Chapter 5
Combinational Logic Analysis
Basic Combinational Logic Circuits

- We know that we can create a logic expression using SOP and POS expressions that use AND and OR gates to implement.
- Now we want to explore the logic functions in more detail.
- AND-OR Logic
  - Used to implement SOP expressions

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Basic Combinational Logic Circuits (continued)

- **AND-OR-Invert Logic**
  - The output of an AND-OR circuit is complemented (inverted)
  - We can implement POS expressions with AND-OR-Invert logic
  - Use DeMorgan's theorems to convert for illustration:

  \[ X = (\overline{A + B})(\overline{C + D}) = (\overline{AB})(\overline{CD}) = (\overline{AB})(\overline{CD}) = \overline{AB + CD} = \overline{AB + CD} \]

  ![Diagram](a)

  ![Diagram](b)

  For a 4-input AND-OR-Invert logic circuit, the output X is LOW (0) if both input A and input B are HIGH (1) or both input C and Input D are High (1).
Basic Combinational Logic Circuits (continued)

- **Exclusive-OR Logic**

![Diagram of Exclusive-OR Logic](image)

- **Exclusive-NOR Logic**

![Diagram of Exclusive-NOR Logic](image)
Implementing Combinational Logic

- From a Boolean Expression to a Logic Circuit
  - Use the order of precedence for and, or, and parentheses
  - Combine the and, or, and inverter gates to create expression
  - e.g. Implement the following: \( X = AB(C \overline{D} + EF) \)
    - First, invert \( D \)
    - Then use AND gates for \( C \overline{D} \) and \( EF \)
    - Then use an OR gate for \( C \overline{D} + EF \)
    - Finally, AND together the outputs of the inputs and or gate

Note that propagation delays may cause problems, in that case, reduce the expression to get the SOP expression.
Implementing Combinational Logic

- From a Truth Table to a Logic Circuit
  - Create the product term from the truth table
  - OR together the product terms to get expression

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

\[ X = \overline{A} B C + A \overline{B} \overline{C} \]
The Universal Property of NAND and NOR Gates

- The NAND Gate as a Universal Logic Element

Note that we use the NAND gate to create any of the primitive operations. We may want to do this to minimize the number of different types of logic ICs on a circuit board.
The Universal Property of NAND and NOR Gates

- The NOR Gate as a Universal Logic Element

(a) One NOR gate used as an inverter

(b) Two NOR gates used as an OR gate

(c) Three NOR gates used as an AND gate

(d) Four NOR gates used as a NAND gate
Combinational Logic Using NAND and NOR Gates

- **NAND Logic**
  
  Using DeMorgan’s theorems to show how to create an SOP expression from three NAND gates.

\[
X = AB + CD
\]

Use the bubble notation as a handy way to do DeMorgan's Theorems.
Combinational Logic Using NAND and NOR Gates

- Use of Appropriate Dual Symbols in NAND Logic Diagram

Red arrows indicate use of the DeMorgan equivalent gate.
Combinational Logic Using NAND and NOR Gates

- **NOR Logic**
  - The NOR gate can create POS expression:

  \[ X = (A + B) + (C + D) \]

  Reduce expression using DeMorgan’s Theorems

  \[ X = (A + B)(C + D) \]
Logic Circuit Operation With Pulse Waveform Inputs

- We can represent each stage in a logic circuit as a response to a waveform input:

\[ Y = B + C \]

- \( Y \) is 1 when either \( B \) or \( C \) is 1
- \( X \) is 0 when both \( Y \) and \( A \) are 1

\[ X = \overline{A} (B + C) = \overline{A}B + \overline{A}C \]
Combinational Logic with VHDL

• Structural Approach to VHDL Programming
  - We can define new logic functions to use as blocks in creating VHDL definitions.
  - In our labs we do this by including a schematic as a part of the new design and create a symbol for it
  - In VHDL you can use a software approach to define these complex blocks.
• **VHDL Components**

Assume we want to define this circuit:

```vhdl
architecture LogicOperation of AND_OR_Logic is
  component AND_gate is
    port(A, B: in bit); X: out bit);
  end component AND_gate;
  component OR_gate is
    port(A, B: in bit); X: out bit);
  end component OR_gate;
  signal OUT1, OUT2: bit;
begin
  G1: AND_gate port map(A=>In1, B=>In2, X=>OUT1);
  G2: AND_gate port map(A=>In3, B=>In4, X=>OUT2);
  G3: OR_gate port map(A=>OUT1, B=>OUT2, X=>OUT3);
end architecture LogicOperation;
```
Combinational Logic with VHDL

- VHDL Concurrency
  - If we define two logic statements in a VHDL definition they are executed concurrently.

```vhdl
entity combinational is
  port(A, B, C, D: in bit; X, Y: out bit);
end entity combinational;

architecture Example of Combinational is
begin
  x<=(A and B) or not C;
  y<=C or not D;
end architecture Example;
```

Concurrent statements. They are executed at the same time.
Combinational Logic with VHDL

- Applying Software Development Tools

Screen shot of Xilinx IDE for a project with module definitions from Lab 7