



信息科学与技术学院

School of Information Science and Technology

# CS 110

# Computer Architecture

# Heterogeneous Computing & Summary

Instructors: Siting Liu & Yuan Xiao

Course website: <https://faculty.sist.shanghaitech.edu.cn/liust/courses/CS110.html>

School of Information Science and Technology (SIST)

ShanghaiTech University

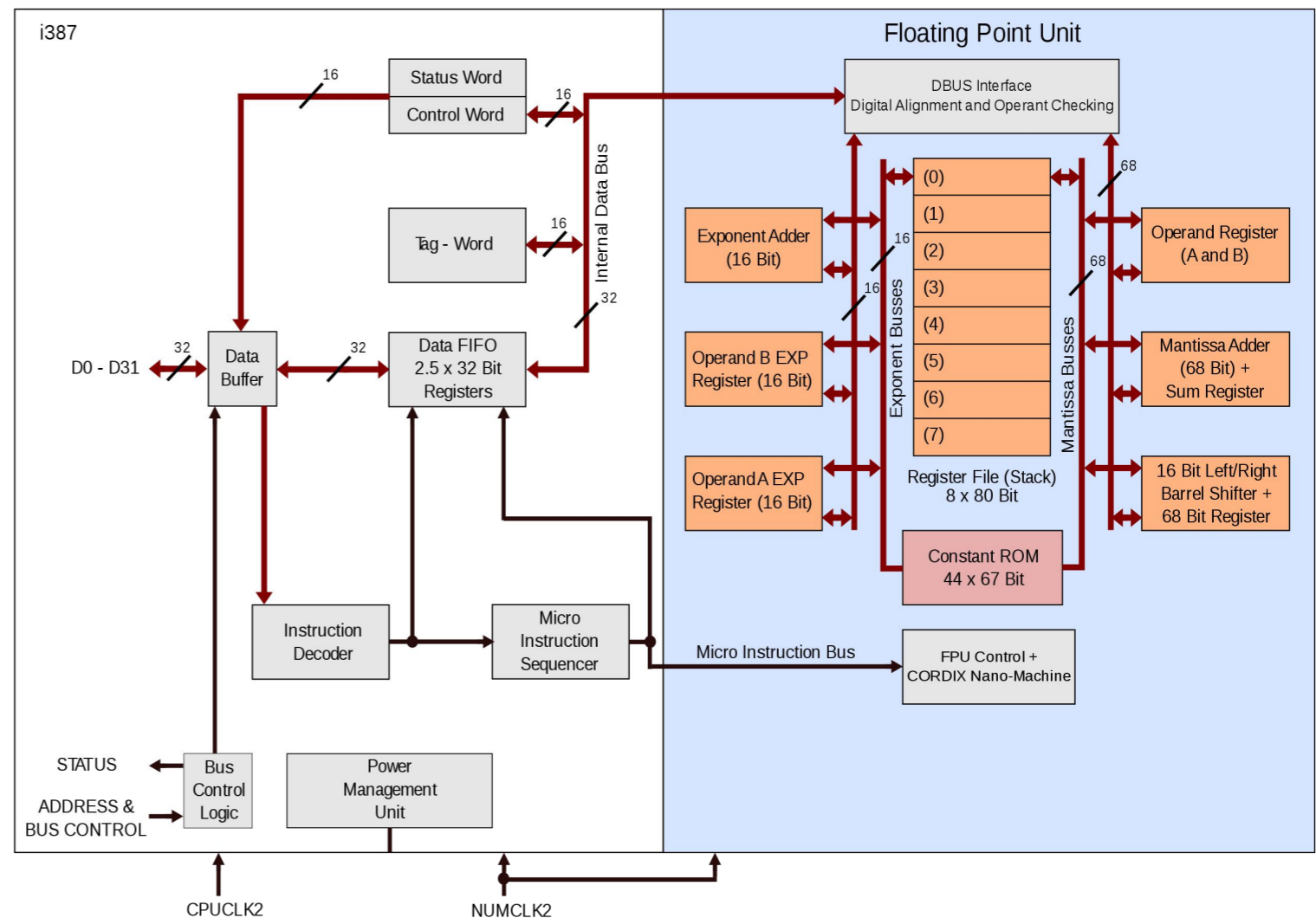
2026/6/18

# Outline

- Heterogeneous computing
  - Modern SoCs (accelerator-level parallelism)
  - Introduction to FPGA
- Summary

# Heterogeneous Computing

- Computing systems that use **more than one kind of processors or cores**.
- Usually by adding dissimilar coprocessors, incorporating specialized processing capabilities to handle particular tasks.
- Historically, Intel uses math-coprocessor to accelerate floating-point arithmetics.
- Otherwise, floating-point arithmetics are performed with integer assembly.
- More coprocessors with different microarchitecture emerges targeting certain domain (a.k.a. domain-specific architectures)



Intel 80387 microarchitecture from Wikipedia.

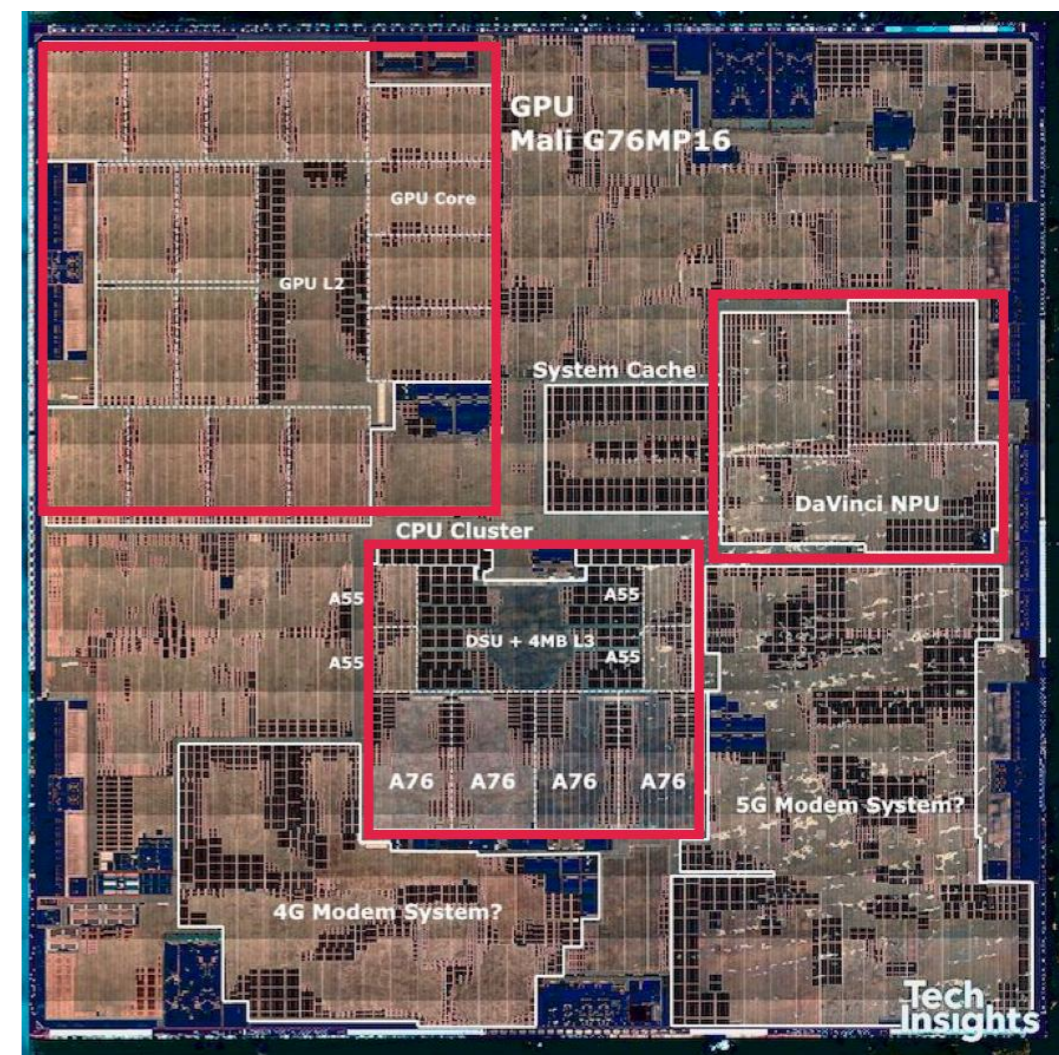
# Heterogeneous computing (SoC)

- “Coprocessor” is less mentioned since they are mostly integrated on-chip nowadays, leads to SoC (system-on-chip);
- The “coprocessor” in an SoC is often referred to as accelerator these days, accelerating certain applications or domain



