Introduction

• Objective: To understand how the simple model computer from the previous lecture could be implemented using logic gates.

• Review of Boolean functions and expressions
• Review of logic gates
• Decoders, Encoders, and Multiplexors

References: Dewdney, The New Turing Omnibus (Ch. 3, 13, and 28).
### Boolean Functions

- A Boolean variable has two possible values (true/false) (1/0).
- A Boolean function has a number of Boolean input variables and has a Boolean valued output.
- A Boolean function can be described using a truth table.
- There are $2^n$ Boolean functions of $n$ variables.

#### Multiplexor function

<table>
<thead>
<tr>
<th>$s$</th>
<th>$x_0$</th>
<th>$x_1$</th>
<th>$f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Boolean Expressions

- An expression built up from variables, and, or, and not.

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>x • y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>x + y</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>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>x</th>
<th>x̅</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

and  or  not
Boolean Expressions

• A Boolean expression is a Boolean function.

• Any Boolean function can be written as a Boolean expression
  – Disjunctive normal form (sums of products)
  – For each row in the truth table where the output is true, write a product such that the corresponding input is the only input combination that is true
  – Not unique

• E.G. (multiplexor function)

\[
\overline{s} \cdot \overline{x_0} \cdot \overline{x_1} + \overline{s} \cdot x_0 \cdot x_1 + s \cdot \overline{x_0} \cdot x_1 + s \cdot x_0 \cdot x_1
\]
Boolean Logic

- Boolean expressions can be simplified using rules of Boolean logic
  - Identity law: \( A + 0 = A \) and \( A \cdot 1 = A \).
  - Zero and One laws: \( A + 1 = 1 \) and \( A \cdot 0 = 0 \).
  - Inverse laws: \( A + \overline{A} = 1 \) and \( A \cdot \overline{A} = 0 \).
  - Commutative laws: \( A + B = B + A \) and \( A \cdot B = B \cdot A \).
  - Associative laws: \( A + (B + C) = (A + B) + C \) and \( A \cdot (B \cdot C) = (A \cdot B) \cdot C \).
  - Distributive laws: \( A \cdot (B + C) = (A \cdot B) + (A \cdot C) \) and \( A + (B \cdot C) = (A + B) \cdot (A + C) \).
  - DeMorgan’s laws: \( A + B = A \cdot \overline{B} \) and \( A \cdot B = A + \overline{B} \).

- The reason for simplifying is to obtain shorter expressions, which leads to simpler logic circuits.
Simplification of Boolean Expressions

- Simplifying multiplexor expression using Boolean algebra

\[
\overline{s} \cdot x_0 \cdot \overline{x_1} + \overline{s} \cdot x_0 \cdot x_1 + s \cdot \overline{x_0} \cdot x_1 + s \cdot x_0 \cdot x_1
\]

\[
= \overline{s} \cdot x_0 \cdot \overline{x_1} + \overline{s} \cdot x_0 \cdot x_1 + s \cdot x_1 \cdot \overline{x_0} + s \cdot x_1 \cdot x_0 \quad \text{(commutative law)}
\]

\[
= \overline{s} \cdot x_0 \cdot (x_1 + \overline{x_1}) + s \cdot x_1 \cdot (x_0 + \overline{x_0}) \quad \text{(distributive law)}
\]

\[
= \overline{s} \cdot x_0 \cdot 1 + s \cdot x_1 \cdot 1 \quad \text{(inverse law)}
\]

\[
= \overline{s} \cdot x_0 + s \cdot x_1 \quad \text{(identity law)}
\]

- Verify that the Boolean function corresponding to this expression has the same truth table as the original function.
Logic Circuits

- A single line labeled \( x \) is a logic circuit. One end is the input and the other is the output. If \( A \) and \( B \) are logic circuits so are:
  - and gate

\[
\begin{array}{c}
A \\
B
\end{array}
\]

- or gate

\[
\begin{array}{c}
A \\
B
\end{array}
\]

- inverter (not)

\[
A
\]
Logic Circuits

• Given a Boolean expression it is easy to write down the corresponding logic circuit
• Here is the circuit for the original multiplexor expression
Logic Circuits

• Here is the circuit for the simplified multiplexor expression
Nand Gates

• A nand gate is an inverted and gate

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

nand

• All Boolean functions can be implemented using nand gates (and, or, and not can be implemented using nand)
Decoder

- A decoder is a logic circuit that has n inputs (think of this as a binary number) and $2^n$ outputs. The output corresponding to the binary input is set to 1 and all other outputs are set to 0.
Encoder

- An encoder is the opposite of a decoder. It is a logic circuit that has $2^n$ inputs and n outputs. The output equal to the input line (in binary) that is set to 1 is set to 1.
A multiplexor is a switch which routes n inputs to one output. The input is selected using a decoder.
Implementing Logic Gates with Transistors

A Transistor NOT Gate

A Transistor NAND Gate
Exercises

• Design an OR gate from NAND gates.

• Prove De Morgan’s laws.

• Conjunctive normal form consists of products of sums. Obtain a conjunctive normal form for the multiplexor on slide 5 and draw the corresponding circuit. How does the number of gates compare with the circuit on slide 9.

• Design a 3 × 8 decoder.

• Design an 8 × 3 encoder.

• Redesign the multiplexor on slide 14 using only inverters, three-input NAND gates, and a single four-input NAND gate.

• Show a transistor NOR gate.