We Can Use Logical Completeness to Express Functions

Let the truth table to the right define the function $F$.

Recall that we can use the logical completeness construction to write $F$ as a Boolean expression:

- This row is... $AB'C$
- And this is... $ABC'$
- And this is... $ABC$

What's the Best Way to Write Function $F$?

So $F = AB'C + ABC' + ABC$.

But we can also write $F = AB + AC$.

What about $F = A(B + C)$?

Which one is best?

Your Answer Is Wrong! Choose a Metric First

The answer depends on our choice of metric!

How do we measure good?

Common answers for circuit design:

- area / size / cost, OR
- performance / speed, OR
- power / energy consumption, OR
- complexity / reliability.
We Use Heuristics for These Metrics

In practice, measuring exactly is expensive (~$50-100M for a full design, and ~$2-5M just for trying something.) Instead, we use heuristics, which are ways of estimating a metric.

A good heuristic is
- reasonably accurate, and
- monotonic relative to a real measurement
- (so that bigger estimates mean bigger measurements).

An Area Heuristic for ECE120

Here’s a heuristic for area:
- Count literals (A, A’, B, B’, C, C’), then
- Add the number of operations (not including complemented literals).

Why does it work? Remember gate structures?
- each input (literal) → two transistors
- operators into operators → two transistors

So it gives an approximate transistor count.

But wires also take space!

The Area Heuristic Favors F = A (B + C)

Let’s calculate the area heuristic for our three forms of F.
So F = A (B + C) is the smallest design.

<table>
<thead>
<tr>
<th>Form of F</th>
<th>Lits</th>
<th>Ops</th>
<th>Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>AB’C + ABC’ + ABC</td>
<td>9</td>
<td>4</td>
<td>13</td>
</tr>
<tr>
<td>AB + AC</td>
<td>4</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>A (B + C)</td>
<td>3</td>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>

A Delay (Speed) Heuristic for ECE120

Here’s a heuristic for delay / speed:
- Find the maximum number of gates between any input and any output.
- Include complemented literals if the complemented values are not available.

Why does it work?
- Each gate takes time switch its output on/off.
- We call this time a gate delay.

So it gives an approximate delay between inputs changing and outputs changing.
The Delay Heuristic Favors the Last Two Forms

Now let’s calculate the delay heuristic.
So \( F = AB + AC \) and \( F = A (B + C) \) are the fastest designs.

<table>
<thead>
<tr>
<th>Form of F</th>
<th>Lits</th>
<th>Ops</th>
<th>Area</th>
<th>Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>( ABC + ABC' + ABC )</td>
<td>9</td>
<td>4</td>
<td>13</td>
<td>2 or 3</td>
</tr>
<tr>
<td>( AB + AC )</td>
<td>4</td>
<td>3</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>( A (B + C) )</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>

We Have a Winner: \( F = A (B + C) \)

\( F = A (B + C) \) is best by both metrics.
But the answers are not always so simple.
Sometimes no solution is best by both metrics.
- See Section 2.1.1 for a simple example.
- Later in our class, we will explore more space/time tradeoffs in design.
- In practice, tradeoffs are commonplace.
- Take a look at Section 2.1.6* for more.

What About Power and Complexity?

These two metrics are beyond our class’ scope.
You’ll see power in ECE385.
One heuristic for power
- uses the fact that current flows when a transistor switches on/off
- and uses simulation to estimate the number of times that happens.

Complexity is hard to measure, and is usually based on experience.