

Logic Gates

CS 350: Computer Organization & Assembler Language Programming

A. Why?

- Logic gates are the lowest level of hardware that deal with logical values.

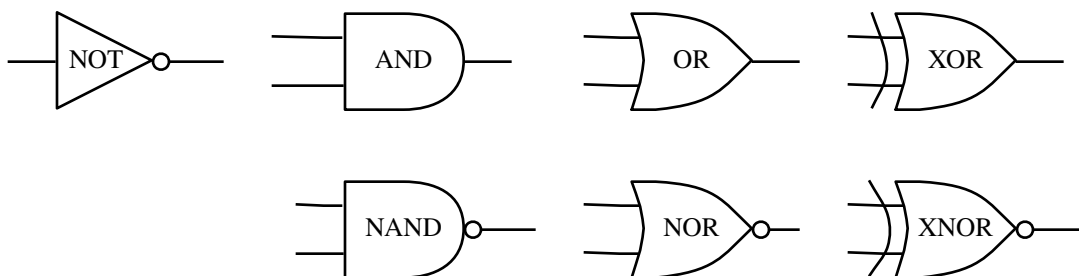
B. Outcomes

After this lecture, you should

- Know the symbols for the logic gates.
- Be able to trace the execution of a simple logic circuit.
- Be able to convert between truth tables, logical formulas, and loop-free logic circuits.

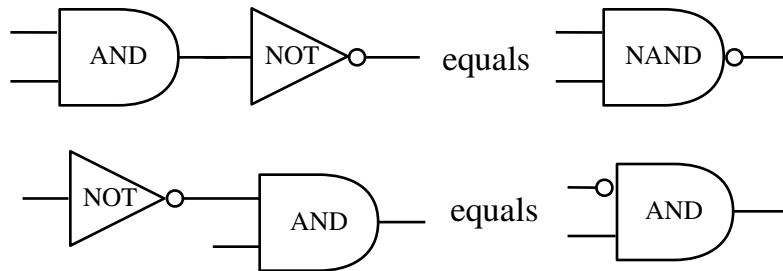
C. Logic Gates

- A logic gate is a device that performs a logical calculation by taking logical inputs and producing logical output(s).
- (We're interested in electronic logic gates.)
- Symbols for gates



- Binary *XNOR* (“eks-nore”) is logical equivalence (i.e., iff).
- We can extend binary gates to n -ary gates (n inputs, 1 output).
 - E.g., $AND(X, Y, Z, U) = ((X AND Y) AND Z) AND U$.
 - $OR(X, Y, Z, U) = ((X OR Y) OR Z) OR U$.

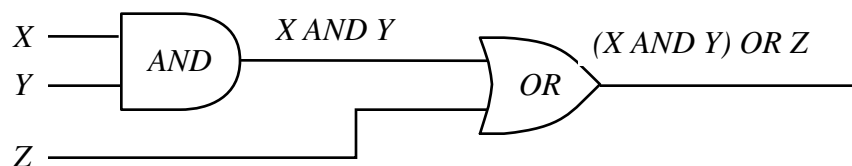
- Since *AND* and *OR* are associative, parenthesization isn't important.
- n -way *NAND*/*NOR*/*XNOR* is the *NOT* of n -way *AND*/*OR*/*XOR*
- n -way *XOR* yields true iff an odd number of inputs are true.
- n -way *XNOR* yields true iff an even number of inputs are true.
- Adding a circle to an output or input has the same effect as a *NOT* gate:



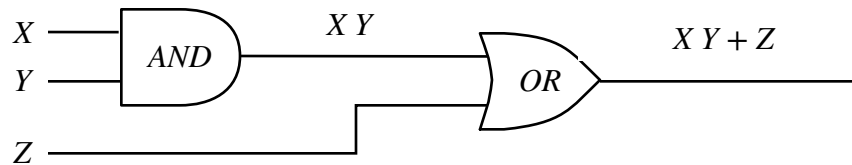
- It's straightforward to combine gates to calculate simple boolean expressions.
- Note the time it takes for a circuit to stabilize depends on the length of the longest path through the circuit. (The shorter the longest path, the faster the circuit, in general.)

