AND with inputs *a* and *b* is shown in the chart.**
- **AND is defined as follows: *a* AND *b* = 1 if and only if (iff) *a* = 1 and *b* = 1.**
- **Mathematically, we represent "*a* AND *b*" as *a*·*b* (an unfortunate choice).**
- **AND may have more than two inputs, i. e.: *a* AND *b* AND *c* AND *d*.**
- **The electronic circuit symbols for 2- and 4-input ANDs are shown at the right.**
- **Regardless of the number of inputs, the output of AND is 1 iff <u>all inputs are 1</u>.**

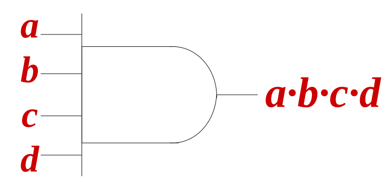
AND Truth Table

a	b	a AND b
0	0	0
0	1	0
1	0	0
1	1	1

2-Input AND



4-Input AND



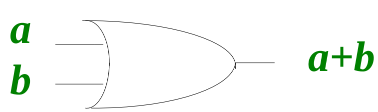
Third Boolean Function: OR

- **OR has <u>two or more inputs</u>.**
- **The OR truth table for two inputs *a*, *b* is shown in the adjacent chart.**
- **OR is defined as follows: *a* OR *b* = 1 if either *a* or *b* or both *a* and *b* = 1.**
- **Mathematically, we represent "*a* OR *b*" as a + b (another bad choice).**
- **OR may have more than two inputs, i. e.: *a* OR *b* OR *c* OR *d*.**
- **The electronic circuit symbols for 2- and 4- input ORs are shown at the right.**
- **Regardless of the number of inputs, the output of OR is 0 iff all inputs are 0.**

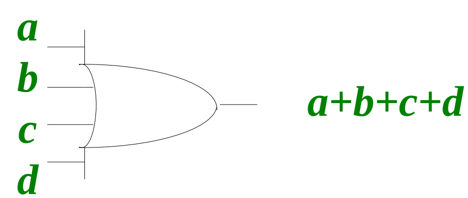
OR Truth Table

a	b	a OR b
0	0	0
0	1	1
1	0	1
1	1	1

2-Input OR

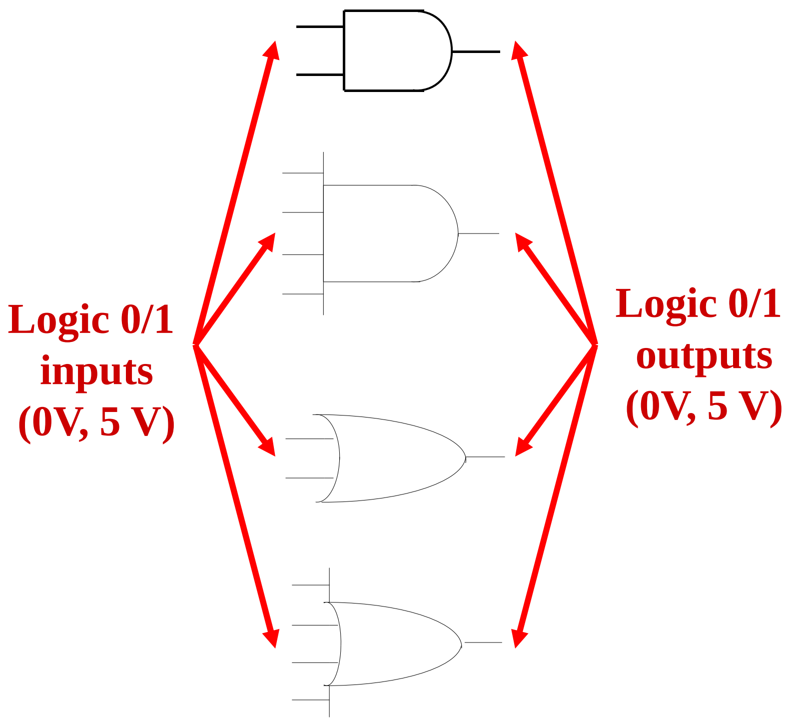


4-Input OR



Logic “1” and “0”

- Electronic circuits don’t manipulate logic 1 and 0 literally.
- In digital circuits, the values “1” and “0” are levels of voltage, and the logic circuits that we use are technically “inverting amplifiers with saturated outputs.”
- In the circuits we will use, logic 0 is 0 volts, and logic 1 is 5 volts.

