

Combinatorial circuits - arithmetics

Lecturer: Guillaume Beslon
(Lecture adapted from Lionel Morel)

Computer Science and Information Technologies - INSA Lyon

Fall 2026

Blackboard:
How to add two integers?

Half-Adder

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

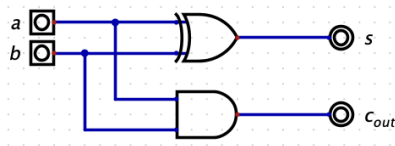
- ▶ inputs : two bits a and b that we want to add;
- ▶ outputs : a sum bit s and an output carry c_{out} .

a	b	s	c_{out}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

We see that:

$$s = a \oplus b, \quad \text{et} \quad c_{out} = a \cdot b.$$

We build the corresponding circuit:



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_{in} ;
- ▶ outputs: one bit for the sum s and one bit for the output carry c_{out}

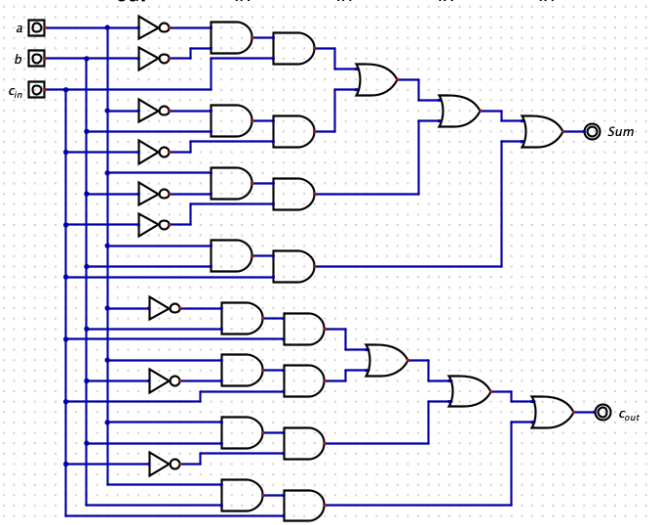
a	b	c_{in}	s	c_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

- ▶ The normal form gives:

$$s = \bar{a}\bar{b}c_{in} + \bar{a}b\bar{c}_{in} + a\bar{b}\bar{c}_{in} + abc_{in}$$
$$c_{out} = \bar{a}bc_{in} + a\bar{b}c_{in} + ab\bar{c}_{in} + abc_{in}$$

Full-Adder

$$s = \overline{a}\overline{b}c_{in} + \overline{a}b\overline{c}_{in} + a\overline{b}\overline{c}_{in} + abc_{in}$$
$$c_{out} = \overline{a}bc_{in} + a\overline{b}c_{in} + ab\overline{c}_{in} + abc_{in}$$



But the normal form of the full adder can be simplified!

- ▶ $s = \overline{a}\overline{b}c_{in} + \overline{a}b\overline{c_{in}} + a\overline{b}\overline{c_{in}} + abc_{in}$ becomes:

$$s = \overline{a}(b \oplus c_{in}) + a(\overline{b \oplus c_{in}})$$

- ▶ Since $x \oplus y = \overline{x}y + x\overline{y}$, we get:

$$s = a \oplus (b \oplus c_{in}) = a \oplus b \oplus c_{in}.$$

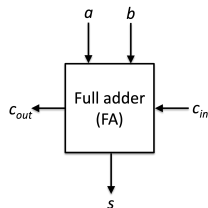
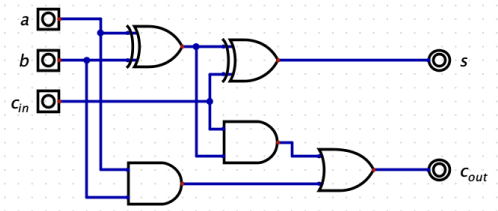
- ▶ For the carry, $c_{out} = \overline{a}bc_{in} + a\overline{b}c_{in} + ab\overline{c_{in}} + abc_{in}$ becomes:

$$c_{out} = (\overline{a}b + a\overline{b})c_{in} + ab(\overline{c_{in}} + c_{in})$$

$$c_{out} = (a \oplus b)c_{in} + ab.$$

Full-Adder (contd)

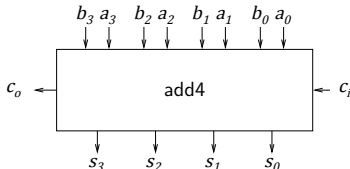
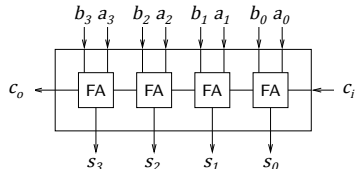
We get a much simpler circuit for a 1-bit *full adder* :



n-bits Adder

We can now cascade k *Full Adders* (FA) to build a circuit that adds up 2 k -bits natural numbers.

