



Electronics Systems (938II)

Lecture 4.2

Programmable Logic Devices – FPGA overview

Introduction to FPGA*



- Latest and most advanced PLD
 - **FPGA = Field-Programmable Gate Array**

*Latest evos in terms of
Programmable logic devices.*



Introduction to FPGA*

- FPGAs are similar to PALs
 - Programmable device for implementing different logic functions at hardware level
 - The same piece of hardware, but different configurations → different circuits
 - Memory-like programming approach
 - OTP
 - Anti-fuses
 - Reprogrammable
 - SRAM-based

Introduction to FPGA*

- How FPGAs are programmed
 - From computer through USB
 - Dedicated software
 - Each vendor use a different EDA software
 - Using latest HDLs
 - Verilog
 - VHDL
 - SystemVerilog (*sor of Verilog evb*)

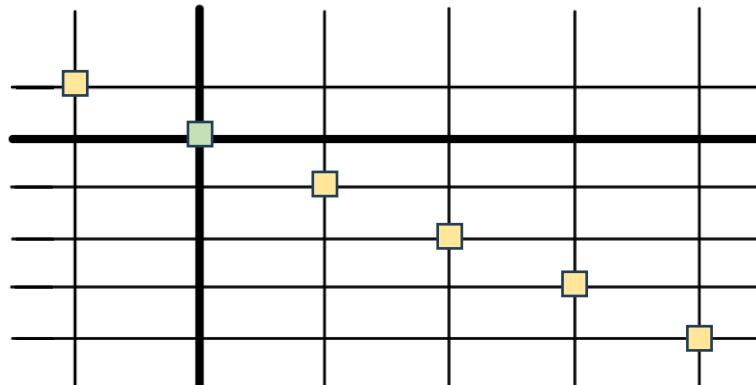
Introduction to FPGA*

- However, FPGAs show some differences from PALs
 - Routing
 - Logic function block

Routing*

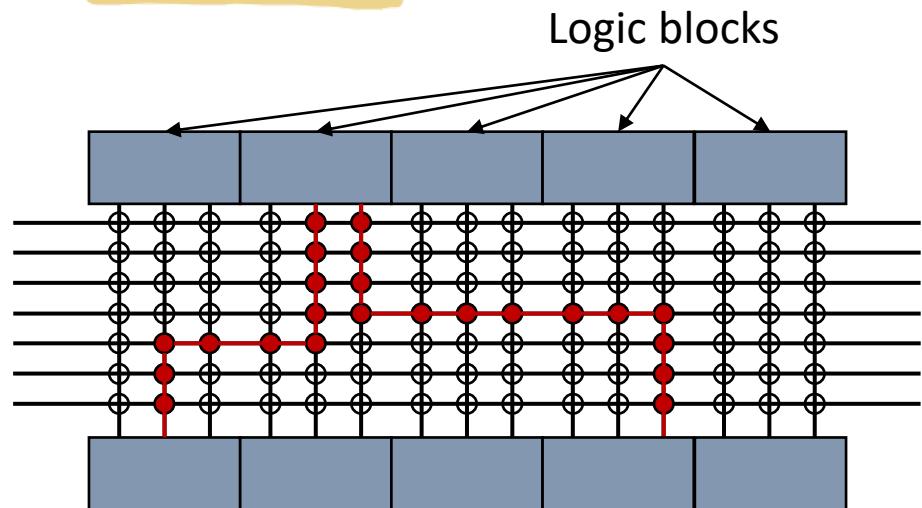
PAL

- Fixed routing wires
 - Programmable nodes
 - Non-segmented wires



Segmented wires: connection between blocks not predefined. Matrix of FPGA connections to create a path from one block to the other one.

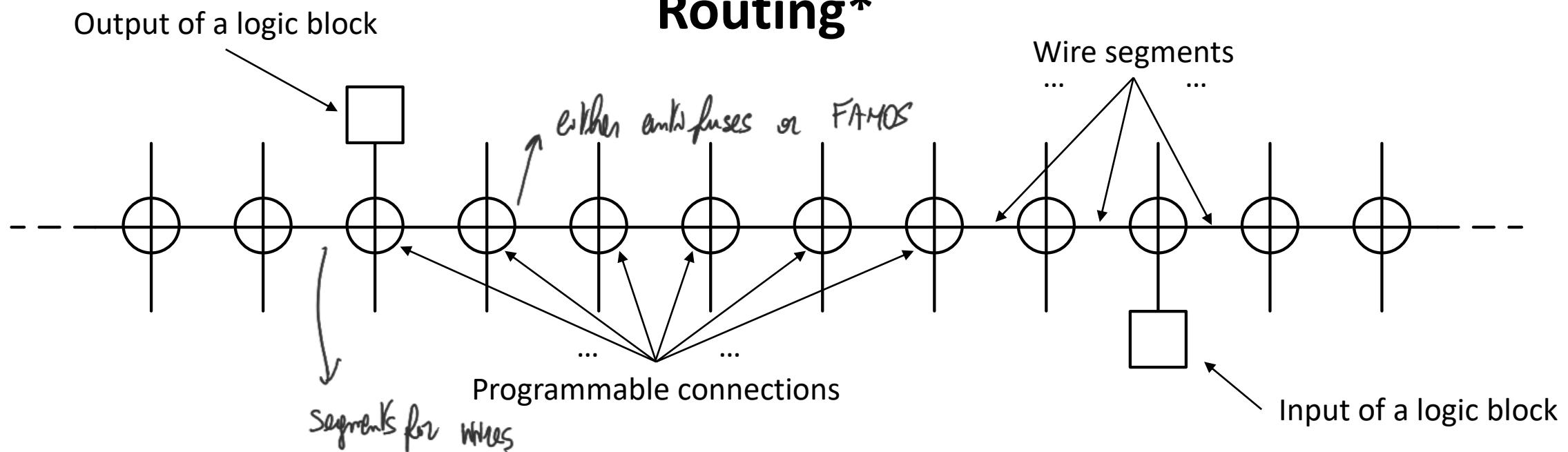
- Programmable routing wires
 - Segmented wires



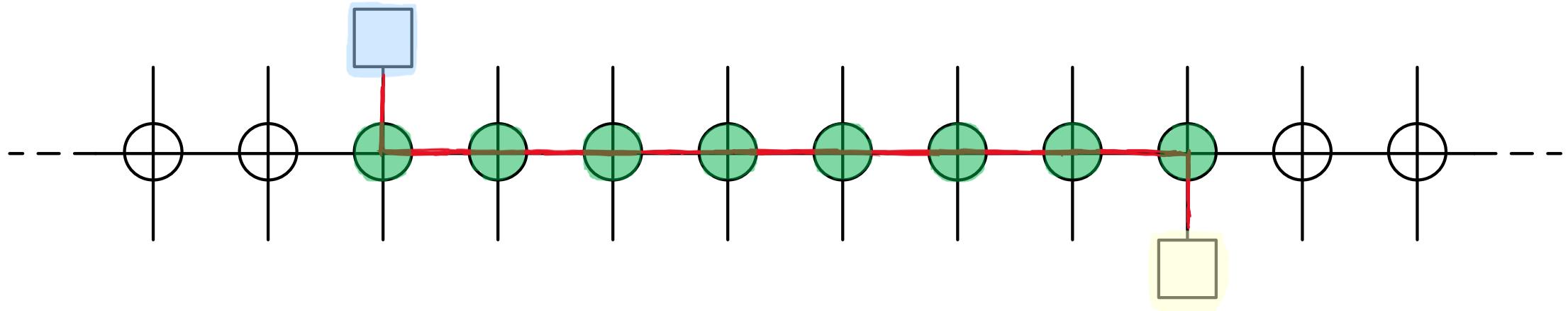
Routing*

- Segmented routing in FPGAs
 - Higher flexibility Advantages: move without predefined paths.
 - Higher wire delay
 - Can have an even greater impact on circuit speed (propagation delay) than the logic block
 - Propagation with segmented approach is higher.

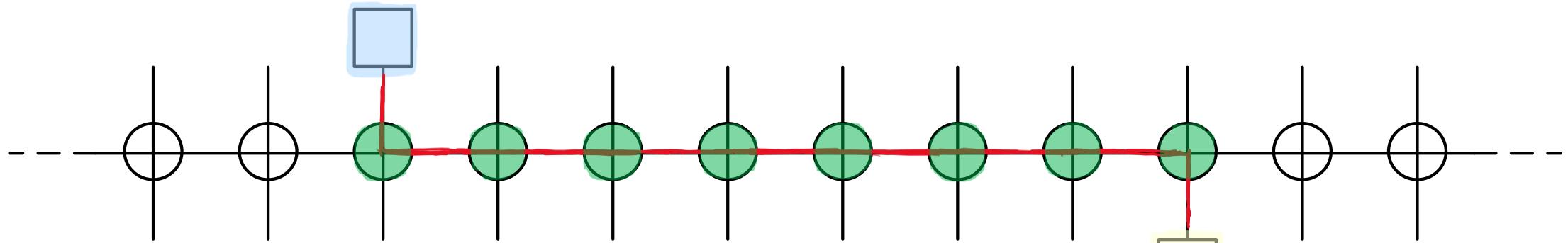
Routing*



Routing*



Routing*



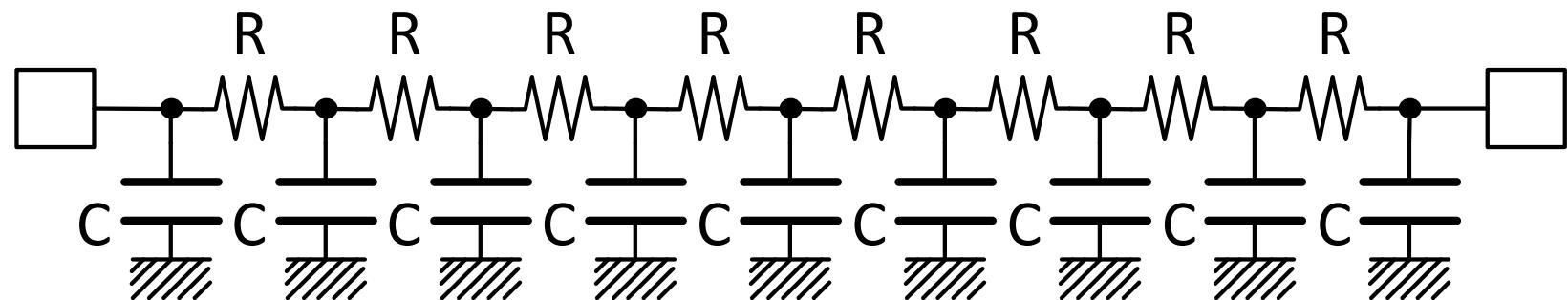
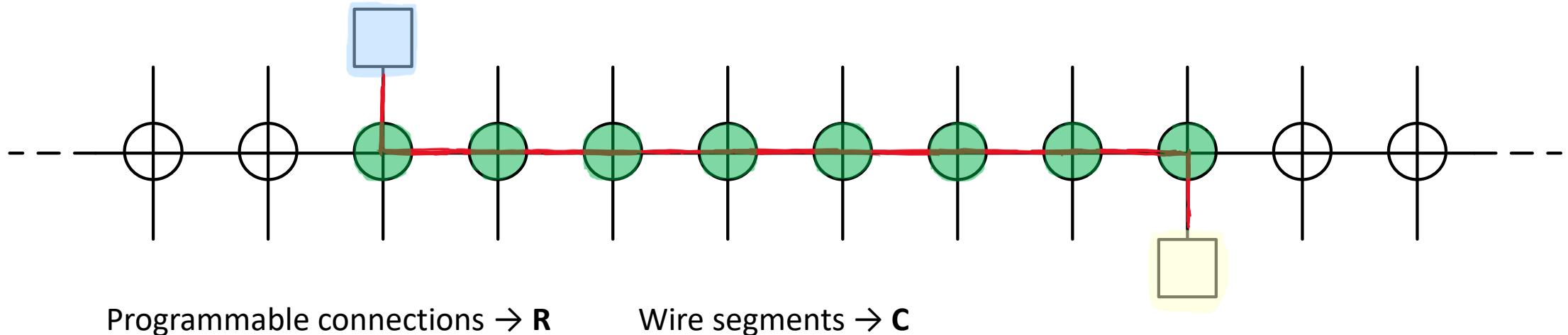
Programmable connections $\rightarrow R$

Can be modeled
as resistance R

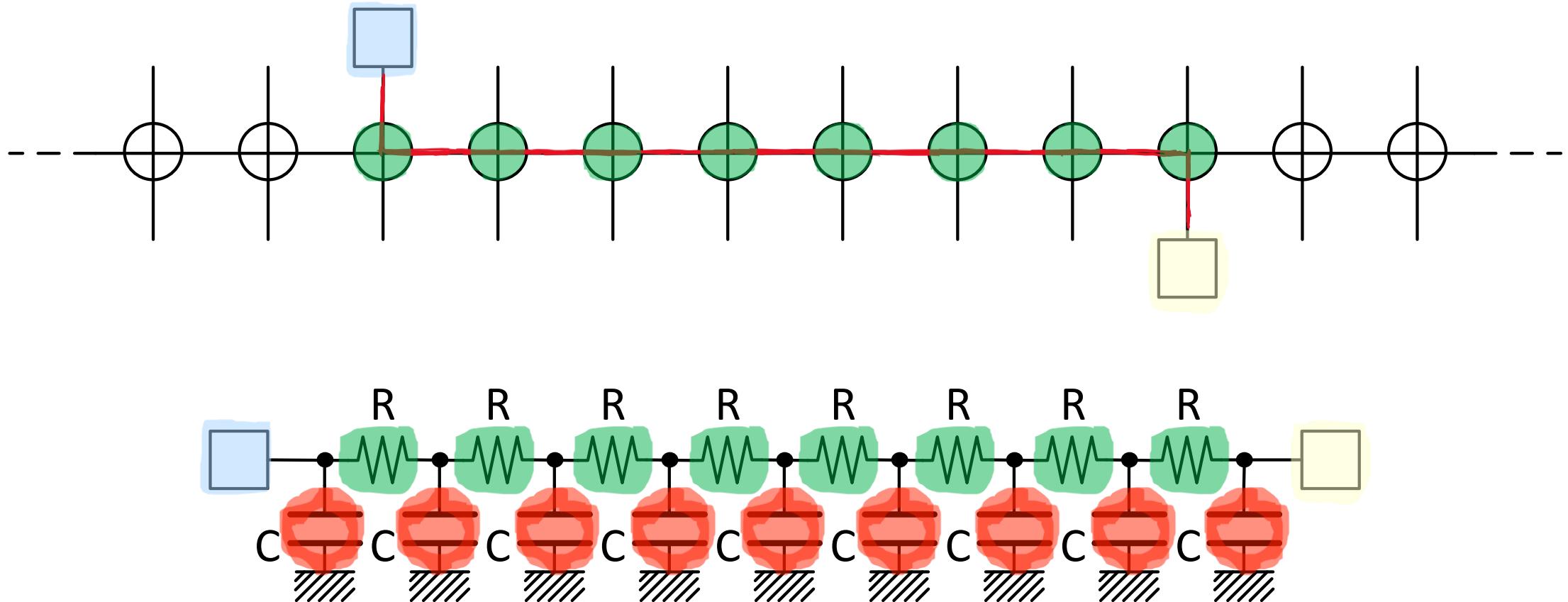
Wire segments $\rightarrow C$

↓
as capacitance C

Routing*

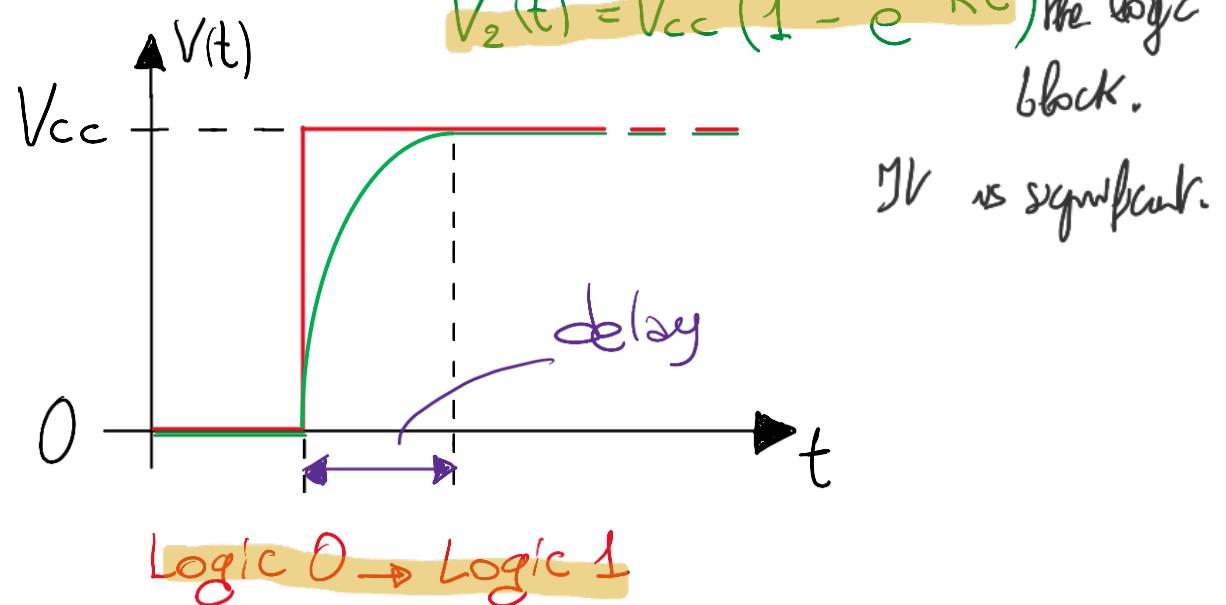
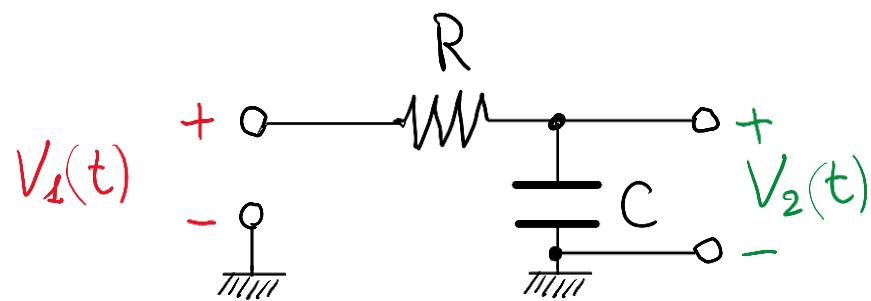