D. Translating From a Simple Logic Circuit To a Propositional Formulas

- A simple logic circuit diagram can be translated to an equivalent propositional formula. It must be a **combinatorial circuit**: It must have no loop.
 - Start with the gate(s) that connect directly to the circuit's inputs and label the output of each gate with the proposition for that gate and its inputs.
 - Continue finding gates that have two labeled inputs and labeling their outputs.
- Here's an example:



or



E. Translating Between Truth Tables and Propositional Formulas

- The trick in converting from a truth table to a corresponding propositional formula is to notice that each row of a truth table corresponds to a unique conjunction of the variables (with each variable possibly negated).
- Example:** $\bar{X}\bar{Y} + X\bar{Y}$ describes the truth table below

X	Y	<i>Proposition</i>	Z
0	0	$\bar{X}\bar{Y}$	1
0	1	$\bar{X}Y$	0
1	0	$X\bar{Y}$	1
1	1	XY	0

- (Note: In the trivial cases, all the outputs are 0 or 1 and we can use 0 or 1 as the expression.)
- Disjunctive normal form:** The propositional formula that corresponds to a truth table has a specific format: It's the disjunction (*OR*) of some terms, where each term is the conjunction (*AND*) of some possibly-negated variables (a.k.a. “**atoms**” or “**literals**”). This format is called disjunctive normal form. (A normal form is just a standard way of writing something.)
- Examples:** $\bar{X}Y + XZ$ and $XZ + YZ$ are both in DNF, but $(X + Y)Z$ is not in DNF even though it's equivalent to $XZ + YZ$.
- A DNF expression is in **Full DNF** if each term includes all the variables of the expression.
 - Example:** $XYZ + X\bar{Y}Z + XY\bar{Z} + \bar{X}YZ$ is in full DNF.

- **Example:** $XZ + YZ$ is in DNF but not full DNF (the first term is missing Y or \bar{Y} ; the second is missing X or \bar{X}).
- You can convert an expression from DNF to full DNF by inserting $(V + \bar{V})$ for missing variables V and expanding by distributing AND over OR
 - **Example:** $XZ + YZ$

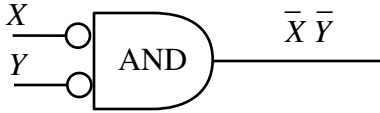
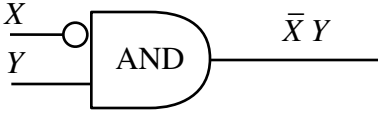
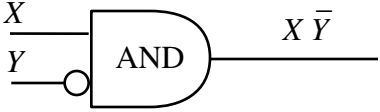
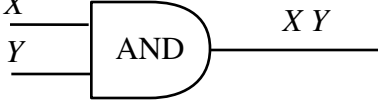
$$= X(Y + \bar{Y})Z + (X + \bar{X})YZ$$

$$= XYZ + X\bar{Y}Z + XYZ + \bar{X}YZ$$

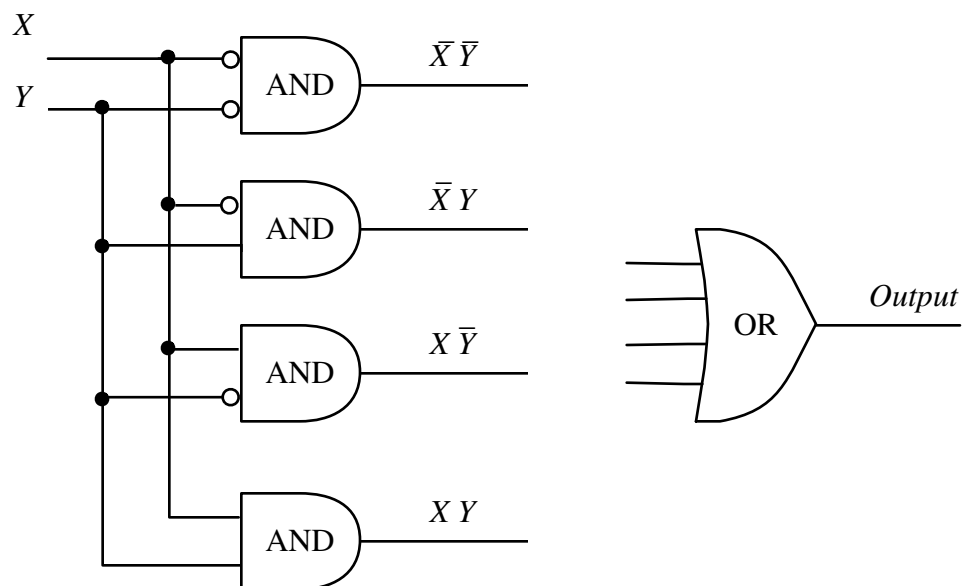
$$= XYZ + X\bar{Y}Z + \bar{X}YZ$$
- This technique can produce long propositions that have shorter equivalents. Algebraic manipulations can be used to simplify the expression.
- One way to convert from a propositional formula to a truth table is to use DNF: Use rules like DeMorgan's laws and distribution to expand the formula and then use excluded middle, contradiction, maybe double negation elimination (a.k.a. Pierce's law), and domination ($FX = F$; $T + X = T$) to simplify.

