Course Info

- No Lab/discussion this week
- Project 2.2 will be available soon.
- HW5 available.
- MID-TERM II this Thursday!!!

信息科学与技术学院
School of Information Science and Technology

CS 110
Computer Architecture
Heterogeneous Computing & FPGA

Instructors:

Siting Liu & Chundong Wang

Course website: https://toast-lab.sist.shanghaitech.edu.cn/courses/CS110@ShanghaiTech/Spring-2023/index.html

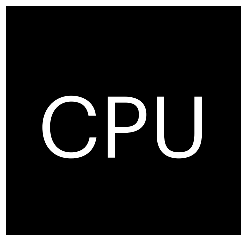
School of Information Science and Technology (SIST)

ShanghaiTech University

2023/2/7

Hardware vs. Software

- How to perform an addition?




```
lw t1,address
lw t2,address
add t0,t1,t2
sw t0,address
```

IF → ID → EX → MEM → WB

300 ps 100 ps 200 ps 300 ps 100 ps

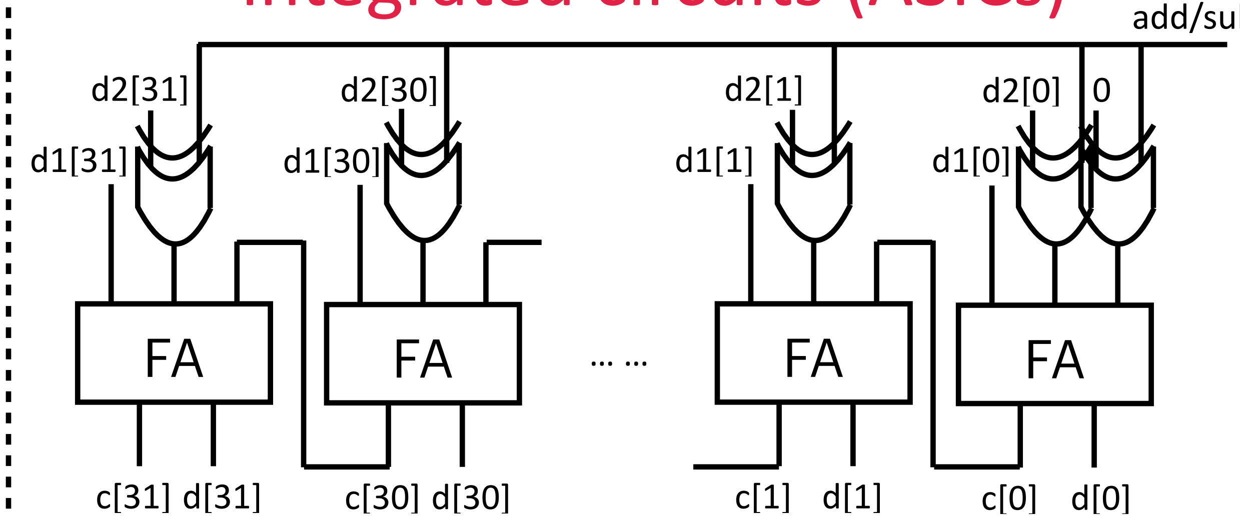
Very complex circuitry

Relatively high power consumption

High flexibility

Software-programmable

Application-specific integrated circuits (ASICs)



Only EX

200 ps

Only use hundreds of gates

Low-power

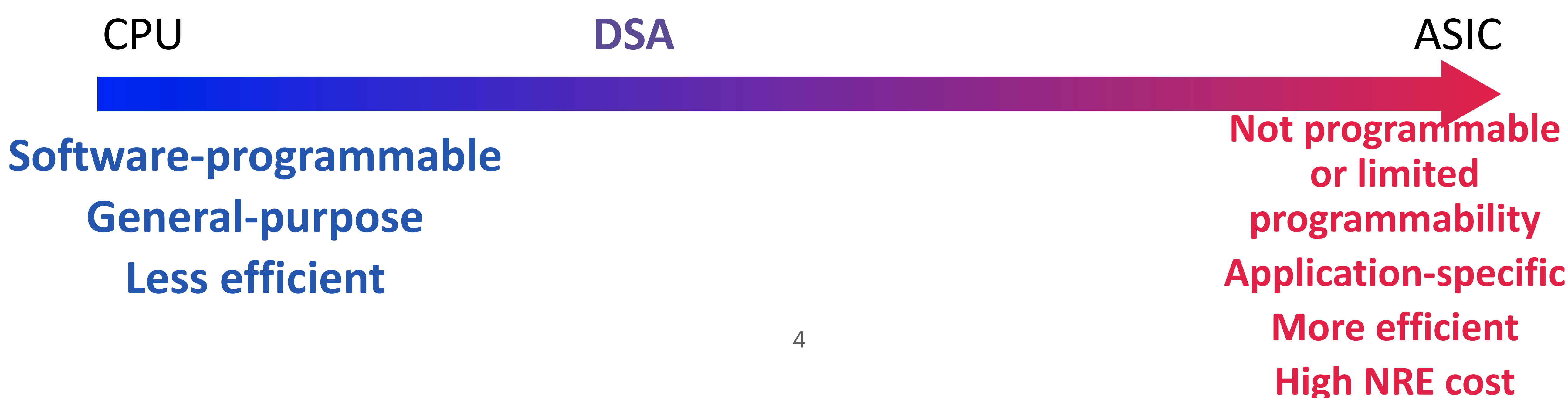
Only perform addition/subtraction

Not programmable

Hardware vs. Software

- For a task, we can always write software code or build ASIC hardware.

Software solution	Hardware solution
+/add instruction	Build an adder
Shift-and-add	RV-M extension & add multiplier
Insert “nop” to avoid stall	Add hardware to avoid stall
… …	… …



Domain-Specific Architecture

- Build hardware for an application "domain" instead of a certain task
- Moderate software-programmable, with relatively higher efficiency compared with CPU

GPU

For graphics
Rasterization/texture/rendering, etc.
Vector/Matrix operations
CUDA (2006), openCL, etc.
GPGPU

AI-accelerators/NPU/AI chips

For neural networks
Tensor operations, etc.
No unified programming language yet
Still a fast-developing field

CPU

Software-programmable
General-purpose
Less efficient
Low NRE cost

DSA

Software-programmable
Domain-specific
Efficient
Low NRE cost

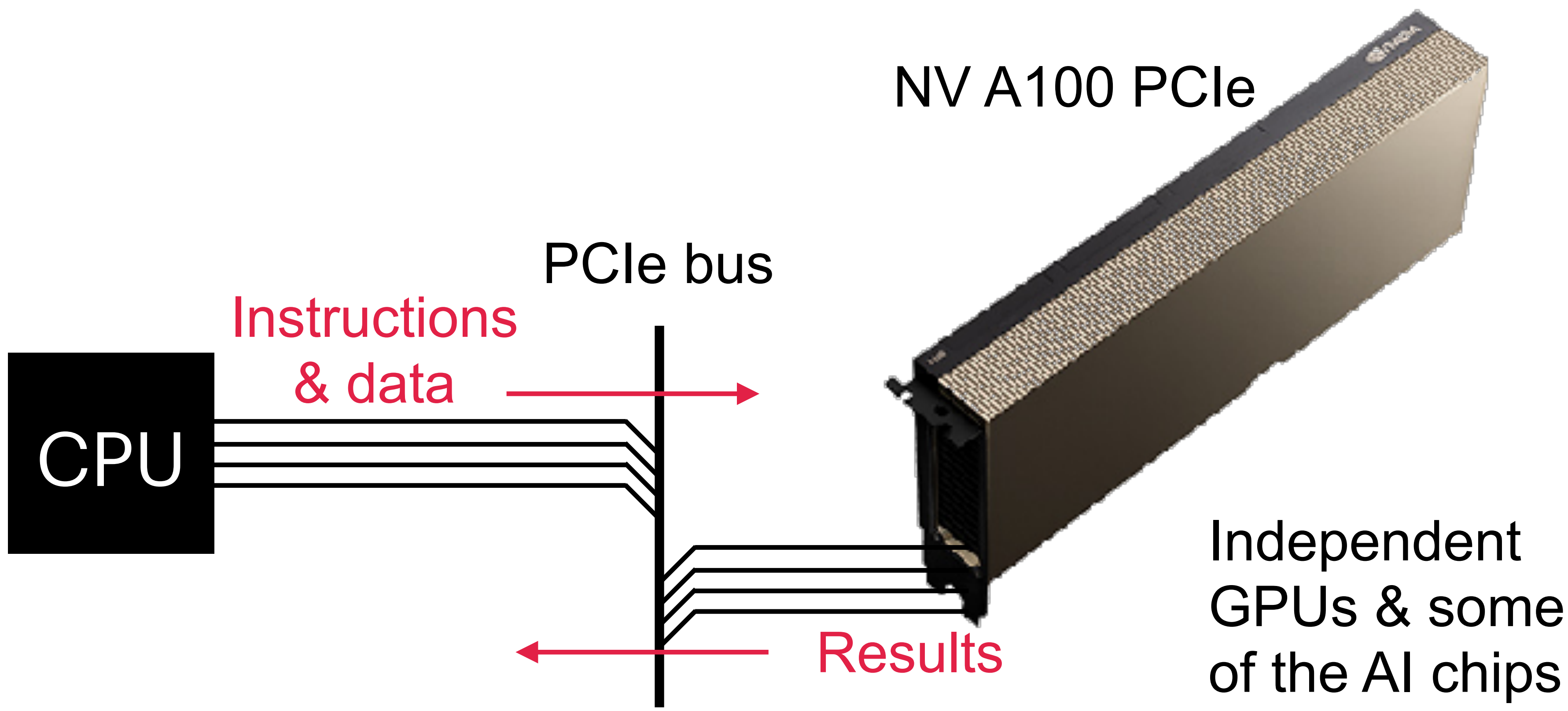
ASIC

Not programmable or limited programmability
Application-specific
More efficient
High NRE cost

Heterogeneous Computing

- Usually cannot work independently
  - **Parallel to CPU, as an I/O device**
  - Integrated in an SoC, as a co-processor

Heterogeneous computing refers to systems that use more than one kind of processor or core.



