Outline – Logic Circuits

- Introduction
- Logic Systems
  - TTL
  - CMOS
- Logic Gates
  - NOT, OR, AND
  - NOR, NAND, XOR
  - Implementation
- Boolean Algebra
- Combinatorial Circuits
  - Multiplexer
  - Demultiplexer
  - Arithmetic Circuits
- Simplifying Logic Circuits

Digital- or Logic Circuits

- Logic circuits constitute the major branch of electronics that deal with the design and implementation of discrete-time computing systems and their components:
  - Main elements are logic gates: NOT, AND, OR
  - Action of logic (or “digital”) circuits can be depicted by
    - Standard logic symbols
    - Truth tables
    - Boolean algebra
    - Formal VLSI circuit design languages:
      - Verilog and VHDL

Binary Representation

- In digital circuits, information is stored in binary form, consisting of an ordered array of two state devices.
- Base two numbering system is utilized:
  - Numbers composed of 1’s and 0’s which are called bits (= Acronym of binary digits):
    - Example: $11100_2 = 1\cdot2^4 +1\cdot2^3 +1\cdot2^2 +0\cdot2^1 +0\cdot2^0 = 28_{10}$

Logic Voltage Levels

**Transistor-Transistor Logic (TTL):**

<table>
<thead>
<tr>
<th>Input Voltage</th>
<th>Output Voltage</th>
</tr>
</thead>
<tbody>
<tr>
<td>+5</td>
<td>+4</td>
</tr>
<tr>
<td>+4</td>
<td>+3</td>
</tr>
<tr>
<td>+3</td>
<td>+2</td>
</tr>
<tr>
<td>+2</td>
<td>+1</td>
</tr>
<tr>
<td>+1</td>
<td>LOW=0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**CMOS Logic:**

<table>
<thead>
<tr>
<th>Input Voltage</th>
<th>Output Voltage</th>
</tr>
</thead>
<tbody>
<tr>
<td>+19</td>
<td>+18</td>
</tr>
<tr>
<td>+13</td>
<td>+12</td>
</tr>
<tr>
<td>+9</td>
<td>+8</td>
</tr>
<tr>
<td>+5</td>
<td>+4</td>
</tr>
<tr>
<td>+1</td>
<td>LOW=0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- In TTL:
  - $+5V$ → High-State (H) → 1 → True
  - $0V$ → Low-State (L) → 0 → False
- In CMOS:
  - Supply Voltage → High-State (H) → 1 → True
  - $0V$ → Low-State (L) → 0 → False
Logic Gates

• All digital devices can be built using only three basic gates:
  – NOT (Called “Inverter” rather than NOT gate!)
  – OR
  – AND
• In digital circuit design, various combinations of these gates appear frequently:
  – NOR (OR + NOT)
  – NAND (AND + NOT)
  – XOR (Exclusive AND)
  – XNOR (Exclusive OR)

Chapter 7a ME 534 6

Basic Logic Gates

NOT:

\[ Y = A' \]

AND:

\[ Y = A \cdot B \]

OR:

\[ Y = A + B \]

The combination of basic operations AND, OR, and NOT, makes it relatively easy to describe the logical expressions using Boolean Algebra.

Chapter 7a ME 534 7

Exclusive OR (XOR)

\[ Y = A \oplus B \]

- XOR operation appears in frequently arithmetic circuits.
  - Can be formed via basic gates.
- As the truth table indicates, the operation corresponds to binary “addition.”
- The expression \( Y = A \oplus B \) reads as “Y equals A XOR B.”

Chapter 7a ME 534 8

Universal Gates

NOR:

\[ Y = (A+B)' \]

NAND:

\[ Y = (AB)' \]

Chapter 7a ME 534 9
Equivalency of Universal Gates

**NOR Gate:**

\[ A \oplus B = \overline{A \land B} \]

**NAND Gate:**

\[ A \land B = \overline{A} \lor \overline{B} \]

Construction of Gates with UGs

- Common logic functions (NOT, AND, OR) can be built out of universal gates.

Logic-Gate ICs (TTL: 74xxx)

Leading chip manufacturers market integrated circuits that carry out various logic functions.