Apple A15 SoC 107.68 mm<sup>2</sup>

<https://www.semianalysis.com/p/apple-m2-die-shot-and-architecture>



Huawei Kirin 990

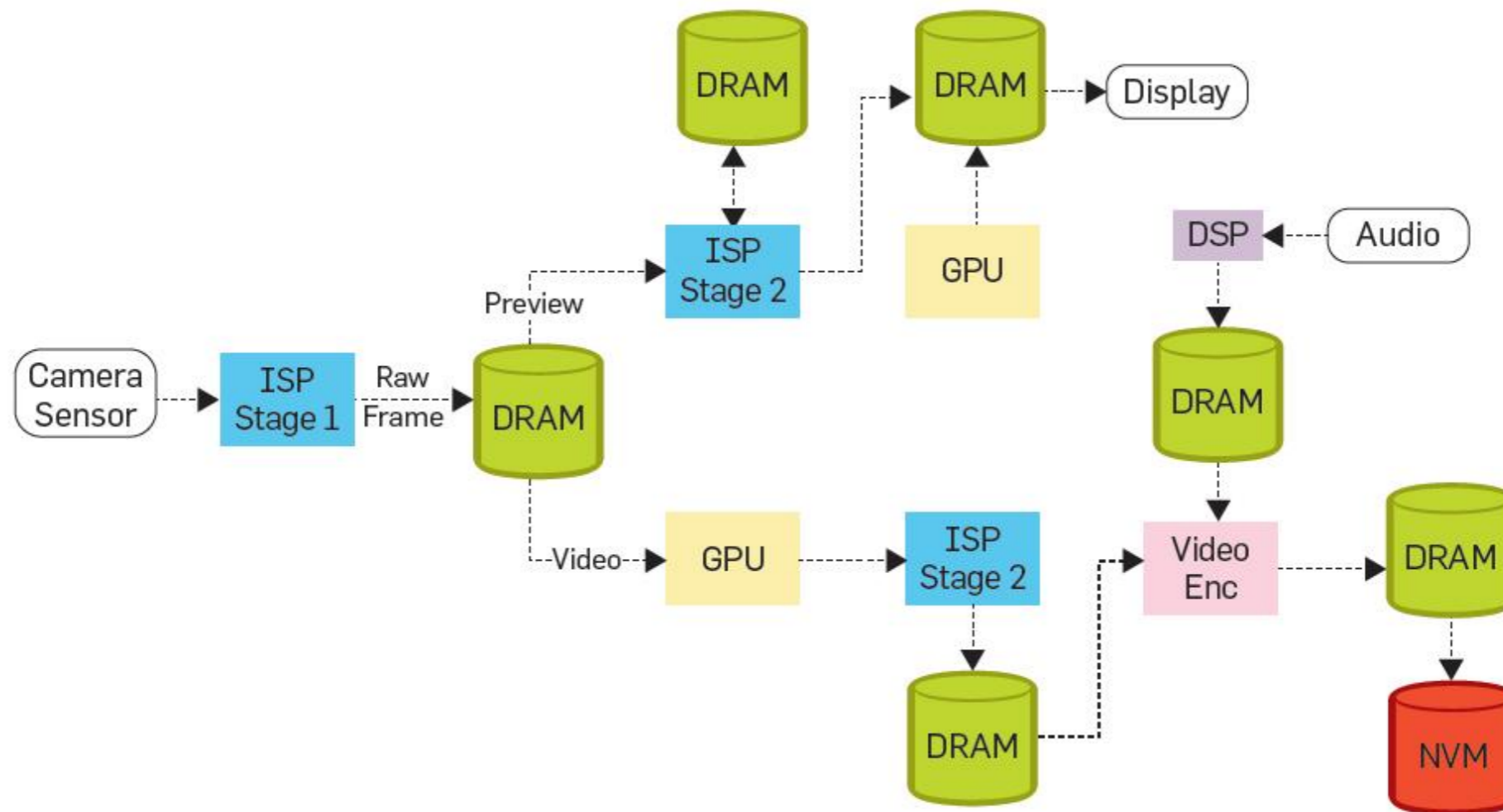
From Techinsights & AnandTech

- 2 performance & 4 efficient CPU cores
- 6 GPU cores
- 16 neural engine cores
- Hardware video encoder & decoder
- Image signal processor (ISP)
- ... ..

# Heterogeneous computing (SoC)

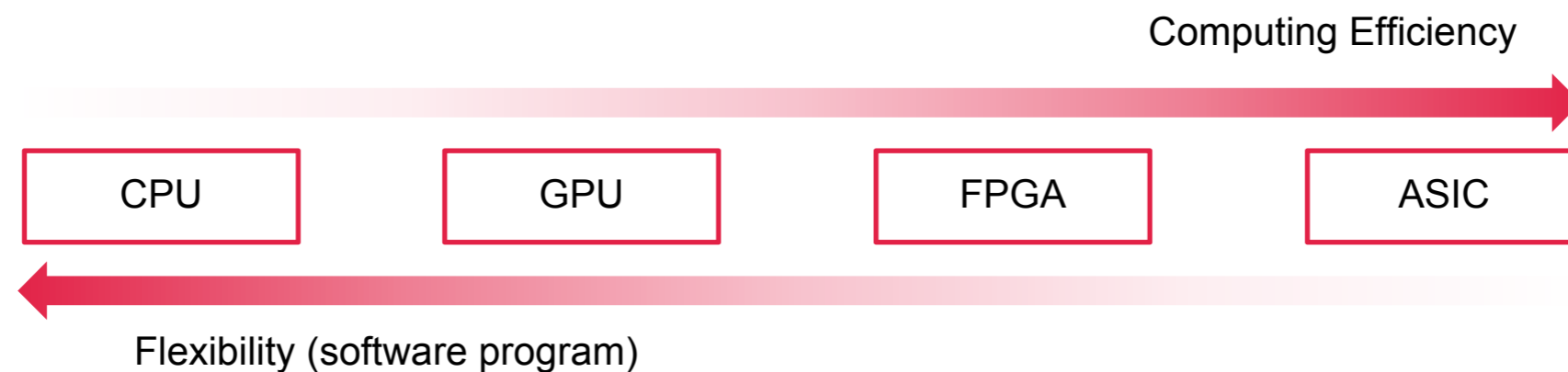
- Accelerator-level parallelism example

A smartphone performing video recording  
The usage of the SoC



# Heterogeneous Computing on Cloud

Providers	CPU	GPU	FPGA	ASIC (DSA)
Alibaba Cloud	X86/ARM/RISC-V	Nvidia/AMD	Intel (Altera)/AMD (Xilinx)	AliNPU
AWS (Amazon)	AWS Graviton (ARM)/X86	Nvidia/AMD	AMD (Xilinx)	AWS Trainium
Google Cloud	X86	Nvidia	N/A	TPU
Huawei Cloud	Kunpeng (ARM)/X86	Nvidia & Ascend	AMD (Xilinx)	Ascend



# Field programmable gate array (FPGA)

- Motivation
  - Make common case fast (via hardware accelerators)
  - But, it is **expensive** to make chips (NRE cost)
  - **FPGA: general-purpose hardware accelerator, hardware reconfigurable**
- Used to implement accelerator for certain algorithms, e.g.,
  - Microsoft Bing search engine
  - High-frequency trading
  - Communication systems (encoders and decoders)
  - Many embedded systems
- Also used for fast prototype of digital ICs

# FPGA vs. CPU

- Logics are described by hardware description language (HDL), and mapped to LUT (look-up table), which is the basic building block in an FPGA
- Instruction-based serial execution of programs

```
int sum=a+b;
```

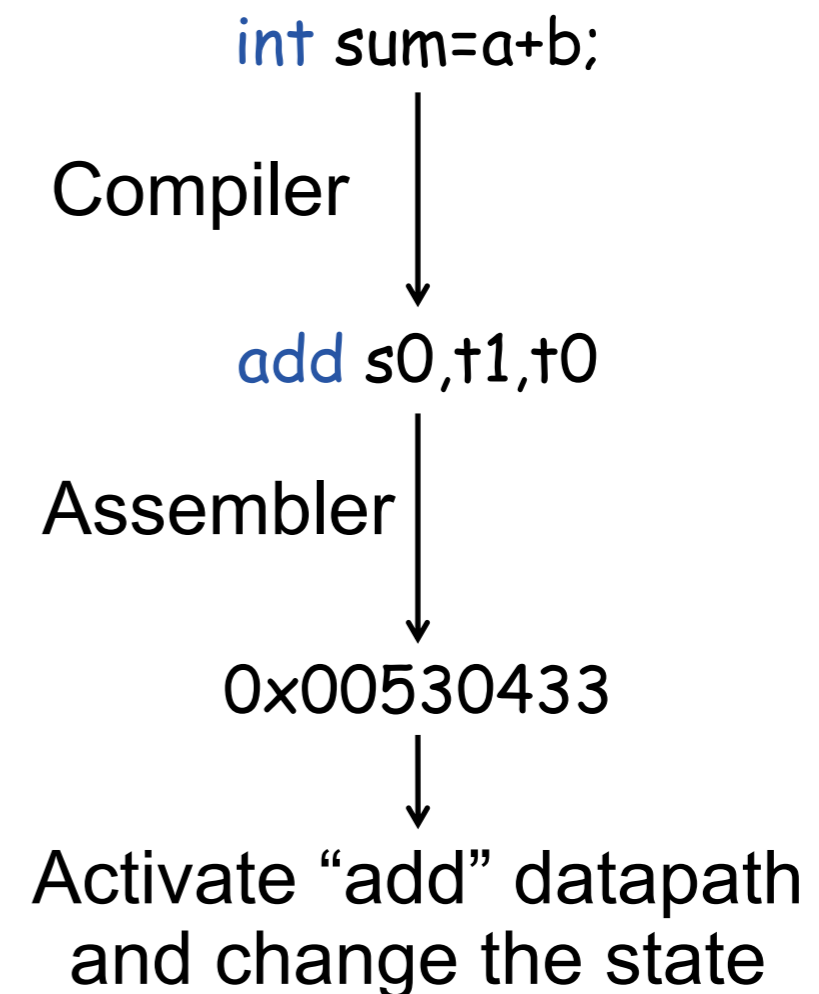
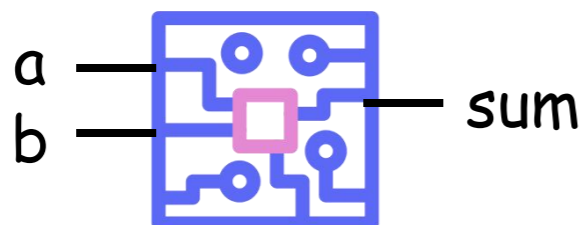
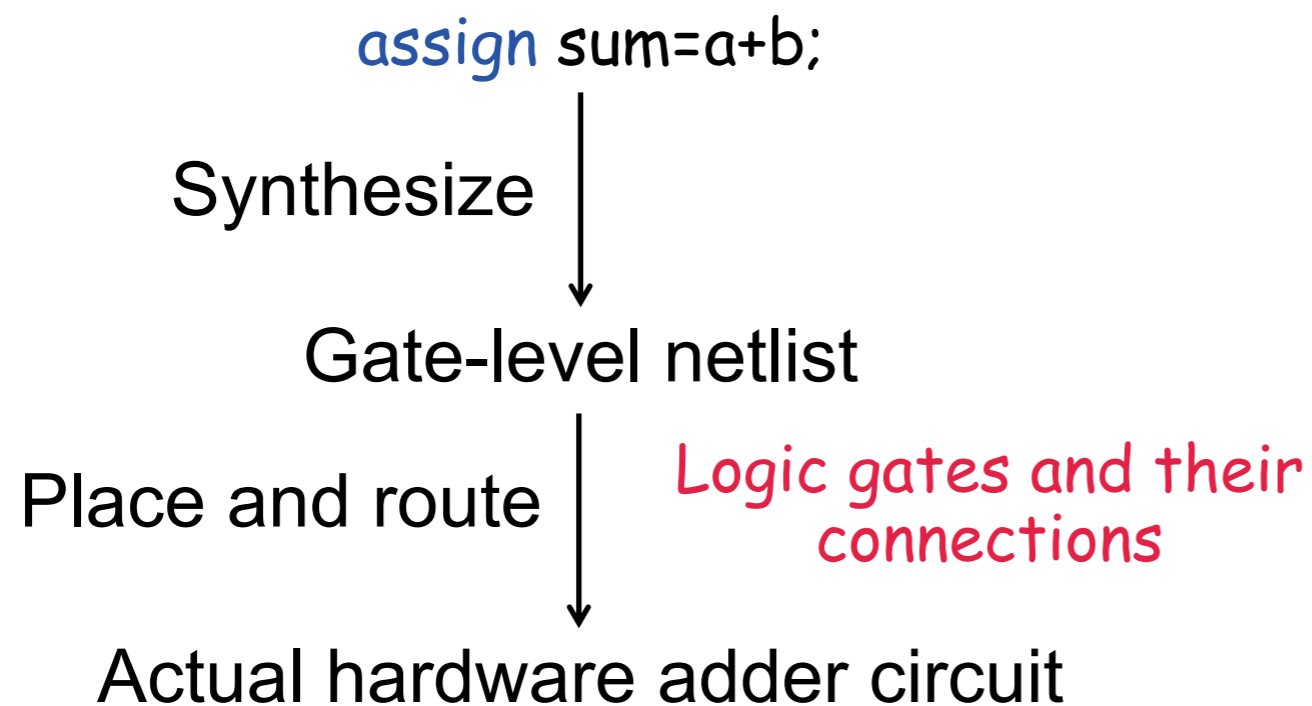
Verilog HDL example of a 2-bit adder (structural model)

```
module fa2(input [1:0] a, input [1:0] b, output [1:0] sum);  
  wire [2:0] c;  
  wire [1:0] s;  
  fa fa1(.cout(c[1]), .sum(s[0]), .a(a[0]), .b(b[0]), .cin(c[0]));  
  fa fa2(.cout(c[2]), .sum(s[1]), .a(a[1]), .b(b[1]), .cin(c[1]));  
  assign sum=s;  
  assign c[0]=1'b0;  
endmodule
```

