Logic Gates

CS 350: Computer Organization & Assembler Language Programming
Lecture 22, Wed 4/17

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, of course. There are some standard symbols for logic gates. Note: 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 XOR yields true iff an odd number of its inputs are true.
  • n-way NAND, NOR, and XNOR are the NOT of n-way AND, OR, or XOR
    • n-way NAND yields true iff at least one of its inputs is false.
    • n-way NOR yields true iff none of its inputs are true (or equivalently, all of its inputs are false)
    • n-way XNOR yields true iff an even number of its 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.)
**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:

![Logic Circuit Diagram](image)

(Remember, you can use other notations of you like, as in:)

![Alternative Notation](image)

**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**: $\overline{X} \overline{Y} + X \overline{Y}$ describes the truth table below

<table>
<thead>
<tr>
<th>$X$</th>
<th>$Y$</th>
<th>Proposition</th>
<th>$Z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>$\overline{X} \overline{Y}$</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>$\overline{X} Y$</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>$X \overline{Y}$</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>$XY$</td>
<td>0</td>
</tr>
</tbody>
</table>

- (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**: $\overline{X} Y + X Z$ and $X Z + Y Z$ are both in DNF, but $(X + Y) Z$ is not in DNF even though it's equivalent to $X Z + Y Z$.

- A DNF expression is in **Full DNF** if each term includes all the variables of the expression.
• **Example:** \( XYZ + X \overline{Y} Z + X Y Z + \overline{X} Y Z \) is in full DNF.

• **Example:** \( XZ + YZ \) is in DNF but not full DNF (the first term is missing \( Y \) or \( \overline{Y} \); the second is missing \( X \) or \( \overline{X} \)).

• You can convert an expression from DNF to full DNF by inserting \((V+\overline{V})\) for missing variables \( V \) and expanding by distributing AND over OR

• **Example:**
  \[
  XZ + YZ
  = X(Y + \overline{Y})Z + (X + \overline{X})YZ
  = XYZ + X\overline{Y}Z + XY\overline{Z} + \overline{X}YZ
  = XYZ + X\overline{Y}Z + \overline{X}YZ
  \]

• 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.

### 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.

  • As an example, here’s a truth table for \( \overline{X}Y + XY \):

<table>
<thead>
<tr>
<th>( X )</th>
<th>( Y )</th>
<th>Proposition</th>
<th>Gate</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>( \overline{X}\overline{Y} )</td>
<td>![Gateway](X OR Y.png)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>( \overline{X}Y )</td>
<td>![Gateway](X OR Y.png)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>( X\overline{Y} )</td>
<td>![Gateway](X OR Y.png)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>( XY )</td>
<td>![Gateway](X OR Y.png)</td>
</tr>
</tbody>
</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 \( \text{NOT} \) gate before the input goes to the \( \text{AND} \) gate.

- We begin with none of the \( \text{AND} \) gate outputs sent to the \( \text{OR} \) gate:

![Diagram of AND and OR gates](image)

- To program a PLA, we connect the \( \text{AND} \) gates that correspond to output \( T \) to the \( \text{OR} \) gate; the \( \text{AND} \) gates that correspond to output \( F \) don’t get connected. E.g., from the truth table above for \( \overline{X} \, \overline{Y} + X \, Y \), we get:

![Diagram of AND and OR gates](image)
• If we’re designing a circuit, we can simplify it tossing out the unused AND gates (and OR inputs). We can also simply redraw the inputs to the AND gates. E.g., for $\overline{X}Y + XY$ above, we get

![Diagram of simplified circuit](image)

• (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. The figure to the right shows a PLA with 3-bits of input, and 4-bits of output.)

![Diagram of PLA](image)

(From Patt & Patel)
**Activity Problems**

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

   ![Logic gate circuit diagram]

2. Let \( E \) be the expression \( \overline{X} Y + \overline{\overline{X} + \overline{Y}} + \overline{\overline{X} + Y} \).
   
   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 \( \overline{Y} (\overline{X} Y Z) + (X \oplus Y) Z + \overline{\overline{X} \rightarrow \overline{Y}} Z \), where \( \oplus \) means XOR. For \( \overline{\overline{X} \rightarrow \overline{Y}} \), you can use \( \overline{\overline{X} v \overline{Y}} \) or \( XY \).

**Solution to Activity Problems**

1. The expression is \( \overline{\overline{\overline{(\overline{P} + Q + QR)} (QR + R)}} \). Using \( \overline{\overline{X}} \) and \( \overline{Y} \) for NAND and NOR, we get \( ((\overline{P} + Q) \overline{QR}) (QR + R) \). The expressions for the different parts of the diagram are shown in red below.

   ![Logic gate circuit diagram solution]
2. (The expression \( E = \overline{X} \overline{Y} + \neg(X \oplus Y) + \neg(X + Y) \))

   a. A direct translation of \( E \) to a logic circuit:

   ![Logic Circuit Diagram]

   b. Using DeMorgan's laws, \( \overline{X} \overline{Y} + \neg(X \oplus Y) + \neg(X + Y) \)
      \[= \overline{X} \overline{Y} + XY + \overline{X} \overline{Y}\]
3. (The expression $\overline{Y} (\overline{X}YZ) + (X \oplus Y)Z + \neg(X \rightarrow \overline{Y}) Z$, where $\oplus$ is XOR)
   
a. A direct translation to a logic gate circuit: (Note $\neg(X \rightarrow \overline{Y}) = XY$.)

```
\[ \text{Diagram of logic gate circuit} \]
```

b. For the expansion to full DNF, it might be easiest to do this in parts:
   
   $\overline{Y} (\overline{X}YZ) = 0$ (because of $\overline{Y}$ and $Y$)
   
   $(X \oplus Y) Z = (X \overline{Y} + \overline{X}Y)Z = X\overline{Y}Z + \overline{X}YZ$
   
   $\neg(X \rightarrow \overline{Y}) Z = XYZ$
   
   Combining,
   
   $\overline{Y} (\overline{X}YZ) + (X \oplus Y)Z + \neg(X \rightarrow \overline{Y}) Z
   
   = 0 + X\overline{Y}Z + \overline{X}YZ + XYZ$
   
   $= \overline{X}YZ + X\overline{Y}Z + XYZ$
The PLA circuit is

![Diagram of PLA circuit]