# Combinatorial circuits - arithmetics 

## Lecturer: Guillaume Beslon (Lecture adapted from Lionel Morel)

Computer Science and Information Technologies - INSA Lyon
Fall 2023

## 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 $c_{o}$.

| $a$ | $b$ | $s$ | $c_{o}$ |
| :---: | :---: | :---: | :---: |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

## 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 $c_{0}$.

| $a$ | $b$ | $c_{i}$ | $s$ | $c_{o}$ |
| :---: | :---: | :---: | :---: | :---: |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

We see that:

$$
s=\bar{a}\left(b \oplus c_{i}\right)+a\left(\overline{b \oplus c_{i}}\right)
$$

Since $x \oplus y=\bar{x} y+x \bar{y}$, we get

$$
s=a \oplus\left(b \oplus c_{i}\right)=a \oplus b \oplus c_{i}
$$

For the carry:

$$
\begin{aligned}
c_{o} & =\bar{a} b c_{i}+a \bar{b} c_{i}+a b \overline{c_{i}}+a b c_{i} \\
& =(\bar{a} b+a \bar{b}) c_{i}+a b\left(\overline{c_{i}}+c_{i}\right) \\
& =(a \oplus b) c_{i}+a b .
\end{aligned}
$$

## Full-Adder (contd)

We get the circuit for a 1-bit full adder :




## n-bits 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.

## Blackboard: From addition to substraction

## Optimization criteria

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

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

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

Example: Let's see how we can reduce a circuit's delay by parallelizing it.

## Propagation Delay

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 ${ }^{1}$


${ }^{1}$ 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 $\mathbf{2 t}$.


For a 4-bits adder, the critical path is the one from $c_{i}$ to $c_{0}$ : delay $=8 \mathbf{t}$.


For a 8-bits adder, delay is $16 \mathbf{t}$.

$\Rightarrow$ 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 2 t 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 $c_{i}$ à $c_{o} \mathbf{1 0 t}$.

- We have just introduced parallelism.
- This reduces delay
- but it increases surface
- aka space-time tradeoff ${ }^{2}$....

[^0]
[^0]:    ${ }^{2}$ classic in CS, check out:
    https://en.wikipedia.org/wiki/Space\%E2\%80\%93time_tradeoff