or directly (behavioral model) `assign sum=a+b;`

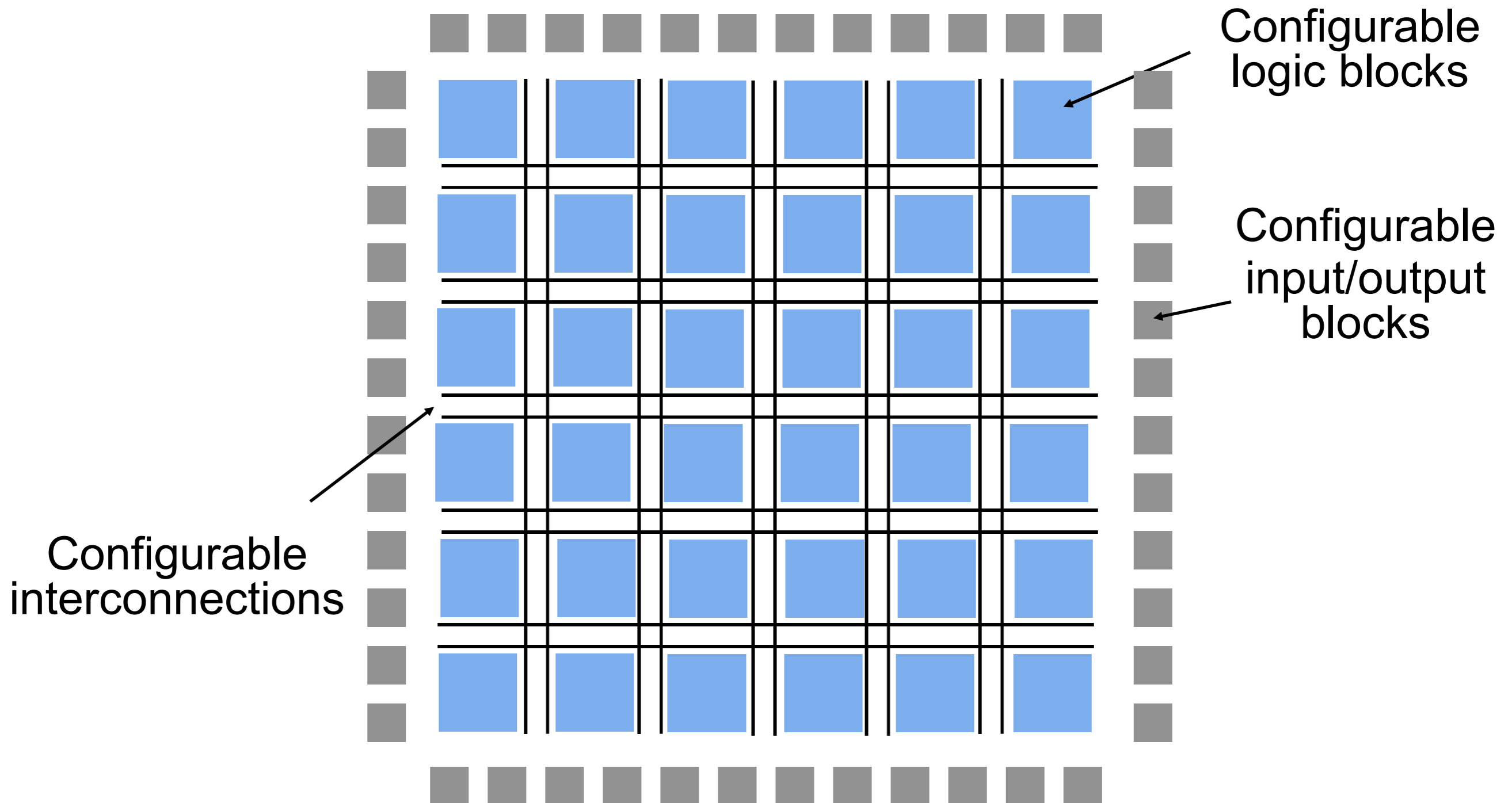
# FPGA vs. CPU

- Logics are described by hardware description language (HDL), and mapped to LUT (look-up table), which is the basic building block in an FPGA
- Instruction-based serial execution of programs

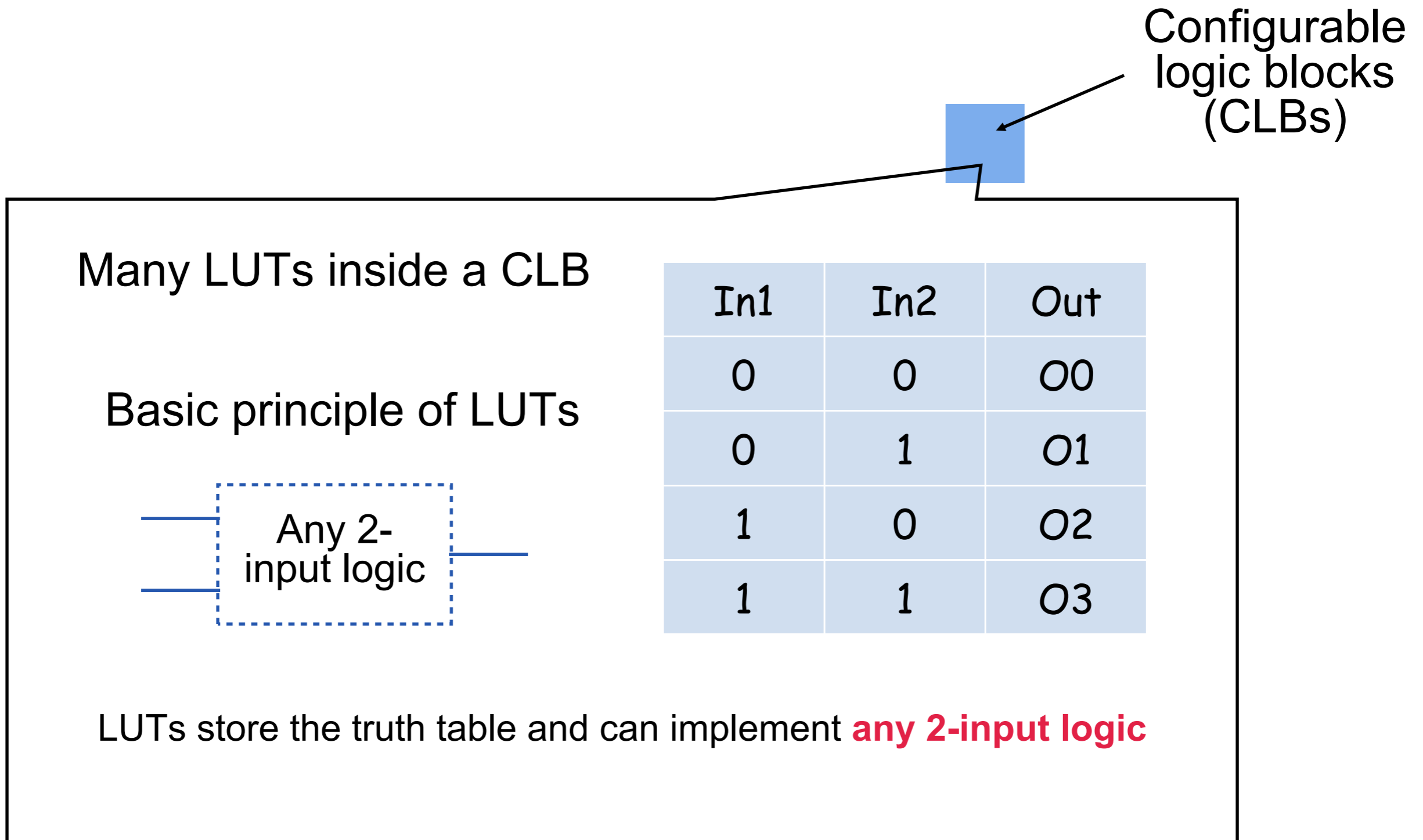


# FPGA implements any logics

(given enough hardware resources)