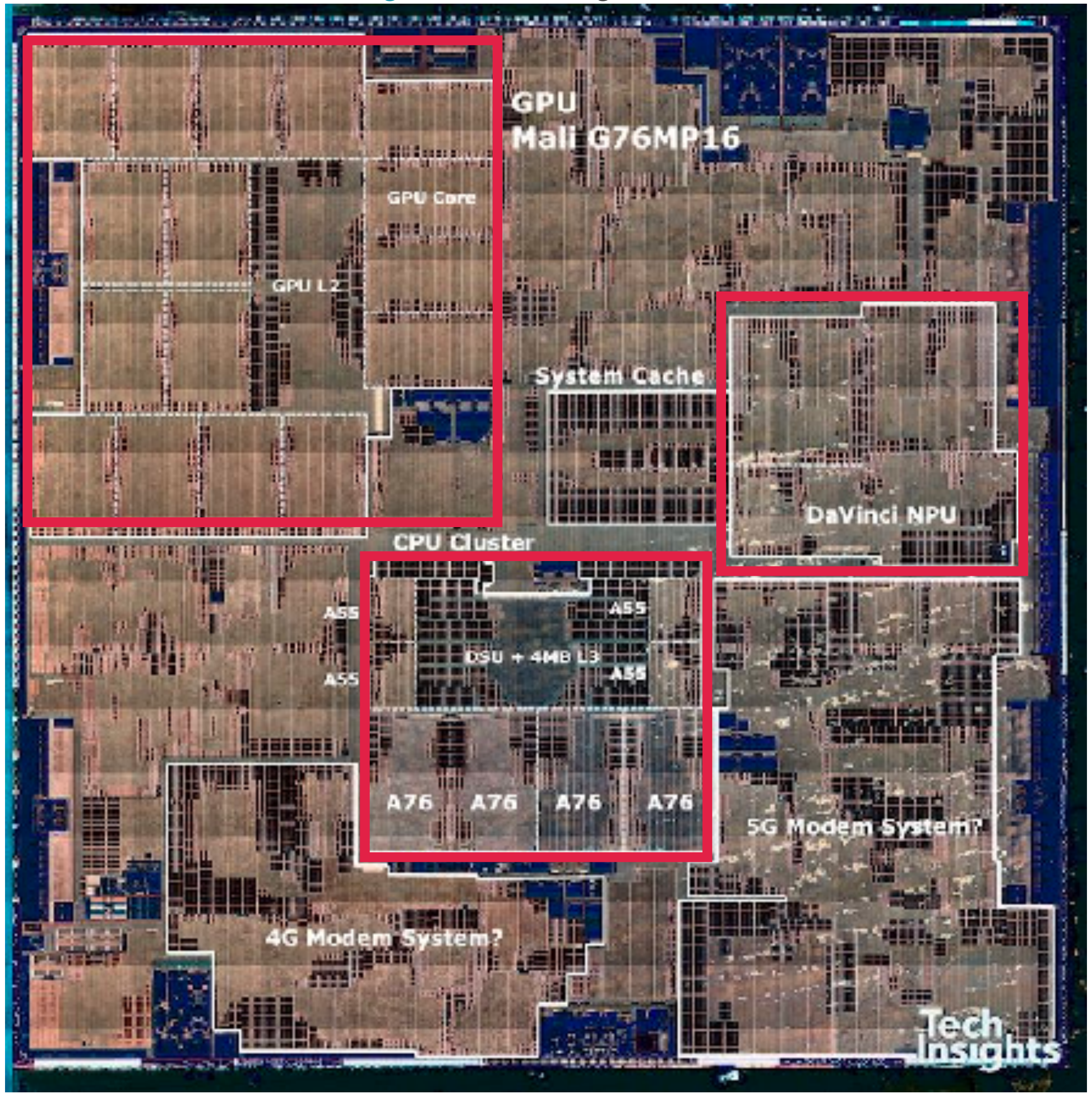
Heterogeneous Computing

- Usually cannot work independently
  - Parallel to CPU, as an I/O device
  - **Integrated in an SoC, as a co-processor**
    - Historically, math processor
    - Integrated GPU in CPU
    - etc.

Image signal processors (ISP)

Huawei Kirin 990

Die shot source: TechInsights - Labelling & Custom contrast: AnandTech



Heterogeneous Computing

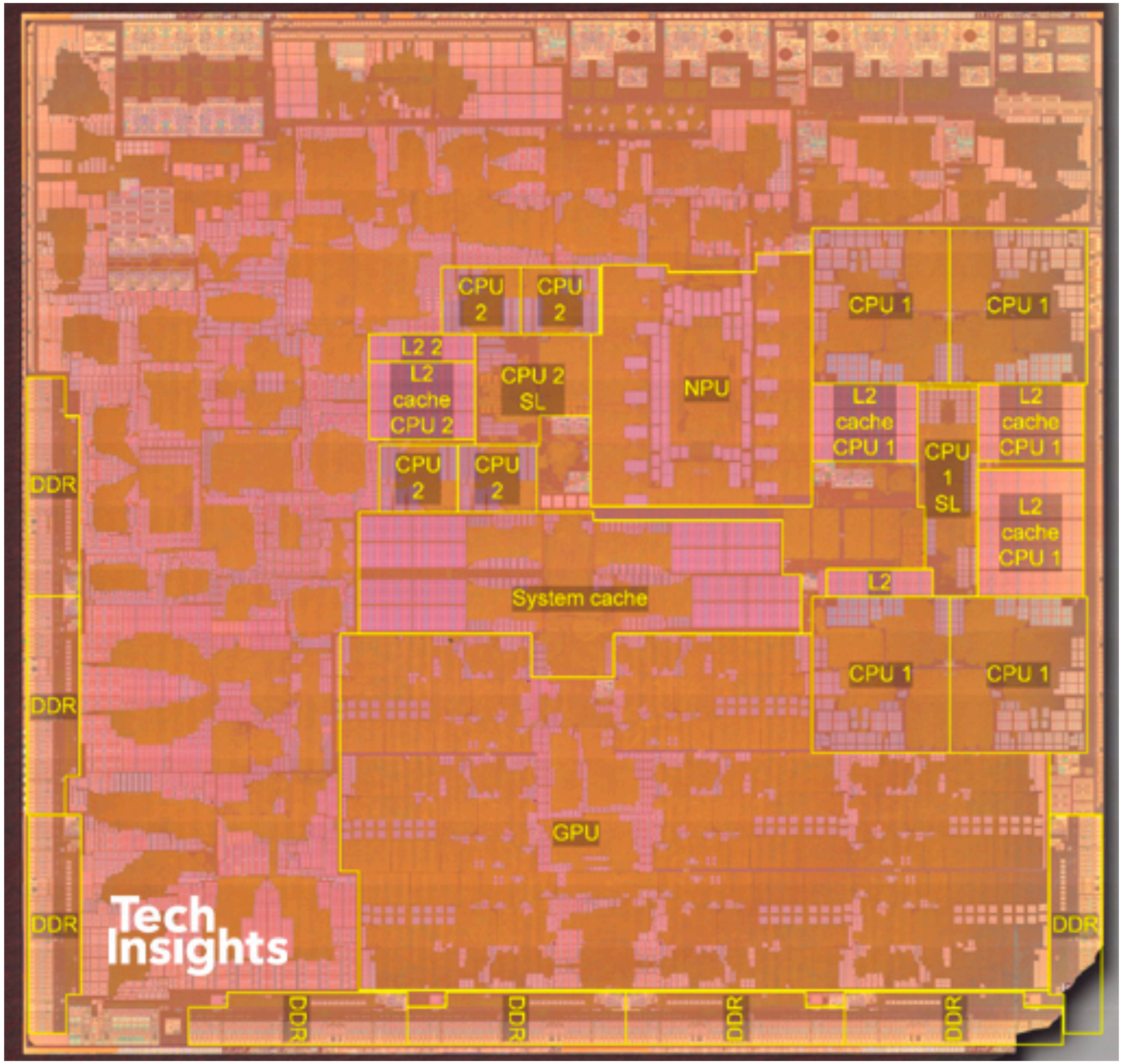
Image signal processors (ISP)

AES encryption/security

Video decoder/encoder

Co-processor examples

Apple M1
Die shot source: TechInsights



Heterogeneous Computing

Company	Name	Process	CPU	GPU	NPU
HiSilicon	Kirin 980 [1]	TSMC 7nm	2 big Cortex-A76 @ 1.92 GHz 2 big Cortex-A76 @ 2.6 GHz 4 little Cortex-A55 @ 1.8 GHz	ARM 10-core Mali-G76 (FP32 480 GFLOPs)	Dual NPU
Apple	M1Max [2][3]	TSMC 5nm	2 HE cores @ 2.064 GHz 8 HP cores @ 3.228 GHz	24/32-core GPU (FP32 10.4 TFLOPs)	16-core neural engine (11 TOPs)

Special cases: SoC with different CPU cores also considered heterogeneous

Another Dimension—Hardware Programmable

Image signal processors (ISP)

AES encryption/security

Video decoder/encoder

- Cannot continue the list forever (algorithms are evolving)
- NRE cost non-negligible

Co-processor examples

CPU

Software-programmable
General-purpose
Less efficient
Low NRE cost

DSA

Software-programmable
Domain-specific
Efficient
Low NRE cost

Hardware-programmable devices

ASIC

Not programmable or limited programmability
Application-specific
More efficient
High NRE cost

Field-Programmable Gate Array (FPGA)

- The hardware functionality can be changed by programming (mostly HDLs such as Verilog & VHDL)
- FPGA can implement any digital circuits with a certain size
- Shorter time-to-market vs. ASIC
- Heterogeneous w.r.t. CPU/GPU

CPU	DSA	FPGA	ASIC
Software-programmable	Software-programmable	HW-programmable	Not programmable or limited programmability
General-purpose	Domain-specific	Versatile	Application-specific
Less efficient	Efficient	More efficient	More efficient
Low NRE cost	Low NRE cost	Low NRE cost	High NRE cost

Another Dimension—Hardware Programmable

Hardware-programmable

Providers	CPU	GPU	FPGA	ASIC (DSA)
Alibaba Cloud	X86/ARM/RISC-V	Nvidia/AMD	Intel (Altera)/ AMD (Xilinx)	AliNPU
AWS (Amazon)	Graviton (ARM) / X86	Nvidia/AMD	Xilinx	AWS Trainium
Azure (MS)	X86	Nvidia	Certain DNN models	
Baidu Cloud	X86	Nvidia	Xilinx	Kunlun
Google Cloud	X86	Nvidia	N/A	TPU
Huawei Cloud	Kunpeng (ARM)/X86	Nvidia & Ascend	Xilinx	Ascend
Tecent Cloud	X86	Nvidia & Xinghai	Xilinx	Enflame-tech (燧原)

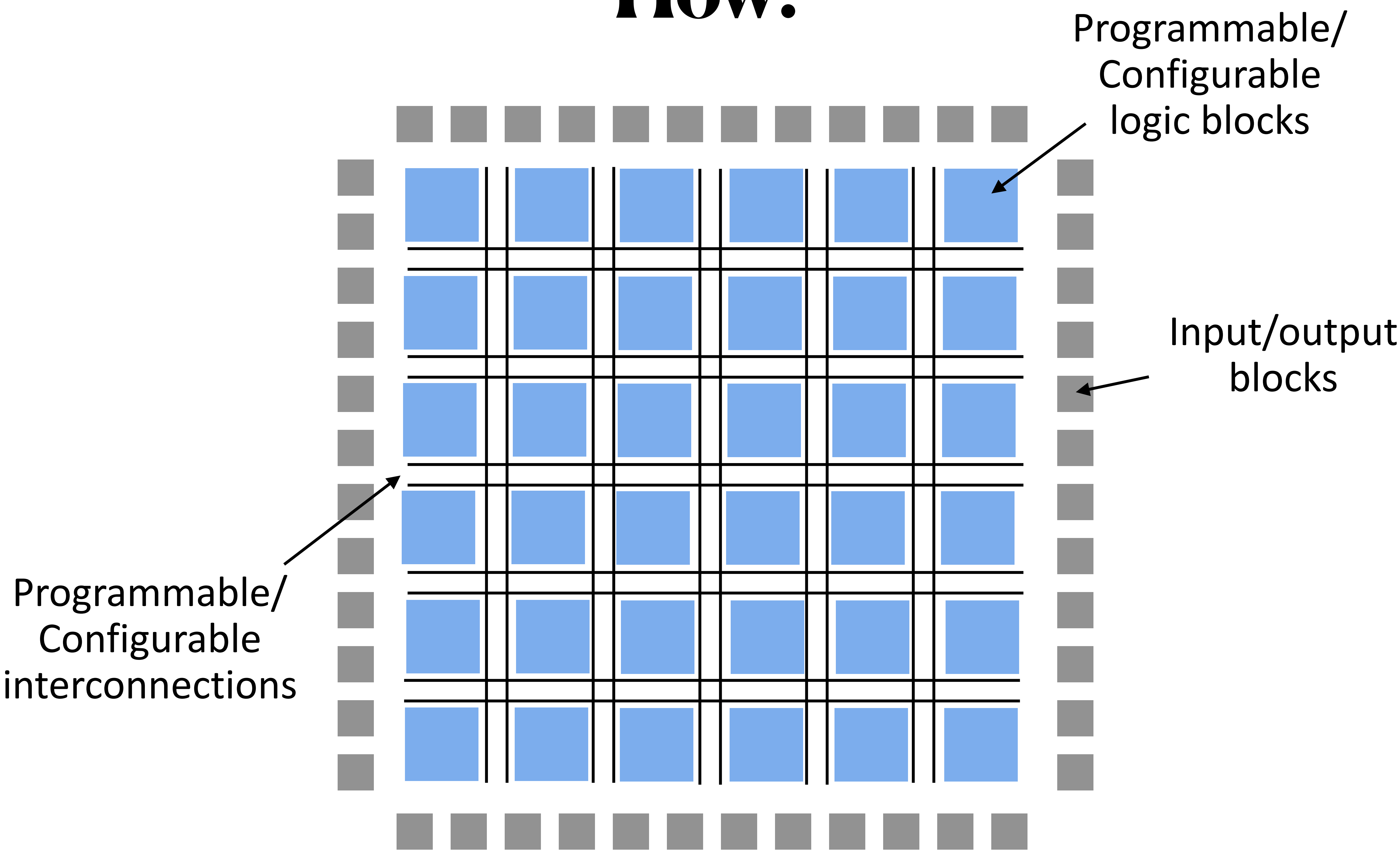
FPGA Applications

- Communication (decoding/encoding algorithms, etc.)
  - Smart networking device
  - SmartNIC