F. Translating From a Truth Table to a Simple Logic Circuit

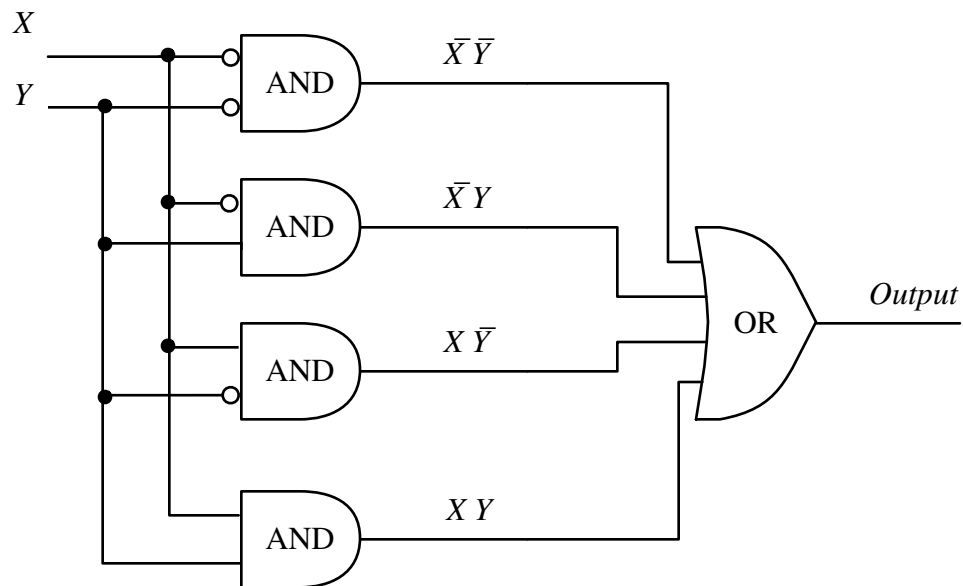
- A **Programming Logic Array (PLA)** is a general structure for implementing a truth table using logic gates. It relies on the disjunctive normal form representation of a truth table as a logical expression.
 - Each conjunct corresponds to an *AND* gate with literals as its inputs. The outputs of selected *AND* gates are all sent to an *OR* gate, and the output of the *OR* gate corresponds to the value in the truth table.
- A table with n logical inputs has 2^n rows (one for each gate). E.g., A table with inputs X and Y has $2^2 = 4$ rows. Note that for each input that is negated, we insert a *NOT* gate before the input goes to the *AND* gate.