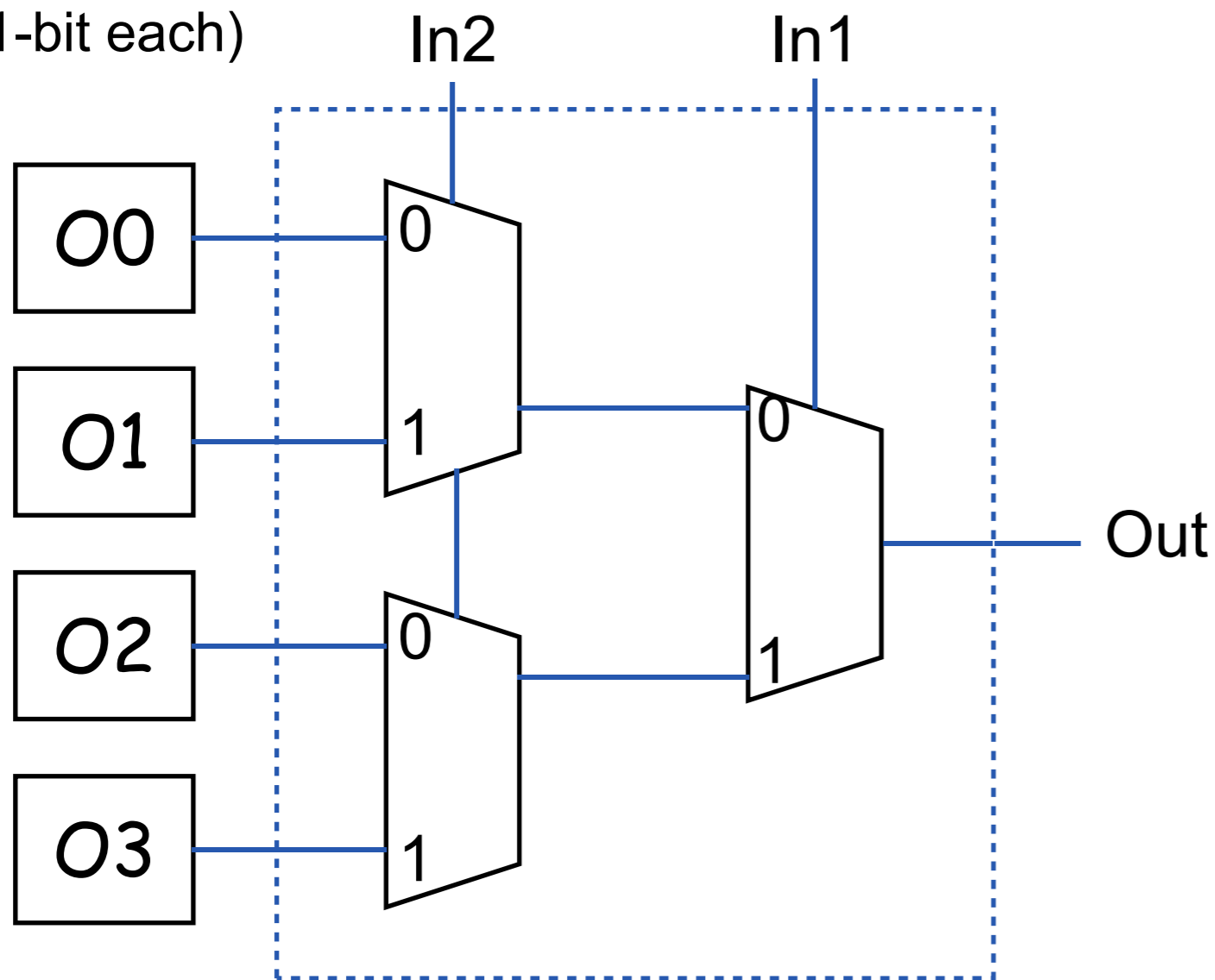
# LUTs inside configurable logic blocks



# LUTs inside configurable logic blocks

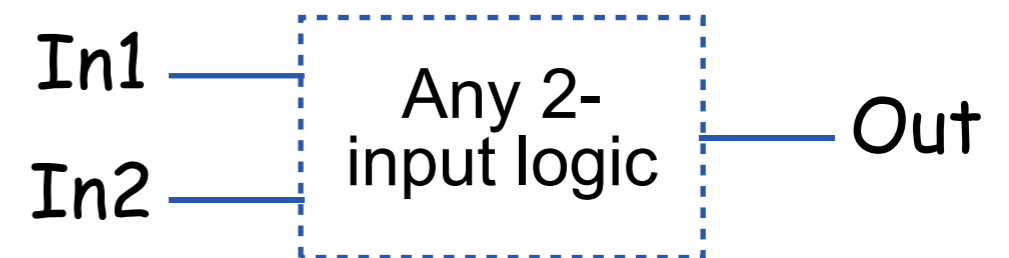
- Look-up table (LUT)

SRAM cells  
(1-bit each)



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

Equivalent to

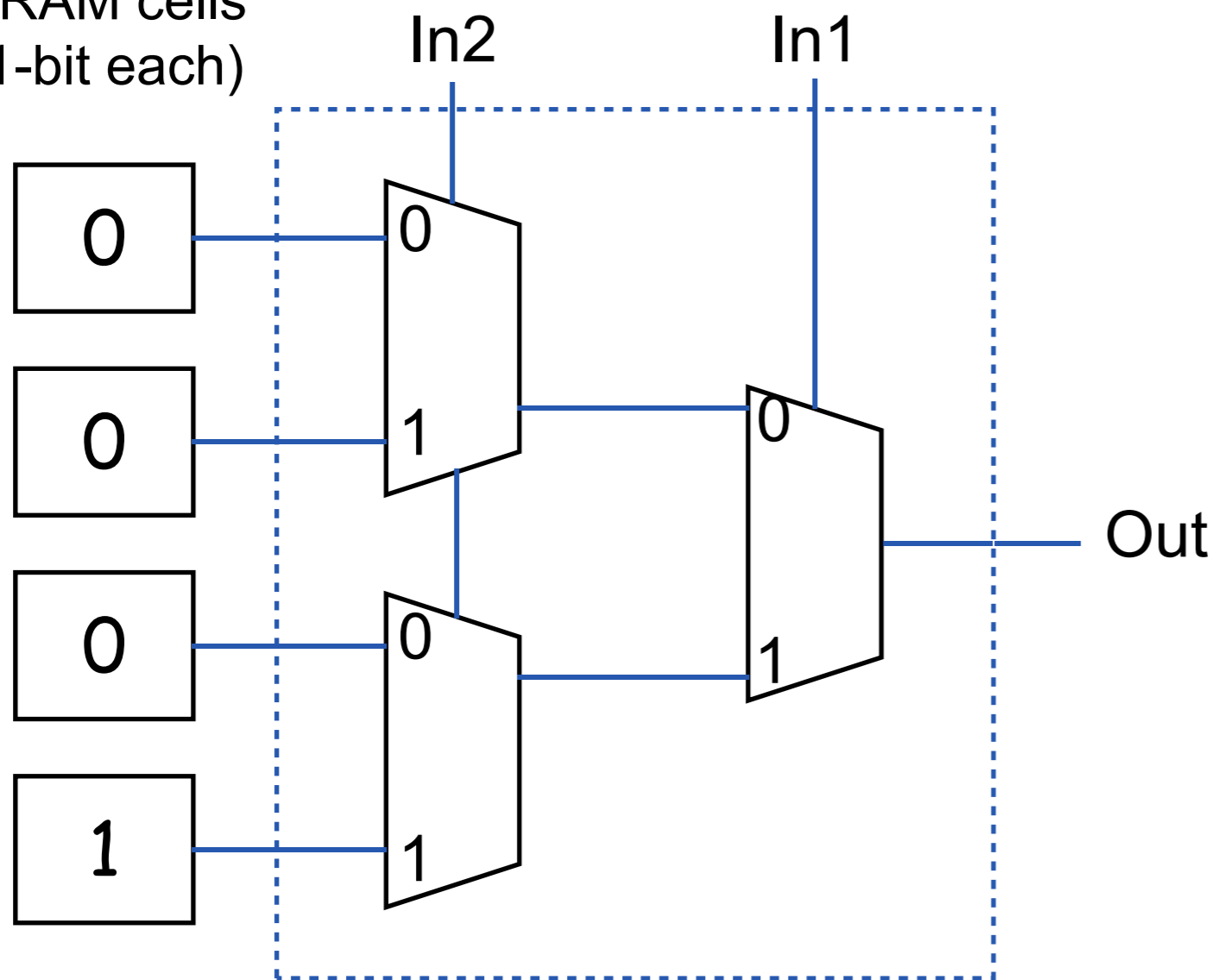


- Similarly, we can build n-input LUT implementing any n-input logic with  $2^n$  single-bit SRAM cells

# LUTs inside configurable logic blocks

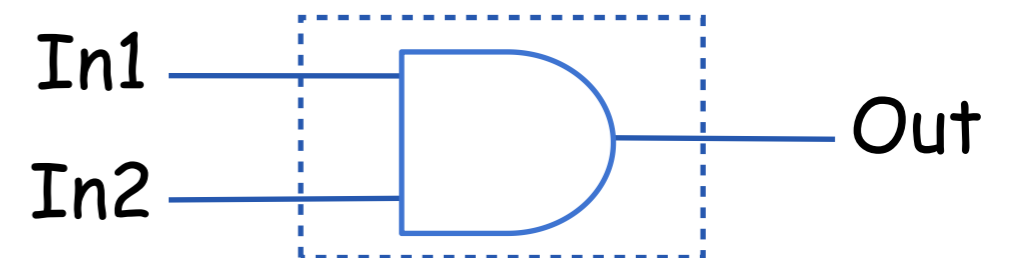
- Example, 2-input LUT implements an AND gate

SRAM cells  
(1-bit each)