- FPGA trading systems (High-frequency trading)
- AI tasks (MS Project Brainwave/Xilinx Vitis AI)
- Also in embedded systems
  - Digital signal processing
  - Image signal processing
  - Control logics

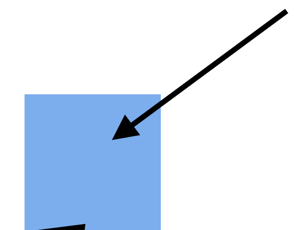
- Also in IC design, for hardware emulation

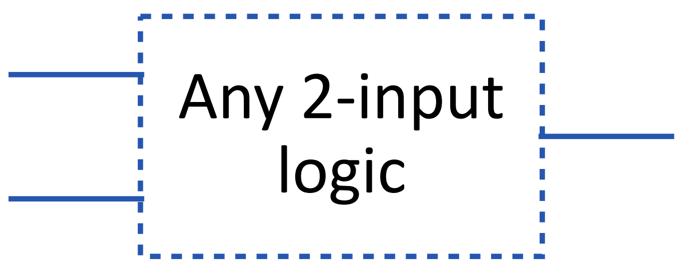
How?


Programmable/
Configurable
logic blocks
Input/output
blocks
Programmable/
Configurable
interconnections

Configurable Logic Blocks

Programmable/
Configurable
logic blocks





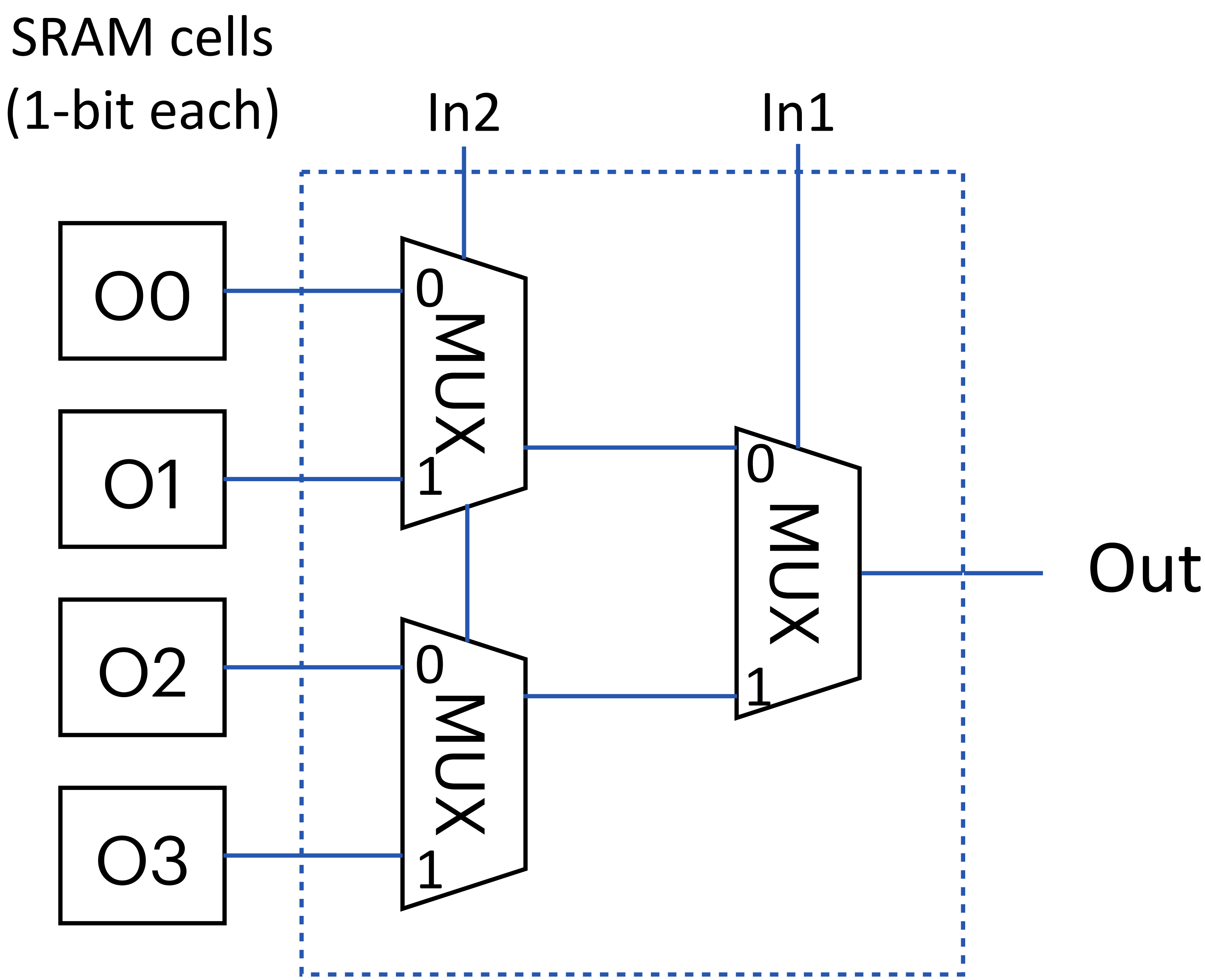
Combinational circuits

In1	In2	Out
0	0	O0
0	1	O1
1	0	O2
1	1	O3

We store the truth table and it can implement any 2-input logic

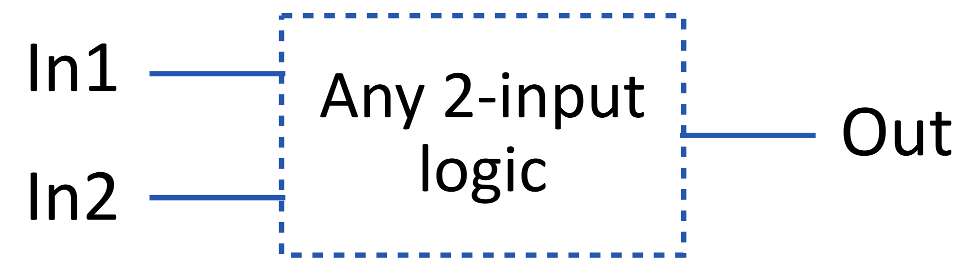
SRAM-based FPGA

- Look-up table (LUT)



