Announcements

- HW #1 due next Wednesday 4/13 at 11:59 PM
- HW #2 out, due Wednesday 4/20 at 11:59 PM
- Midterm 1 coming up soon
Today’s Agenda

- Consensus Theorem
- Shannon’s Expansion
- Full Adder and Subtractor
- Karnaugh Maps
Consensus Theorem

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


Proof: \(AB + A'C + BC = AB + A'C + (1)BC\)

\[ = AB + A'C + (A + A')BC \]
\[ = AB + A'C + ABC + A'BC \]
\[ = AB(1 + C) + A'C(1 + B) \]
\[ = AB + A'C \]
Consensus Examples

1. \( a\bar{b} + a'b\bar{c}d + b'd \)
   - Consensus of \( a \) and \( a' \) (\( a\bar{b}, a'b\bar{c}d \)) is \( bcde \)
   - Consensus of \( b \) and \( b' \) (\( a\bar{b}, b'd \)) is \( ade \)
   - Consensus of \( b \) and \( b' \) (\( a'b\bar{c}d, b'd \)) is \( a'cd \)

2. \((a + c'd')(a' + bc'd')(bc'd' + c'd') = (a + c'd')(a' + bc'd')\)
   - \( A = a, B = c'd', A' = a', C = bc'd' \)
   - Consensus of \( a \) and \( a' \) is \((bc'd' + c'd')\)
Shannon’s Expansion

- Choose a variable to decompose the equation into simpler terms

Sum of Products Form: \( f(x, y, z) = x f(1, y, z) + x' f(0, y, z) \)

Product of Sums Form: \( f(x, y, z) = (x + f(0, y, z))(x' + f(1, y, z)) \)

- If we were to choose \( y \), we would have:
  - \( f(x, y, z) = y f(x, 1, z) + y' f(x, 0, z) \)
  - \( f(x, y, z) = (y + f(x, 0, z))(y' + f(x, 1, z)) \)
Shannon’s Expansion Example #1

\[ f(x, y, z) = xyz + x'y + z'y \]
\[ f(x, y, z) = x \cdot f(1, y, z) + x' \cdot f(0, y, z) \]
\[ = x(yz + 0 + z'y) + x'(0 + y + z'y) \]
\[ = x(y(z+z')) + x'(y(1+z')) \]
\[ = xy + x'y \]
\[ = y \]
Shannon’s Expansion Example #2

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

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

\[
= (x + (y + z')(1)(z + y))(x' + (1)(y' + z)(z + y))
\]

\[
= (x + (y + z')(y + z))(x' + (y' + z)(y + z))
\]

\[
= (x + y)(x' + z)
\]
Full Adder

- Takes 1-bit A, B, \( C_{\text{in}} \)
- Outputs \( C_{\text{out}} \), \( S \)
- \( C_{\text{in}} \) = Carry In
- \( C_{\text{out}} \) = Carry Out
- \( S = (A+B+C_{\text{in}}) \mod 2 \)
  - If \( S \) exceeds 1, then \( C_{\text{out}} = 1 \)

<table>
<thead>
<tr>
<th>id</th>
<th>A</th>
<th>B</th>
<th>( C_{\text{in}} )</th>
<th>( C_{\text{out}} )</th>
<th>S</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>
## Full Subtractor

- Takes 1-bit $A$, $B$, $B_{\text{in}}$
- Outputs $B_{\text{out}}$, $D$
- $B_{\text{in}} = \text{Borrow In}$
- $B_{\text{out}} = \text{Borrow Out}$
- $D = A - B - B_{\text{in}}$
  - If we ever have to perform $0 - 1$, then $B_{\text{out}} = 1$

<table>
<thead>
<tr>
<th>id</th>
<th>$A$</th>
<th>$B$</th>
<th>$B_{\text{in}}$</th>
<th>$B_{\text{out}}$</th>
<th>$D$</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>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</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>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</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>
Karnaugh Maps

- Reduce boolean expressions by writing the entries of the truth table and reducing the expression to either SOP or POS terms
- Row and column entries ordered in gray code (i.e. 00, 01, 11, 10)
- **Prime Implicant**: Obtained by combining all possible maximum number of squares/cells
- **Essential Prime Implicant**: Prime Implicant is essential if it is the only prime implicant that only covers the minterm
  - Must be included in every simplified boolean expression