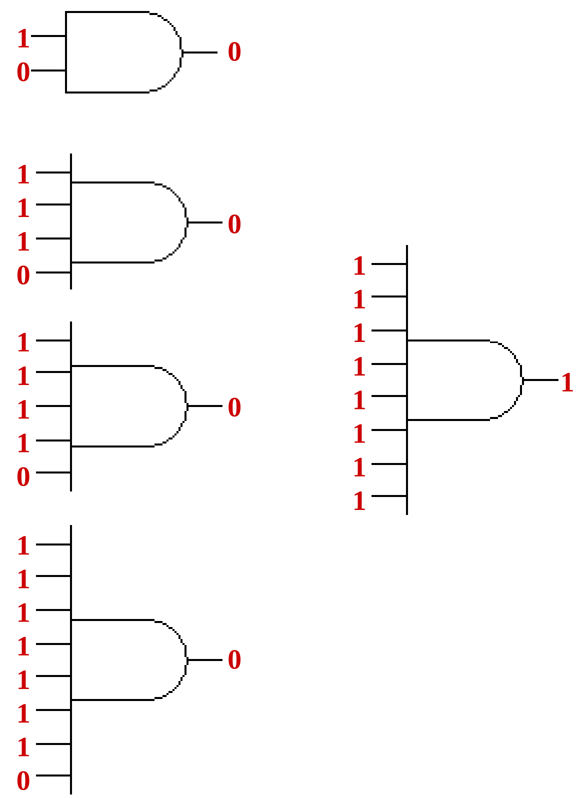


Making More Complex Boolean Functions

- **The three Boolean functions discussed above can be used to form more complex functions.**
- **ANY computer function can be performed using combinations of AND, OR, and NOT.**
- **To simplify the definition of combinational logic (the logic of the computer CPU), any logic function can be composed of a level of AND gates followed by a single OR gate.**
- **There are a few other ways to form Boolean circuits, but we will cover only this one method in Lab 3.**

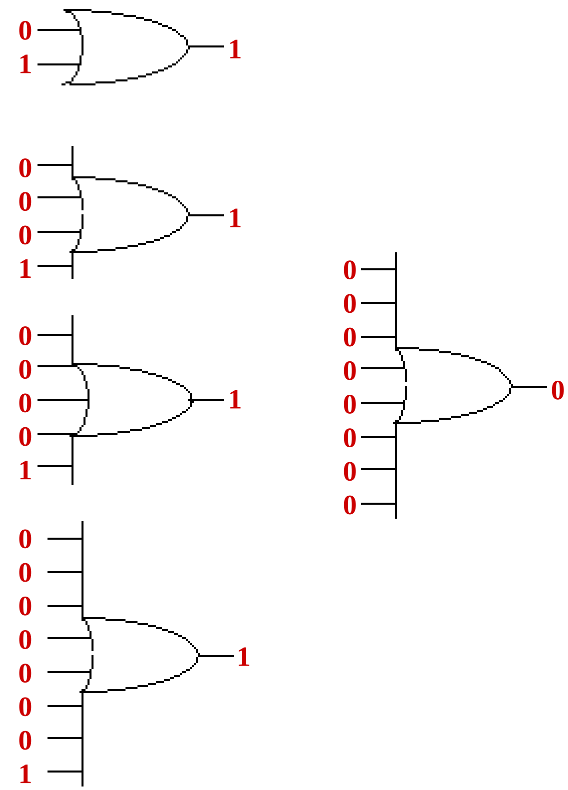
Uniqueness of AND

- The uniqueness of the AND function is that the output of AND is 0 except when EVERY input = 1.
- In the 4 gates to the right, a SINGLE 0 input into each gate forces the output to 0.
- The output of AND is 1 only when ALL inputs = 1 (8-input gate to right).



The “Any 1” Quality of OR

- **The output of OR = 1 if ANY input = 1.**
- **OR outputs a 0 iff ALL inputs = 0.**
- **We can use the ability of OR to “pass” any 1 and the unique 1- outputs of the AND to create Boolean functions.**



Digital Design

- In circuit design, inputs and outputs are defined by a "spec."
- Since computer circuits use only binary numbers, inputs are always 0 and 1, and the output is always 0 and 1.
- The engineer designs the circuit between input and output by:
  - Making a truth table to represent the input/output relationship.
  - Defining a Boolean expression that satisfies the truth table.
  - Constructing a circuit that represents the Boolean function.