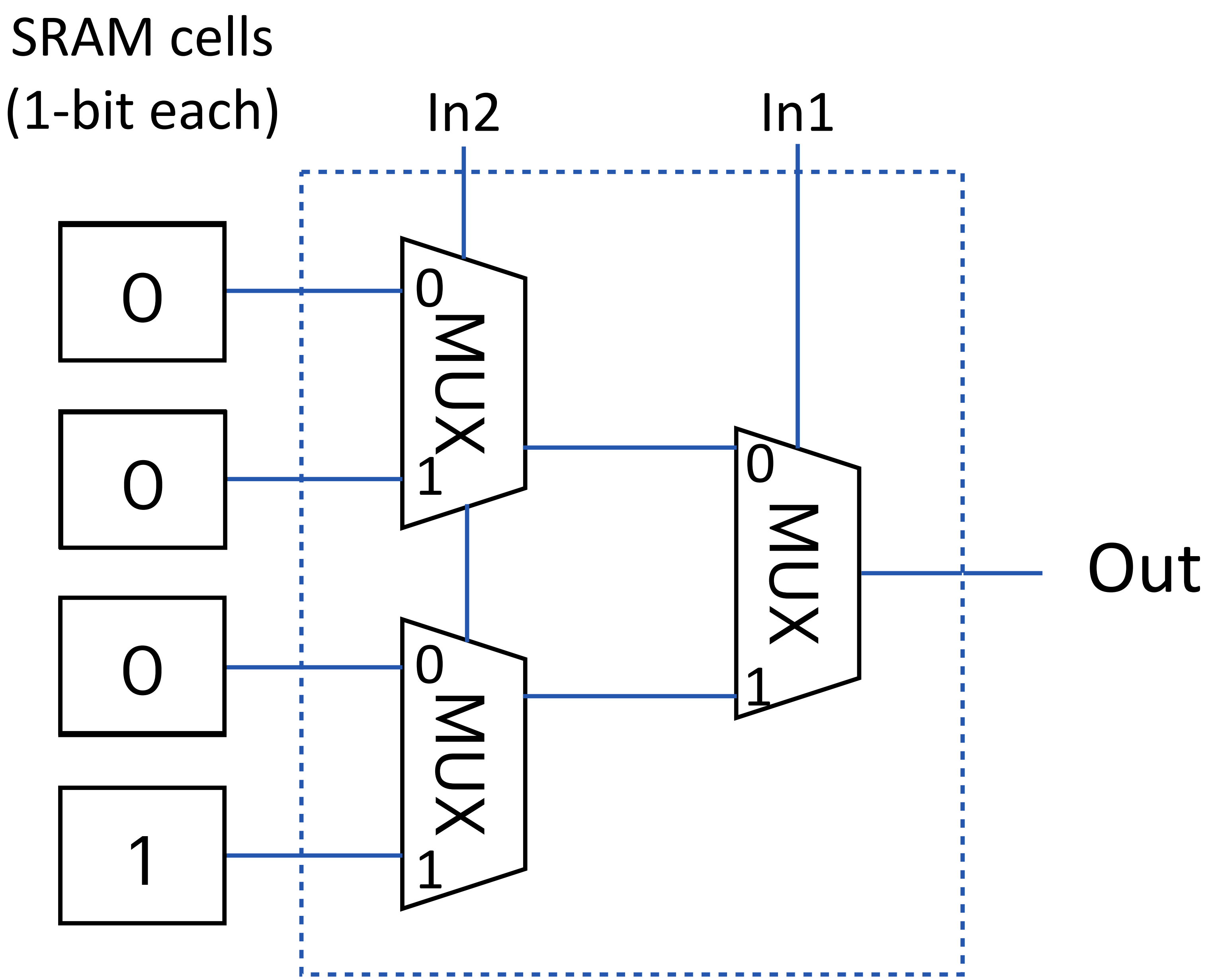
In1	In2	Out
0	0	O0
0	1	O1
1	0	O2
1	1	O3

Equivalent to



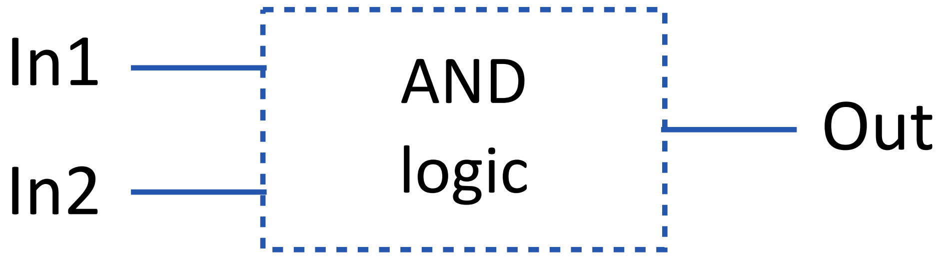
SRAM-based FPGA

- Example: 2-input LUT implementing AND logic



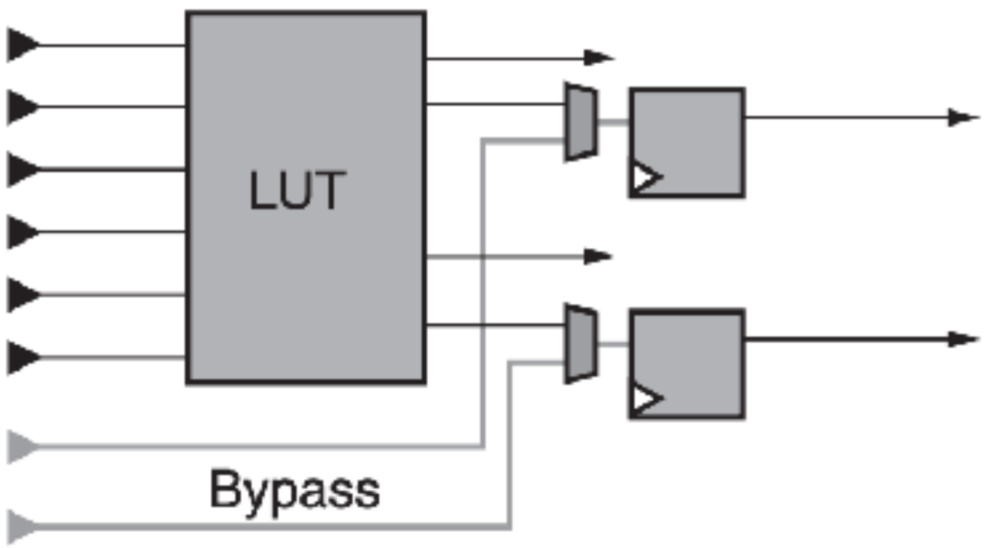
In1	In2	Out
0	0	0
0	1	0
1	0	0
1	1	1

Equivalent to



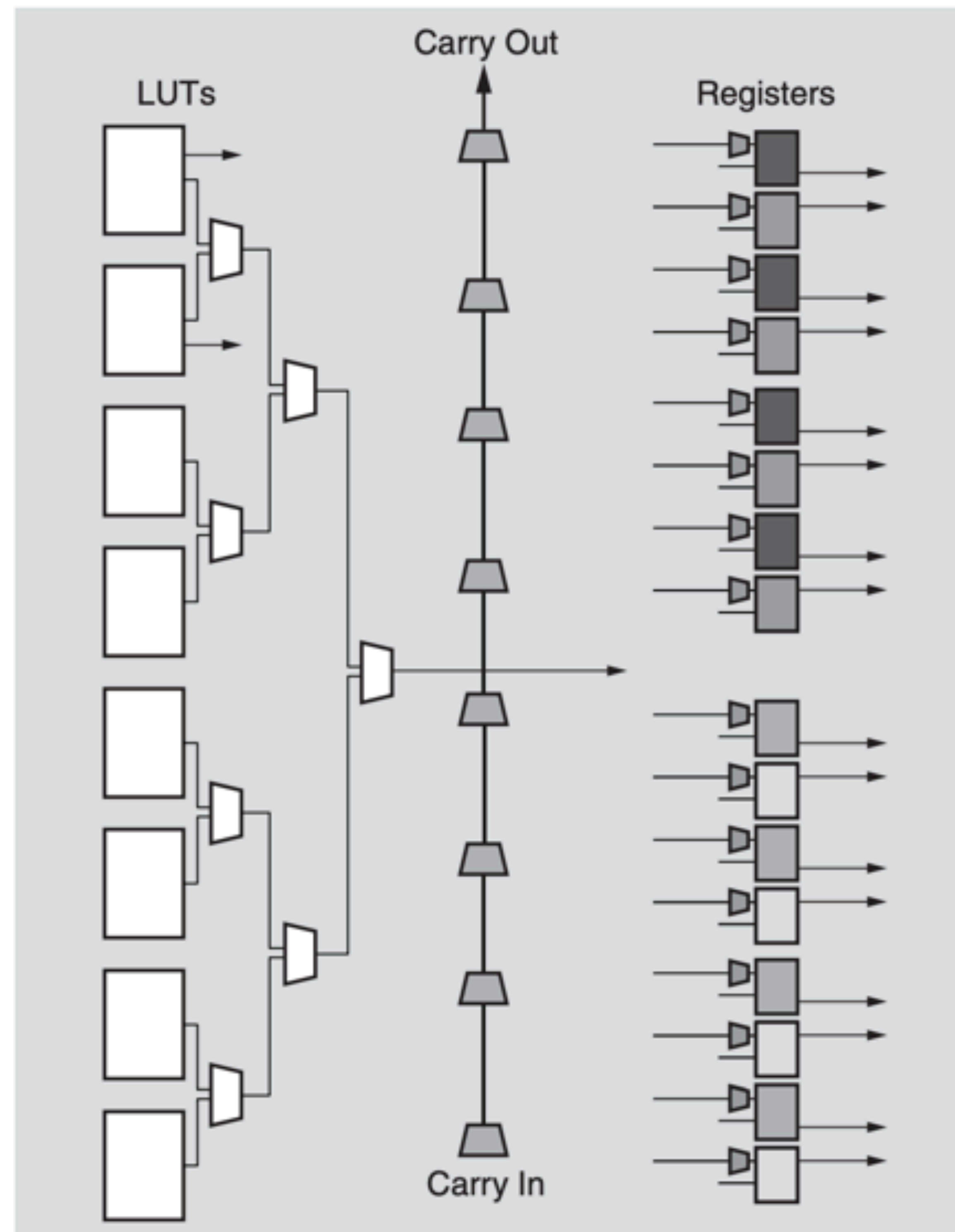
SRAM-based FPGA

- Reality: LUT with larger number of inputs are more capable



6-input LUT
with bypass path & registers

One configurable logic block (CLB in Xilinx/AMD FPGA) consists of many LUTs, registers and carry chain (for arithmetic)