- 7432: OR (four gates per chip)
- 7408: AND (four gates per chip)
- 7404: Inverter (six NOT gates per chip)
- 7402: NOR (four gates per chip)
- 7400: NAND (four gates per chip)
- 7486: XOR (four gates per chip)

Example – AND Gate

![And gate circuit diagram](image)
Applications with ICs (1)

- Do NOT directly connect supply voltage ($V_{DD}$) to the input of (TTL logic family) ICs to represent high-logic state.
  - Instead, connect $V_{DD}$ via a "pull-up" resistor.

Applications with ICs (2)

- A small by-pass (decoupling) capacitor is placed close to the supply-voltage pin:
  - Reduces noise and voltage fluctuations.

Applications with ICs (3)

- TTL outputs of a chip can be directly connected to the input(s) of the same- or a different chip.

Applications with ICs (4)

- To avoid unstability, the unused INPUTs of the chip are oftentimes tied to the ground.
  - Leave the corresponding OUTPUTs alone!!!
Applications with ICs (5)

- Consider the use of universal gates (NAND, NOR) to reduce the number of logic gates ICs.

Open Collector (OC) Gates

- In 74xxx family, there are some logic gates with OC outputs.
- Gate’s output must be connected to $V_{DD}$ via pull-up resistors to get a legitimate logic level signal.

$7415$: AND (three 3-input OC gates)
$7405$: Inverter (six NOT gates w/ OC)
$7403/7422/7438$: NAND (four OC gates)

Quite useful for logic-voltage level shifting!

Boolean Algebra

- In logic circuits, Boolean algebra is used to express / manipulate functional relationships.
- A summary of the basic rules is given here:
  - $\neg, \neg\neg, \neg\neg\neg$ correspond to logical OR, AND, NOT operations.
  - $x, y, z \in \{0, 1\}$.

$\neg(x') = x$
$x + 0 = x$
$x \cdot 0 = 0$
$x + x' = 1$
$x \cdot x' = 0$
$x + x = x$
$x \cdot x = x$
$x + 1 = 1$
$x \cdot 1 = x$

$x + y = y + x$
$x + (y + z) = (x + y) + z$
$x(y + z) = xy + xz$
$x + yz = (x + y)(x + z)$
$x(x + y) = x$
$(xy)' = x' + y' \quad (DeMorgan)$

Combinatorial Circuits

- By combining a number of these basic gates, one can implement any desired function in binary domain.
- Output(s) of the resulting function solely depends on its inputs.
  - Such circuits are referred to as “combinatorial” (or “combinational”) logic circuits.
Combinatorial Circuits (Cont’d)

• Some examples of useful combinational circuits will be illustrated:
  – Multiplexer: A circuit used to select one of its inputs and channel it to its output.
  – Demultiplexer: A selection circuit for diverting its input to one of the output channels.
  – Arithmetic Circuits:
    • Addition / Subtraction
    • Multiplication

Demultiplexer (DEMUX)

Disabled output are held on LOW!

Implementation of Mux / Demux

74150: 16x1 Mux
74151: 8x1 Mux
74153: Dual 4x1 Mux
74154: 1x16 Demux (decoder)
74139: Dual 1x4 Demux
74155: Dual 2x4 Demux
Arithmetic Circuits

- An important application field of combinatorial logic is *binary arithmetic* operations:
  - Addition / Subtraction
  - Multiplication / Division
- *Arithmetic logic unit* (ALU) of all microprocessors / micro-controllers do possess such circuits to perform these functions.
- We shall take a look at only some of the most simplest circuits to illustrate their use.

Binary Addition

<table>
<thead>
<tr>
<th>P</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>R</td>
<td>+</td>
<td>0</td>
<td>+</td>
<td>1</td>
</tr>
<tr>
<td>Carry</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>R</th>
<th>Carry</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

XOR AND

Half-Adder (HA)

N-bit Adder

Circuit, which is also called *ripple adder*, operates in serial fashion as each full adder requires the carry issued by the previous one so as to compute its result.

Binary Subtraction

Binary subtraction is very similar to addition except that a *negative* number is added to a *positive* one. The *two’s complement* method is utilized to represent negative numbers.