Creating a Computer ("Boolean") Function

- **A "spec" for a function f of two variables x and y is that $f = 1$ when x and y are different, and 0 otherwise.**
- **The truth table charts f per the "spec."**
- **How can we describe this behavior with a Boolean expression?**
- **For the first 1, we can create an AND function: $\bar{x} \cdot y$. Note that this expression is 1 ONLY when $x = 0, y = 1$.**
- **For the second 1, we create $x \cdot \bar{y}$, which is only 1 for $x = 1, y = 0$.**

x	y	f
0	0	0
0	1	1
1	0	1
1	1	0

$\bar{x} \cdot y$ $x \cdot \bar{y}$

Boolean Function (2)

- **First 1 AND function: $\bar{x} \cdot y$.**
- **Second 1 AND function: $x \cdot \bar{y}$.**
- **The two Boolean AND functions each describe one of the two conditions in which *f* is 1.**
- **How do we create a Boolean function that describes BOTH conditions of *f* = 1?**
- **Recall that any 1 is passed through OR.**
- **Then if we send both ones through a single OR, its output will match the specified performance of *f*.**

x	*y*	*f*
0	0	0
0	1	1
1	0	1
1	1	0

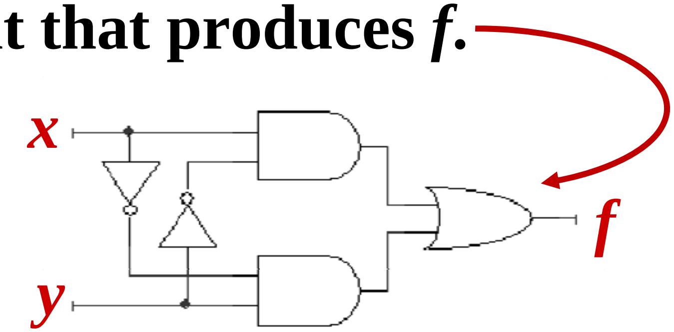
$\bar{x} \cdot y$ $x \cdot \bar{y}$

Boolean Function (3)

- **We OR the two AND functions:**

$$f = x \cdot \bar{y} + \bar{x} \cdot y$$

- **We now have a complete description (Boolean expression) for the function *f*.**
- **Since we know what AND and OR circuits look like, we can build a digital circuit that produces *f*.**



x	y	f
0	0	0
0	1	1
1	0	1
1	1	0

$\bar{x} \cdot y$ $x \cdot \bar{y}$

Building Boolean Functions

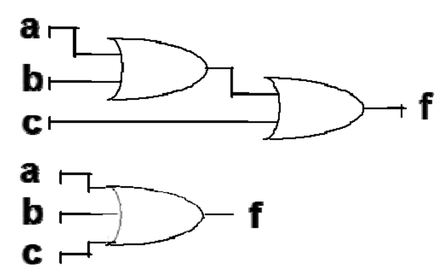
- As we have just seen, if we have Boolean functions that result from a truth table and "spec," we can convert the Boolean functions to computer circuits.
- Consider these functions:

$$a+b+c=f$$

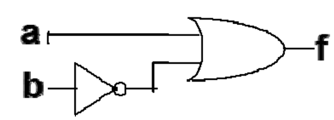
$$a+\bar{b}=f$$

$$(a \cdot b)+c=f$$

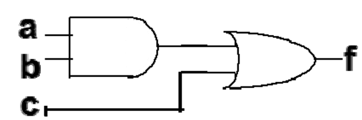
1:



2:



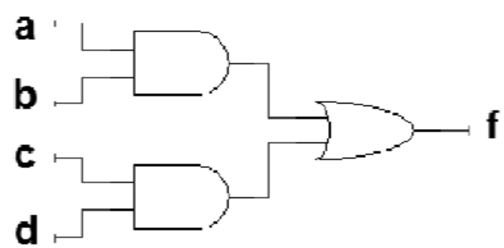
3:



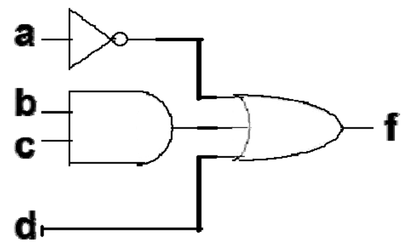
Quick Exercise

- **Designing a circuit from the Boolean expressions:**

$$(a \cdot b) + (c \cdot d) = f$$



$$\overline{a} + (b \cdot c) + d = f$$



Design: "Spec" to Truth Table to Circuit

- **Say you have the following "design spec" for function *f*:**
  - **There are three inputs *x*, *y*, and *z*, to a digital circuit.**
  - **The circuits must have an output of 1 when *y* = *z* =1 and *x* = 0; and when *x* = *z* = 1 and *y* = 0.**
  - **Design the circuit using AND, OR, and NOT gates.**

x	y	z	$f(x, y, z)$	AND's
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$f = \overline{x}yz$
1	0	0	0	
1	0	1	1	$f = x\overline{y}z$
1	1	0	0	
1	1	1	0	

The truth table above shows the desired output:

f = 1 when x=0, y, z = 1,

f = 1 when y=0, x, z = 1.