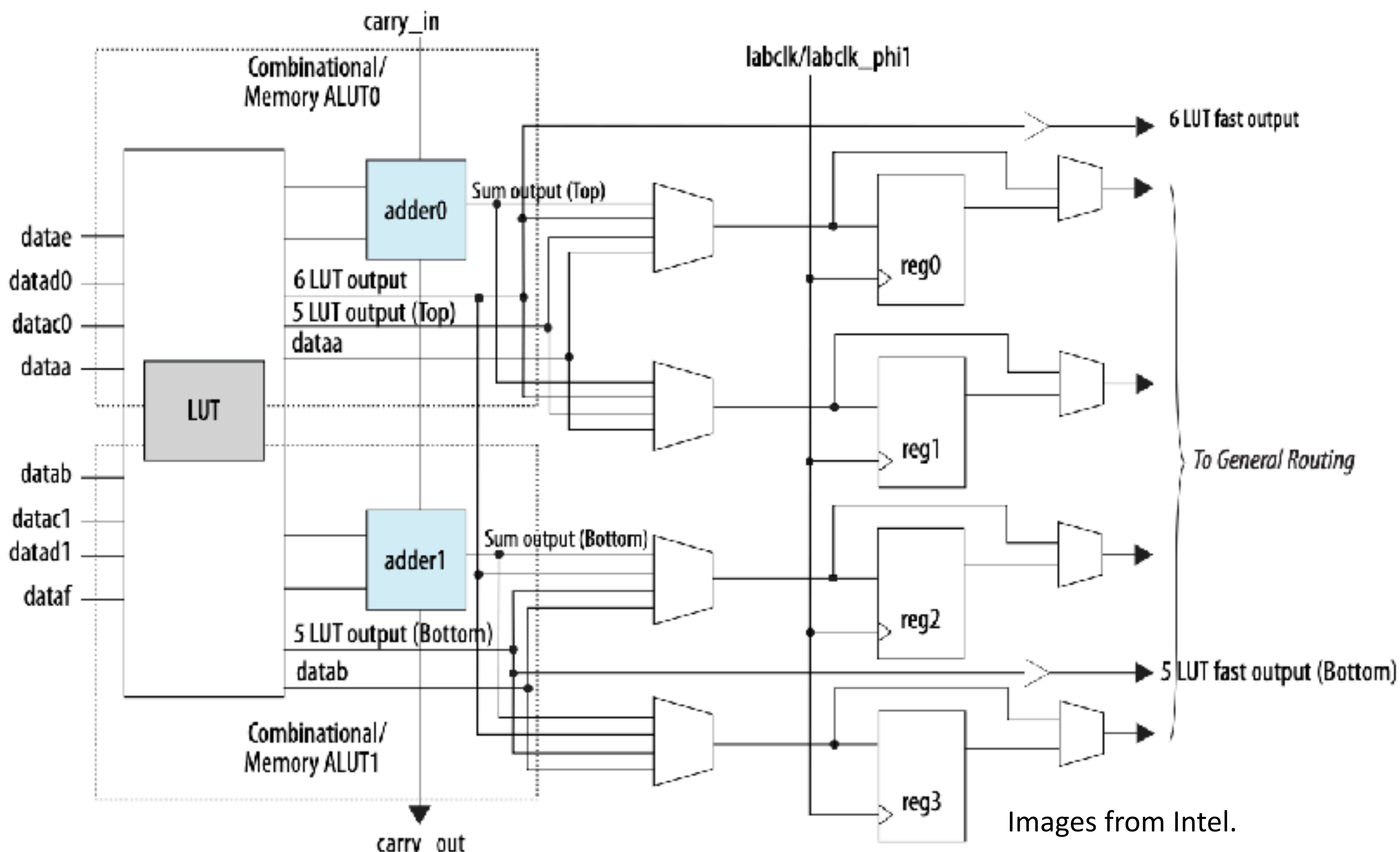


Images from AMD

SRAM-based FPGA

- Reality: LUT with larger number of inputs are more capable

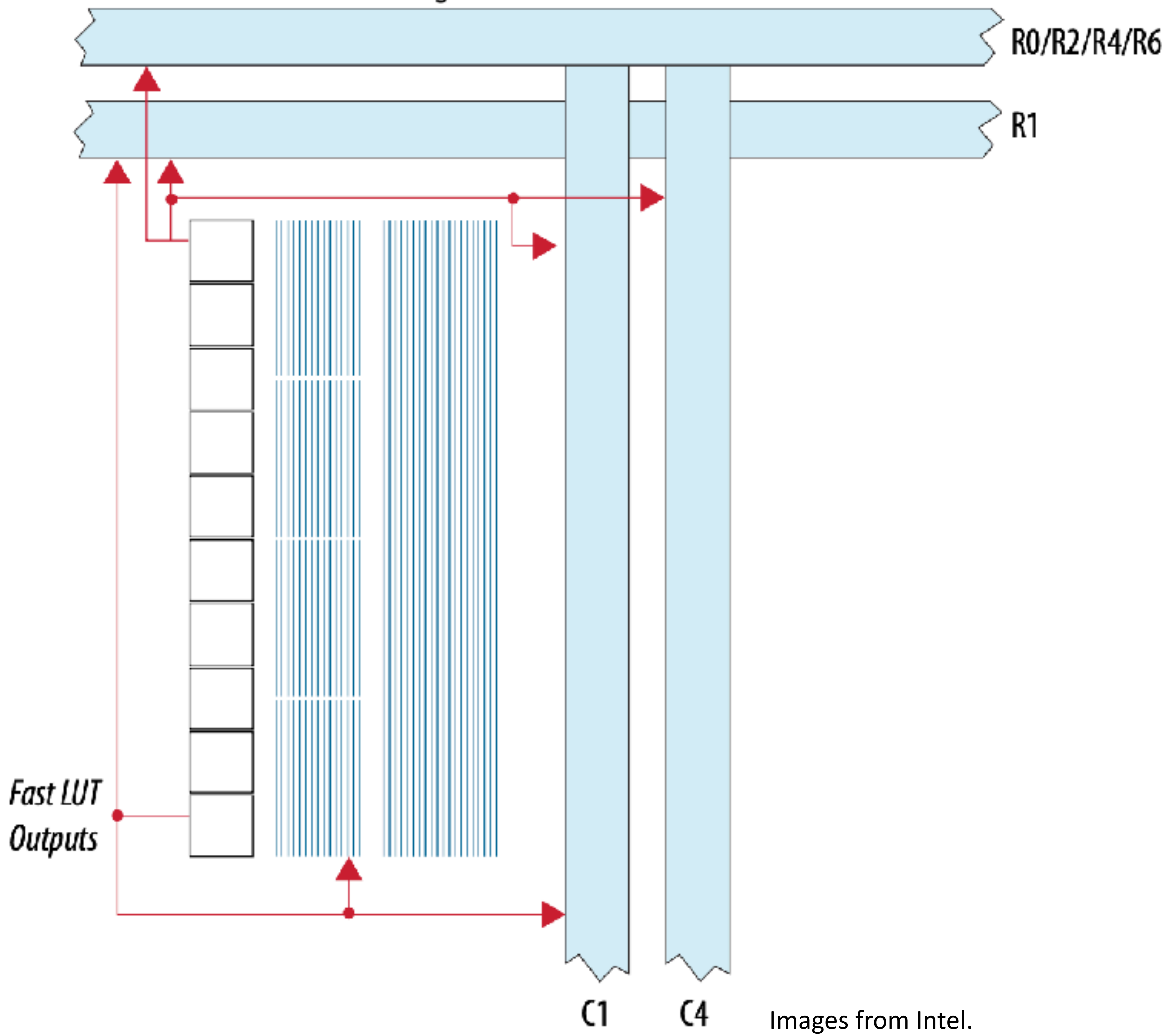
Intel Agilex 7 ALM High-Level Block Diagram (ALM in Altera/Intel FPGA)



Images from Intel.

Intel Agilex 7 LAB Structure and Interconnects Overview

This figure shows an overview of the Intel Agilex 7 LAB and MLAB structure with the LAB interconnects.



Images from Intel.

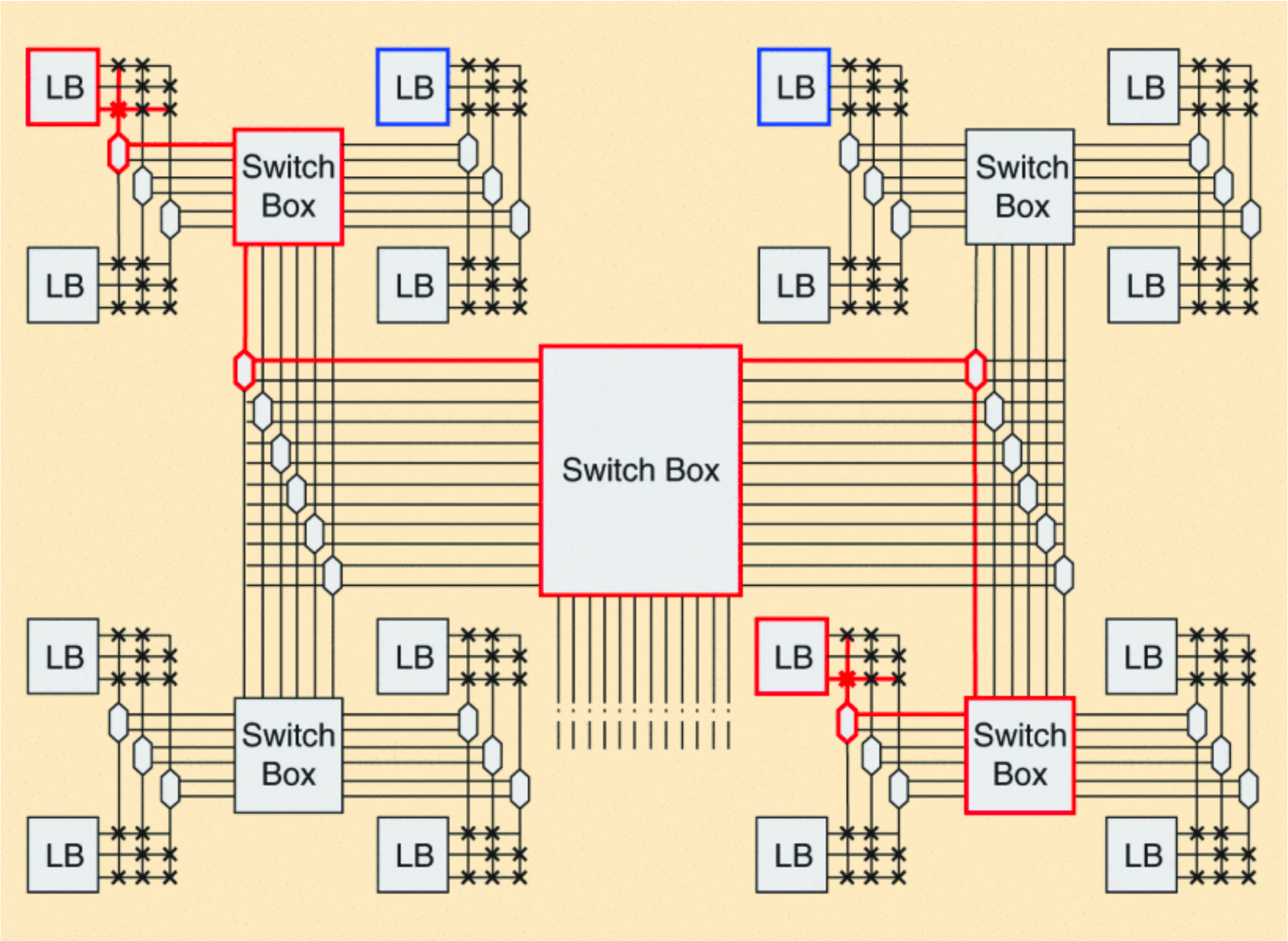
Other than CLB/LAB

- Routing, aka, interconnecting
  - Through programmable wires and switches
  - Between logic blocks (CLB/ALMs), and between I/O blocks and logic blocks
- Routing is a challenging problem
  - Routing technique used in an FPGA largely decides the amount of area used by wire segments and programmable switches, as compared to area consumed by functional blocks.
  - Inferior routing may lead to congestion or failure of signals.

Other than CLB/LAB

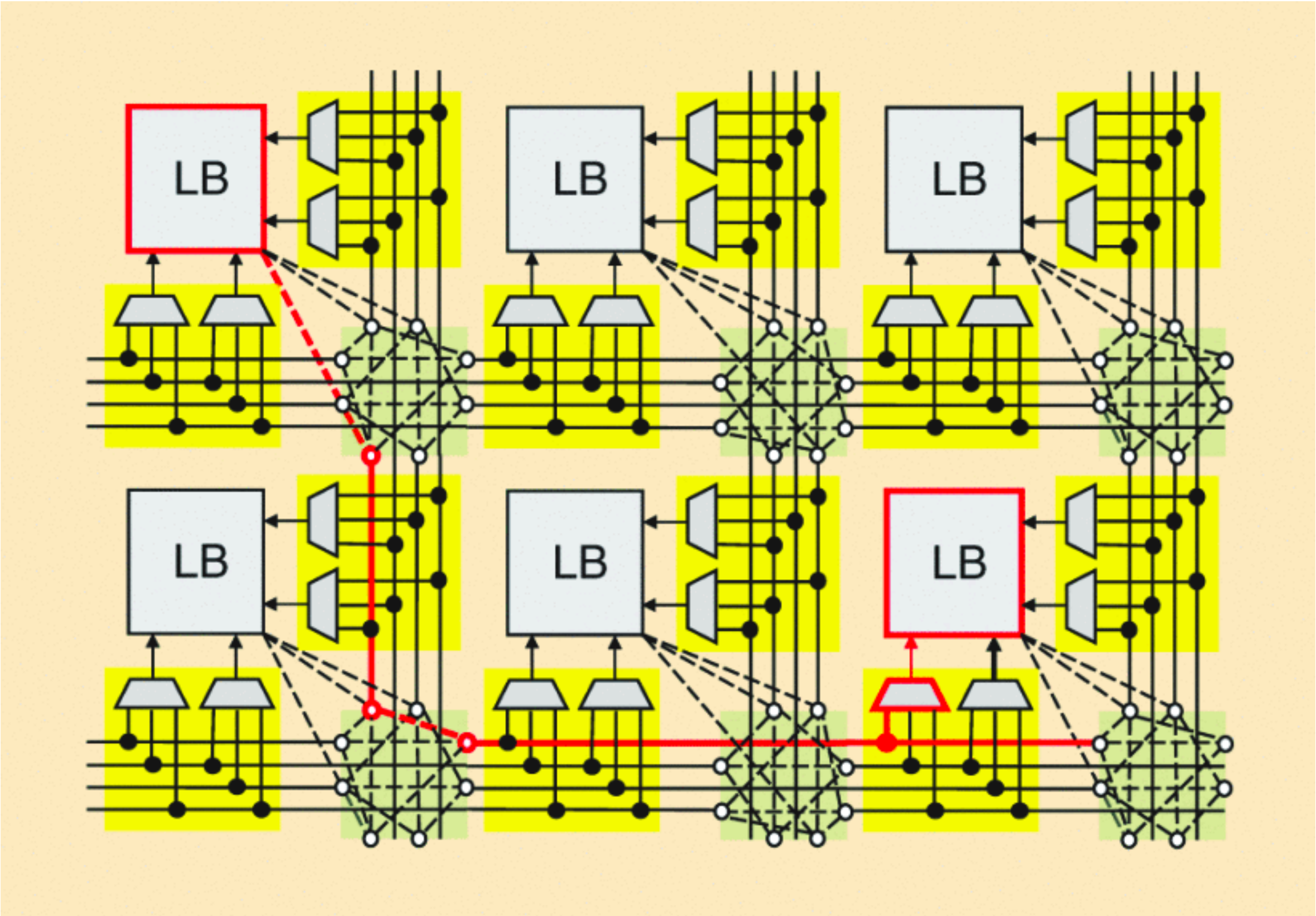
- Routing, aka, interconnecting
  - Through programmable wires and switches
  - Between logic blocks (CLB/ALMs), and between I/O blocks and logic blocks
- Routing is a challenging problem
  - Routing technique used in an FPGA largely decides the amount of area used by wire segments and programmable switches, as compared to area consumed by functional blocks.
  - Inferior routing may lead to congestion or failure of signals.
- Different FPGA routing architecture
  - Hierarchical FPGA
  - Island-style routing architecture