X	Y	<i>Proposition</i>	<i>Gate</i>
0	0	$\bar{X} \bar{Y}$	
0	1	$\bar{X} Y$	
1	0	$X \bar{Y}$	
1	1	$X Y$	

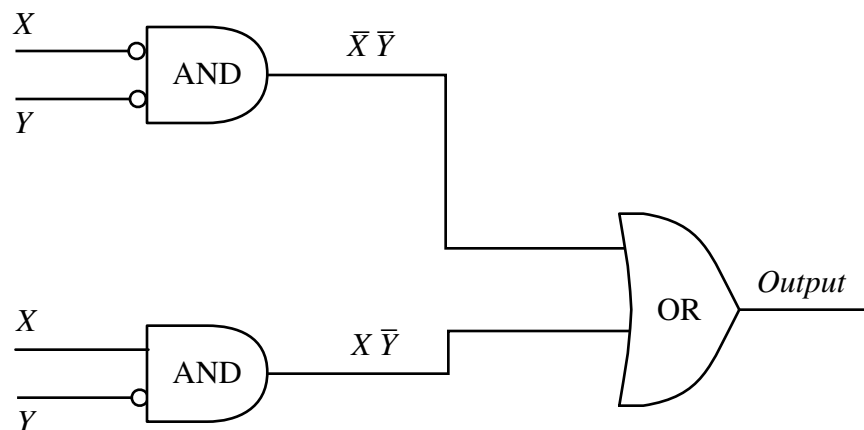
We begin with none of the *AND* gate outputs sent to the *OR* gate:



- To program a PLA, we connect the *AND* gates that correspond to output T to the *OR* gate; the *AND* gates that correspond to output F don't get connected. E.g., for the earlier truth table, we get:

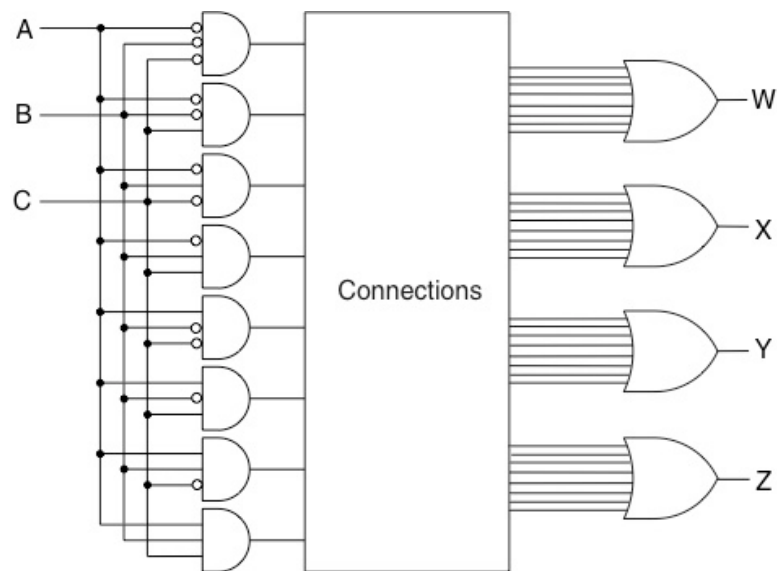


- If we're designing a circuit, we can simplify it by tossing out the unused *AND* gates (and *OR* inputs). We can also simply redraw the inputs to the *AND* gates. E.g., For the earlier truth table, we get



- (For a physical PLA, the unused *AND* gates and *OR* inputs don't get deleted, we just don't use them.)
- An actual PLA can have multiple outputs in addition to multiple inputs. (For an example, see Patt and Patel's Figure 3.17 below.)

Fig 3.17: 3-bit input, 4-bit output PLA



Logic Gates

CS 350: Computer Organization & Assembler Language Programming

A. Why?

- Logic gates are the lowest level of hardware that deal with logical values.

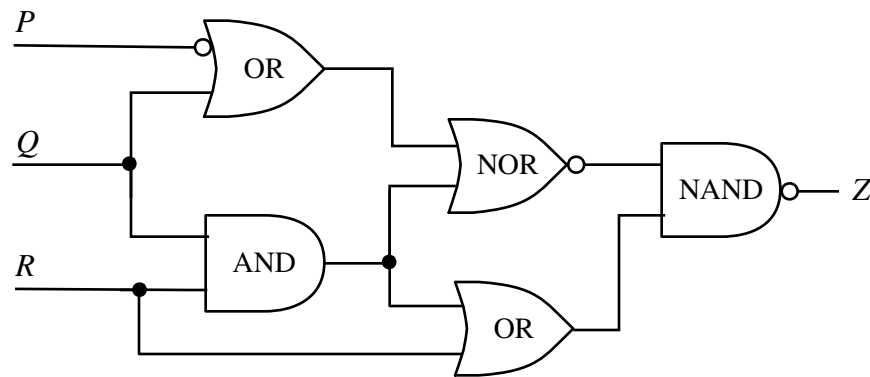
B. Outcomes

After this activity, you should be able to:

- Read and write the symbols for the logic gates.
- Trace the execution of a simple logic circuit.
- Convert between truth tables, logical formulas, and loop-free logic circuits.