Getting the Boolean Function

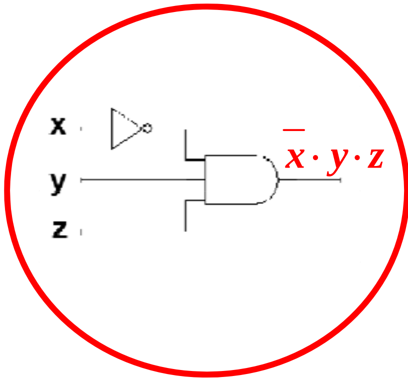
- We can create a Boolean function for each of the two "1" conditions:
  - Inverting x and ANDing it with y and z creates a 1 for the first condition.
  - Inverting y and ANDing it with x and z creates the other 1.
  - Notice that each AND function produces a 1 ONLY for that combination of variables.
- According to the definition of OR, any 1 goes through that gate.
  - Therefore OR the two AND functions together to get a function that is 1 for both cases!

x	y	z	f(x, y, z)	AND's
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$f = \bar{x}yz$
1	0	0	0	
1	0	1	1	$f = x\bar{y}z$
1	1	0	0	
1	1	1	0	

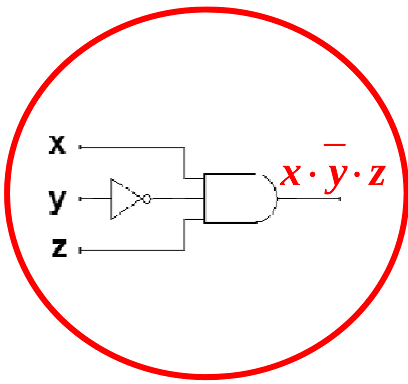
This is a 1 ONLY for (0,1,1)

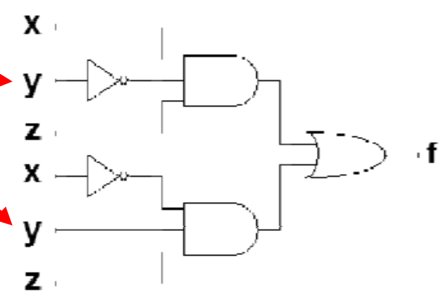
This is a 1 ONLY for (1,0,1)

Boolean Function and Circuit



x	y	z	f(x, y, z)	AND's
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$\bar{x}\cdot y\cdot z$
1	0	0	0	
1	0	1	1	$x\cdot \bar{y}\cdot z$
1	1	0	0	
1	1	1	0	

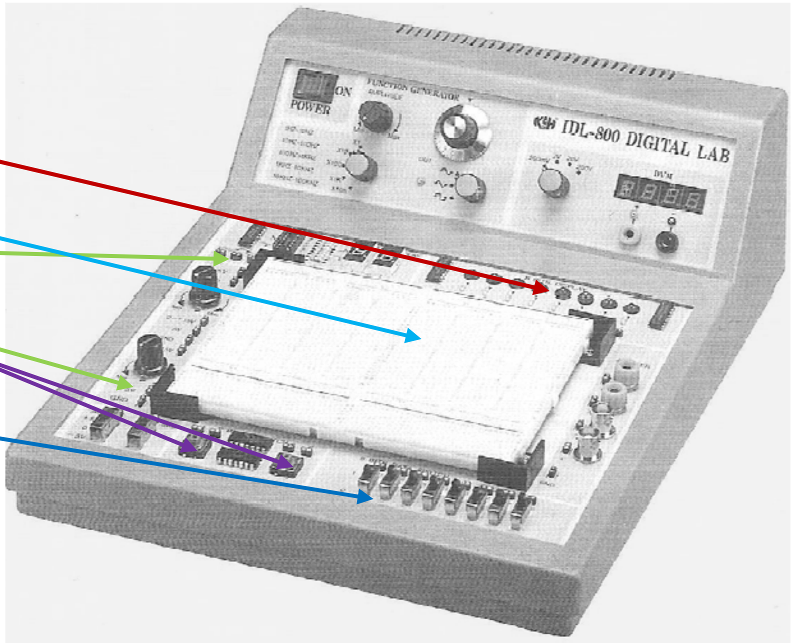




The OR function (*f*) completely satisfies the spec and truth table!