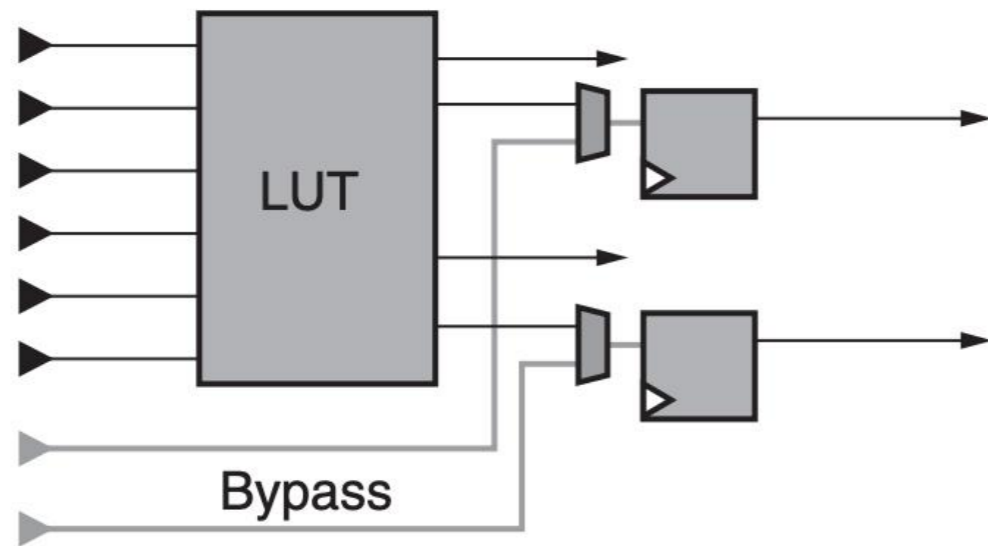
In1	In2	Out
0	0	0
0	1	0
1	0	0
1	1	1

Equivalent to



# LUTs inside configurable logic blocks

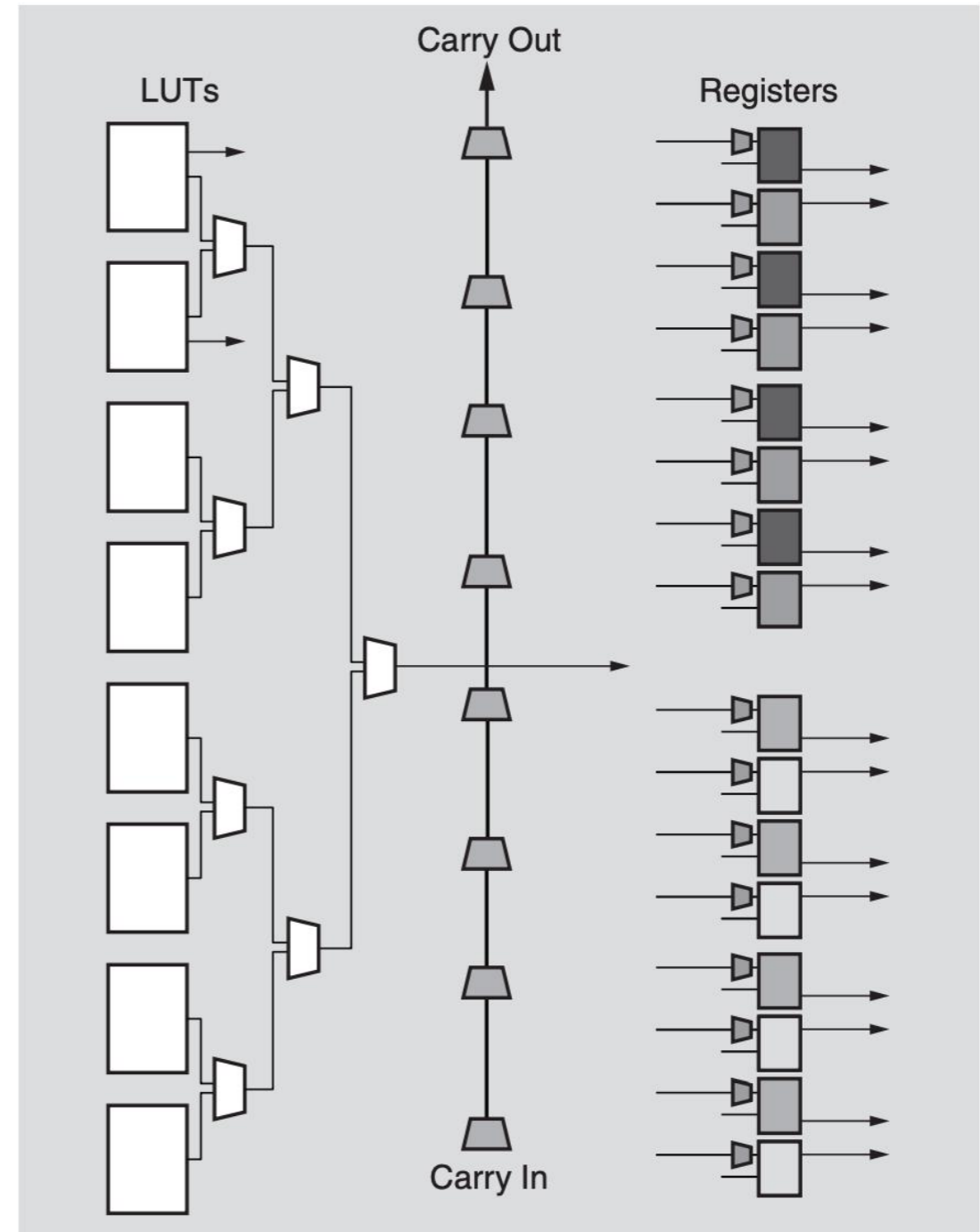
- In reality: LUT with larger number of inputs are more capable



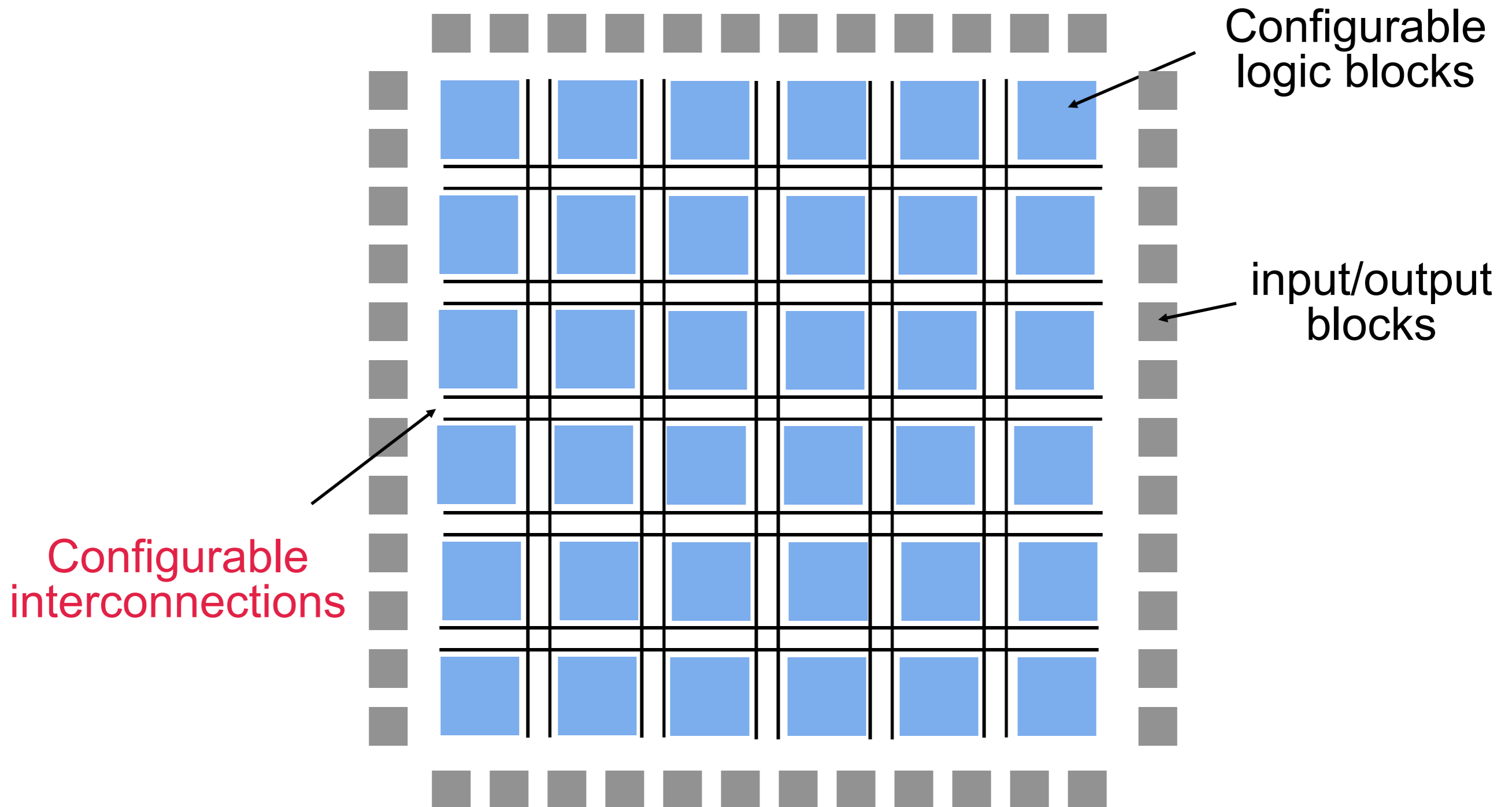
6-input LUT with bypass path & registers