Routing*



Routing*

- Propagation delay of routing \propto (number of RC cells)² [Proven] Long path can introduce significant delay that can be even longer than the logic block.
- Each RC cell introduces a delay



Routing*

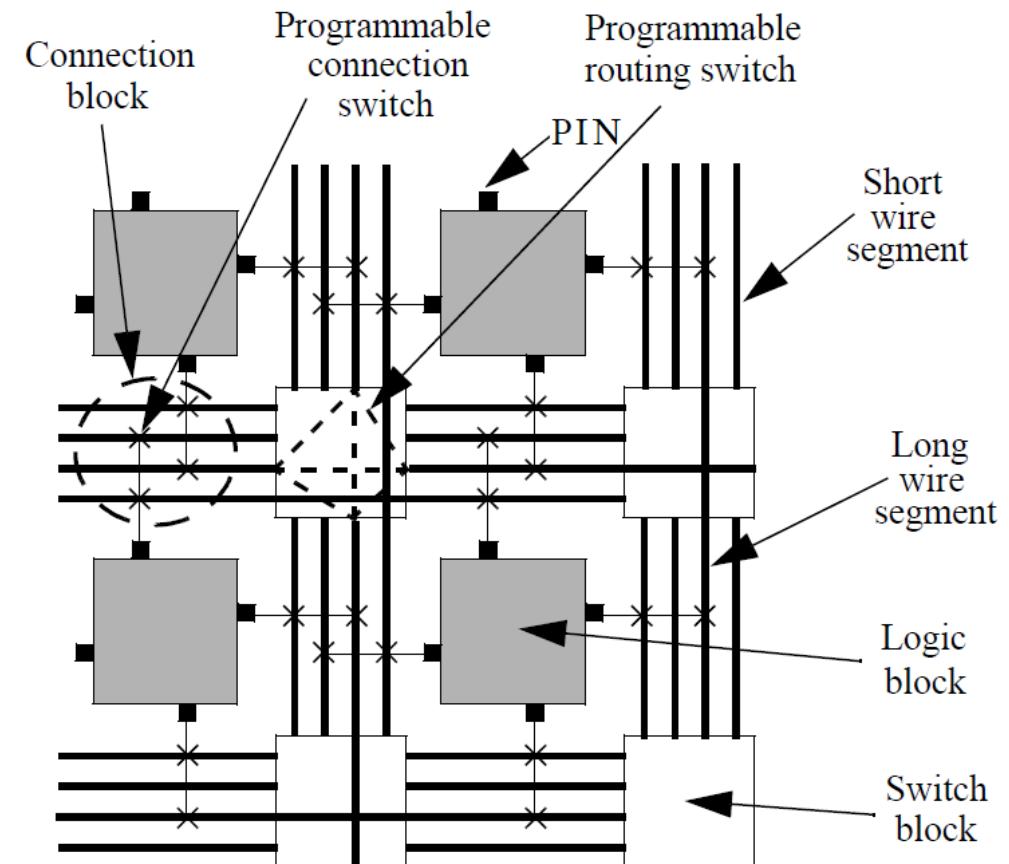
- Segmented routing in FPGA includes

- Switching matrices**

*Lots of logic blocks
that can be
connected together.*

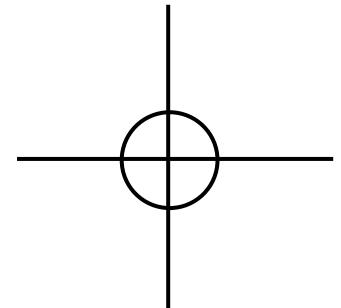
- Local and global routing wires**

*Long wires reach globally all logic blocks, while short
wire segments don't.*



Routing*

- How programmable connections are realized?



Routing*

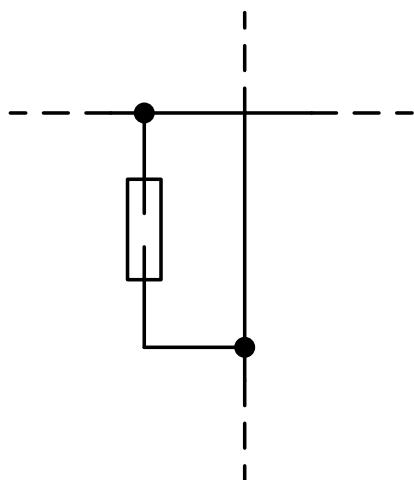
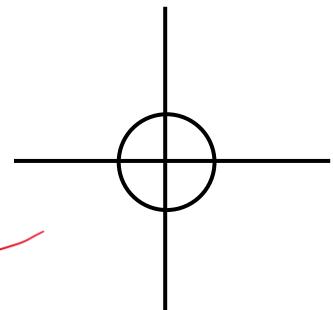
- How programmable connections are realized?
 - Anti-fuse (OTP FPGA)

- Anti-fuse symbol



Routing*

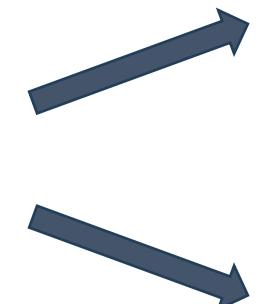
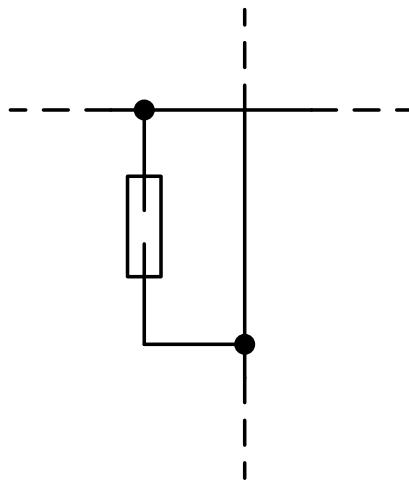
- How programmable connections are realized?
 - Anti-fuse (OTP FPGA)



Routing*

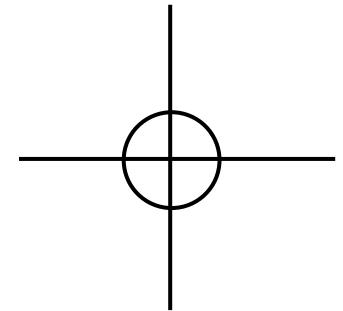
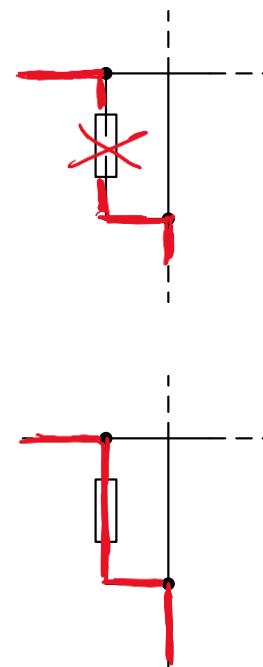
- How programmable connections are realized?

- Anti-fuse (OTP FPGA)



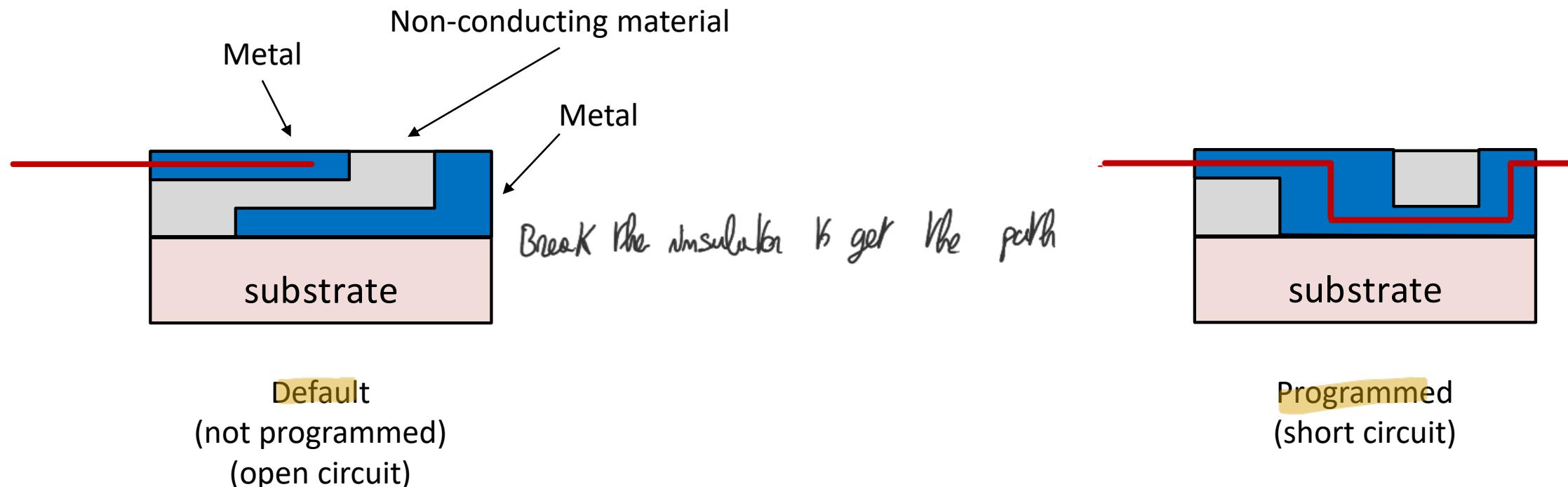
Default
(not programmed)
(open circuit)

Programmed
(short circuit)



Anti-fuse (in brief)*

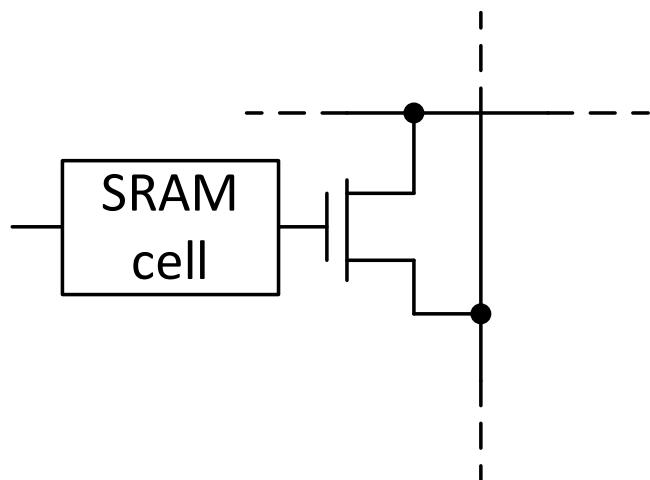
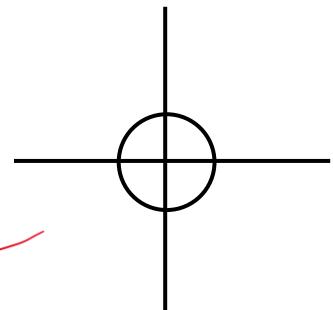
- Programming = application of a very high voltage (between two metal plates)



Routing*

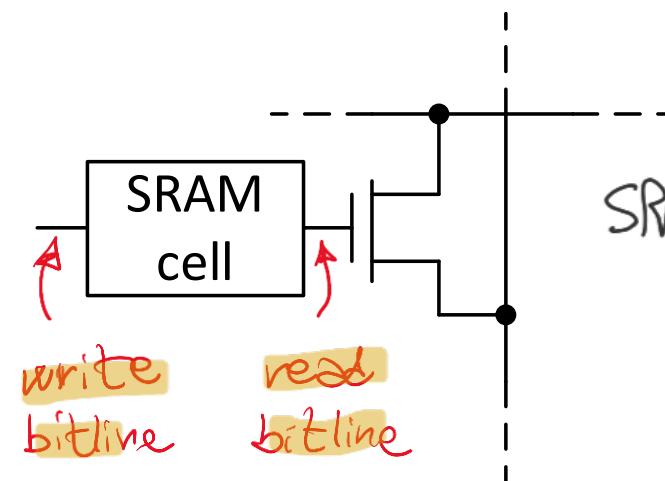
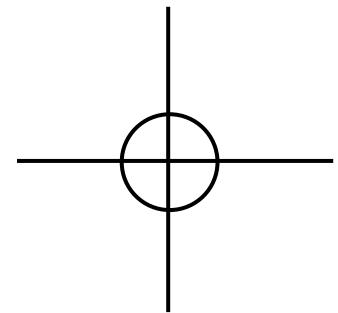
- How programmable connections are realized?

- SRAM (Reprogrammable FPGA)



Routing*

- How programmable connections are realized?
 - SRAM (Reprogrammable FPGA)

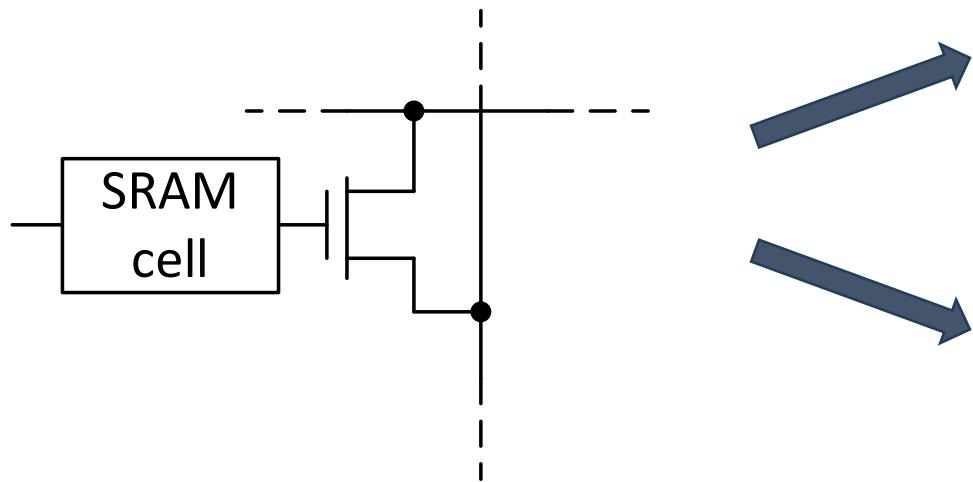


SRAM has 2 buffers. One is used to program, the second to drive the pass transistors.

Routing*

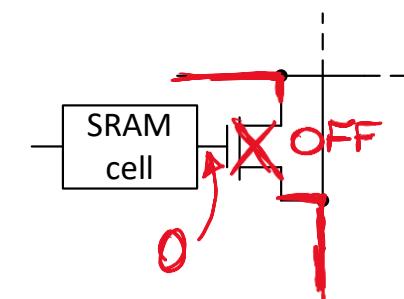
- How programmable connections are realized?