$f(a,b,c) = \Sigma_m(0, 2, 4, 5) = ab' + a'c' + b'c'$

Prime Implicants: $AB'$, $A'C'$, $B'C'$

Essential Prime Implicants: $AB'$, $A'C'$
### 3 way K-map with Don’t Cares

**Function:**

\[ f(a,b,c) = \sum_m(0, 2, 4, 5) + \sum_d(1, 6) = b' + c' \]

**Prime Implicants:**

- \( B' \)
- \( C' \)

**Essential Prime Implicants:**

- \( B' \)
- \( C' \)

<table>
<thead>
<tr>
<th>C/AB</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
3 way K-map for Full Adders

K-map for $S$:

\[
\begin{array}{c|cccc}
C_{in}/AB & 00 & 01 & 11 & 10 \\
\hline
0 & 0 & 1 & 0 & 1 \\
1 & 1 & 0 & 1 & 0 \\
\end{array}
\]

\[
S = A'B'C_{in} + A'BC_{in}' + ABC_{in} + AB'C_{in}'
\]

K-map for $C_{out}$:

\[
\begin{array}{c|cccc}
C_{in}/AB & 00 & 01 & 11 & 10 \\
\hline
0 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 1 & 1 \\
\end{array}
\]

\[
C_{out} = BC_{in} + AB + AC_{in}
\]
Recall

\[ AB' + A'B = A \text{ XOR } B, \]
\[ AB + A'B' = A \text{ XNOR } B = (A \text{ XOR } B)' \]

Then:

\[ S = A'B'C + A'BC' + ABC' + AB'C' \]
\[ = B(AC + A'C') + B'(A'C + AC') \]
\[ = B(A \text{ XOR } C)' + B'(A \text{ XOR } C) \]
\[ = B \text{ XOR } (A \text{ XOR } C) \]
\[ = B \text{ XOR } A \text{ XOR } C_{\text{in}} \]
Full Adder Diagram
3 way K-map for Full Subtractors

K-map for D:

<table>
<thead>
<tr>
<th>B_{in}/AB</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

\[ D = A'B'B_{in} + A'BB_{in}' + ABB_{in} + AB'B_{in} \]

K-map for B_{out}:

<table>
<thead>
<tr>
<th>B_{in}/AB</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

\[ B_{out} = A'B_{in} + A'B + BB_{in} \]
Recall
\[ AB' + A'B = A \text{ XOR } B, \]
\[ AB + A'B' = A \text{ XNOR } B = (A \text{ XOR } B)' \]

\[ D = A'B'B_{\text{in}} + A'BB_{\text{in}}' + ABB_{\text{in}} + AB'B_{\text{in}}' \]
\[ = A(BB_{\text{in}} + B'B_{\text{in}}') + A'(B'B_{\text{in}} + BB_{\text{in}}') \]
\[ = A(B \text{ XOR } B_{\text{in}})' + A'(B \text{ XOR } B_{\text{in}}) \]
\[ = A \text{ XOR } B \text{ XOR } B_{\text{in}} \]
Full Subtractor Diagram
### 4 way K-map

The function $f(a, b, c, d) = \sum_m(0, 1, 8, 9, 10) + \sum_d(2, 4, 5, 11, 13)$ is represented using a K-map. Here is the K-map:

<table>
<thead>
<tr>
<th>$cd \setminus ab$</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>1</td>
<td>X</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>01</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>10</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**Prime Implicants:**
- $a'c'$
- $b'd'$
- $b'c'$
- $c'd$
- $ab'$

**Essential Prime Implicants:**
- None
A 4-way K-map is shown with the function:

\[ f(a,b,c,d) = \sum_m(0, 1, 8, 9, 10) + \sum_d(2, 4, 5, 11, 13) \]

The simplified expressions for the function are:

\[ f = b'd' + b'c' \]
\[ f = b'c' + ab' \]
\[ f = c'd + b'd' \]
\[ f = ab' + b'c' \]