Binary Data Transmission

• Data transmission for \( n \)-bit data words
  › Parallel
    • all bits at once
    • 1 time step to get all data
  › Serial
    • one bit at a time
    • \( n \) time steps to get all data
  › Serial-parallel
    • both serial & parallel components
    • \( m \) time steps to get all \( n \) bits, \( k \) bits at a time
  › Trade-off:
    • \# inputs/outputs (I/O)
    • speed of data transmission
    • Combinational logic has parallel input data and output data
What is Combinational Logic?

• A collection of logic gates in which there are **NO** feedback loops
  › No feedback loops means there is no path in the circuit on which you will pass through a given gate more than once
  › Also defined as a circuit that can be represented by a directed acyclic graph (no cycles in the graph)
    • Gates represented by *vertices* (aka *nodes*)
    • Connections represented by *edges*
Boolean (Logic) Equations

- Any $n$-input, $m$-output combinational logic circuit can be completely described by a set of $m$ logic equations
  - One logic equation for each output
  - Gives the output responses to all $2^n$ possible combinations of input values

\[
\begin{align*}
O_1 &= f_1(I_1, I_2, \ldots, I_n) \\
O_2 &= f_2(I_1, I_2, \ldots, I_n) \\
& \quad \vdots \\
O_m &= f_m(I_1, I_2, \ldots, I_n)
\end{align*}
\]
Truth Tables

• Any $n$-input, $m$-output combinational logic circuit can be completely described by a truth table
  › Gives the output responses to all $2^n$ possible combinations of input values
  › Therefore, truth tables and logic equations contain the same information

• Two logic equations (or two combinational logic circuits) are *equivalent* if they produce the same truth tables

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>00...00</td>
<td>$v_1...v_m$</td>
</tr>
<tr>
<td>00...01</td>
<td>$v_1...v_m$</td>
</tr>
<tr>
<td>00...10</td>
<td>$v_1...v_m$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>11...11</td>
<td>$v_1...v_m$</td>
</tr>
</tbody>
</table>
Representations of Logic Functions

- Truth Table
- Boolean (or logic) equations
- Sum-of-Products (SOP)
  - \( Z = A \overline{B} + A'C \)
  - AND is product
  - OR is sum
- SOP canonical form
  - \( Z = A'B'C + A'BC + ABC' + ABC \)
  - All literals are present in all product terms
- Minterm (a 1 in a TT row)
  - \( Z = \Sigma_{A,B,C}(1,3,6,7) \)

**Literal** – a single variable or the complement of a variable

**Product term** – a single literal or a logic product of multiple literals

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>Z</th>
<th>Row value</th>
<th>Minterm</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>A’B’C’</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>A’B’C</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>A’BC’</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>A’BC</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>AB’C’</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>5</td>
<td>AB’C</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>6</td>
<td>ABC’</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>ABC</td>
</tr>
</tbody>
</table>
Other Representations

• Product-of-Sums (POS)
  \[ Z = (A+C) \cdot (A'+B) \]

• POS canonical form
  \[ Z = (A+B+C) \cdot (A+B'+C) \cdot (A'+B+C) \cdot (A'+B+C') \]
  All literals are present in all sum terms

• Maxterm (a 0 in a TT row)
  \[ Z = \Pi_{A,B,C}(0,2,4,5) \]
  - Note this is all TT rows not in minterm expression for this example

• POS representations are less often used than SOP

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>Z</th>
<th>Row value</th>
<th>Maxterm</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>A+B+C</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>A+B+C'</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>A+B'+C</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>A+B'+C'</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>A'+B+C</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>5</td>
<td>A'+B+C'</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>6</td>
<td>A'+B'+C</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>A'+B'+C'</td>
</tr>
</tbody>
</table>

*Sum term – a single literal or a logic sum of multiple literals*
Conversion Between Representations

- minterm
- canonical SOP
- canonical POS
- SOP
- POS
- Truth Table
- maxterm
- non-SOP
- non-POS
- P&Ts = Boolean Postulates & Theorems

Easy
Harder
Conversion Between Representations

• Minterm to truth table
  › convert decimal minterm to binary
    \[ Z = \Sigma_{A,B,C}(1,3,6,7) \]
    \[ = \Sigma_{A,B,C}(001,011,110,111) \]
  › place 1s in truth table entry for each minterm
    • Pay attention to input ordering
  › place 0s in all other entries

• Maxterm to truth table
  › convert decimal maxterm to binary
    \[ Z = \Pi_{A,B,C}(0,2,4,5) \]
    \[ = \Pi_{A,B,C}(000,010,100,101) \]
  › place 0s in truth table entry for each maxterm
    • Pay attention to input ordering
  › place 1s in all other entries
Conversion Between Representations