- SRAM (Reprogrammable FPGA)

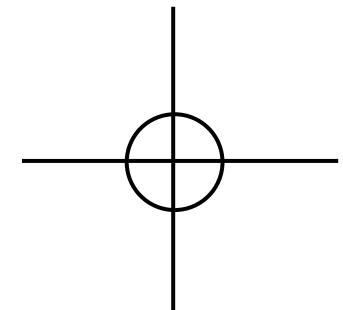
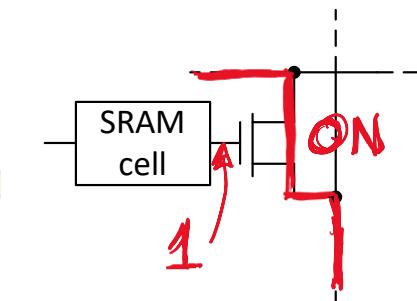


Not connected
(open circuit)

Connected
(short circuit)



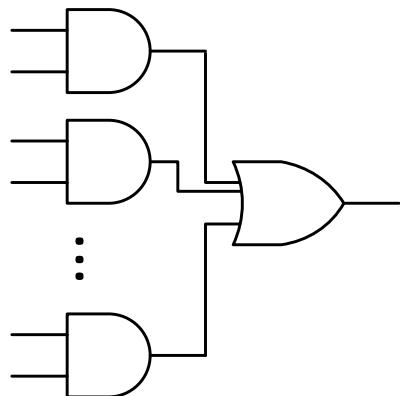
Behave little am open circuit.



Logic function block*

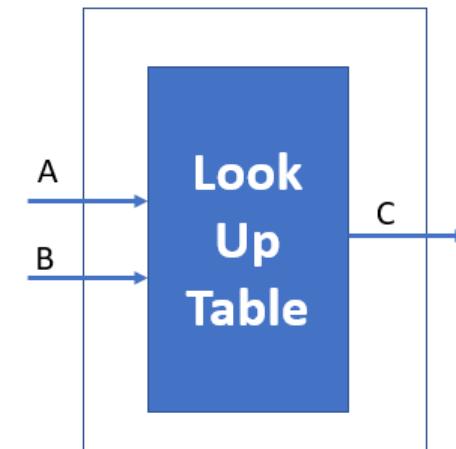
PAL

- Fixed
 - AND planes + OR



FPGA

- Configurable
 - Based on Look-Up Table (LUT)



Logic function block*

- LUT \approx programmable memory
 - Each memory location contains one of the output values of the function
 - The inputs of the function form the memory address
 - Each combination of the inputs addresses a different memory cell, so a different output value
- Let's see an example

Logic function block*

- Example
 - Logic function $Y = f(A_0, A_1)$ defined as it follows:

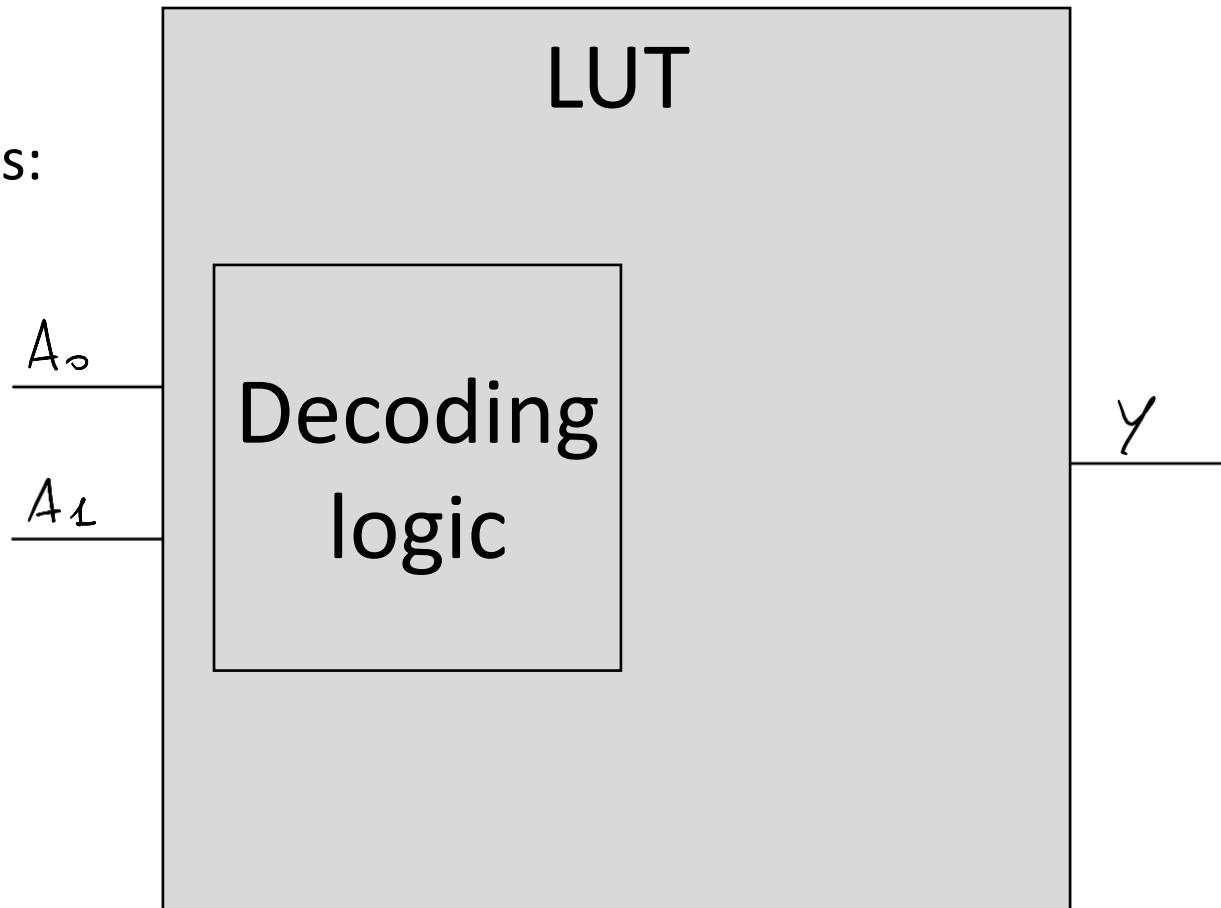
A_0	A_1	Y
0	0	1
0	1	0
1	0	1
1	1	1

- It is enough programming the LUT so that for each combination of the inputs A_0 and A_1 the memory output is the one corresponding to the definition above

Logic function block*

- Rationale
 - $Y = f(A_0, A_1)$ defined as it follows:

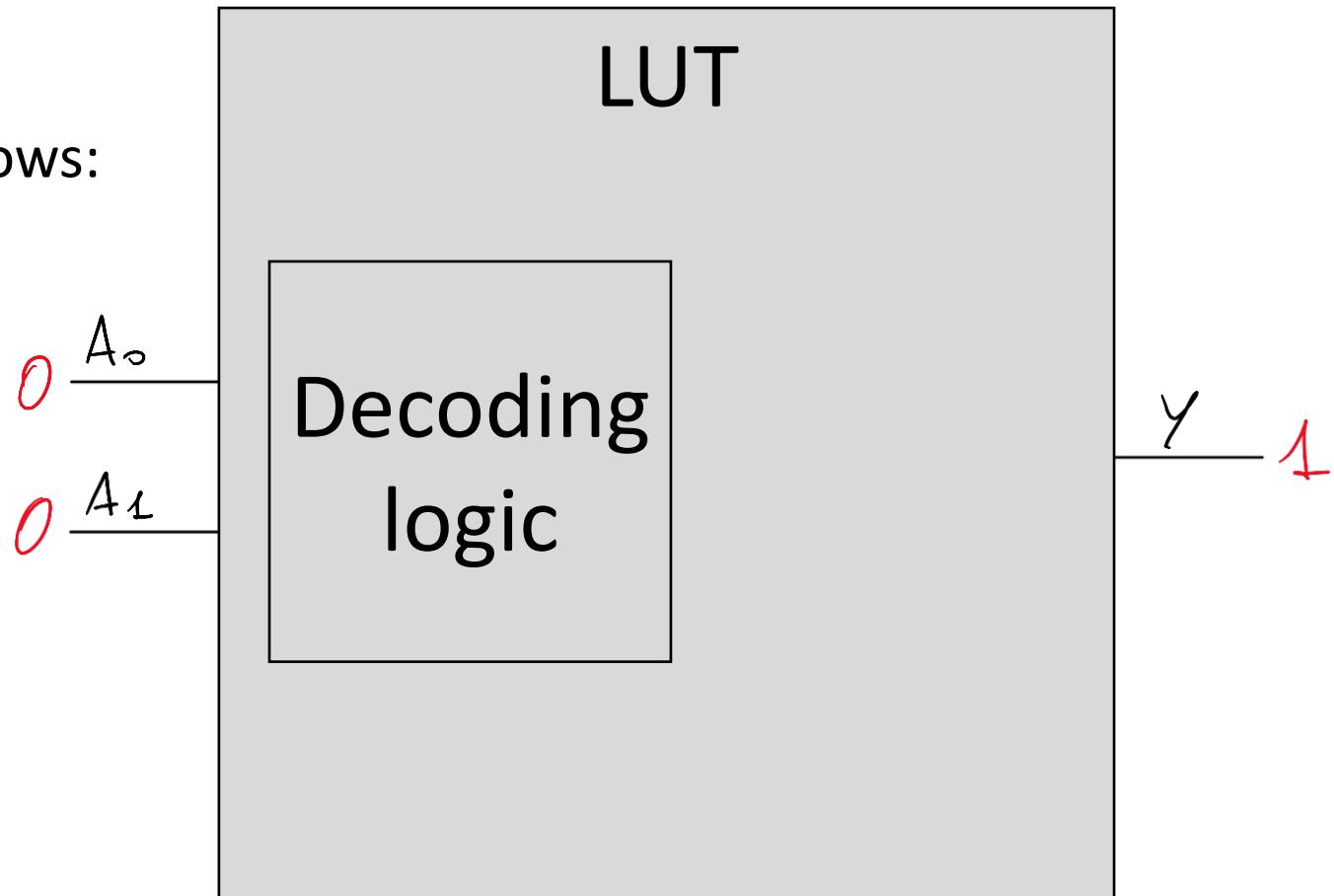
A_0	A_1	Y
0	0	1
0	1	0
1	0	1
1	1	1



Logic function block*

- Rationale
 - $Y = f(A_0, A_1)$ defined as it follows:

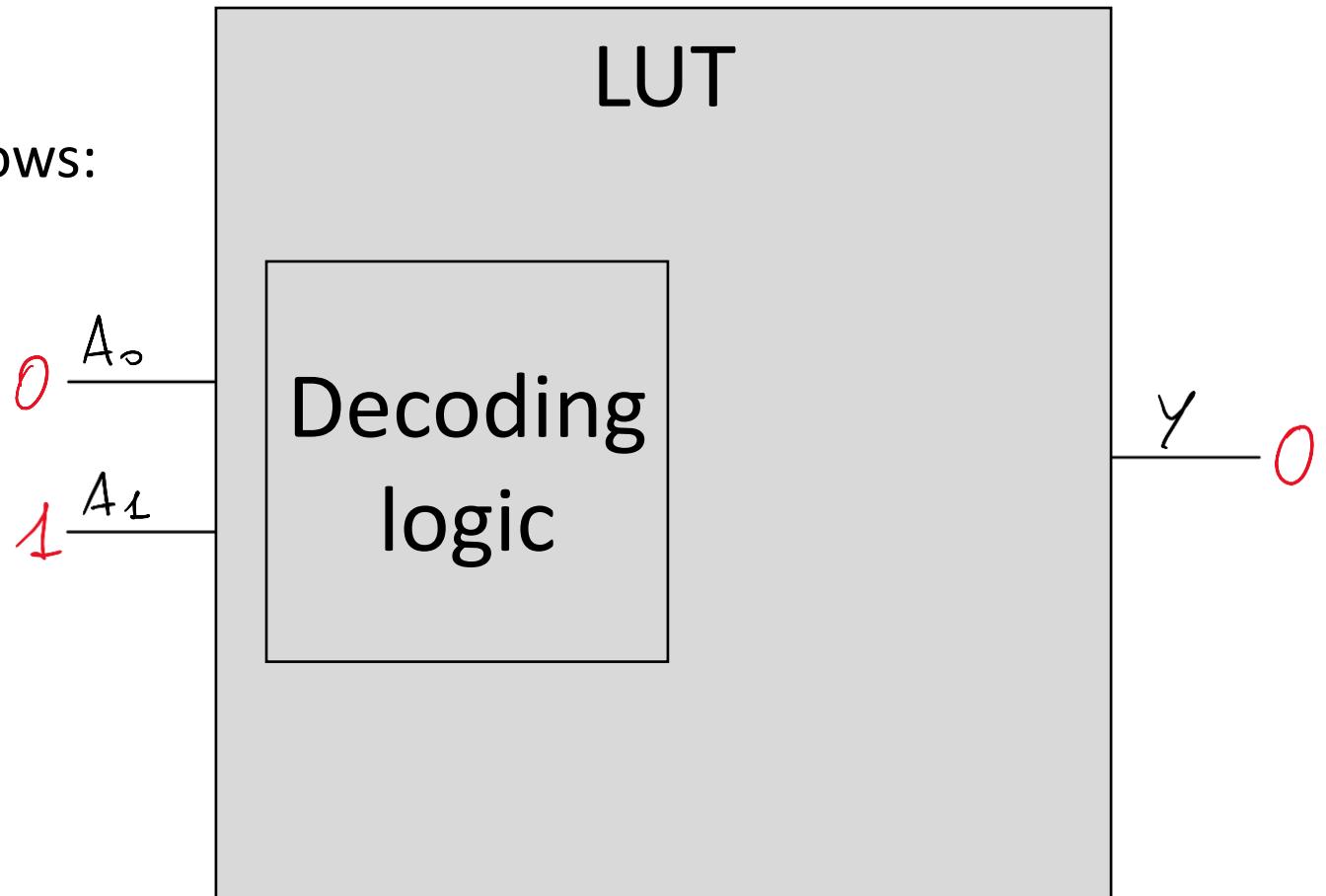
A_0	A_1	Y
0	0	1
0	1	0
1	0	1
1	1	1



Logic function block*

- Rationale
 - $Y = f(A_0, A_1)$ defined as it follows:

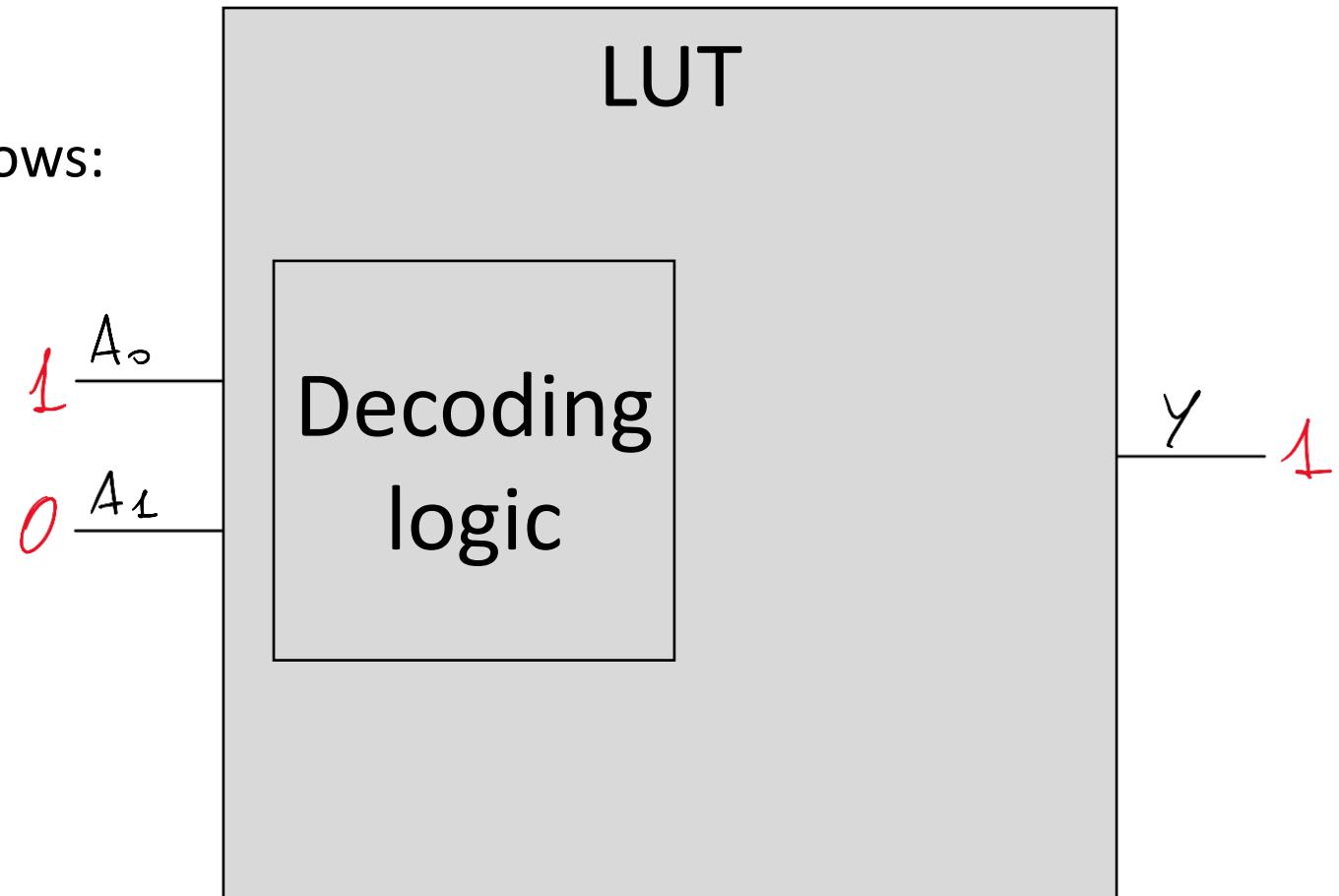
A_0	A_1	Y
0	0	1
0	1	0
1	0	1
1	1	1



Logic function block*

- Rationale
 - $Y = f(A_0, A_1)$ defined as it follows:

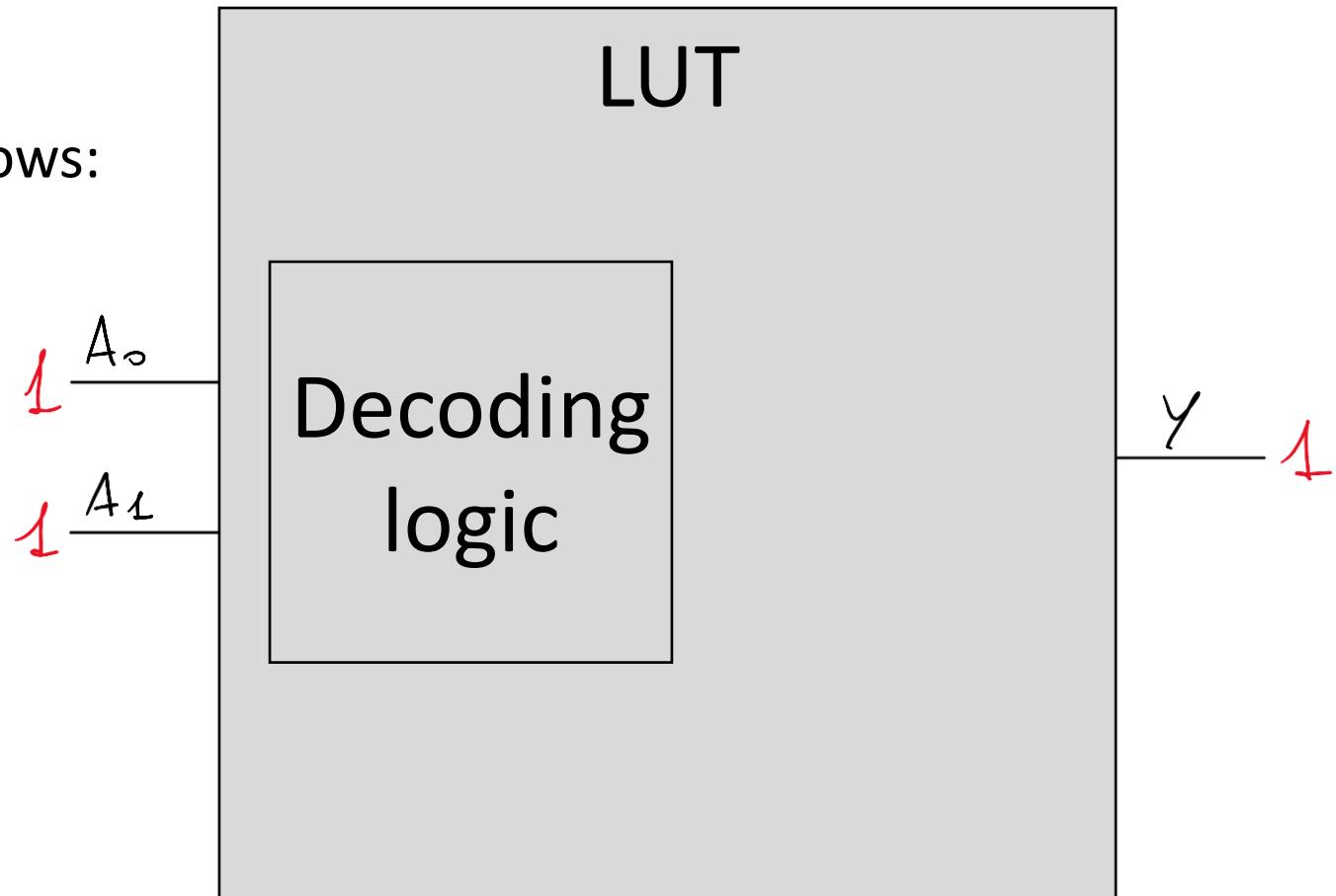
A_0	A_1	Y
0	0	1
0	1	0
1	0	1
1	1	1



Logic function block*

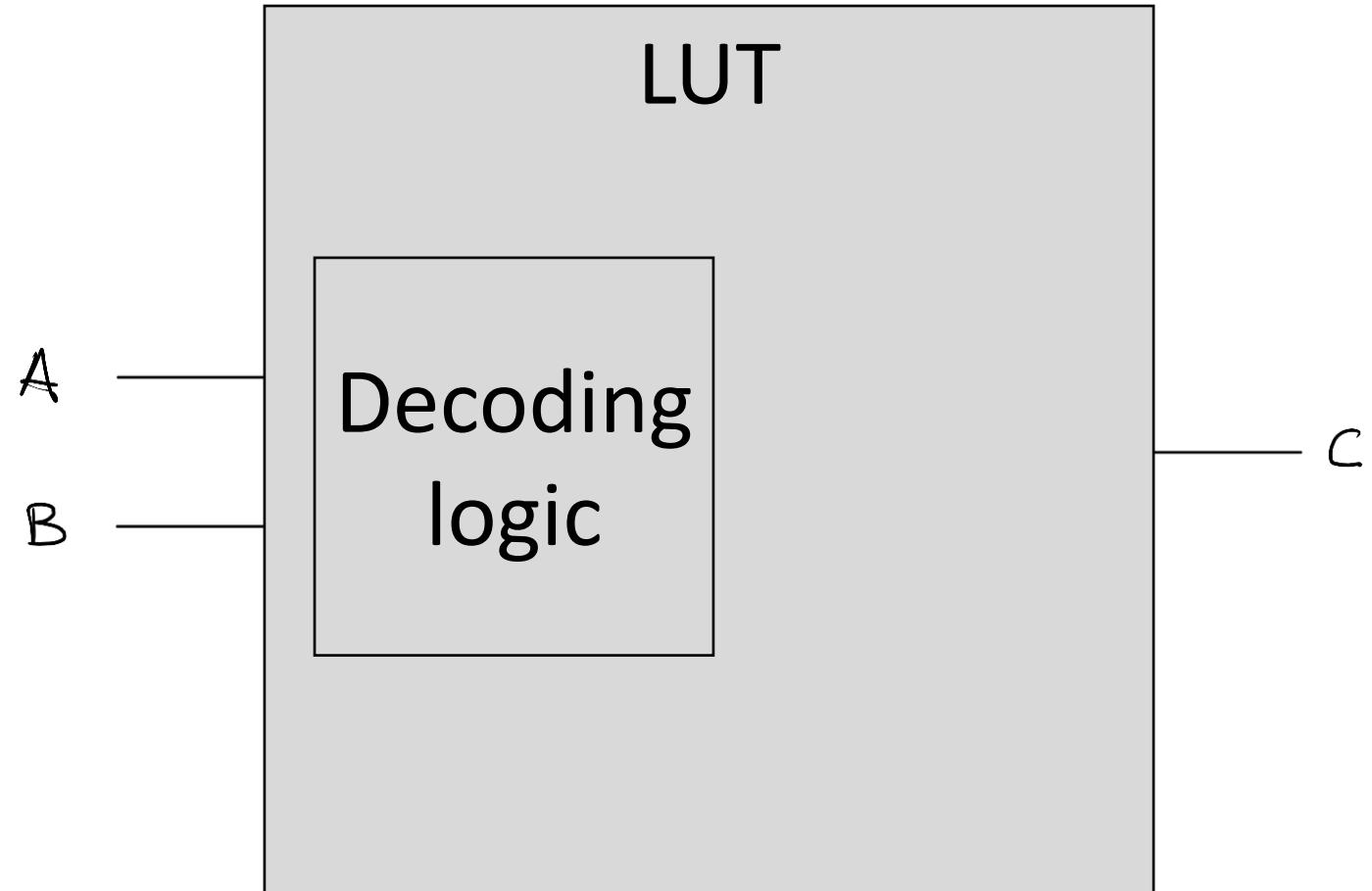
- Rationale
 - $Y = f(A_0, A_1)$ defined as it follows:

A_0	A_1	Y
0	0	1
0	1	0
1	0	1
1	1	1



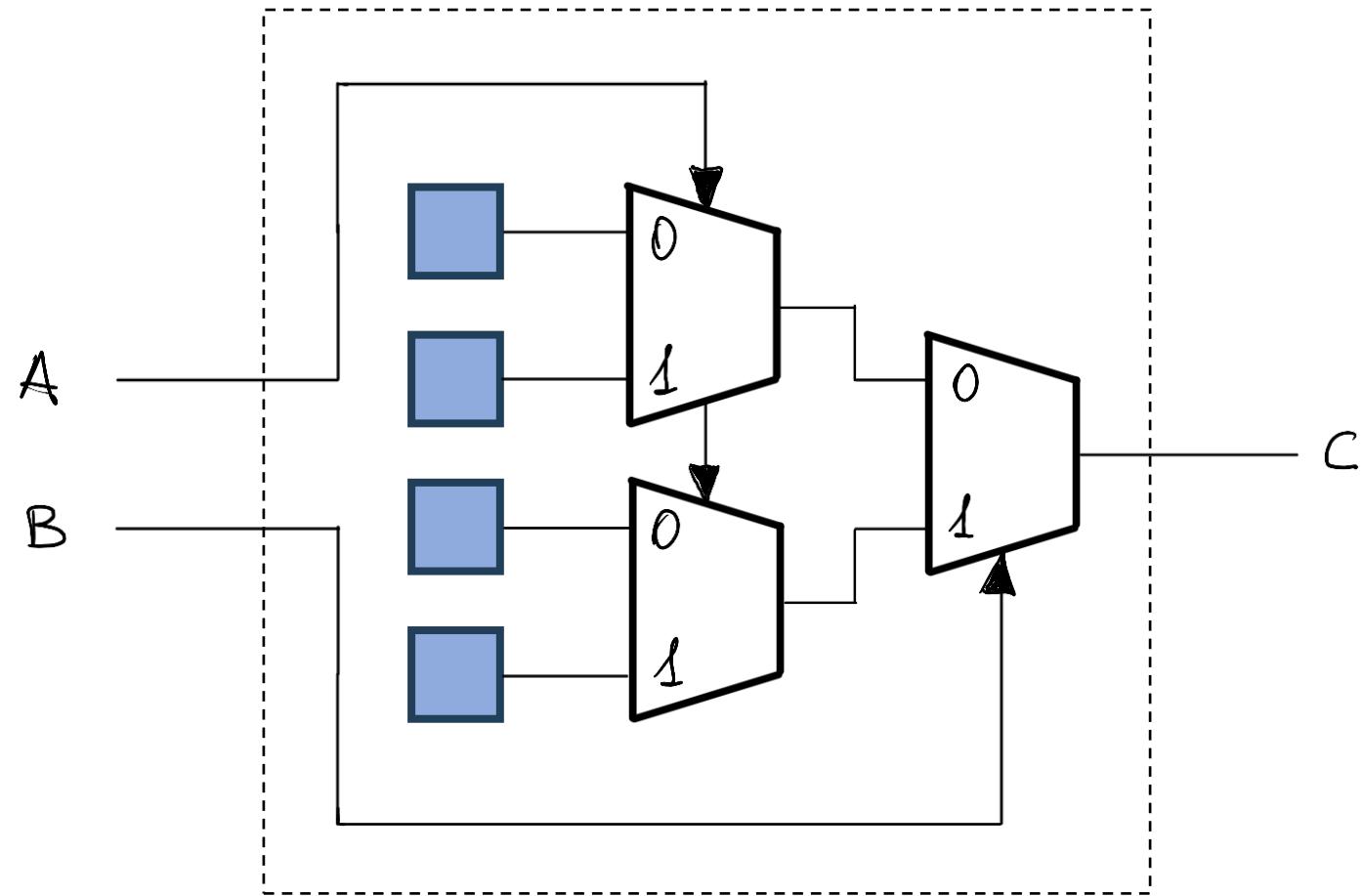
Logic function block*

How is a LUT made?



Logic function block*

- LUT architecture outline



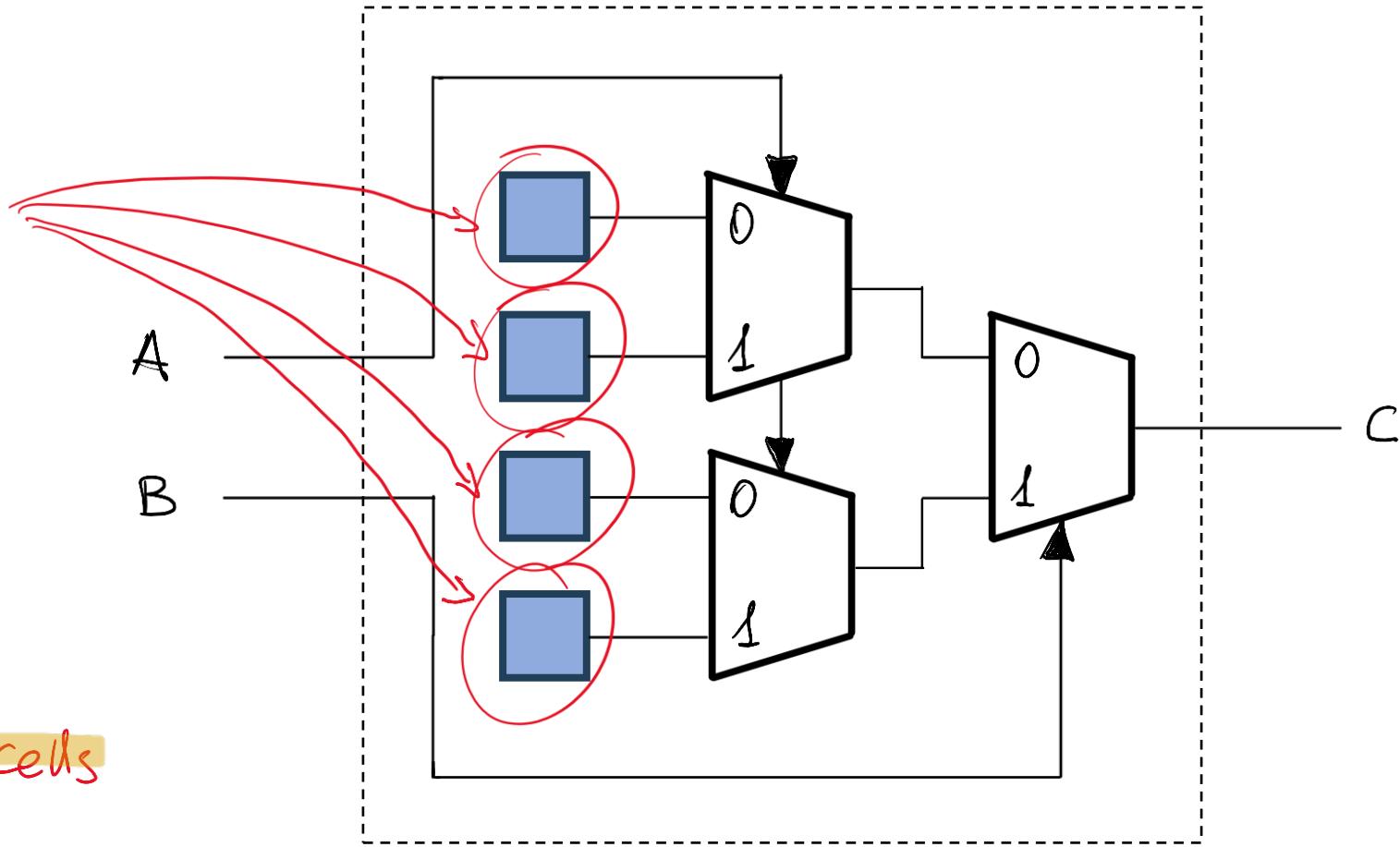
Logic function block*

- LUT architecture outline

Memory cells (1 bit)