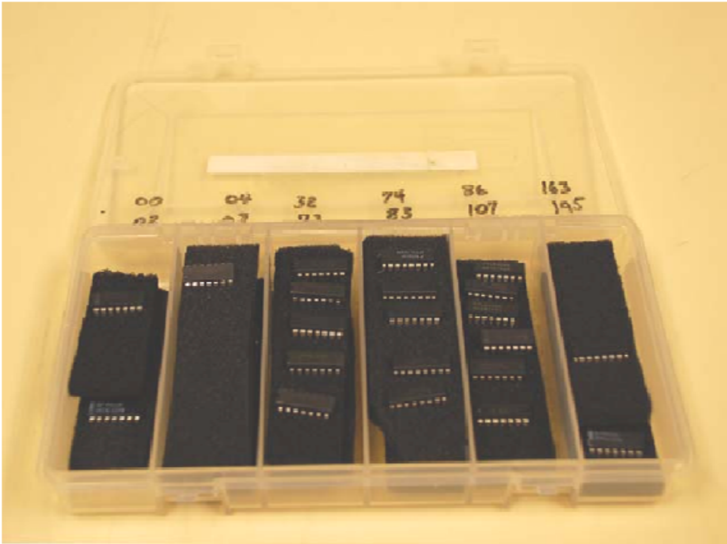
Lab Instrument: IDL-800

- **Key parts:**
  - **LED indicators**
  - **Circuit board**
  - **+5V power**
  - **Momentary 0-1 switches**
  - **0-1 input switches**



Digital Circuit Kit

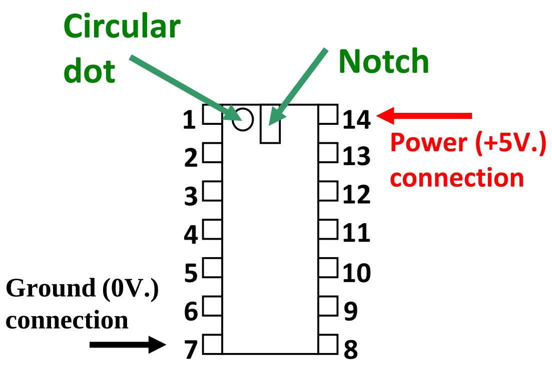
- The digital circuit kits are also used in EE 2310.
- You will only be using AND, OR, and NOT circuits (see below).
- NEVER replace a bad or broken circuit in the kit. Give to the TA to be replaced.
- ALWAYS put up the kit when you are done.



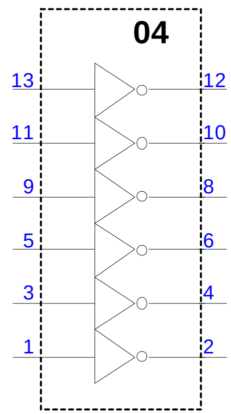
Plugging in a Digital IC and Wiring Up a Circuit

- **Remember: The circuit is ALWAYS plugged in so that it spans a channel in the circuit board.**
- **Therefore a wire plugged into any of the parallel holes into which a chip leg is plugged is connected to that leg of the chip.**

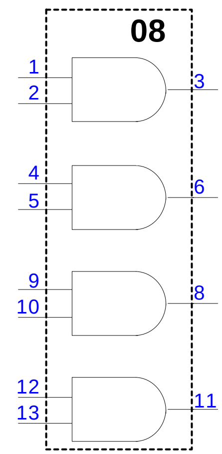
Digital IC Circuit Diagrams



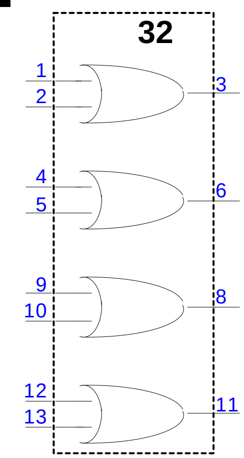
74 LS XXX Chip Outline



SN 74LS04 Hex inverter gate



SN 74LS08 Quad 2-input AND gate



SN 74LS32 Quad 2-input OR gate

- You will be using the 74LS04, 74LS08, and 74LS32 digital integrated circuits.
- The diagram above (also in your manual) shows the outline of the chip, with power/ground inputs.
- The three chip schematics show how the circuits in each chip connect to the input pins.