Example: a 4-bits adder:



NB: The carry is propagated exactly as we do in the hand-written operation¹.

¹This is exactly the column addition method we learned in elementary school!

Blackboard: From addition to subtraction

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** τ , 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.

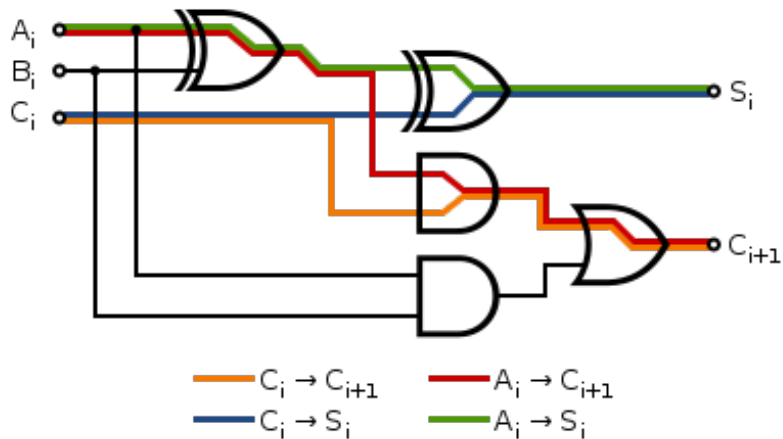
To determine a circuit's propagation delay, we need:

- ▶ 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.

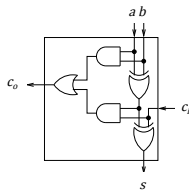
Example²



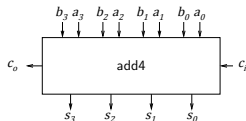
²https://en.wikipedia.org/wiki/Propagation_delay

Carry Lookahead

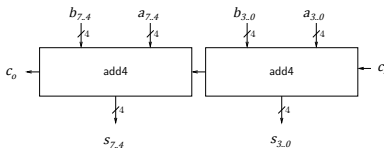
With a hypothetical 1 t propag. delay for all logical gates, the *full adder's* delay is, (from a or b to c_{out}): 3 t . Similarly, the delay from c_{in} to c_{out} is 2 t.



For a 4-bits adder, the critical path is the one from c_{in} to c_{out} : delay = **9 t**.



For a 8-bits adder, delay is **17 t**.

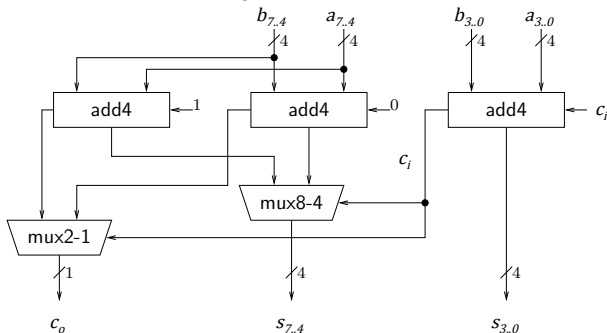


⇒ Delay is proportional to the length of the carry propagation path.

Carry Lookahead

Speculate possible results:

- ▶ Compute the 4 LSbits of the result s , together with c_{in}
- ▶ In parallel, compute two versions of the 4 MSbits of s
 - ▶ One assuming that $c_{in} = 0$
 - ▶ One assuming that $c_{in} = 1$
- ▶ Choose $s_{7..4}$ according to c_{in}



Optimizing the (8-bits) adder

If mux have a $2t$ delay. After $9t$,

- ▶ c_{in} is known,
- ▶ the two versions (one for $c_{in} = 0$, one for $c_{in} = 1$)
- ▶ The selection can now occur, based on c_{in} .

Delay from c_{in} to c_{out} falls to **$11t$** .

- ▶ We have just introduced **parallelism**.
- ▶ This reduces delay
- ▶ but it increases surface
- ▶ aka **space-time tradeoff**³....

³classic in CS, check out:

https://en.wikipedia.org/wiki/Space%E2%80%93time_tradeoff