Hierarchical FPGA



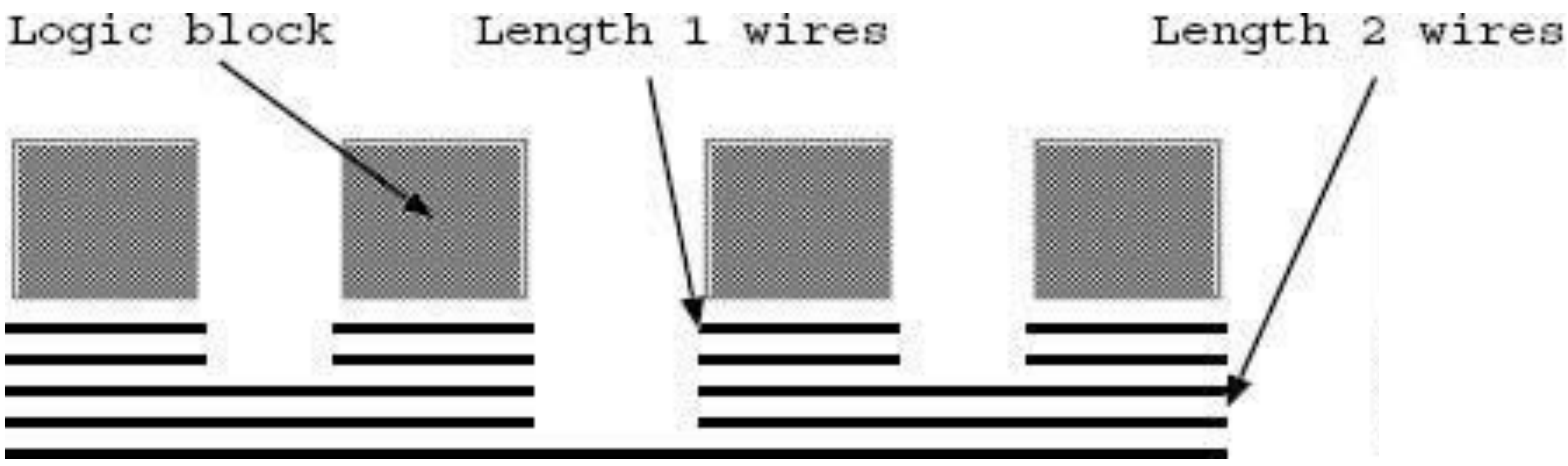
A. Boutros and V. Betz, "FPGA Architecture: Principles and Progression," in *IEEE Circuits and Systems Magazine*, vol. 21, no. 2, pp. 4-29, Secondquarter 2021, doi: 10.1109/MCAS.2021.3071607.

Island-style FPGA



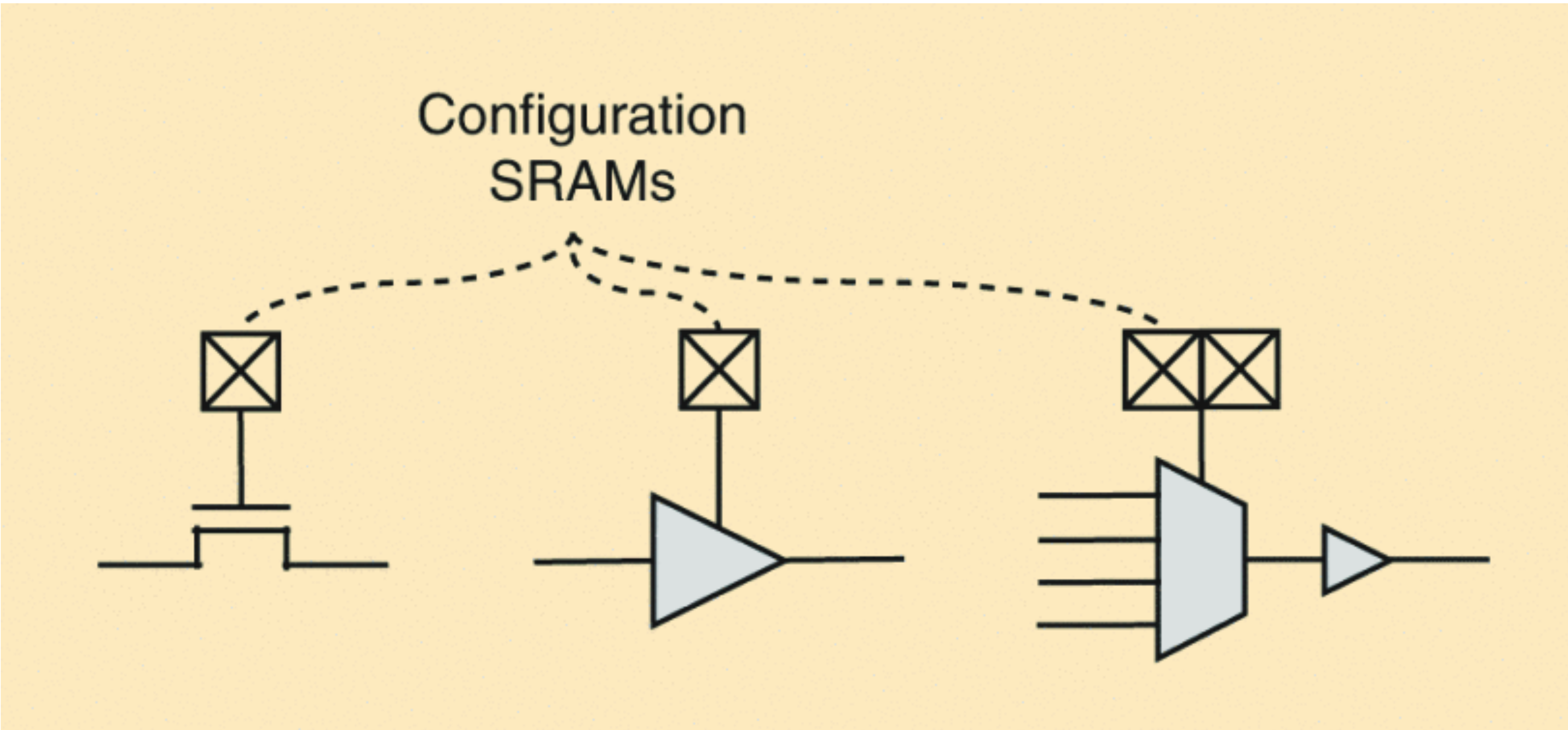
A. Boutros and V. Betz, "FPGA Architecture: Principles and Progression," in *IEEE Circuits and Systems Magazine*, vol. 21, no. 2, pp. 4-29, Secondquarter 2021, doi: 10.1109/MCAS.2021.3071607.

FPGA Routing Wires



- Some FPGAs contain routing architectures that include different lengths of wires.
- The length of a wire is the number of functional blocks it spans.
- Long wires introduce shorter delays for long interconnections since fewer switch blocks will be passed.

Programmable Switches

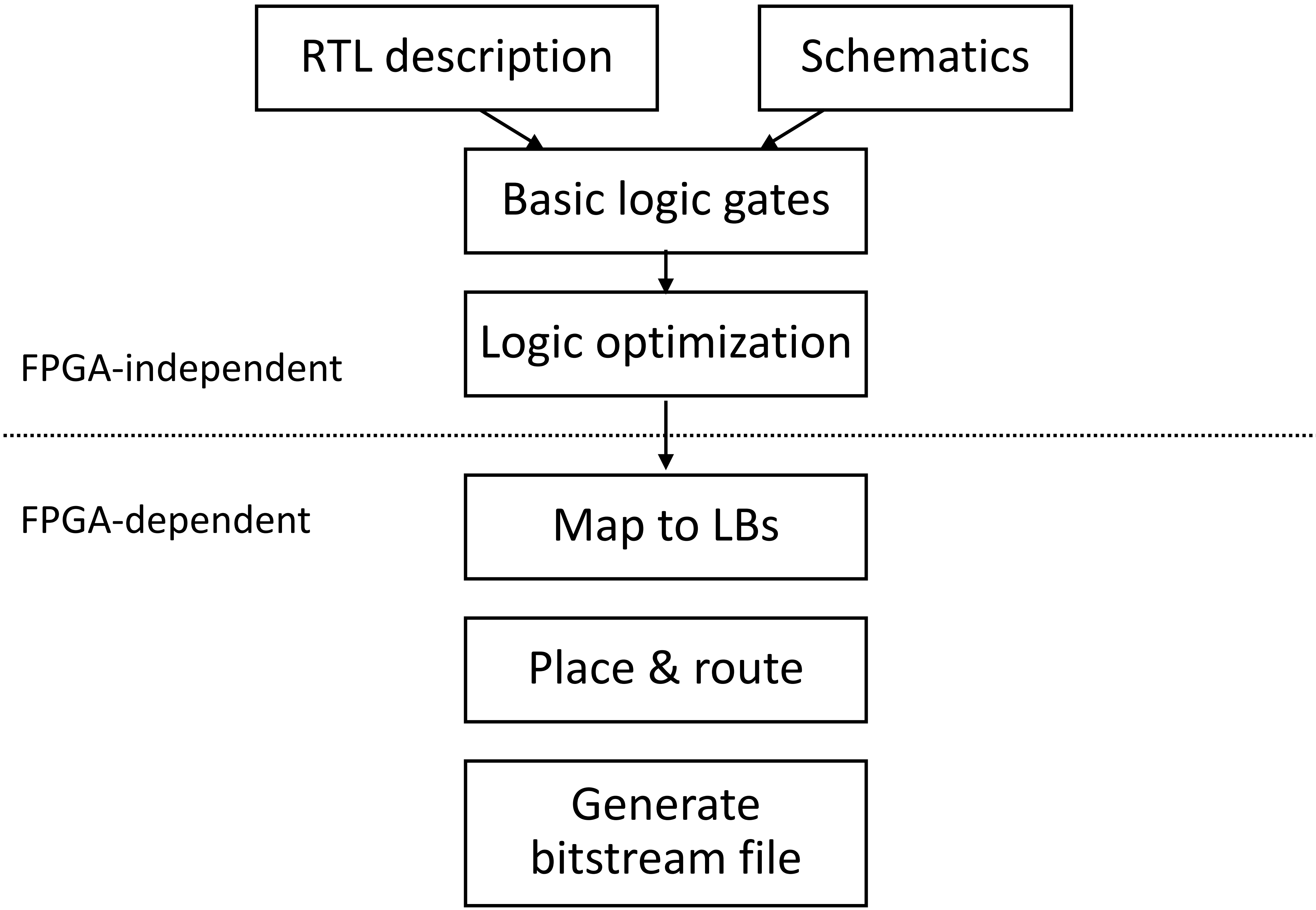


A. Boutros and V. Betz, "FPGA Architecture: Principles and Progression," in *IEEE Circuits and Systems Magazine*, vol. 21, no. 2, pp. 4-29, Secondquarter 2021, doi: 10.1109/MCAS.2021.3071607.