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

Images from AMD



# Configurable interconnections



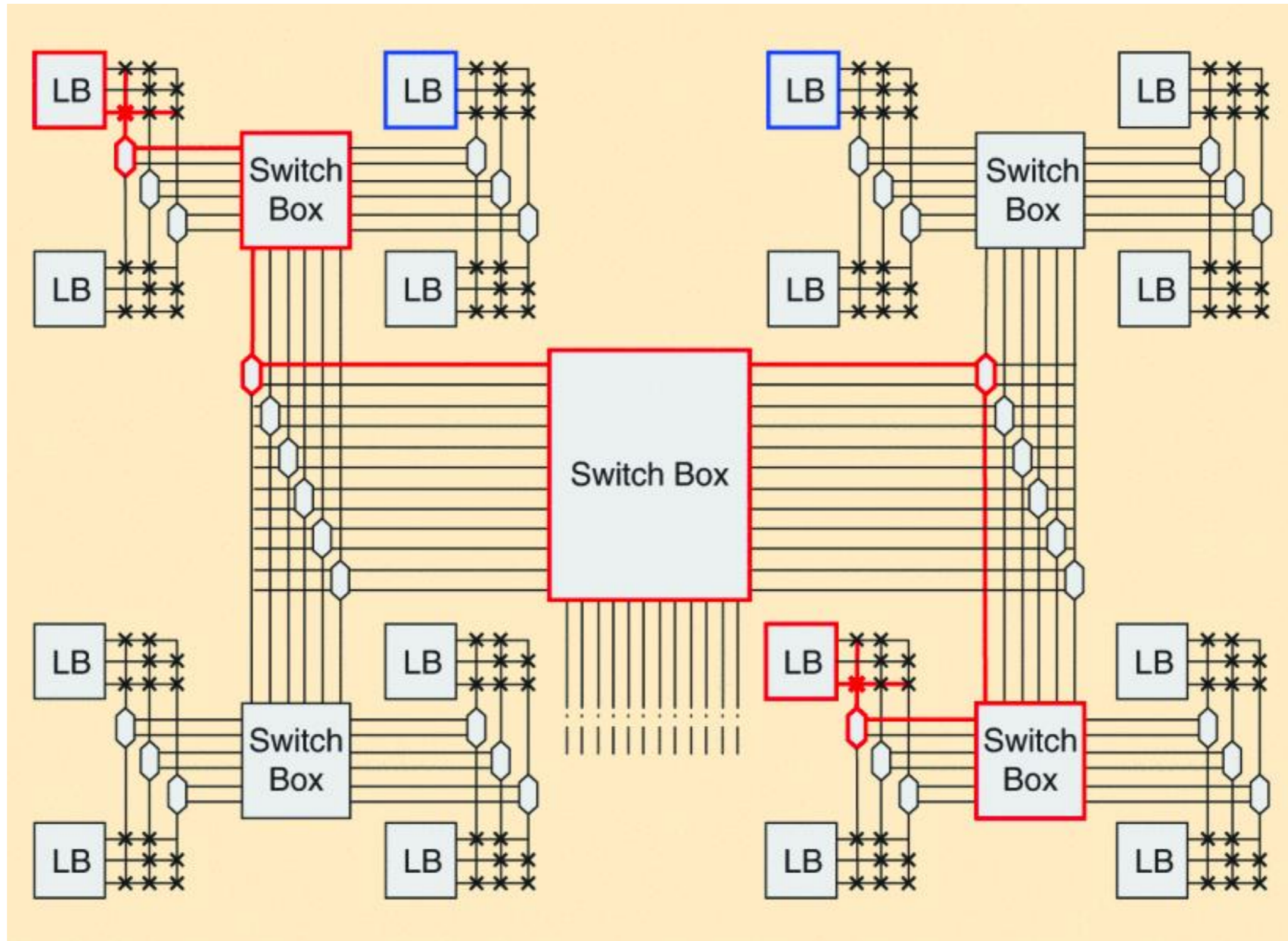
# Configurable interconnections

- Routing, a.k.a., interconnecting
  - Through programmable wires and switches
  - Between logic blocks (CLBs), 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.

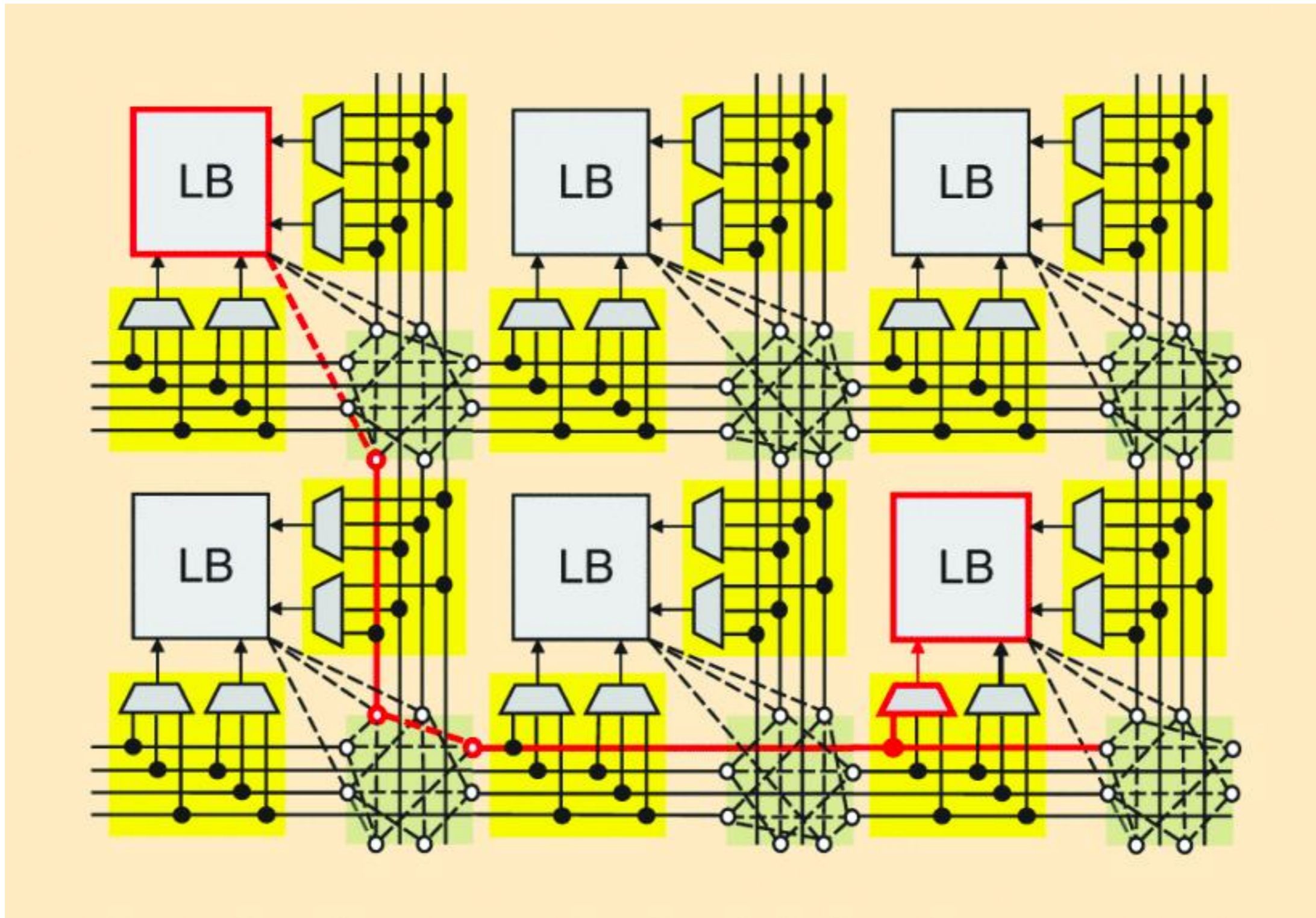
# Configurable interconnections

- Routing, a.k.a., interconnecting
  - Through programmable wires and switches
  - Between logic blocks (CLBs), 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