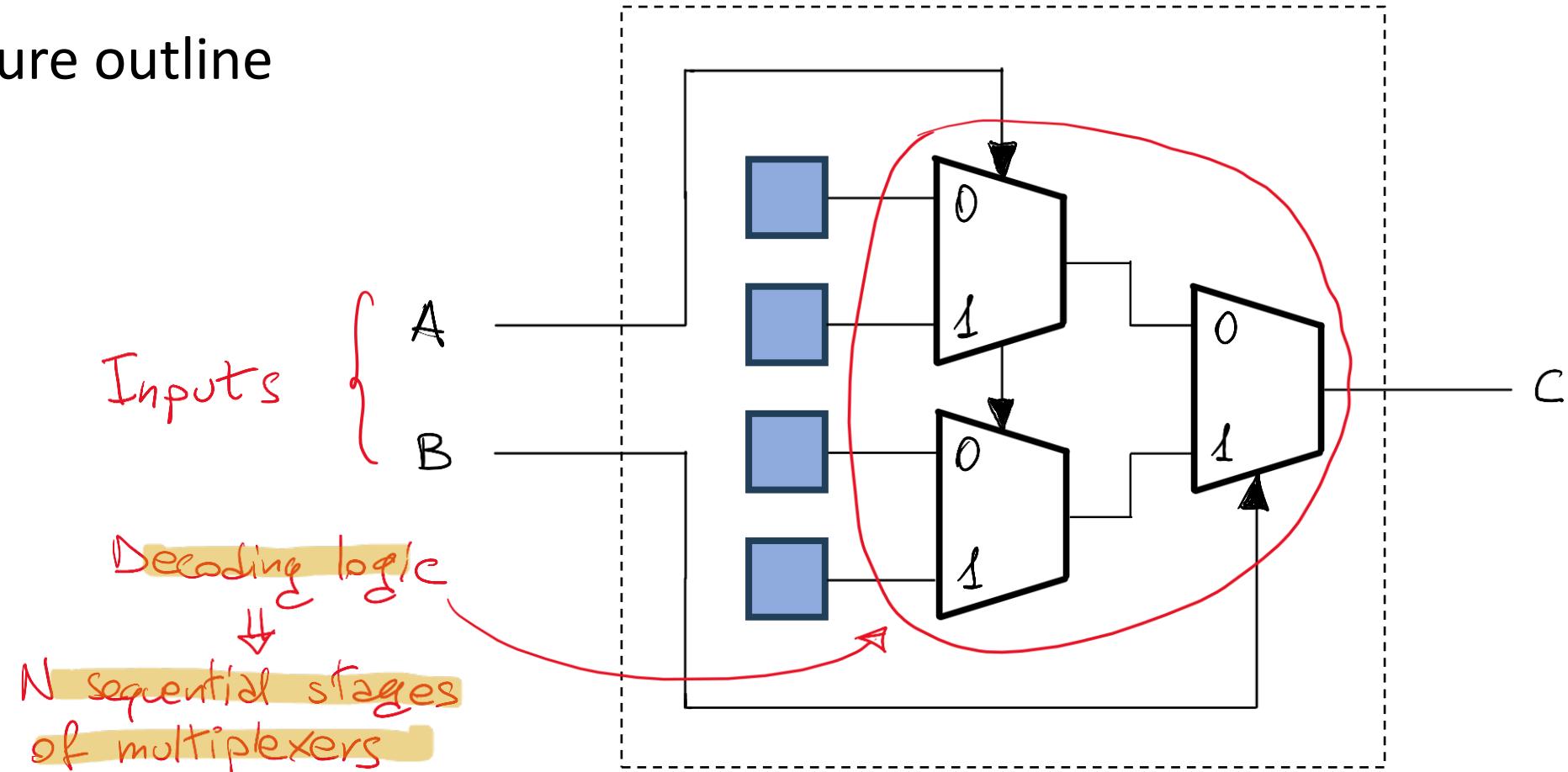
$N = \text{number of inputs}$

2^N (single-bit) memory cells



Logic function block*

- LUT architecture outline

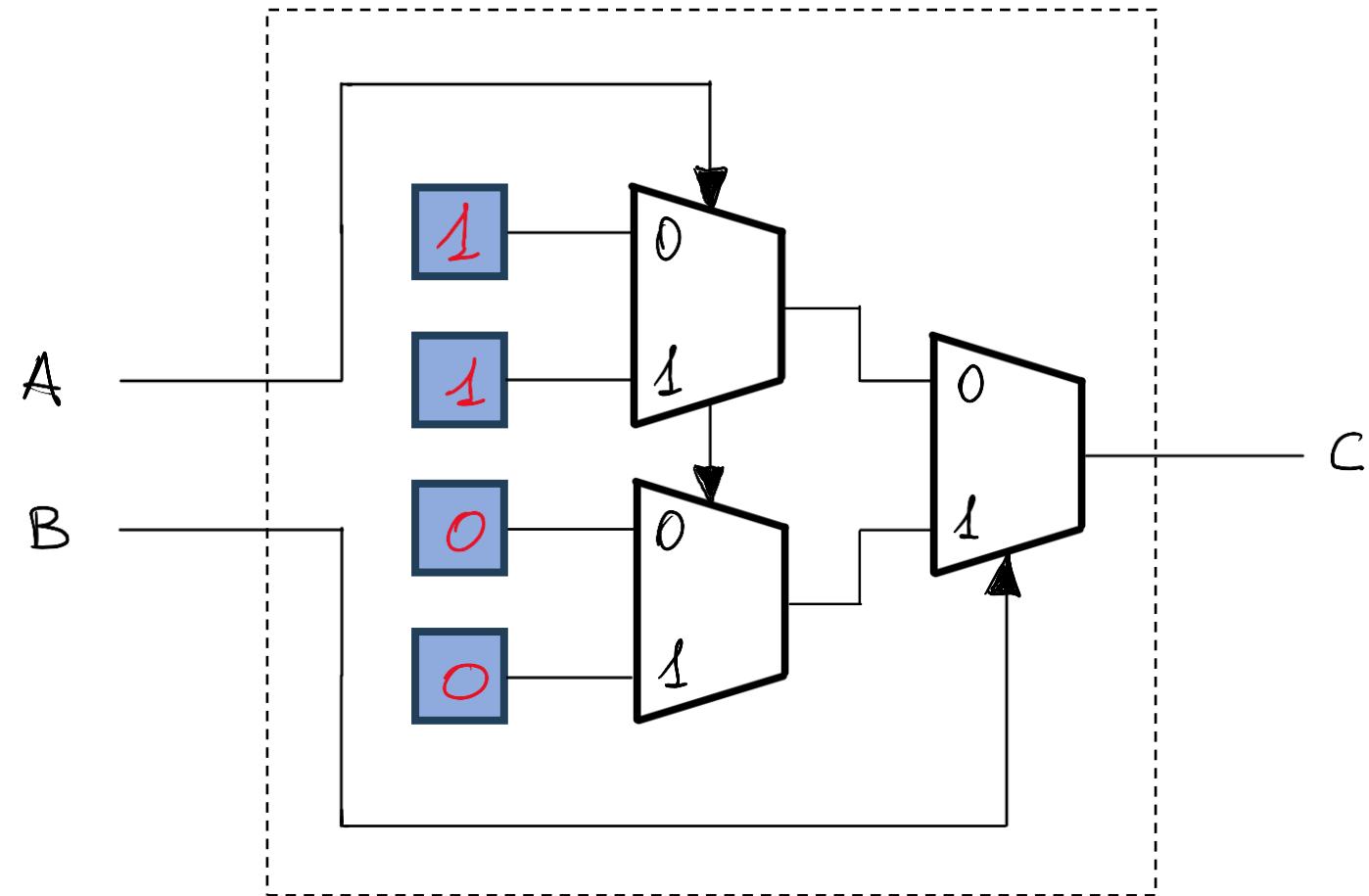


Logic function block*

- LUT architecture outline
 - Example: $Y_2 = f(A_0, A_1)$

A_0	A_1	Y_2
0	0	1
0	1	0
1	0	1
1	1	0

$A = A_0$
 $B = A_1$
 $C = Y_2$



Logic function block*

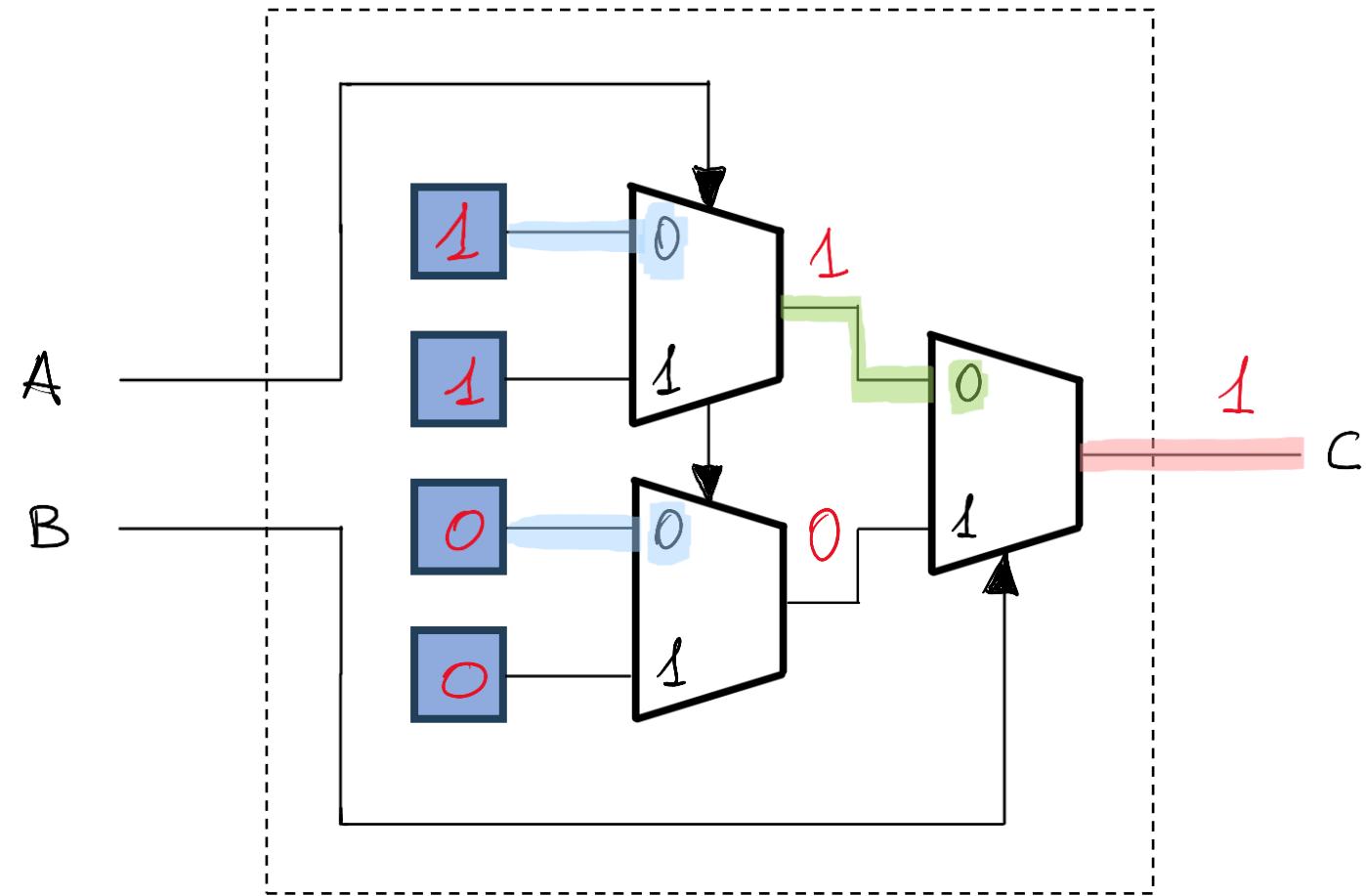
- LUT architecture outline
 - Example: $Y_2 = f(A_0, A_1)$

A_0	A_1	Y_2
0	0	1
0	1	0
1	0	1
1	1	0

$$A = A_0$$

$$B = A_1$$

$$C = Y_2$$



Logic function block*

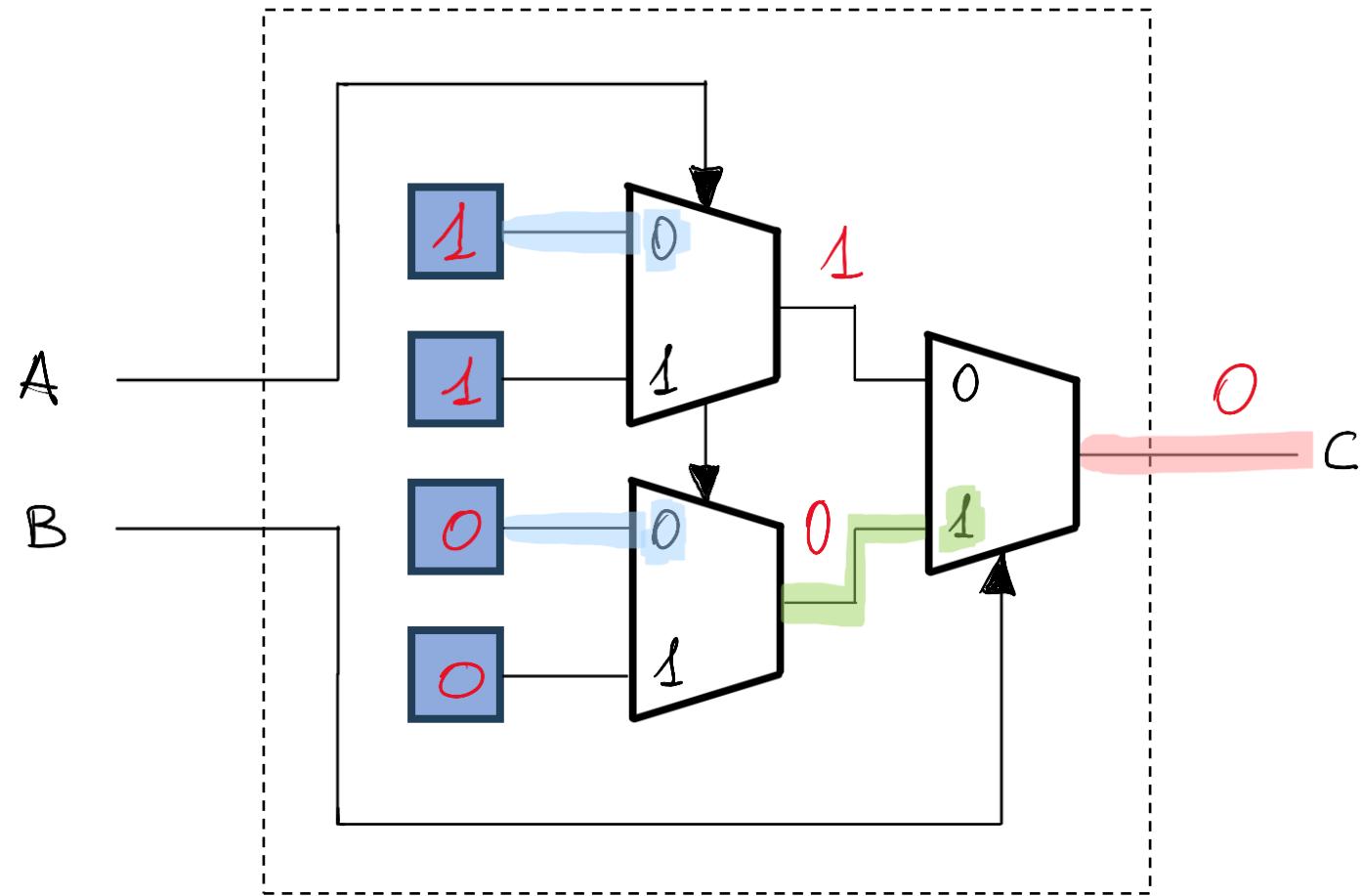
- LUT architecture outline
 - Example: $Y_2 = f(A_0, A_1)$

