Combinatorial circuits - arithmetics

Lecturer: Guillaume Beslon
Original Author: Lionel Morel

Computer Science and Information Technologies - INSA Lyon

Fall 2019
Blackboard:
How to add two integers?
Half-Adder

The simplest one-bit operation used is called a half adder:

- inputs: two bits \( a \) and \( b \) that we want to add;
- outputs: a sum bit \( s \) and an output carry \( c_o \).

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></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>1</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>
</tbody>
</table>

We see that:

\[ s = a \oplus b, \quad \text{et} \quad c_o = a \cdot b. \]

We build the corresponding circuit:

Ah! But we also need to take the input carry into account.
Blackboard
Full-Adder

The full adder has:

- inputs: 2 bits \(a\) and \(b\), and one bit for the incoming carry \(c_i\);
- outputs: one bit for the sum \(s\) and one bit for the output carry \(c_o\).

<table>
<thead>
<tr>
<th>(a)</th>
<th>(b)</th>
<th>(c_i)</th>
<th>(s)</th>
<th>(c_o)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<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>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</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>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

We see that:

\[ s = \overline{a}(b \oplus c_i) + a(b \oplus c_i). \]

Since \(x \oplus y = \overline{x}y + x\overline{y}\), we get

\[ s = a \oplus (b \oplus c_i) = a \oplus b \oplus c_i. \]

For the carry:

\[ c_o = \overline{a}bc_i + a\overline{b}c_i + ab\overline{c_i} + abc_i \]

\[ = (\overline{a}b + a\overline{b})c_i + ab(\overline{c_i} + c_i) \]

\[ = (a \oplus b)c_i + ab.\]
We get the circuit for a 1-bit *full adder*:
We can now cascade $k$ full adders to build a circuit that adds up $2^k$-bits natural numbers.

NB: The carry is propagated exactly as we do in the hand-written operation.
From addition to substraction

Blackboard
Optimization criteria

A boolean function $f$ can be implemented by many $(\to \infty)$ circuits. What choice criteria can we use?

- **number of logical gates** $\to$ circuit (die) area
- **delay** $\to$ circuit frequency
- **power consumption**

Optimization algorithms are exponential (in the number of input variables) $\to$ an optimal circuit might not be found in a practicable time.

Example: Let’s see how we can reduce a circuit’s delay by parallelizing it.
Any circuit has a certain **Propagation delay** $\tau$, defined as the time between a change on the circuit’s inputs (at $t$) and the stabilization of the circuit’s outputs (at $t + \tau$).

During $[t, t + \tau]$, outputs can be in **unpredictable transient states**.
Propagation Delay and Critical Path

Each logical gate has a given (physically-fixed) delay.

To determine a circuit’s propagation delay, we need:

▶ compute the propagation delay associated to each path that connect one input to one output of the circuit;
▶ identify a path whose delay is maximal. This is called a critical path

A circuit’s propagation delay = delay of one of its critical paths.

NB: There may be several critical paths.
Example

![Diagram of logic gates](https://en.wikipedia.org/wiki/Propagation_delay)
Carry Lookahead

With a hypothetical 1 t propag. delay for all logical gates, the full adder’s delay is 3 t. And the delay from \( c_i \) à \( c_o \) is 2 t.

For a 4-bits adder, the critical path is the one from \( c_i \) to \( c_o \): delay = 8 t.

For a 8-bits adder, delay is 16 t.

⇒ Delay is proportional to the length of the carry propagation path.
Carry Lookahead

Speculate possible results:

- Compute the 4 LSbits of the result $s$, together with $c_i$
- In parallel, compute two versions of the 4 MSbits of $s$
  - One assuming that $c_i = 0$
  - One assuming that $c_i = 1$
- Choose $s_{7..4}$ according to $c_i$
Optimizing the (8-bits) adder

If mux have a 2t delay. After 8t,

- $c_i$ is known,
- the two versions (one for $c_i = 0$, one for $c_i = 1$)
- The selection can now occur, based on $c_i$.

Delay from $c_i$ to $c_o$ falls to $c_i \rightarrow c_o 10t$.

- We have just introduced **parallelism**.
- This reduces delay
- but it augments surface
- aka **space-time tradeoff**\(^2\)....

\(^2\)classic in CS, check out: https://en.wikipedia.org/wiki/Space%E2%80%93time_tradeoff
Next lecture...

HAL 9000: Do you want me to repeat the last response?

Or, how to remember stuff that we will (surely) need later.
aka, Registers and Memories.
(but maybe we finish combinatorial circuits first)