C. Questions

1. Translate the logic gate circuit below to its equivalent boolean expression. Do a direct translation — don't simplify the expression.

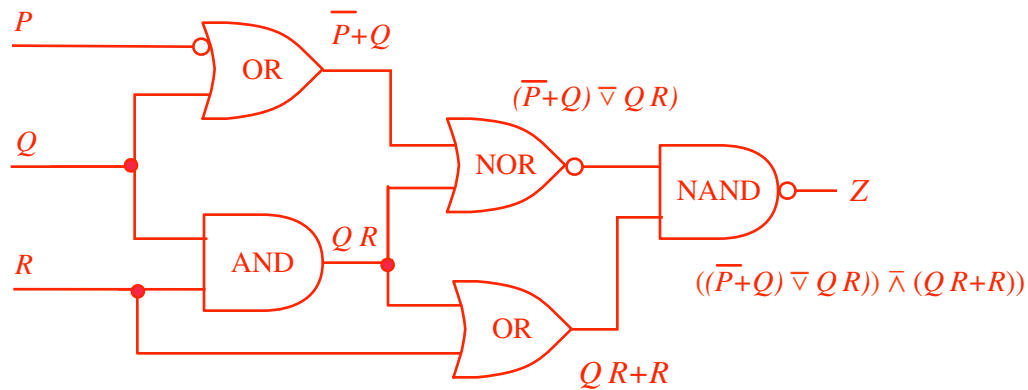


2. Let E be the expression $\overline{X} Y + \neg(\overline{X} + \overline{Y}) + \neg(X + Y)$.
 - (a) Write a logic gate implementation for E . Don't modify or simplify E .
 - (b) Use DeMorgan's laws (and possibly Pierce's law for removing double negations) to simplify E to full DNF. Write a logic gate implementation for this full DNF expression. (I.e., the PLA circuit for E .)
3. Repeat Problem 2 on the expression $\overline{Y}(\overline{X}YZ) + (X \oplus Y)Z + \neg(X \rightarrow \overline{Y})Z$. (Note \oplus means XOR.) [For $\neg(X \rightarrow \overline{Y})$, use XY .]

Activity 16 Solution

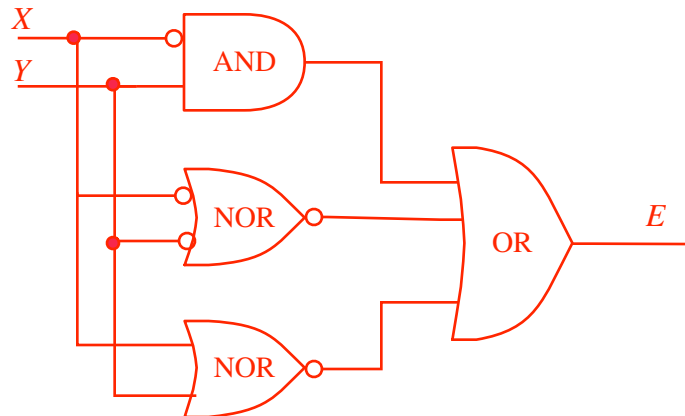
1. The expression is $\neg(\neg(\bar{P} + Q + QR)(QR + R))$. Using $\bar{\wedge}$ and $\bar{\vee}$ for NAND and NOR, we get $((\bar{P} + Q) \bar{\vee} (QR)) \bar{\wedge} (QR + R)$.

The expressions for the different parts of the diagram are shown in red below.