A_0	A_1	Y_2
0	0	1
0	1	0
1	0	1
1	1	0

$$A = A_0$$

$$B = A_1$$

$$C = Y_2$$



Logic function block*

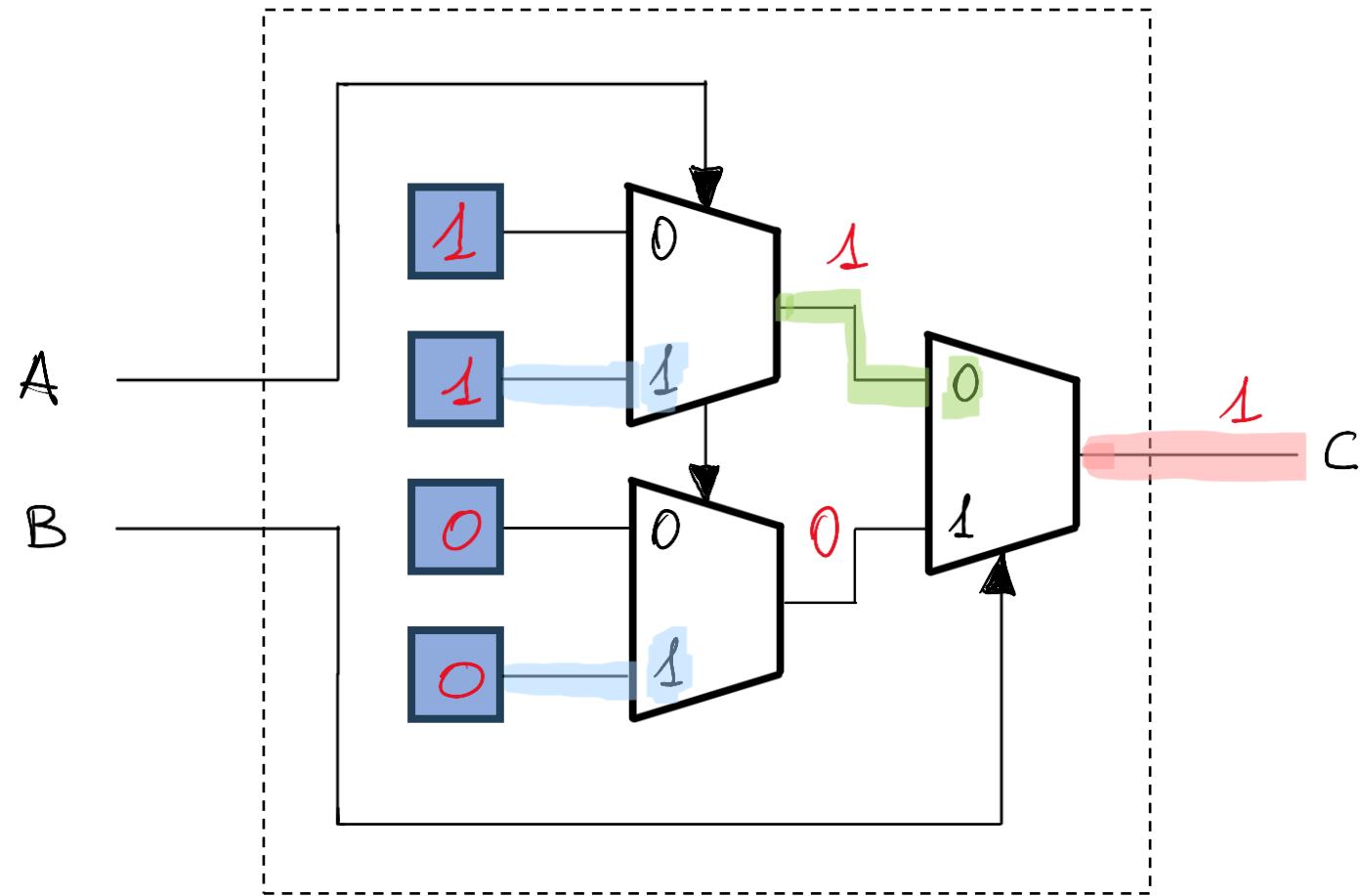
- LUT architecture outline
 - Example: $Y_2 = f(A_0, A_1)$

A_0	A_1	Y_2
0	0	1
0	1	0
1	0	1
1	1	0

$$A = A_0$$

$$B = A_1$$

$$C = Y_2$$



Logic function block*

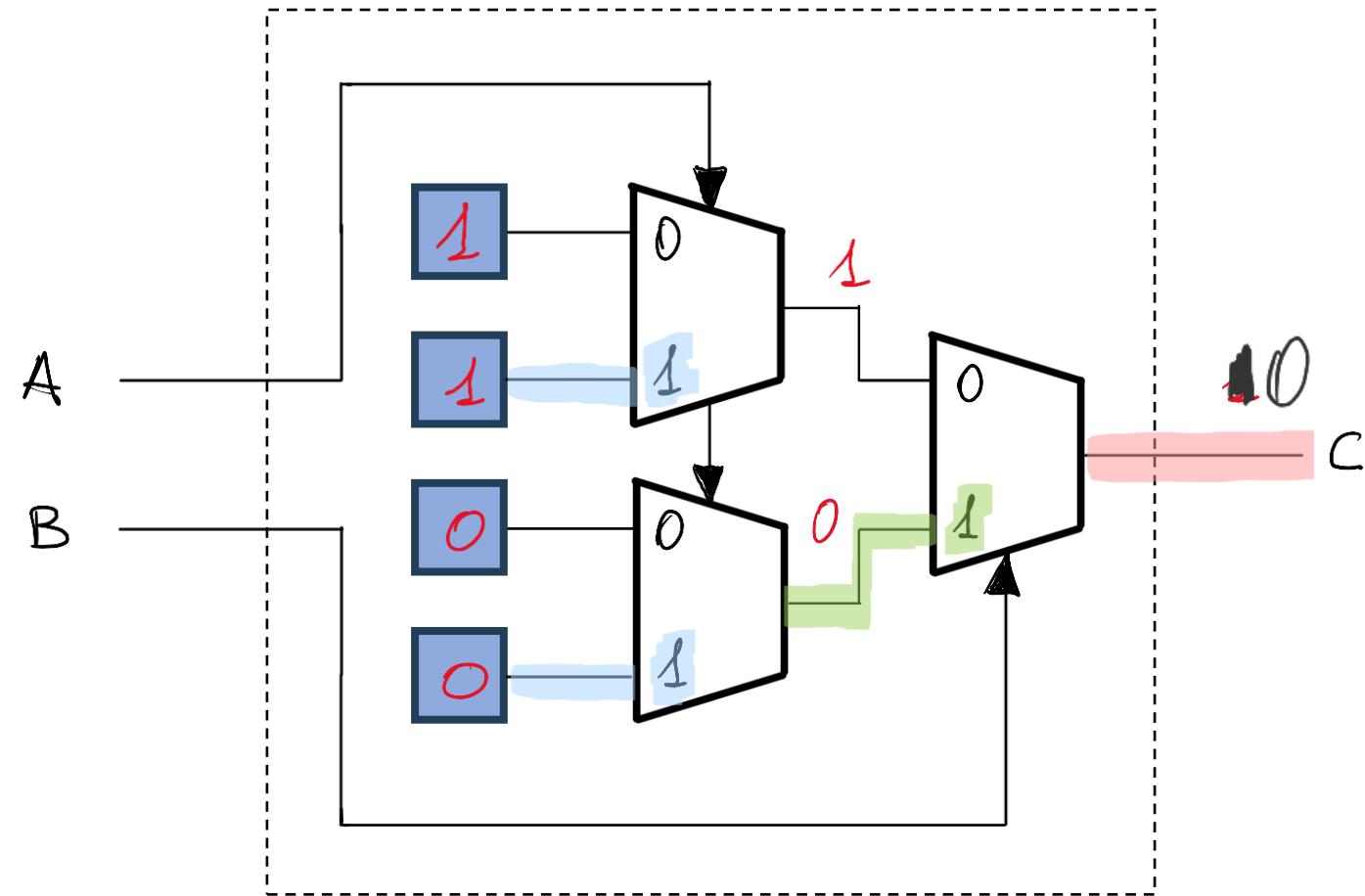
- LUT architecture outline
 - Example: $Y_2 = f(A_0, A_1)$

A_0	A_1	Y_2
0	0	1
0	1	0
1	0	1
1	1	0

$$A = A_0$$

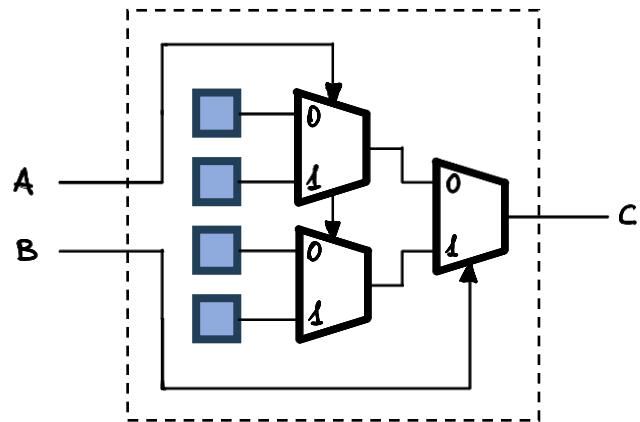
$$B = A_1$$

$$C = Y_2$$



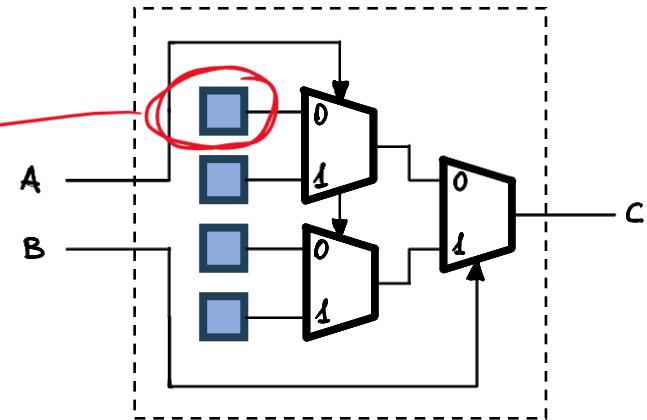
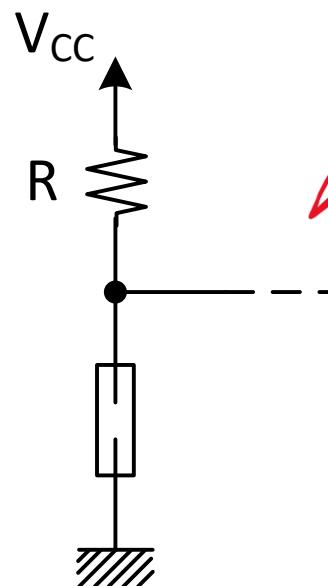
Logic function block*

- How are memory cells realized?



Logic function block*

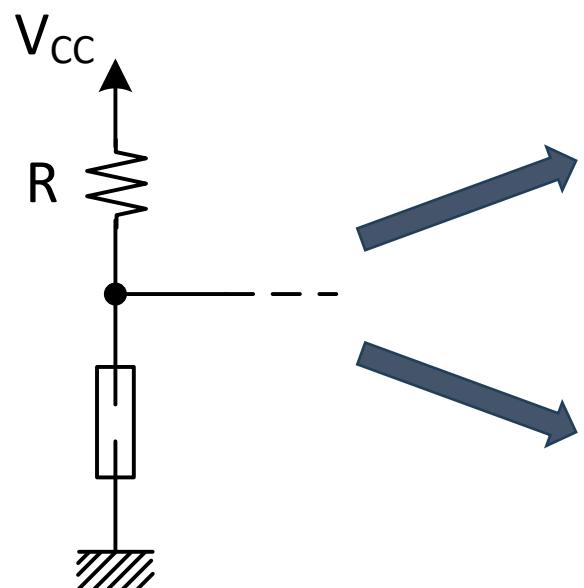
- How are memory cells realized?
 - **Anti-fuse (OTP FPGA)**



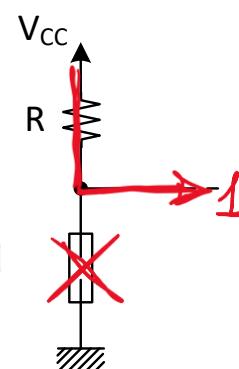
Logic function block*

- How are memory cells realized?

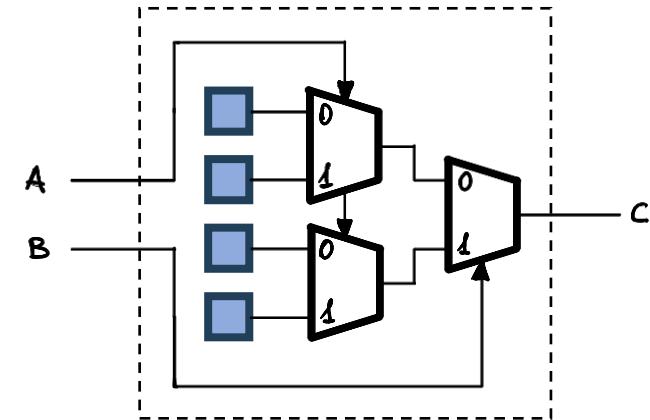
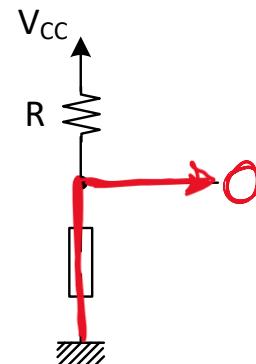
- Anti-fuse (OTP FPGA)



Default
(not programmed)

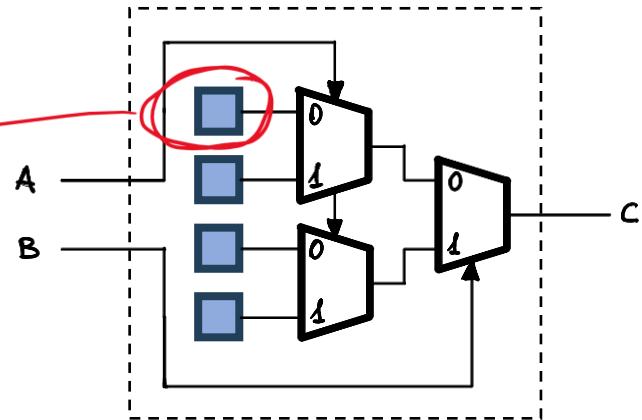
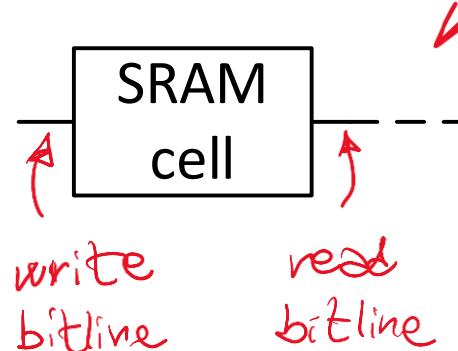


Programmed
(short circuit)