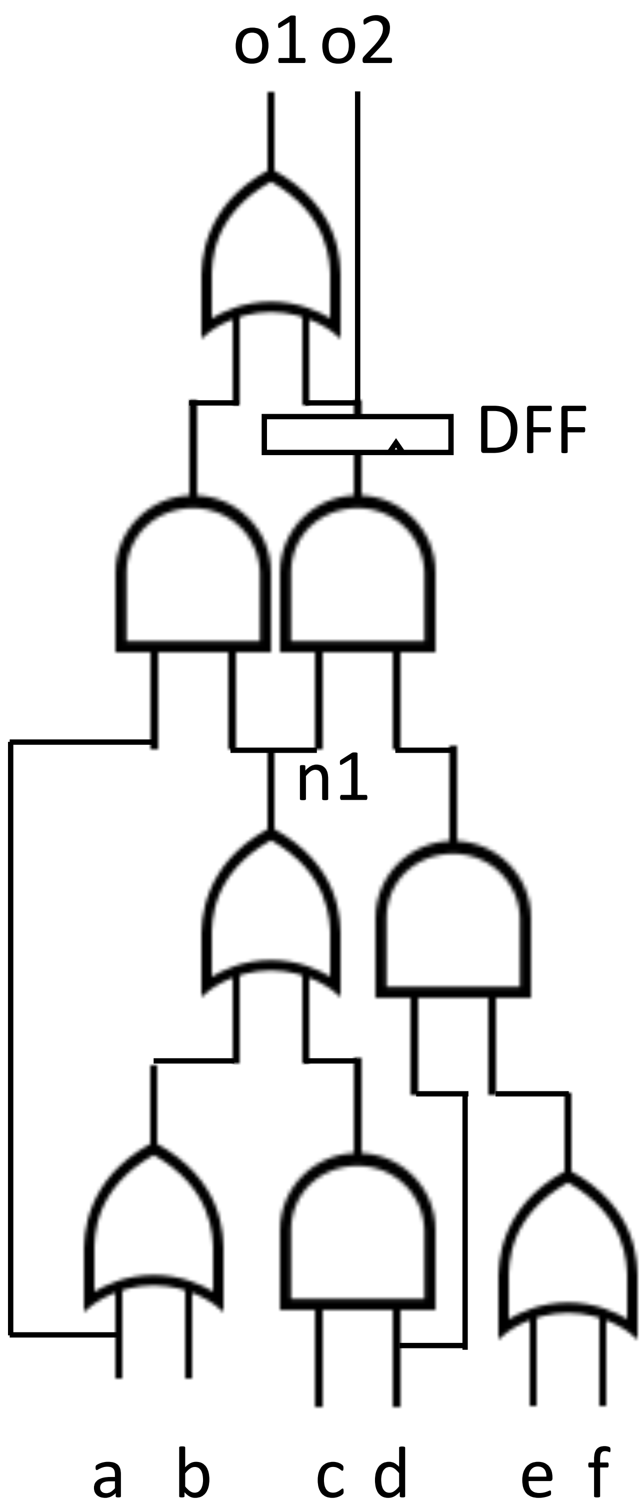
Software Side: FPGA Design Flow

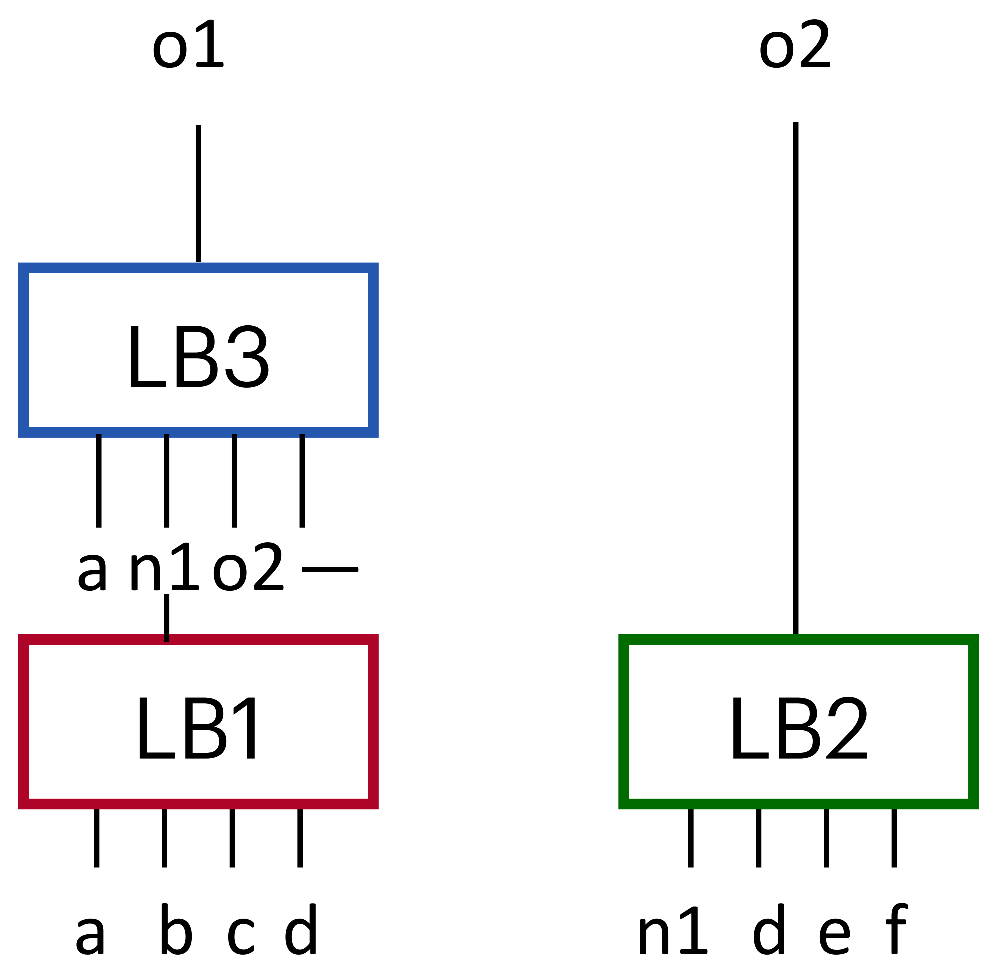
Register-transfer level (RTL) model: describe the dataflow

A visual representation of the design (such as Logisim)