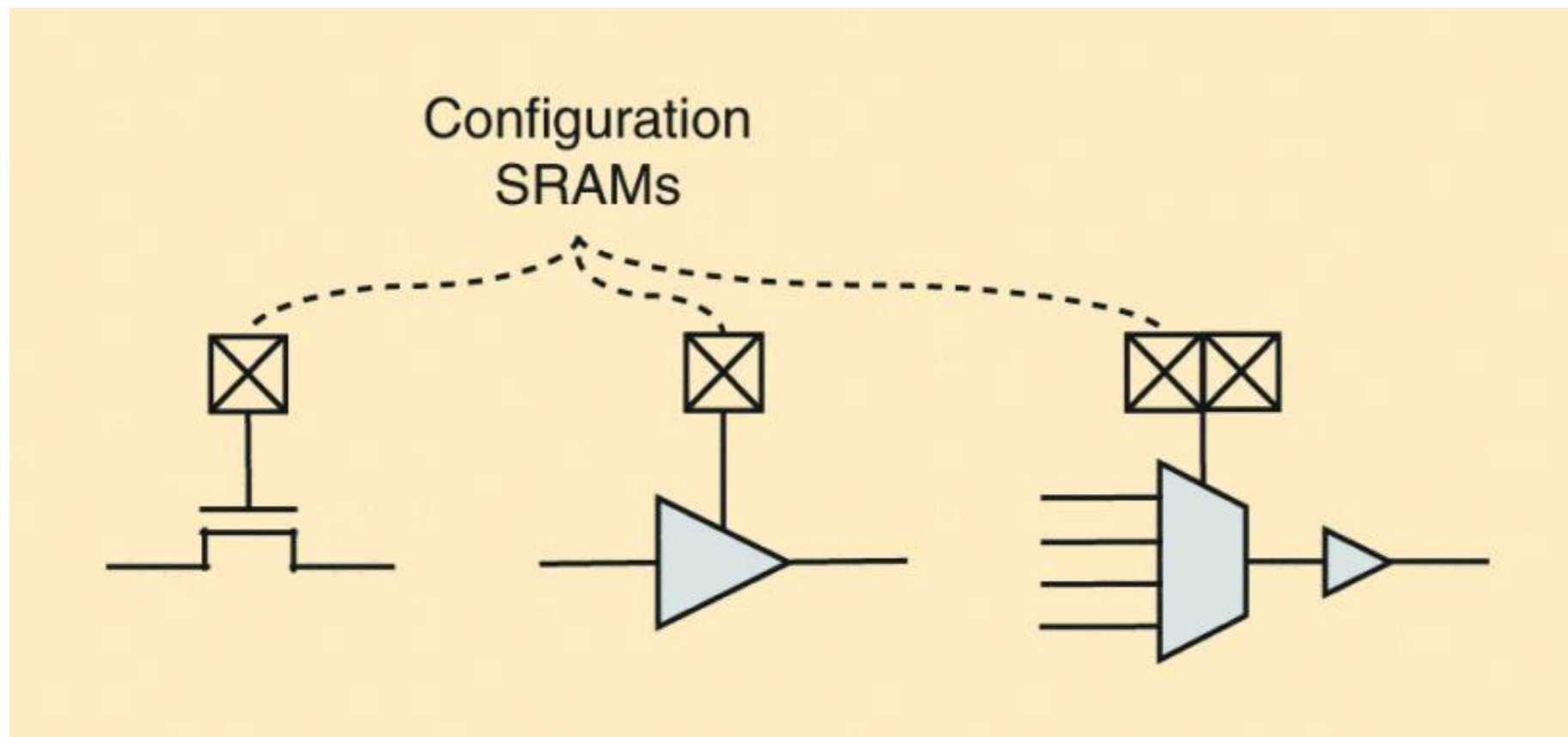


# Island-style FPGA

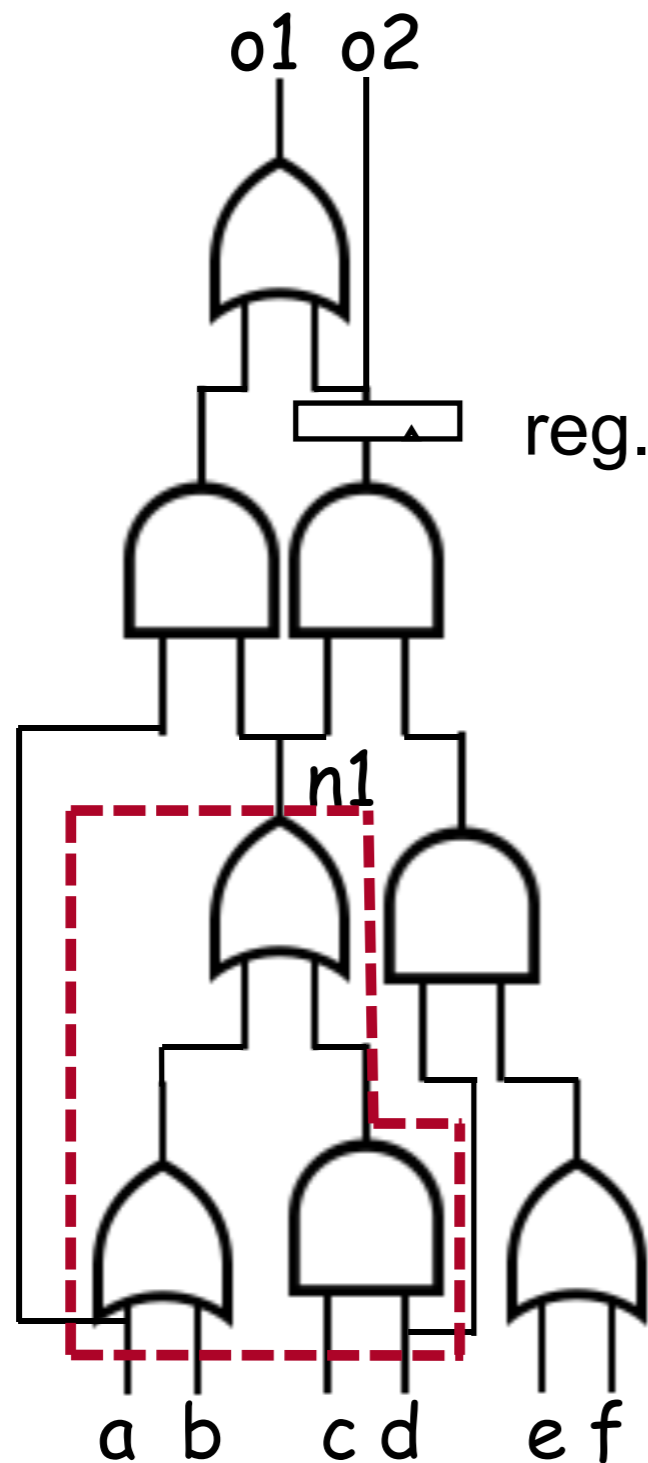


A. Boutros and V. Betz, "FPGA Architecture: Principles and Progression," in *IEEE Circuits and Systems Magazine*, vol. 21, no. 2, pp. 4-29, Secondquarter 2021.

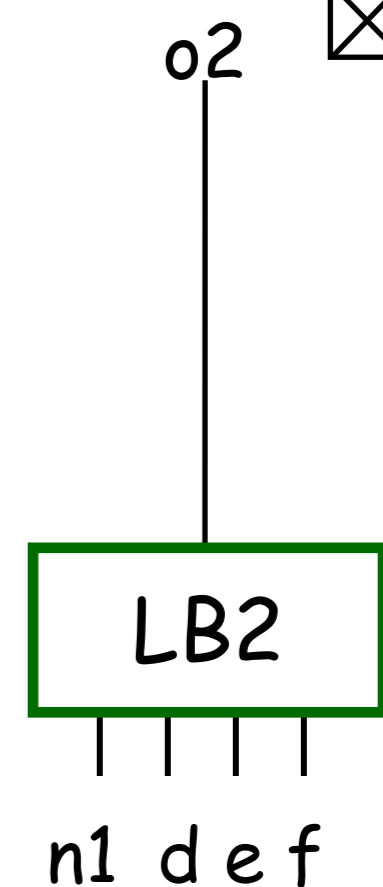
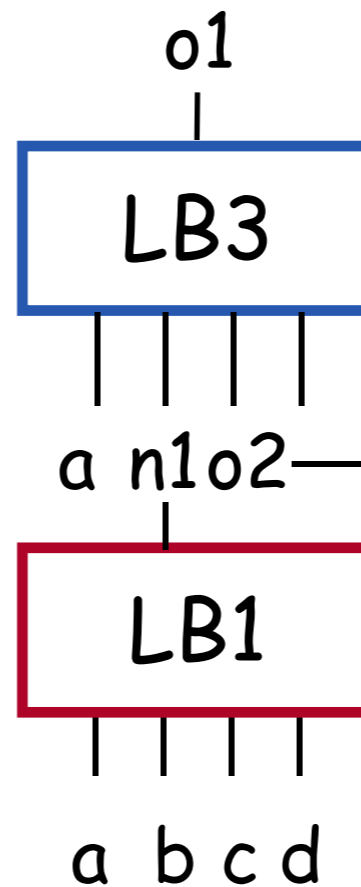
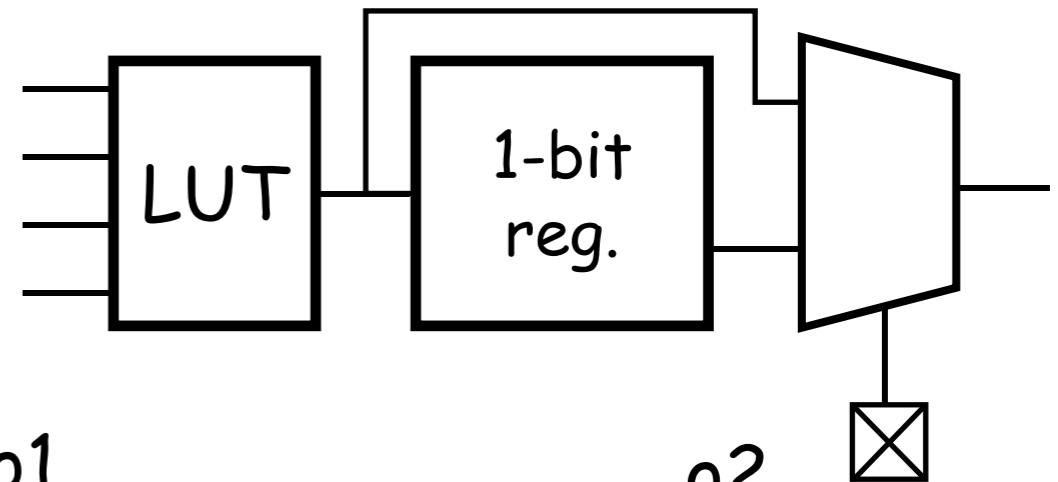
# Programmable switches



# FPGA mapping

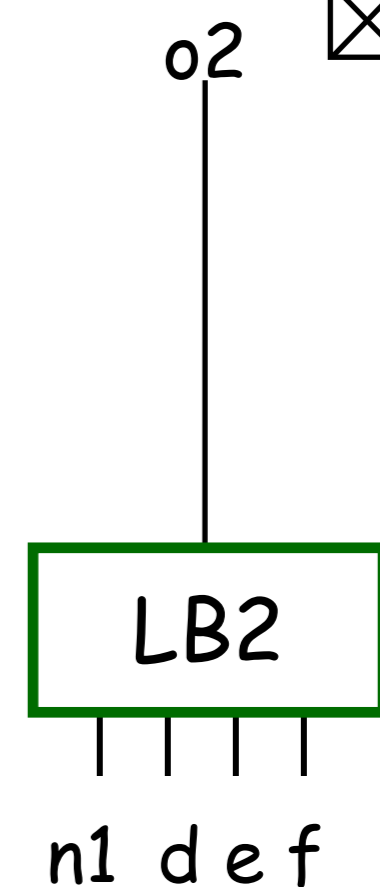
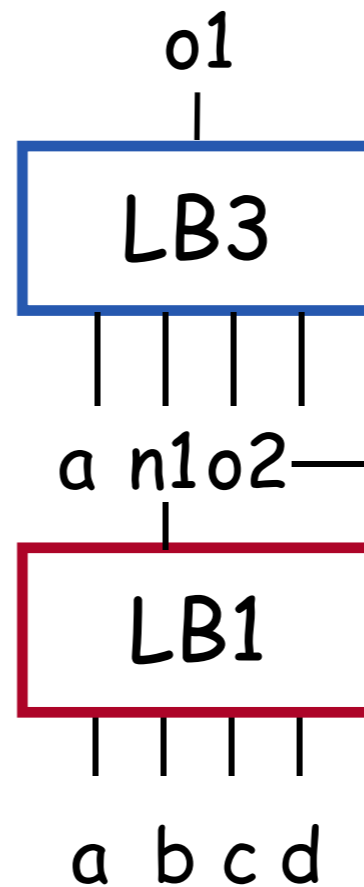
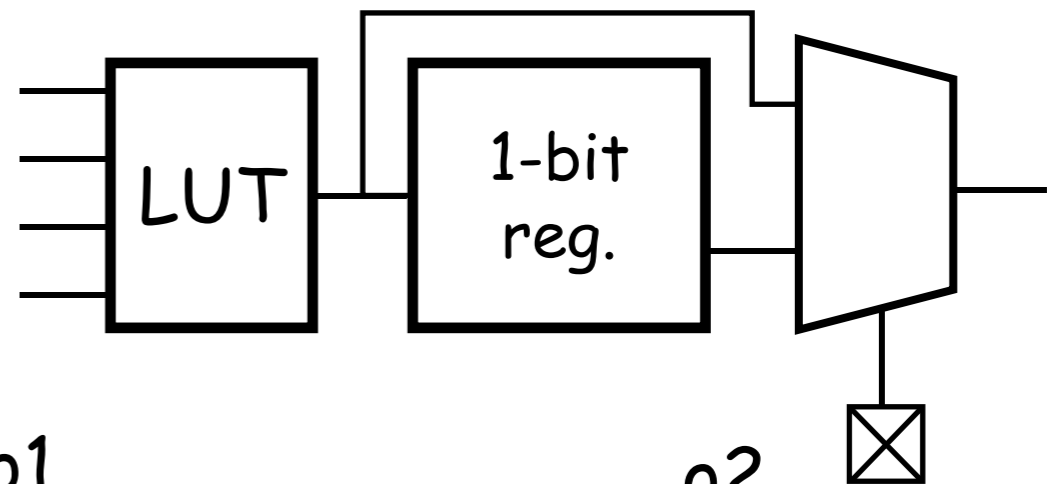
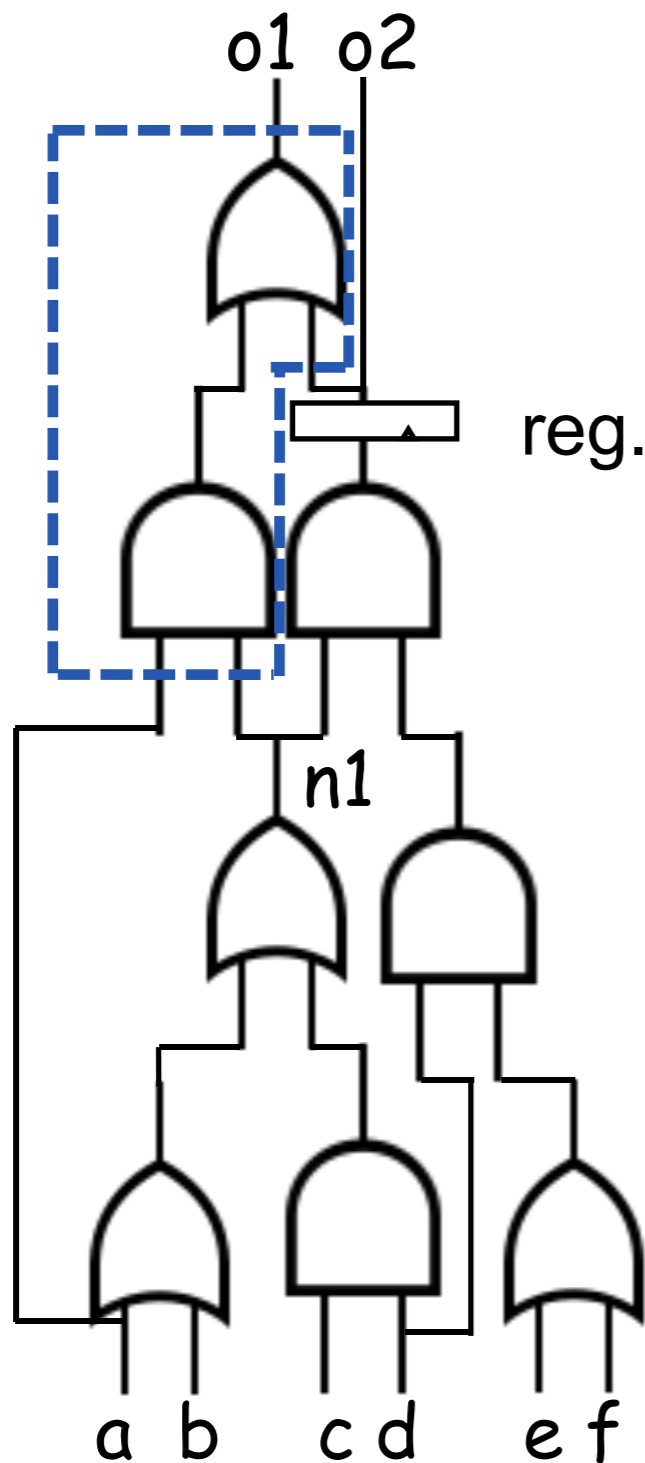