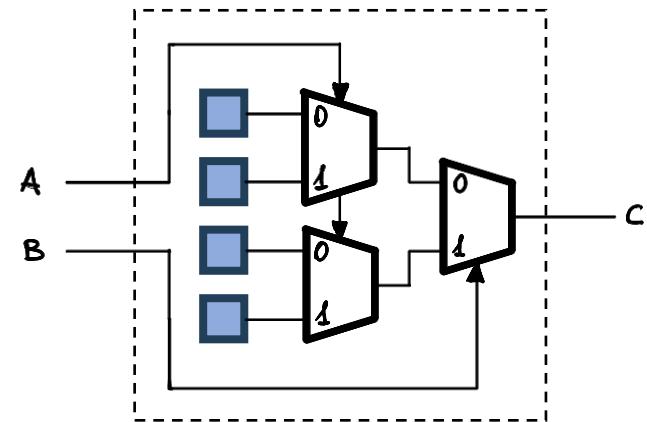
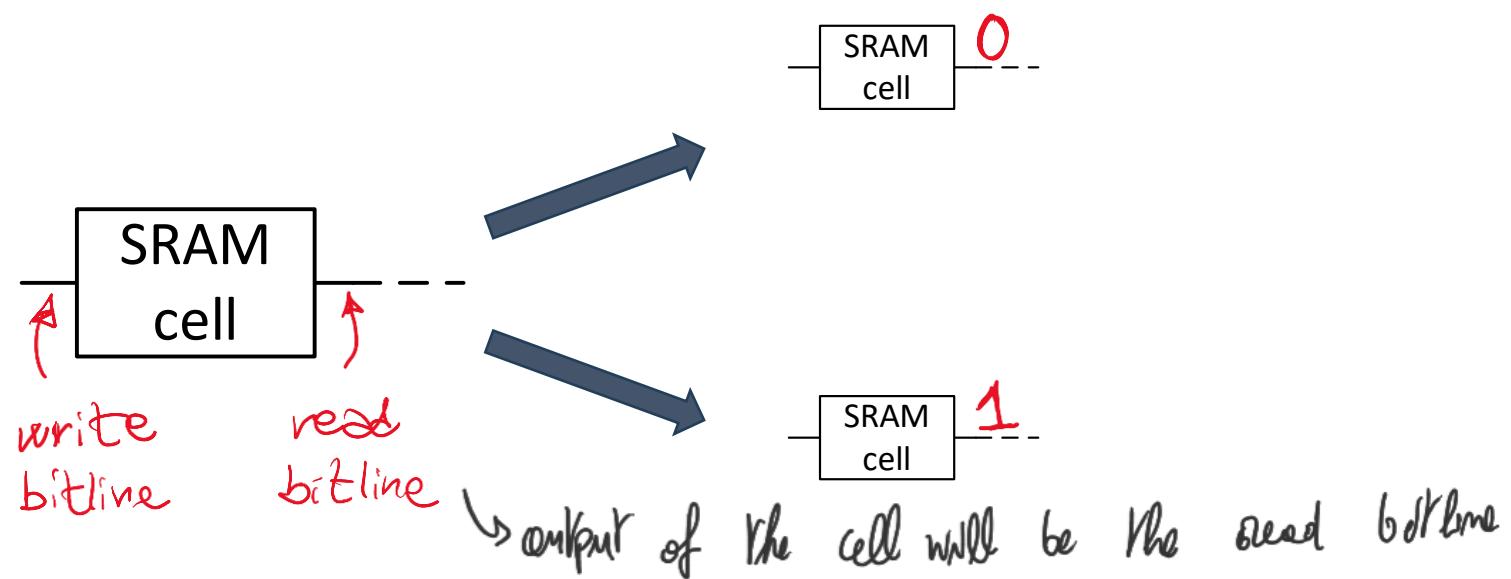
Logic function block*

- How are memory cells realized?
 - SRAM (Reprogrammable FPGA)



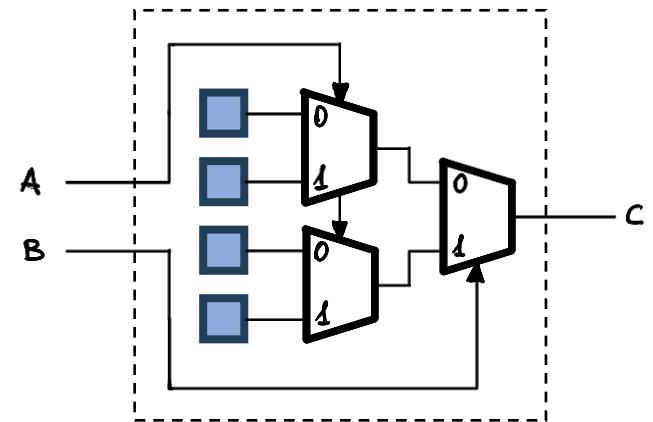
Logic function block*

- How are memory cells realized?
 - **SRAM** (Reprogrammable FPGA)



Logic function block*

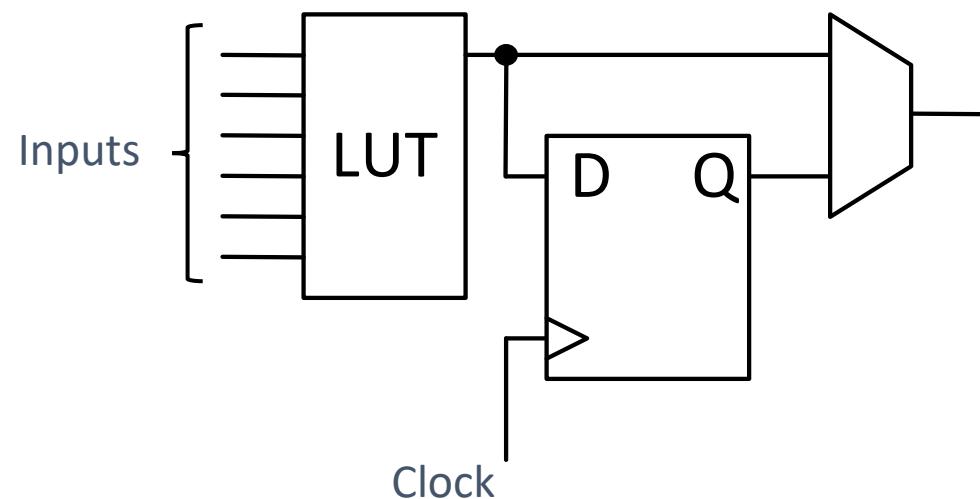
- The one in this example can be called 2-LUT
 - 2 inputs
- Earliest FPGAs were equipped with 3-LUTs
- Modern FPGAs are equipped with 6-LUTs



$\uparrow 2^6$ memory cells

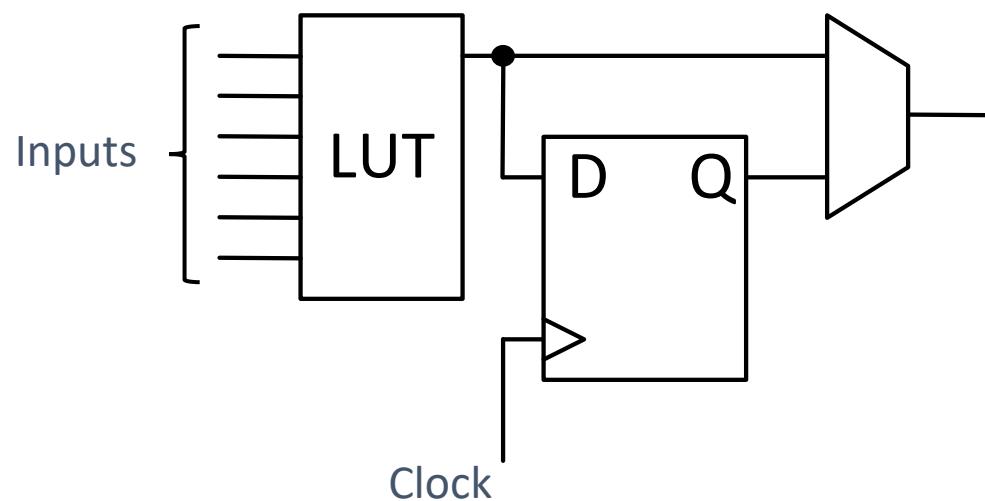
Logic function block*

- **Architecture outline**

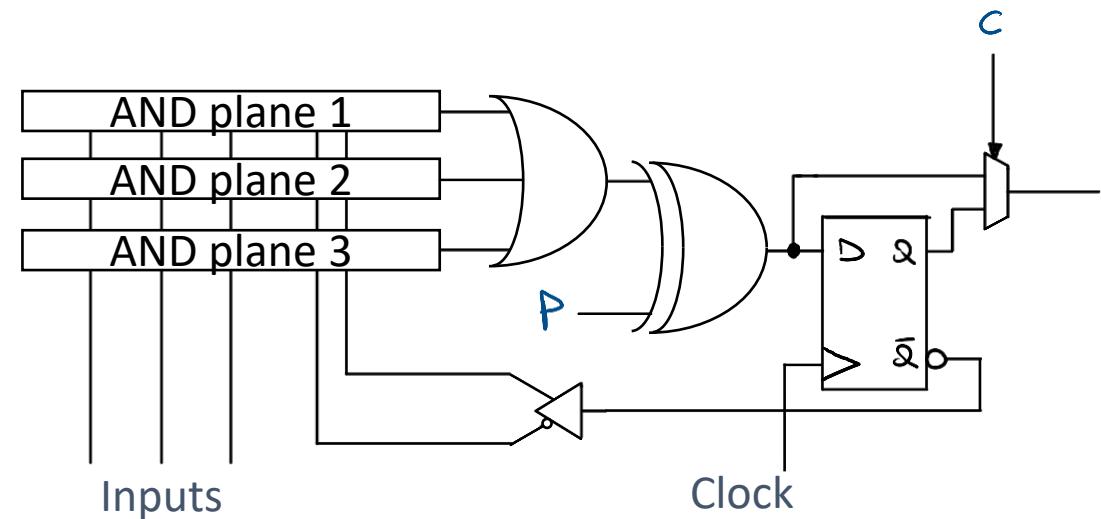


Logic function block*

FPGA



Main difference is in how PAL
we realize the logic function.



CLB*

- The logic function block also embeds other resources to improve flexibility and performance (dedicated to more specific functions)
 - The overall block is called Configurable Logic Block (**CLB**)
 - It assumes different names based on the FPGA vendor
 - Altera/Intel
 - ALM = Adaptive Logic Module
 - Xilinx/AMD
 - CLB = Configurable Logic Block
 - Slice (in older FPGAs)
 - Microsemi/Microchip
 - LE = Logic Elements

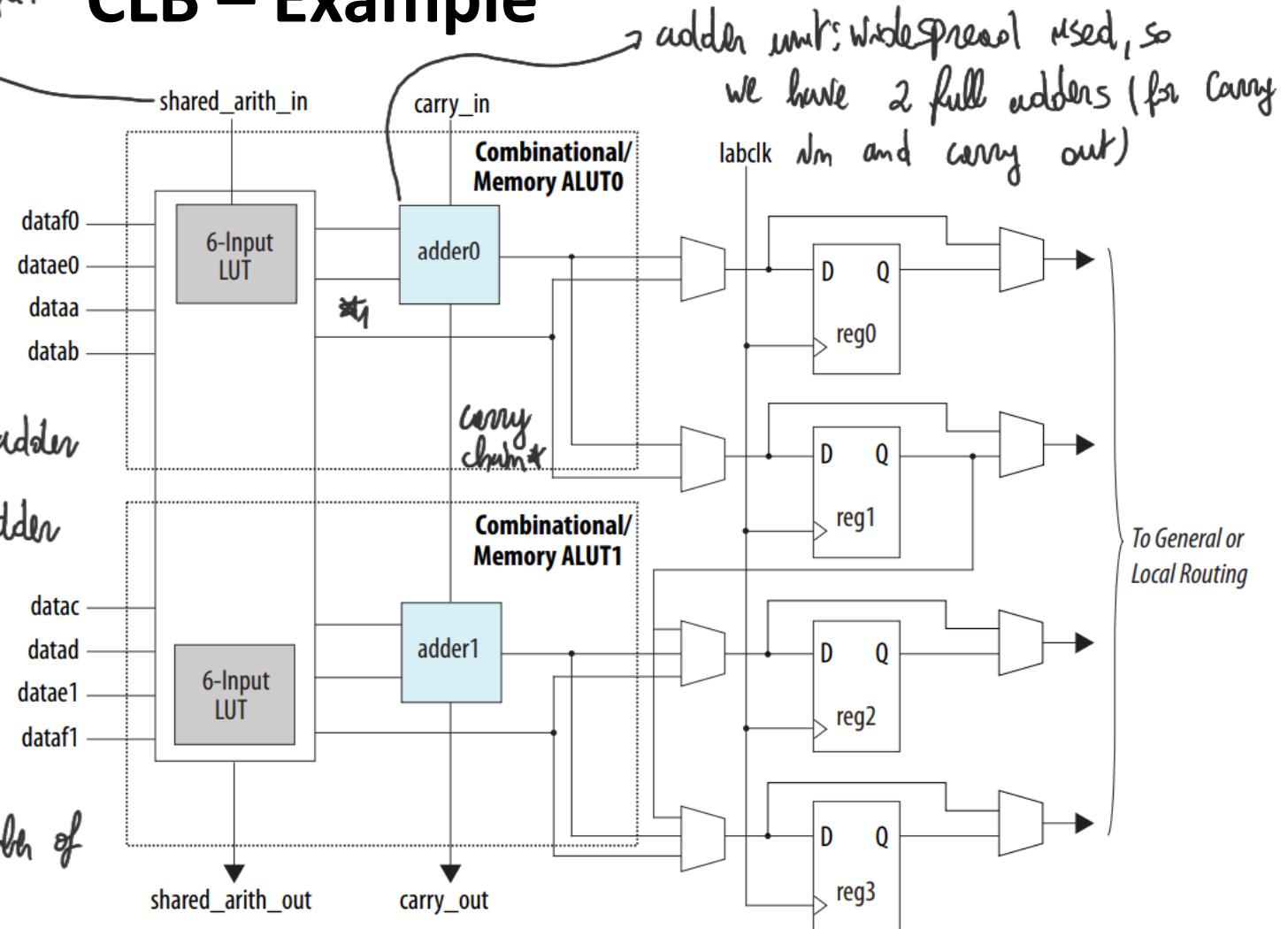
- ALM
 - Altera/Intel
 - Cyclone V FPGA

*₁ 2 possibilities, either
look up or use adder
* Present in ripple carry adder

Reach logic blocks \Rightarrow local wiring
can reach a higher number of
logic elements.