**Example:** Let's carry out the following subtraction based on this method:

\[
egin{array}{cccccccc}
103 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
-18 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
\hline
85 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1
\end{array}
\]

To calculate two’s complement of a positive number, one needs to reverse all bits (0 → 1 or 1 → 0) and add 1 to the resulting number.
Subtraction (Cont’d)

The two's complement of 18 (that is, -18) becomes

\[
\begin{array}{c}
0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
\text{Reverse bits} & \rightarrow & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\
\text{Add 1} & \rightarrow & + & 1 \\
\text{2s Complement} & & 1 & 1 & 1 & 0 & 1 & 1 & 0
\end{array}
\]

Hence, we get

\[
\begin{array}{c}
0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\
+ & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\
\hline
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1
\end{array}
\]

Carry: 1 (Ignore)

Subtractor / Adder

- When \( S = 1 \), subtraction is performed by taking two's complement of \( Q \):
  - XOR gates reverse the bits of \( Q \).
  - 1 is added to \( Q' \) as \( C_{in} = S = 1 \).
- When \( S = 0 \), addition is performed:
  - Bits of \( Q \) are intact.
  - \( C_{in} = S = 0 \).

Multiplication

Binary multiplication is very similar to how we do multiplication manually:

\[
\begin{array}{c}
1 & 0 & 0 & 1 \\
\times & 1 & 1 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & + \\
1 & 0 & 0 & 1 & \text{ (Carry)} \\
+ & 1 & 0 & 0 & 1 \\
\hline
1 & 1 & 0 & 1 & 1 & 0 & 0
\end{array}
\]

2-bit Multiplier

Consider a 2-bit multiplication:

\[
\begin{array}{c}
P_1 & P_0 \\
\times & Q_1 & Q_0 \\
\hline
A_1 & A_0 \\
\times & B_1 & B_0 \\
\hline
R_3 & R_2 & R_1 & R_0
\end{array}
\]
Canonical Forms

- Any Boolean function with arbitrary complexity boils down to two canonical forms:
  - Sum-of-products
  - Product-of-sums

### Example - XOR

**Circuit 1 (Sum-of-products):**

- Boolean Expression: \( Y = AB' + BA' \)

**Circuit 2 (Product-of-sums):**

- Boolean Expression:
  
  \[
  Y = (A + B)(A' + B') \\
  = A(A' + B') + B(A' + B') \\
  = AA' + AB' + BA' + BB'
  \]

Simplifying Logic Circuits

- There exist an infinite number of combinational circuits that may implement a mapping expressed in terms of a truth table.
- There are various techniques to simplify them:
  - Utilize adhoc techniques
    - Search for patterns (common functions)
  - Write Boolean expressions describing the operation of the circuit
  - Make simplifications using the Boolean algebra rules.
  - Apply mapping techniques
    - Karnaugh Maps (up to 5 variables)
  - Use software tools
    - Based on heuristic search techniques and Boolean rules.
    - Most VLSI design packages incorporate such modules.

Creating Circuits using Truth Tables

- Circuit designer generally starts out with a truth table:
  - One output; N inputs.
- To create an efficient logic circuit representation (as a sum-of-products) using the given table, the following procedure is suggested:
  - Find all entries with non-zero ("true") output:
    - Inverting all zero inputs of such entries, write the Boolean expression as a product.
    - Sum all the "product" terms.
  - Make simplifications using the Boolean algebra rules.
- In fact, K-maps geometrically exploits this idea!
Example - Simplification

Consider the four-input truth table and derive a simplified Boolean expression:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

\[ Y = \overline{A'B'C'D}' + \overline{AB'C'D} + AB'C'D' + AB'CD + ABC'D' + ABCD \]

\[ Y = A'C'D' + AC(B' + B) + AB'C(D + D) + ABC(D' + D) \]

\[ Y = A'C'D' + AC(B' + B) = AC + A'C'D' \]

Y does NOT depend on B!

Programmable Logic Devices

• Realizing a specific Boolean function using large number of discrete logic ICs is not convenient in practice.

• Instead, the function is converted to a **canonical** form and is implemented via
  – Programmable Logic Array (PLA)
  – Programmable Array Logic (PAL)
  – Field-programmable Gate Array (FPGA)
  – EEPROM, ROM
  – and more...

Programmable Logic Devices (Cont'd)

In PLAs, the desired function (i.e. connectivity among various gates) is programmed electrically by "burning" the corresponding fuses inside the chip.