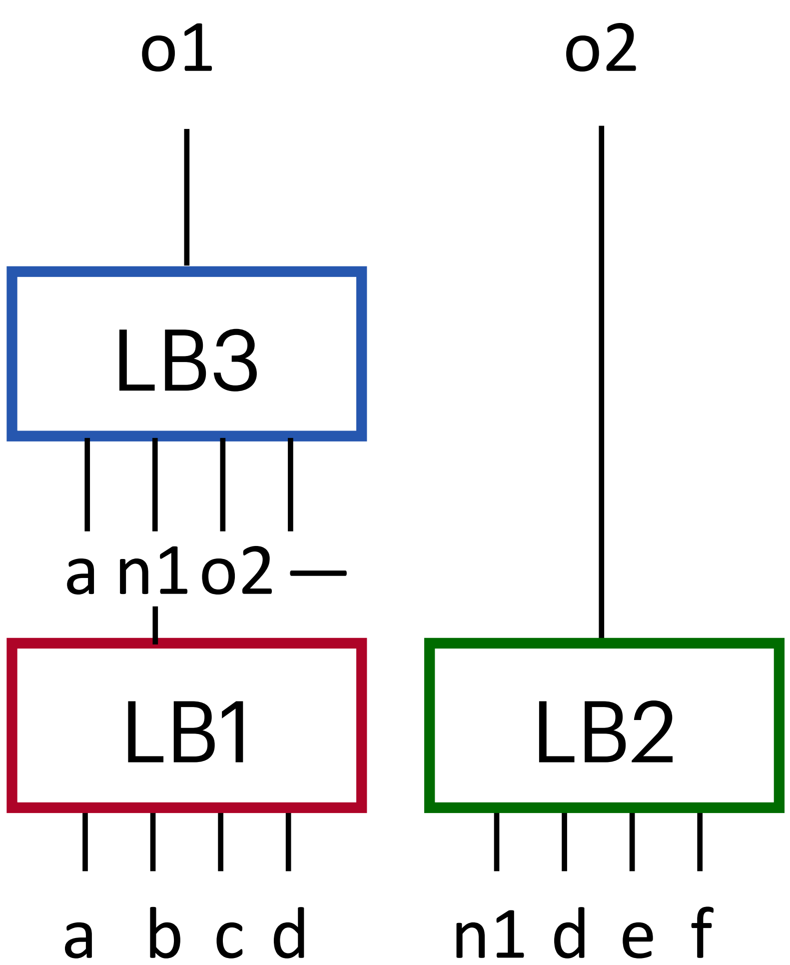


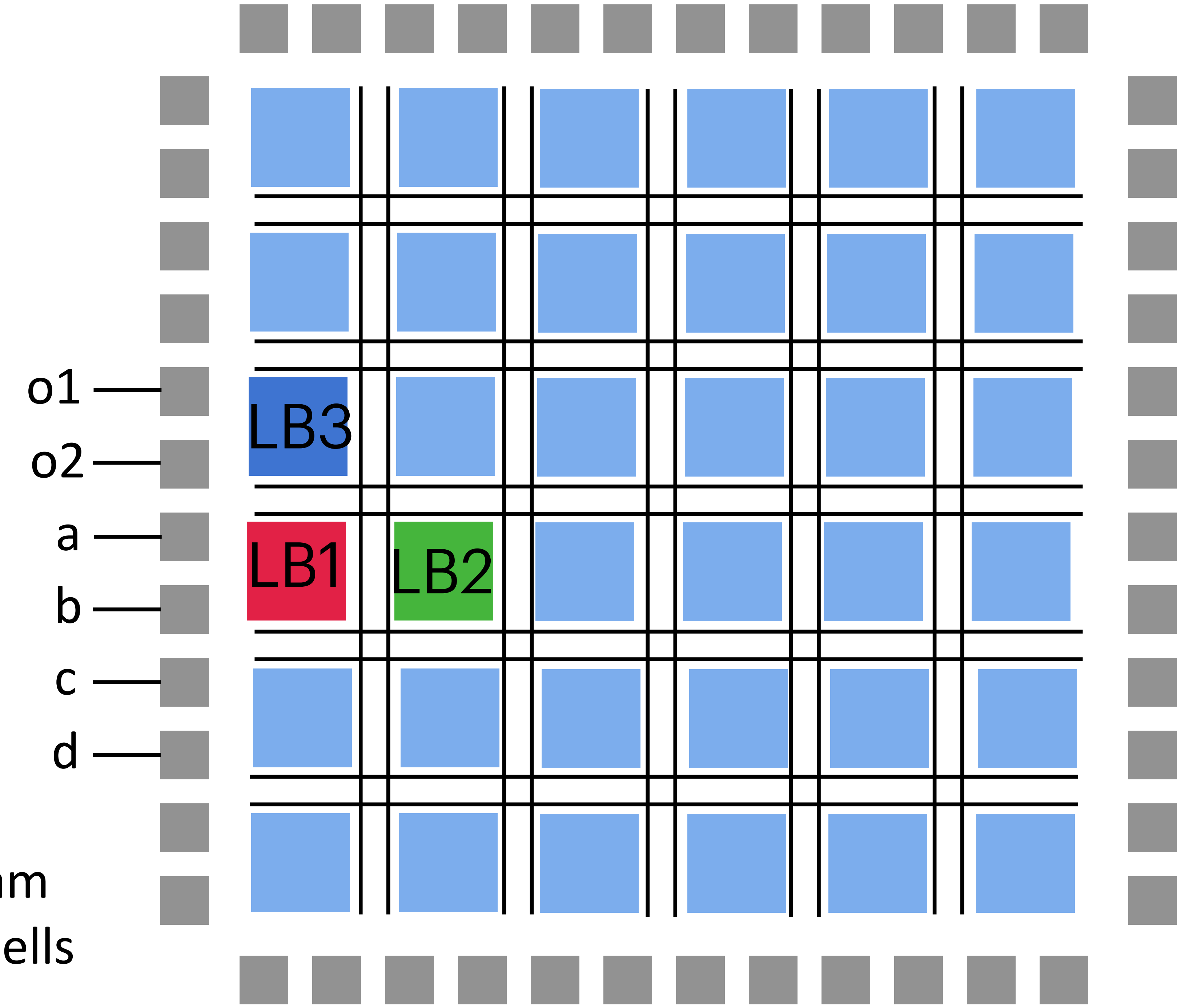
FPGA Mapping





FPGA Placement & Routing





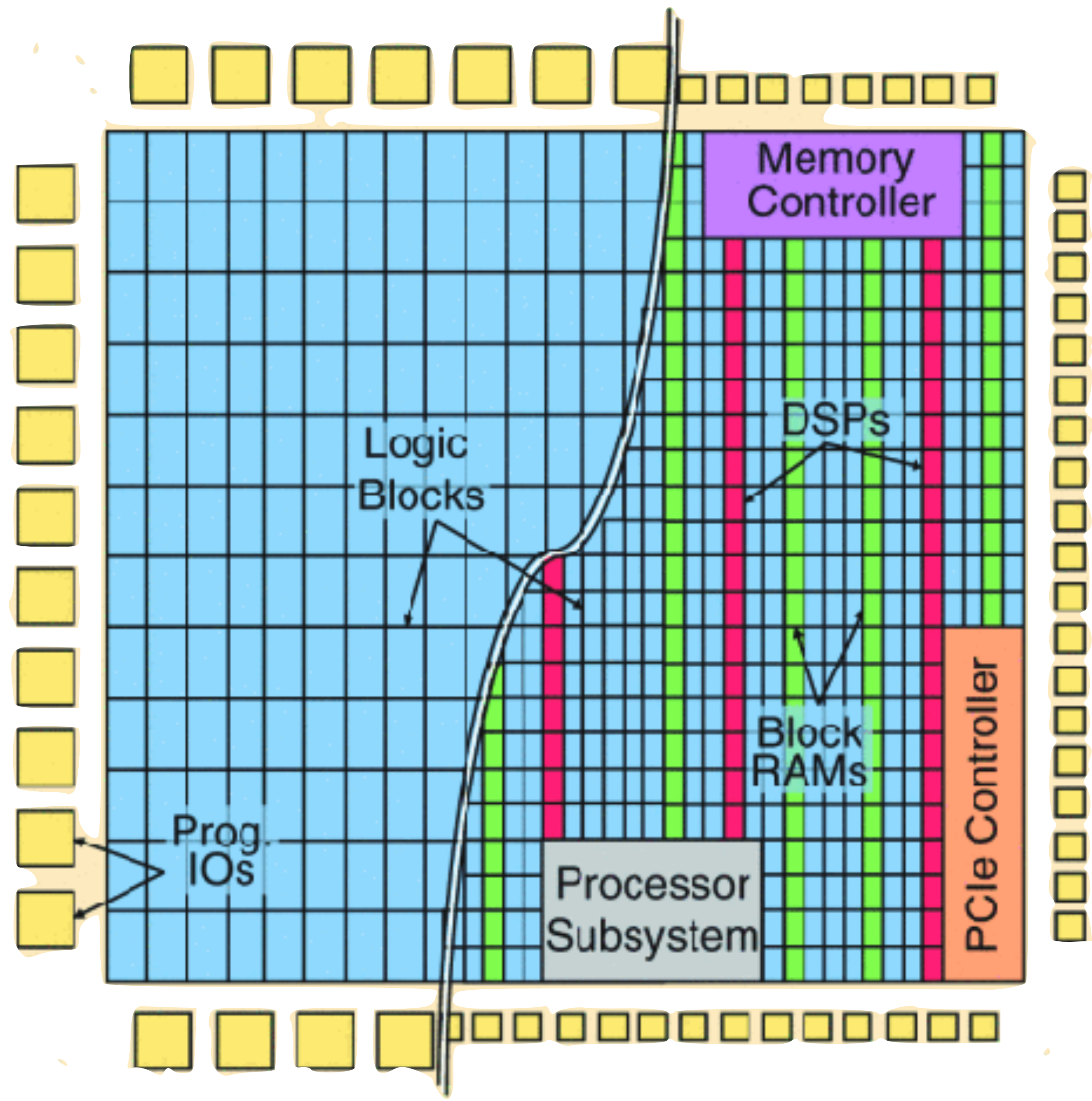
After that, generate bitstream file to configure the SRAM cells

Modern FPGAs

- More like SoC (system-on-chip)
- Logic blocks (functional blocks), DSP slices, block/distributed RAM, I/O and even embedded CPUs (usually ARM core) & GPUs

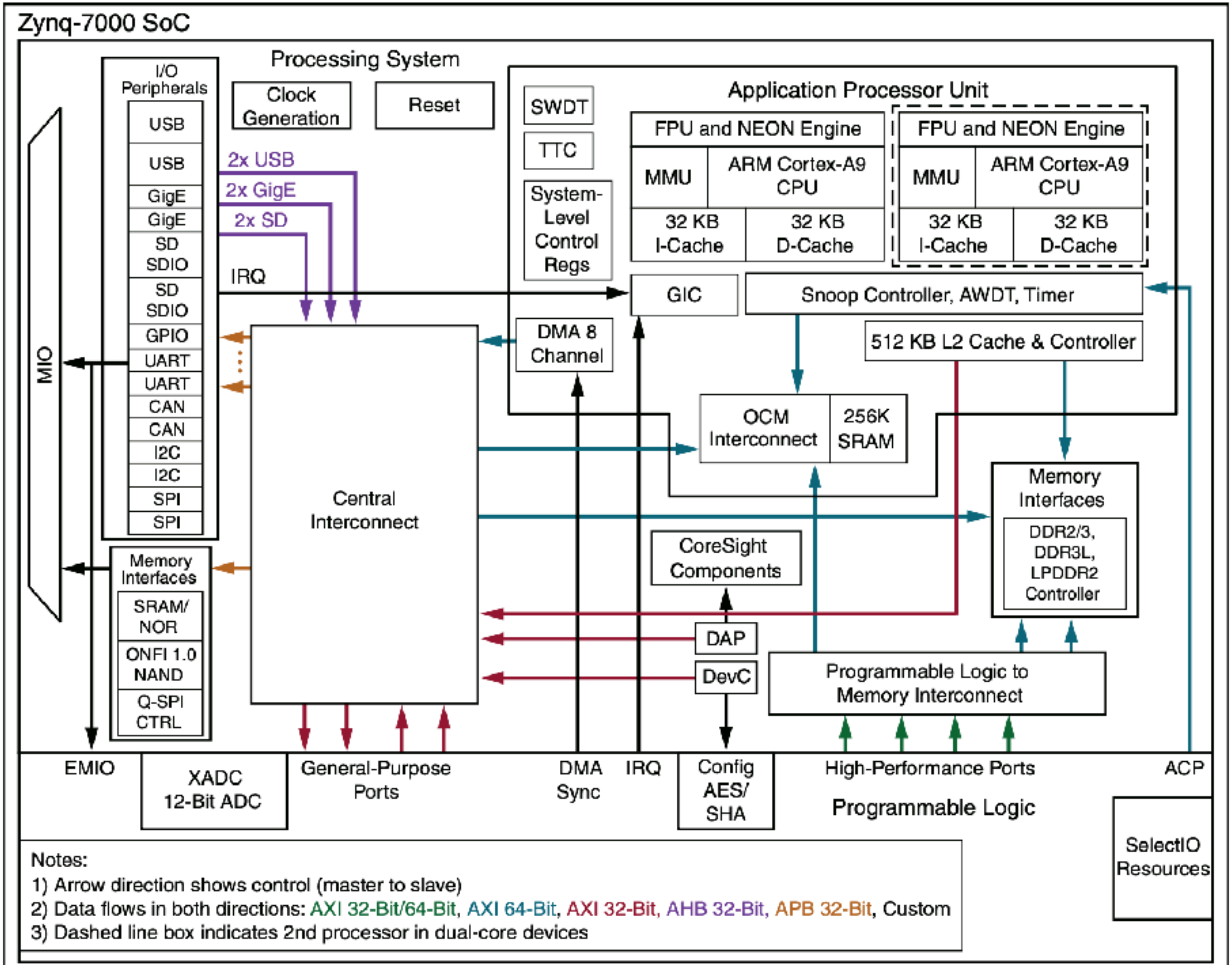
Heterogeneous

Traditional FPGAs



Modern FPGAs

AMD/Xilinx



Zynq-7000 SoC
Processing System
I/O Peripherals
USB
USB
GigE
GigE
SD SDIO
SD SDIO
GPIO
UART
UART
CAN
CAN
I2C
I2C
SPI
SPI
Clock Generation
Reset
2x USB
2x GigE
2x SD
IRQ
MIO
Memory Interfaces
SRAM/ NOR
ONFI 1.0 NAND
Q-SPI CTRL
Central Interconnect
Application Processor Unit
SWDT
TTC
System-Level Control Regs
FPU and NEON Engine
MMU
ARM Cortex-A9 CPU
32 KB I-Cache
32 KB D-Cache
FPU and NEON Engine
MMU
ARM Cortex-A9 CPU
32 KB I-Cache
32 KB D-Cache
GIC
Snoop Controller, AWDT, Timer
DMA 8 Channel
512 KB L2 Cache & Controller
OCM Interconnect
256K SRAM
Memory Interfaces
DDR2/3, DDR3L, LPDDR2 Controller
CoreSight Components
DAP
DevC
Programmable Logic to Memory Interconnect
EMIO
XADC 12-Bit ADC
General-Purpose Ports
DMA Sync
IRQ
Config AES/ SHA
High-Performance Ports
ACP
Programmable Logic
SelectIO Resources
Notes:
1) Arrow direction shows control (master to slave)
2) Data flows in both directions: AXI 32-Bit/64-Bit, AXI 64-Bit, AXI 32-Bit, AHB 32-Bit, APB 32-Bit, Custom
3) Dashed line box indicates 2nd processor in dual-core devices

Question: True or False

- Given enough resources (LUTs, logic blocks and RAMs), an FPGA can implement a RISC-V CPU (e.g., RV32I).

Hardware Description Language

- A way to document the hardware design, which has become IEEE standard (Verilog HDL & VHDL).

```
module alu(opA, opB, aluop, result, zero);
parameter width=32;
input [1:0] aluop; input [width-1:0] opA, opB;
output reg [width-1:0] result; output reg zero;
always @(*) zero = (result == 0);
always @(opA, opB, aluop) begin
case (aluop) 0: result = opA + opB;
1: result = opA - opB;
2: result = opA & opB;
3: result = opA | opB;
default: result = 0;
endcase
end
endmodule
```

An ALU

Hardware Description Language

- A way to document the hardware design, which has become IEEE standard (Verilog HDL & VHDL).

A Regfile

```
module rf(reg1, reg2, wr, data, reg_wr, d1, d2, clk);
parameter reg_width=32, num_reg=32;
input [4:0] reg1, reg2, reg_wr;
input wr, clk; input [reg_width-1:0] data;
output reg [reg_width-1:0] d1, d2;
reg [reg_width-1:0] reg_file [0:num_reg-1];

always @(*)
begin
d1 = reg_file[reg1];
d2 = reg_file[reg2];
end
always @(posedge clk)
begin
if (wr)
reg_file[reg_wr] <= data;
end
initial $readmemh("rf.txt",reg_file);
endmodule
```

Hardware Description Language

- A way to document the hardware design, which has become IEEE standard (Verilog HDL & VHDL).
- Can be used both for FPGA design & ASIC design.
- New HDLs like SpinalHDL & Chisel HDL

HDL vs. Software PL

Hardware	Software
Concurrent execution of tasks. This demands all tasks and events to operate in coherence with a timing reference signal called clock	Sequential execution of tasks and instructions. There is no concept of synchronization to clock reference
Very fast execution. Functional timing in nanosecond scale units is achievable in hardware. And therefore, time critical functions are designed to be in hardware	Slow execution. Minimum timing resolution is 100s of microsecond
Can be parallel	Sequential though it can appear to be parallel for the user
Physical and costs are exorbitant if it has to be redone	Can be recompiled
Need to be first time success.	Can be corrected and recompiled without much effort
Hardware can be one time developed as platform and reused for lifetime if the functionality is the same	Can be redone easily.
Development from paper specification to physical system on chip	Need processing hardware platform for sw development
Need to verify fully imagining all scenario ahead of fabrication and hence verification and validation are unavoidable	Verification is necessary to prove the intent of the design but in the case of minor defects, it can be corrected.

Source: https://link.springer.com/book/10.1007%2F978-3-030-23049-4

FPGA vs. ASIC

Field-programmable gate array	Application-specific integrated circuit
Fast time-to-market	Low cost when production volumes are high
Reconfigured without costly mask changes	Higher efficiency
Testing done by FPGA vendor	High risk due to high cost of correcting design errors
Cannot exploit 100% the hardware	Greater design/verification/production/test cost (Engineering costs, masks, packaging)
Less protection against design theft	
HDL	Domain-specific language

[1] Bruce F. Cockburn and Jie Han. Chapter 1. Review of Classical Sequential Logic Design. ECE 511 2013.