Lecture 3:
Combinational Logic Specification
and Simplification:

CSE 140: Components and Design Techniques for Digital Systems

Diba Mirza
Dept. of Computer Science and Engineering
University of California, San Diego
What you should know at the end of this lecture....

1. How to derive the Sum of Product and Product of Sum canonical forms from the truth table.

2. How to design a combinational circuit starting with the truth table.

3. How to simplify switching functions using Boolean Algebra axioms and theorems
Q: Does the following SOP canonical expression correctly express the above truth table:

\[ Y(A,B) = \Sigma m(2,3) \]

A. Yes
B. No
## Sum of Product Canonical Form

<table>
<thead>
<tr>
<th>Minterm</th>
<th>A</th>
<th>B</th>
<th>Carry</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>A’B’</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A’B</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>AB’</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>AB</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

I. \( \text{sum}(A,B) = \)

II. \( \text{carry}(A,B)= \)
DeMorgan’s Theorem

- $Y = \overline{AB} = \overline{A} + \overline{B}$

- $Y = \overline{A + B} = \overline{A} \cdot \overline{B}$
Product of Sum Canonical Form

<table>
<thead>
<tr>
<th>Max term</th>
<th>A</th>
<th>B</th>
<th>Carry</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0 1 0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1 0 0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1 1 1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

We will derive another canonical form using the SOP expression for the compliment of the outputs: Carry and Sum

carry(A,B)=

Product of Sum Canonical Form

<table>
<thead>
<tr>
<th>Max term</th>
<th>A</th>
<th>B</th>
<th>Carry</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0 1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1 0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1 1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

The POS expression for sum(A,B)
A. \((A' + B)(A + B')\)
B. \(A' B + AB'\)
C. \((A+ B')(A' + B)\)
D. Either A or C
PI Q: When would you use the SOP instead of the POS to express the switching function?

A. When the output of the function is TRUE for most input combinations.
B. When the output of the function is FALSE for most input combinations.
C. We always prefer the SOP form because it's more compact.
PI Q: When would you use the SOP instead of the POS to express the switching function?

A. When the output of the function is TRUE for most input combinations.
B. When the output of the function is FALSE for most input combinations.
C. We always prefer the SOP form because its more compact

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Carry</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>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Re-deriving the truth table

Switching Expressions:

Sum \((A, B) = A'B + AB'\)

Carry \((A, B) = AB\)

Ex:

Sum \((0,0) = 0' \cdot 0 + 0 \cdot 0' = 0 + 0 = 0\)

Sum \((0,1) = 0' \cdot 1 + 0 \cdot 1' = 1 + 0 = 1\)

Sum \((1,1) = 1' \cdot 1 + 1 \cdot 1' = 0 + 0 = 0\)

Truth Table

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

Logic circuit for half adder:
The SOP and POS forms don’t usually give the optimal circuit or the simplest Boolean expression of the switching function.

To optimize the circuit, we need to simplify the Boolean expression using:

1. Boolean Algebra axioms and theorems
2. Karnaugh Maps (K-Maps) (next lecture)
Axioms of Boolean Algebra

1. \( B = 0 \), if \( B \) not equal to 1
2. \( 0' = 1 \)
3. \( 1.1 = 1 \)
4. \( 0.1 = 0 \)
5. \( a+0=a, \ a.1=a \)  \hspace{1cm} \text{Identity law}
6. \( a+a' = 1, \ a.a' = 0 \)  \hspace{1cm} \text{Complement law}
Theorems of Boolean Algebra

I. Commutative Law: \[ A + B = B + A \]
\[ AB = BA \]

II. Distributive Law
\[ A(B+C) = AB + AC \]
\[ A+BC = (A+B)(A+C) \]
III Associativity

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

\[(AB)C = A(BC)\]
IV: Consensus Theorem: \[ AB + B' C + AC = AB + B' C \]
PI Q: Which of the following is $AC' + BC + BA$ equal to?

A. $AB + C'A$
B. $AC' + CB$
C. $BC + AB$
D. None of the above
Proof of consensus Theorem using Boolean Algebra

\[ AB + B'C + AC = AB + B' C \]
V. DeMorgan’s Theorem

- \( Y = \overline{AB} = \overline{A} + \overline{B} \)

- \( Y = \overline{A + B} = \overline{A} \cdot \overline{B} \)
Circuit Transformation: Bubble Pushing

- Pushing bubbles backward (from the output) or forward (from the inputs) changes the body of the gate from AND to OR or vice versa.

- Pushing a bubble from the output back to the inputs puts bubbles on all gate inputs.