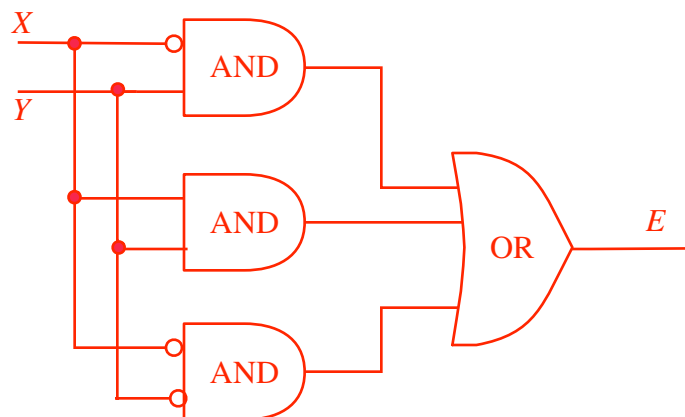


2. (The expression $E = \bar{X}Y + \neg(\bar{X} + \bar{Y}) + \neg(X + Y)$)

a. A direct translation of E to a logic circuit:

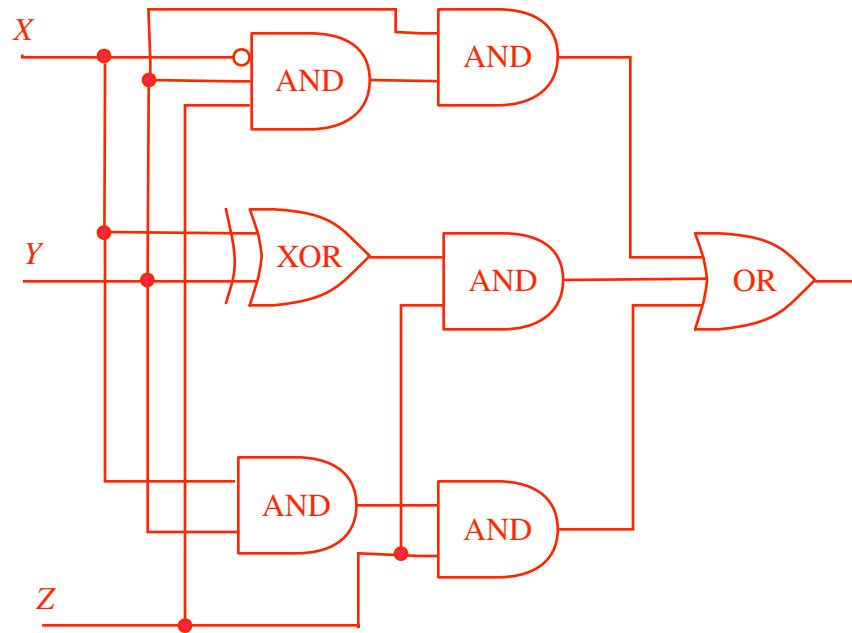


b. Using DeMorgan's laws,
 $\bar{X}Y + \neg(\bar{X} + \bar{Y}) + \neg(X + Y)$
 $= \bar{X}Y + XY + \bar{X}\bar{Y}.$



3. (The expression $\overline{Y} (\overline{X} Y Z) + (X \oplus Y) Z + \neg(X \rightarrow \overline{Y}) Z$, where \oplus is XOR)

a. A direct translation to a logic gate circuit: (Note $\neg(X \rightarrow \overline{Y}) = X Y$.)



b. For the expansion to full DNF, it might be easiest to do this in parts:

- $\overline{Y} (\overline{X} Y Z) = 0$ (because of \overline{Y} and Y)
- $(X \oplus Y) Z = (X \overline{Y} + \overline{X} Y) Z = X \overline{Y} Z + \overline{X} Y Z$
- $\neg(X \rightarrow \overline{Y}) Z = X Y Z$

Combining,

$$\begin{aligned} & \overline{Y} (\overline{X} Y Z) + (X \oplus Y) Z + \neg(X \rightarrow \overline{Y}) Z \\ &= 0 + X \overline{Y} Z + \overline{X} Y Z + X Y Z \\ &= \overline{X} Y Z + X \overline{Y} Z + X Y Z \end{aligned}$$

The PLA circuit is

