EECS 42 Introduction to Electronics for Computer Science
Andrew R. Neureuther

Lecture # 11  Logic Implementation
• Logic Levels and Gate Circuits
• Combination of Logic Functions
• Synthesis from a Truth Table
• NAND Gate Synthesis
• XOR and Introduction to Timing

http://inst.EECS.Berkeley.EDU/~ee42/

First Midterm Exam: Topics
• Basic Circuit Analysis (KVL, KCL)
• Equivalent Circuits and Graphical Solutions for Nonlinear Loads
• Transients in Single Capacitor Circuits
• Node Analysis Technique and Checking Solutions

Exam is in class 3:10-4:03 PM, Closed book, Closed notes, Bring a calculator, Paper provided

Example: Basic Circuit Analysis

Example: Load-Line Method

REMANDER
Midterm March 5th, 3:10-4:03 PM
Closed Book, Closed Notes, Bring Calculator, Paper Provided

Old Exams Are Posted on Web
Review Session 5-7 Mon 3/4/04 in 241 Cory

EE 43 Labs Are Not Cancelled but will be light:
Example: Transient

\[ \begin{align*}
R_1 &= 4 \, \text{k}\Omega \\
R_2 &= 2 \, \text{k}\Omega \\
I_{SS} &= 1 \, \text{mA}
\end{align*} \]

Example: Node Equation method

\[ \begin{align*}
R_1 &= 4 \, \text{k}\Omega \\
R_2 &= 2 \, \text{k}\Omega \\
I_{SS} &= 1 \, \text{mA}
\end{align*} \]

Some Important Logical Functions

- **“AND”**
- **“OR”**
- **“INVERT” or “NOT”** not A (or \( \bar{A} \))
- **“not AND” = NAND**
- **“not OR” = NOR**
- exclusive OR = XOR A \( \oplus \) B (only 1 when A, B differ)

Logic Gates

\[
R_1 = 4 \, \text{k}\Omega \\
R_2 = 2 \, \text{k}\Omega \\
C = 1 \, \text{pF} \\
\text{ISS} = 1 \, \text{mA} \]

\[
\begin{align*}
A & \quad \text{AND} \\
B & \quad C = A \cdot B \\
A & \quad \text{OR} \\
B & \quad C = A + B \\
A & \quad \text{NOT} \\
B & \quad X \\
A & \quad \text{EXCLUSIVE OR} \\
B & \quad C = A \oplus B
\end{align*} \]

Logic Circuits

\[
\begin{array}{cccc}
A & B & X & Y \\
0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 \\
1 & 1 & 0 & 0
\end{array} \]

Interestingly, this is the same truth table as the EXCLUSIVE OR

Logic Gates – How are they used in practice?

\[
\begin{align*}
A \quad \text{AND} \\
B & \quad C = A \cdot B \\
A \quad \text{AND} \\
1.5V & \quad C \quad \text{would have the value of} \\
1.5V & \quad \text{1.5 V (logical 1).} \\
1.5V & \quad \text{But it would have} \\
0V & \quad \text{the value of} \\
0V & \quad \text{0V logical 0 if} \\
\text{either one of the inputs} & \quad \text{were held at zero V.}
\end{align*} \]
Logic Gates – How are they built in practice?

A Valve is a Transistor

Pull up network

Valves in parallel => NOR

Valves in series => NAND

Current flows when VIN is high

(You can learn about building gates in EE 141.)

The most common basic gates are NAND and NOR?

Not-AND = NAND

\[ A \cdot B \rightarrow \overline{A \cdot B} \]

0 0 1
0 1 0
1 0 0
1 1 1

Not-OR = NOR

\[ A + B \rightarrow \overline{A + B} \]

0 0 1
0 1 0
1 0 1
1 1 0

How to Combine Gate to Produce a Desired Logic Function?
(More basic Logical Synthesis)

Suppose we are given a truth table (all logic statements can be represented by a truth table). How can we implement the function?

\[ \text{Answer: There are lots of ways, but one simple way is implementation from “sum of products” formulation.} \]

\[ \text{How to do this: 1) Write sum of products expression from truth table and 2) Implement using standard gates.} \]

(Warning this is probably inefficient – we need to minimize, or simplify the expression. You will learn this in CS 150.)

Example:

\[ \begin{array}{ccc|c}
0 & 0 & 0 & F \\
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 \\
1 & 1 & 1 & 1 \\
\end{array} \]

Clearly: \[ F = \overline{A \cdot B} \cdot C = \overline{A} \cdot \overline{B} \cdot C \]

or \[ AB \cdot C = 1 \]

i.e. \[ F = \overline{A} \cdot \overline{B} \cdot C + AB \cdot \overline{C} \]

How to Combine Gate to Produce a Desired Logic Function?
(More basic Logical Synthesis)

Example:

\[ \begin{array}{ccc|c}
A & B & C & F \\
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 \\
\end{array} \]

F = \overline{A} \cdot \overline{B} \cdot C + AB \cdot \overline{C}
Logical Synthesis
Guided by DeMorgan's Theorem

DeMorgan's Theorem:

\[ A + B + C = \overline{A \cdot B \cdot C} \] or \[ \overline{A + B + C} = \overline{A \cdot B \cdot C} \]

Example of Using DeMorgan's Theorem:

\[ F = A \cdot B + C \cdot D \cdot E = \overline{A \cdot B \cdot C \cdot D \cdot E} \]

Thus any sum of products expression can be immediately synthesized from NAND gates alone.

Logical Synthesis of XOR

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

F = A \cdot B + \overline{A} \cdot B

We Need a Timing Diagram!

Delay 1

Delay 2

Delay 3

Timing Diagram for Delays in Logic