- Pushing bubbles on all gate inputs forward toward the output puts a bubble on the output and changes the gate body.
Example of transforming circuits using bubble pushing
Shannon’s Expansion

• Shannon’s expansion assumes a switching algebra system
• Divide a switching function into smaller functions
• Pick a variable \( x \), partition the switching function into two cases: \( x=1 \) and \( x=0 \)
  
  \[ f(x,y,z,\ldots) = xf(x=1,y,z,\ldots) + x'f(x=0,y,z,\ldots) \]
Shannon’s expansion

- Shannon’s expansion:
  - $f(x) = xf(1) + x' f(0) = (x + f(0)). (x' + f(1))$
  - $f(x,y) = xf(1,y) + x' f(0,y)$
  - $f(x,y,z,\ldots) = xf(1,y,z,\ldots) + x' f(0,y,z,\ldots)$
Proof of Shannon’s Expansion

\[ f(x,y) = (x + f(0,y))(x' + f(1,y)) \] {Enumerative induction}
Shannon’s Expansion: Example

\[ f(x,y,z) = xf(0,y,z) + x' f(1,y,z) \]

Is the above equation correct?

A. Yes
B. No
Shannon’s Expansion

Decompose the switching function into min terms
\[ f(x,y) = xf(1,y) + x'f(0,y) \]
Shannon’s Expansion

Decompose the switching function into max terms
\[ f(x,y) = (x + f(0,y)). (x' + f(1,y)) \]
Reduction of Boolean Expression

\[ AB + AC + B' \cdot C \]
\[ = AB + B' \cdot C \]

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

Prove the reduction using
(1) Boolean algebra,
(2) Logic simulation and
(3) Shannon’s expansion (exercise)
We are in a position to build a circuit to do n-bit Binary Addition

\[
\begin{array}{c}
5 \\
+ 7 \\
\hline
12
\end{array}
\]

Carry  Sum

\[
\begin{array}{c}
1 \\
1 \\
1 \\
+ 1 \\
+ 1 \\
\hline
12
\end{array}
\]

Carryout  Sums
Binary Addition: Hardware

- **Half Adder:** Two inputs \((a,b)\) and two outputs \((\text{carry}, \text{sum})\).

- **Full Adder:** Three inputs \((a,b,c)\) and two outputs \((\text{carry}, \text{sum})\).
Full Adder

Truth Table

<table>
<thead>
<tr>
<th>Id</th>
<th>a</th>
<th>b</th>
<th>c_{in}</th>
<th>carry</th>
<th>sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
## Minterm and Maxterm

<table>
<thead>
<tr>
<th>Id</th>
<th>a</th>
<th>b</th>
<th>c&lt;sub&gt;in&lt;/sub&gt;</th>
<th>carryout</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>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>a+b+c’</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>a+b’ +c</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>a’ b c</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>a’ +b+c</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>a b’ c</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>a b c’</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>a b c</td>
</tr>
</tbody>
</table>

maxterm

minterm
# Minterm and Maxterm

<table>
<thead>
<tr>
<th>Id</th>
<th>a</th>
<th>b</th>
<th>c&lt;sub&gt;in&lt;/sub&gt;</th>
<th>carryout</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ f_1(a,b,c) = a' bc + ab' c + abc' + abc \]

\[ f_2(a,b,c) = (a+b+c)(a+b+c')(a+b'+c)(a' +b+c) \]

\[ f_1(a, b, c) = m_3 + m_5 + m_6 + m_7 = Sm(3,5,6,7) \]

\[ f_2(a, b, c) = M_0M_1M_2M_4 = PM(0, 1, 2, 4) \]

PI Q: Is \( f_1 = f_2 \)?

A. Yes
B. No
\[ f_1(a,b,c) = a'bc + ab'c + abc' + abc \]
\[ f_2(a,b,c) = (a+b+c)(a+b+c')(a+b'+c)(a' +b+c) \]
\[ f_1(a, b, c) = m_3 + m_5 + m_6 + m_7 = Sm(3,5,6,7) \]
\[ f_2(a, b, c) = M_0M_1M_2M_4 = PM(0, 1, 2, 4) \]

**PI Q: Is \( f_1 = f_2 \)?**

**A. Yes**

**B. No**
PIQ: Reduce using Boolean algebra theorems

\[ \text{Carry}(A,B,C) = A'BC + AB'C + ABC' + ABC \]

A. \( A'BC + AB'C + AB \)

B. \( A'BC + AC + AB \)

C. \( AB + AC + CA \)

D. \( ABC \)

E. Cannot be reduced further
A'BC + AB'C + ABC' + ABC
Circuit for Full Adder Carry out

\[ f_1(a,b,c) = a'bc + ab'c + abc' + abc \]
\[ = ab + c(b+a) \]
Building the simplest possible circuit

We can get the simplest circuit in two ways:

1. Truth table –> Canonical POS/SOP –> Most simplified switching function –> Simplest Circuit
   • This can get difficult when we have many inputs variables

2. Truth table or Canonical POS/SOP –> Karnaugh map –> Simplest Circuit (Next lecture)