more complex
functions,
ex multipliers
and dividers

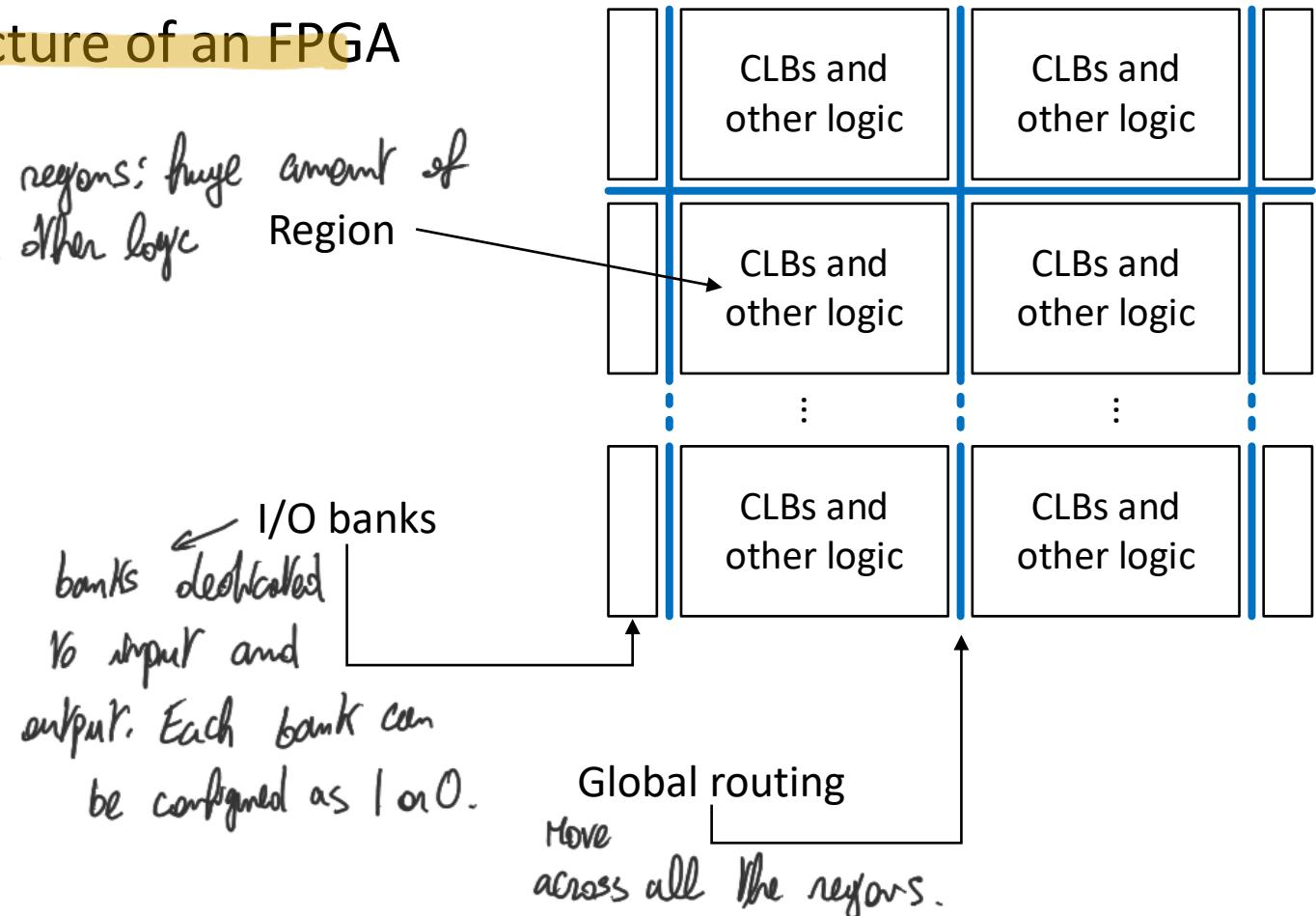
CLB – Example*



Architecture outline*

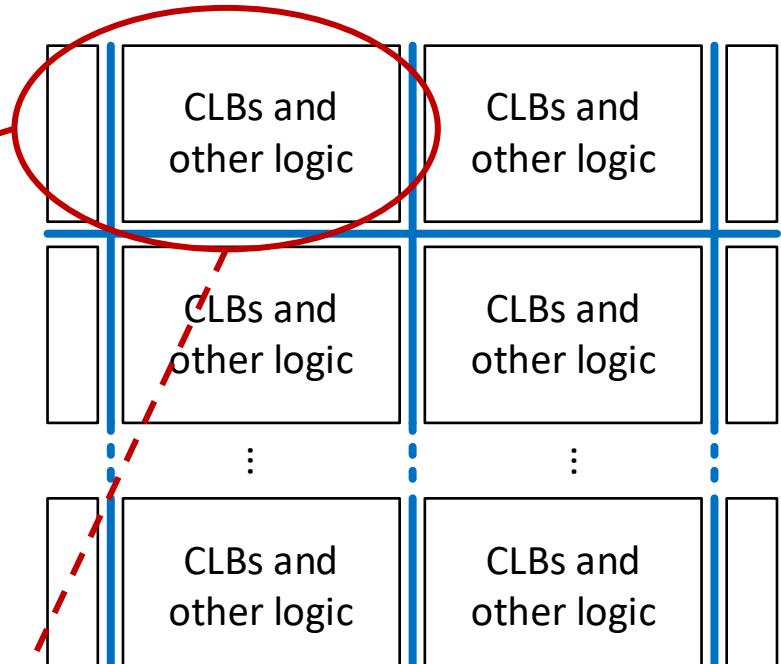
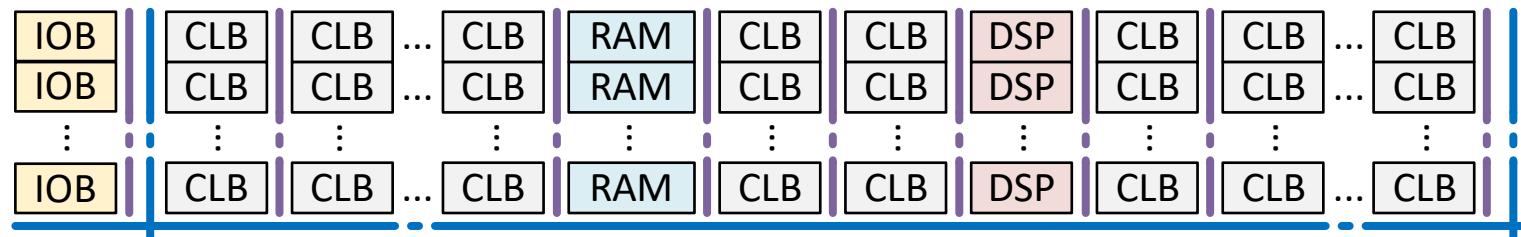
- Finally, we can see the overall architecture of an FPGA
 - Organized in regions

Separated in regions; huge amount of CLBs and other logic elements.



Architecture outline*

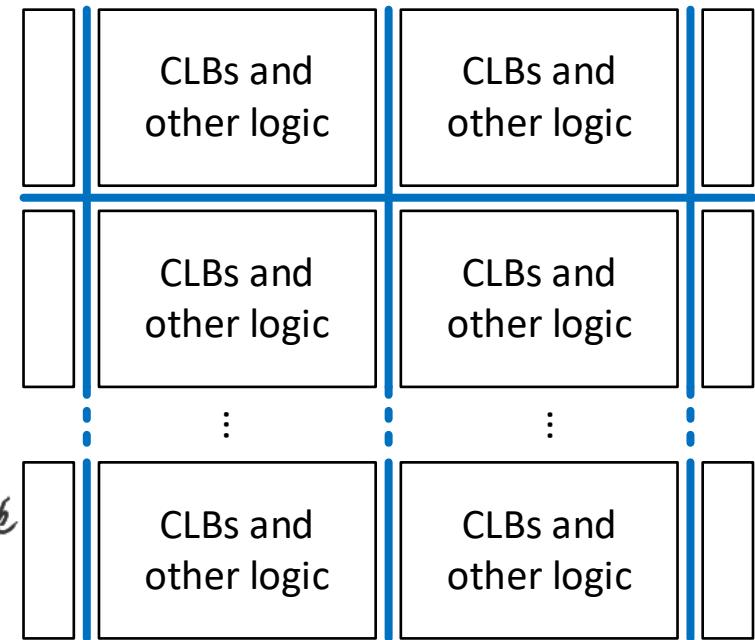
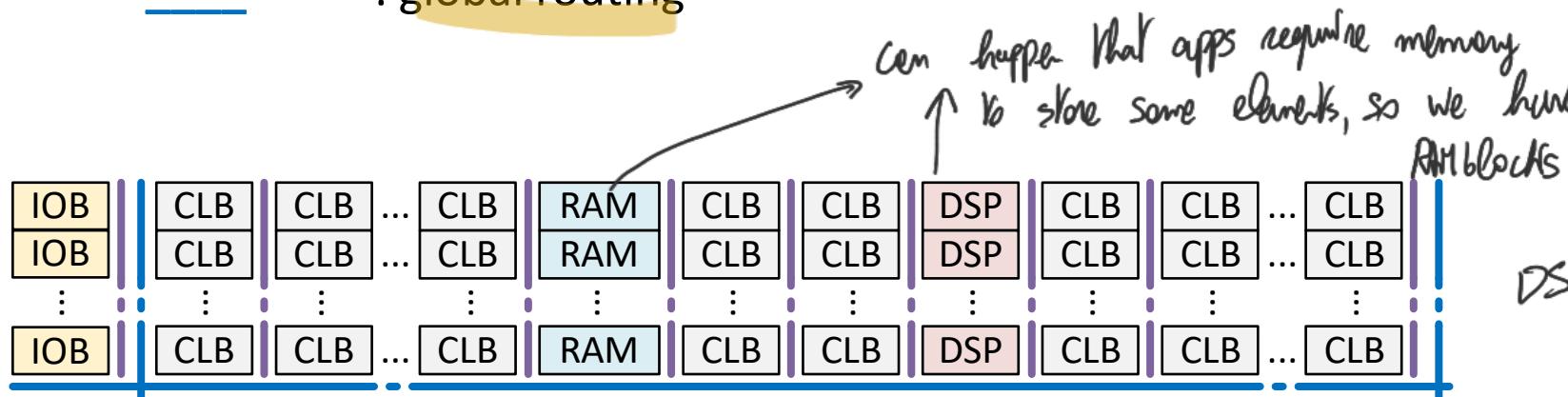
- Finally, we can see the overall architecture of an FPGA



Architecture outline*

- Finally, we can see the overall architecture of an FPGA

- IOB** : Input/Output Block
- RAM** : RAM memory block
- DSP** : Digital Signal Processor
- : local routing
- : global routing



DSP: blocks dedicated to complex arithmetic functions, used e.g. in digital signals.
(for ex. complex multipliers)

Architecture outline*

- Latest FPGAs can embed also a processor

- Processor typically called **HPS = Hard Processing Systems**

- Usually **ARM processor**

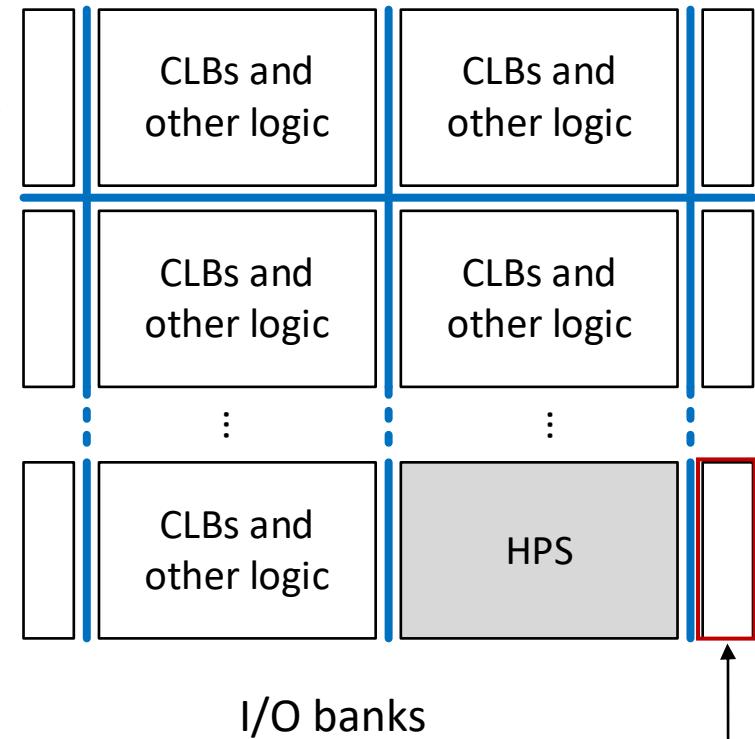
- The overall device typically called **FPGA SoC**

- SoC = System-on-Chip**

→ Corresponding physical hardware circuit for

that:

Latest FPGA are integrating ARM processor to increase flexibility to integrate SW.
 Hard Processor: Since with CLB you can integrate a Processor, in that case you are talking about **Soft Processing Systems**.



I/O banks
reserved for HPS

Anti-fuse vs. SRAM*

Anti-Fuse FPGA

- OTP (One-Time Programmable)
- Permanent programming More oriented to ~~buy free applications~~
 - Do not need to be re-programmed after power-on
- Robust against disturbances
 - E.g.: radiations → suitable for space applications
- It is not possible to test before the release
 - ↳ all bugs found, unfixable

SRAM FPGA

- Can be re-programmed
 - ↳ usually you have SD card + programmer for configuration
- Volatile programming
 - Need to be re-programmed after power-on
- Low resistance against disturbances
 - Electromagnetic disturb may change a SRAM bit, modifying the configuration and the functionality
- Possibility to test before the release

Are FPGA really used in modern applications?*

- Yes! For different reasons and/or a combination of them
 - Flexibility
 - Time-to-market
 - Costs

*They can have lower performances than actual integrated circuits.
But!*

Are FPGA really used in modern applications?*

- Yes! For different reasons and/or a combination of them

- **Flexibility**
- Time-to-market
- Costs

- Possibility to re-program (SRAM FPGA) without changing/substituting the HW
 - Bug fixing
 - System update

Are FPGA really used in modern applications?*

- Yes! For different reasons and/or a combination of them

- Flexibility
- **Time-to-market** When you have specification and when you deploy it.
- Costs

- Shorter time from being conceived until being available for sale

– 3–6 Months vs. 3–5 years (typical time-to-market for ICs)
Much shorter for release of integrated circuit

IC-making functions from scratch (or almost)

Are FPGA really used in modern applications?*

- Yes! For different reasons and/or a combination of them

- Flexibility
- Time-to-market
- Costs

Do you remember of the photolithographic process to produce ICs?

The masks employed the process are strongly expansive

– US \$ 1–10 Million !!! → once you realise the mask you can use it for different circuits.

It is a non-recurring cost (C_{NRE}) that is split over the volume of production (N)

– $\frac{C_{NRE}}{N}$ → lower the volume N , higher the cost per unit → non recurring expense, very high but split over # of chip to realise.

Realising IC or FPGA depends on the volume of production.

Cost of a mask for photolithographic process is expensive.

* Once you buy device, you have final chip.

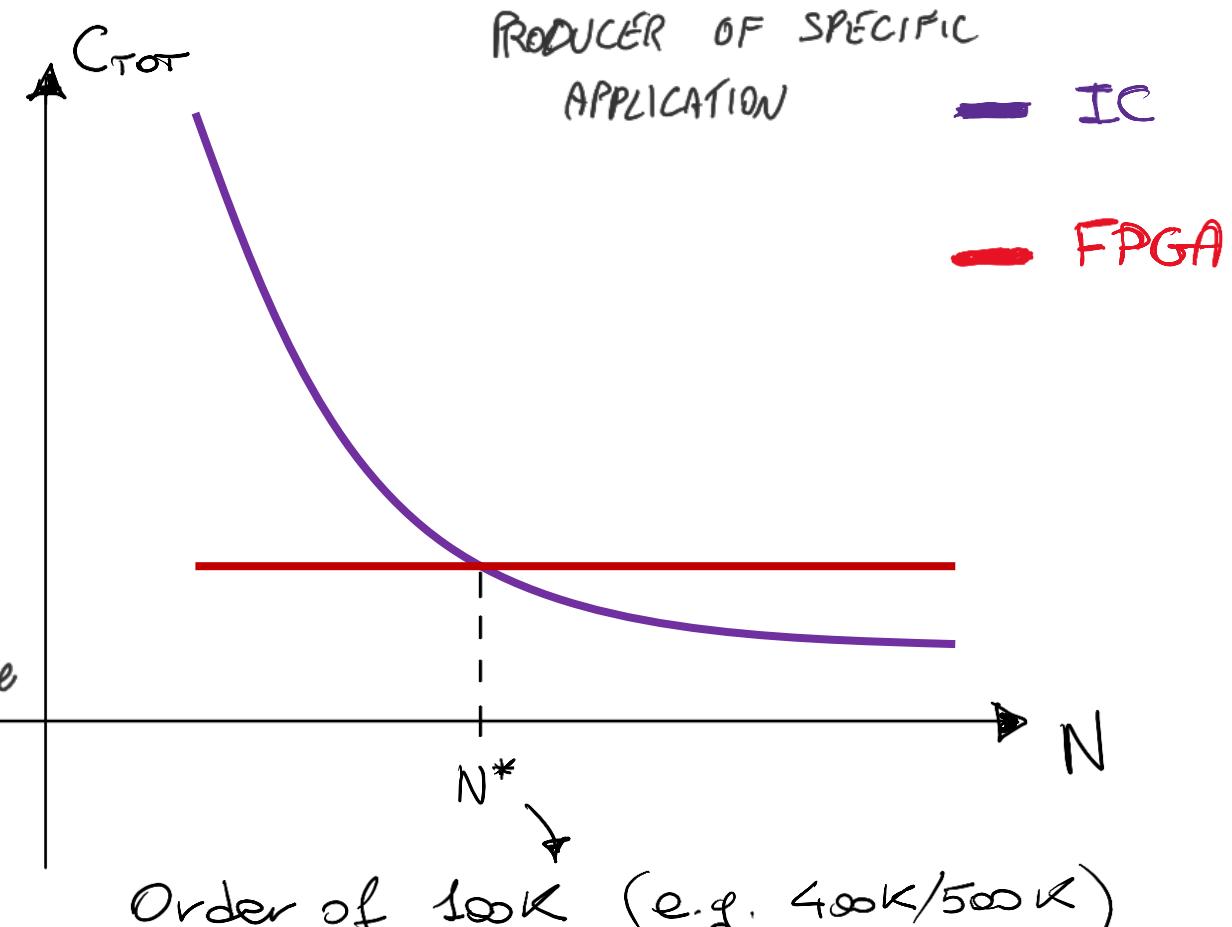
- Assume

- C_{NRE} = Non-recurring costs
- C_{RE} = Recurring costs per unit
 - $C_{RE_{IC}} \ll C_{RE_{FPGA}}$
- C_{TOT} = Total cost per unit
- N = number of units

- It follows that

- $C_{TOT_{FPGA}} = C_{RE_{FPGA}}$ ↑ proportional to number of unverified circuits to realize
- $C_{TOT_{IC}} = C_{RE_{IC}} + \frac{C_{NRE}}{N}$

Aspect for choices: # of devices that will be required for application.



FPGA – Last notes

- Main vendors – EDA and main devices
 - Altera/Intel
 - EDA : Quartus Prime
 - Main FPGA families : Cyclone, Arria, Stratix
 - Xilinx/AMD
 - EDA : Vivado
 - Main FPGA families : Spartan, Kintex, Artix, Virtex
 - Microsemi/Microchip
 - EDA : Libero SoC
 - Main FPGA families : PolarFire, SmartFusion, RT (Radiation Tolerant)



Thank you for your attention

Luca Crocetti
(luca.crocetti@unipi.it)