- **Minterm to canonical SOP**
  - convert decimal minterm to binary
    \[ Z = \Sigma_{A,B,C}(1,3,6,7) \]
    \[ = \Sigma_{A,B,C}(001,011,110,111) \]
  - replace 1s and 0s with variable and complement of variable, respectively
    \[ = \Sigma_{A,B,C}(A'B'C,A'BC,ABC',ABC) \]
  - Be sure to maintain input ordering
    - then sum
      \[ = A'B'C+A'BC+ABC'+ABC \]

- **Canonical SOP to minterm**
  - just reverse the procedure above
Conversion Between Representations

• Maxterm to canonical POS
  › convert decimal maxterm to binary
    \[ Z = \Pi_{A,B,C}(0,2,4,5) \]
    \[ = \Pi_{A,B,C}(000,010,100,101) \]
  › replace 0s and 1s with variable and complement of variable, respectively, and sum
    \[ = \Pi_{A,B,C}(A+B+C, A+B'+C, A'+B+C, A'+B+C') \]
  • Be sure to maintain input ordering
  › then take the product of the individual sum terms
    \[ = (A+B+C)\cdot(A+B'+C)\cdot(A'+B+C)\cdot(A'+B+C') \]
    Note that these last 2 steps are the dual of those for minterm to canonical SOP

• Canonical POS to maxterm
  › just reverse the procedure above
Conversion Between Representations

• POS to SOP

 › multiply like in regular algebra then apply P&Ts

\[
Z = (A+C)\cdot(A'+B)
\]

\[
= A\cdot(A'+B)+C\cdot(A'+B)
\]

using P5b

\[
= AA'+AB+CA'+CB
\]

using P5b (SOP but not minimal)

\[
= 0+AB+A'C+CB
\]

using P6b

\[
= AB+A'C+CB
\]

using P2a

\[
= AB+A'C
\]

using T9a

• Canonical POS to SOP

 › use same procedure
Conversion Between Representations

• SOP to canonical SOP
  › Replace all missing variables in each product term with $X+X'$, where $X$ is the missing variable
  • Recall:
    - $X+X' = 1$, and
    - $Y\cdot1 = Y$, so we don’t change the product term
  \[
  Z = AB + A'C
  \]
  \[
  = A\cdot B\cdot 1 + A'\cdot 1\cdot C \quad \text{using P2b}
  \]
  \[
  = A\cdot B\cdot (C+C') + A'\cdot (B+B')\cdot C \quad \text{using P6a}
  \]
  › then multiply
  \[
  = ABC + ABC' + A'BC + A'B'C \quad \text{using P5b}
  \]
  \[
  = A'B'C + A'BC + ABC' + ABC \quad \text{using P3a}
  \]
Conversion Between Representations

• Canonical SOP to minimal SOP
  › apply P&Ts

\[ Z = A'B'C + A'BC + ABC' + ABC \]

\[ = A' \cdot (B'C + BC) + A \cdot (BC' + BC) \quad \text{using P5b} \]
\[ = A' \cdot (C) + A \cdot (BC' + BC) \quad \text{using T6a} \]
\[ = A' \cdot (C) + A \cdot (B) \quad \text{using T6a} \]
\[ = A'C + AB \quad \text{no change, just removed ( ) and •} \]
\[ = AB + A'C \quad \text{using P3a} \]

• Same for SOP to minimal SOP

• **Problem:** How to know when it’s minimal?
Conversion Between Representations

- Non-SOP/non-POS to SOP

  - apply P&Ts

\[
Z = (((A'B')'C')'+D')' = ((A•B)•C)+\overline{D}
\]

\[
= ((A•B)•C)•\overline{D}
\]

using DeMorgan T8a

\[
= ((\overline{A•B})•C)•D
\]

using T3

\[
= ((\overline{A+B})•C)•D
\]

using DeMorgan T8b

\[
= ((A+B')•C)•D
\]

using T3

\[
= (A+B')•C•D
\]

no change, just removed ( )

\[
= ACD+B'CD
\]

(SOP) using P5b

Note: this is a POS
Conversion Between Representations

• SOP to truth table:
  › Place a logic 1 in each truth table output entry whose input value satisfies a given product term = 1
    • A \( k \)-variable product term will produce \( 2^{n-k} \) 1s in the truth table where \( n \) is the total number of input variables
  › Repeat for all product terms

• POS to truth table:
  › Place a logic 0 in each truth table output entry whose input value satisfies a given sum term = 0
    • A \( k \)-variable sum term will produce \( 2^{n-k} \) 0s in the truth table where \( n \) is the total number of input variables
  › Repeat for all sum terms
Using Truth Tables to Prove Theorems

- Consensus Theorem
  - T9a: \( X \cdot Y + X' \cdot Z + Y \cdot Z = X \cdot Y + X' \cdot Z \)
  - T9b: \( (X+Y) \cdot (X'+Z) \cdot (Y+Z) = (X+Y) \cdot (X'+Z) \)

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
<th>Output</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>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</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>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
<th>Output</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>0</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>