- Assume we use 4-input LUT, each followed by a 1-bit register/bypass path

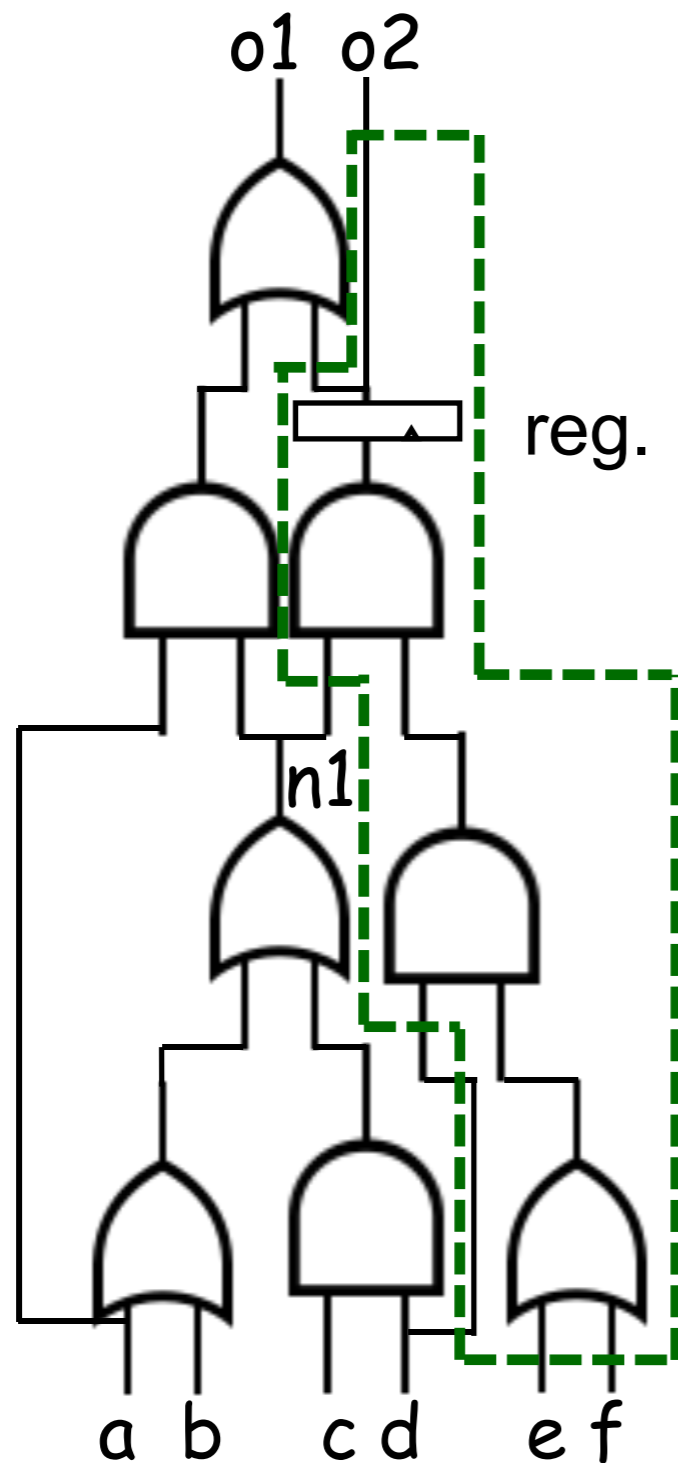


# FPGA mapping

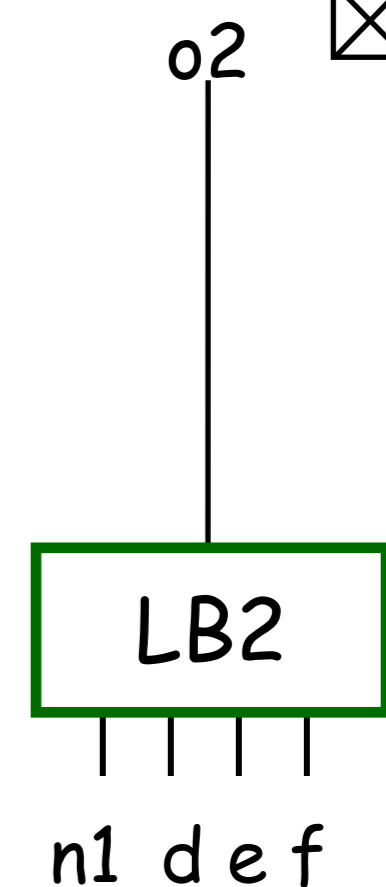
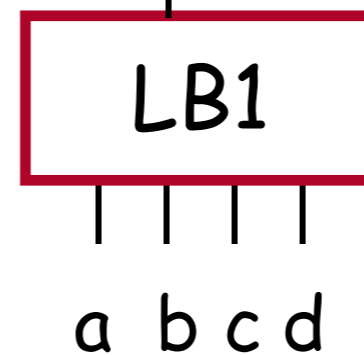
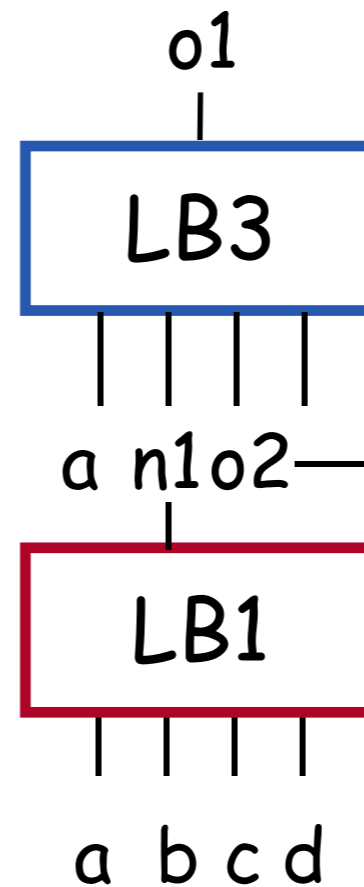
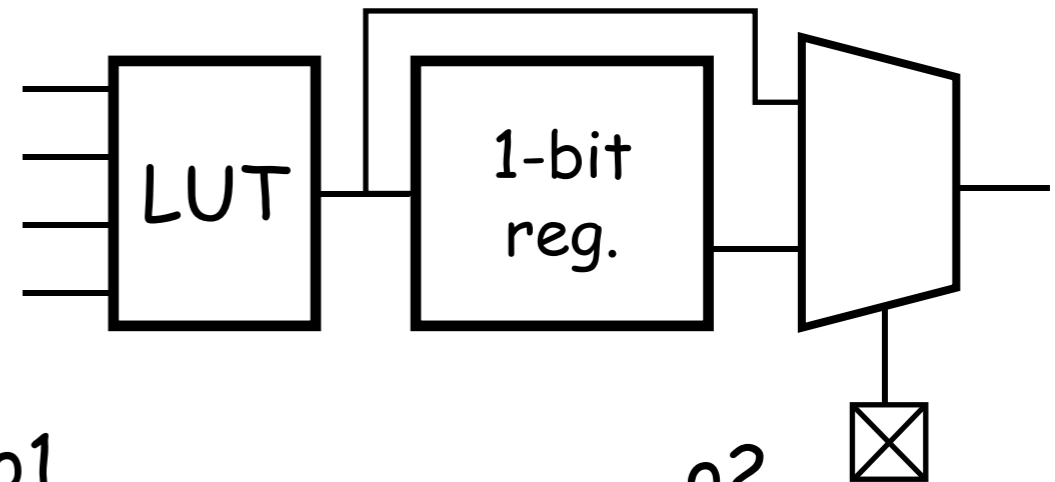
- Assume we use 4-input LUT, each followed by a 1-bit register/bypass path



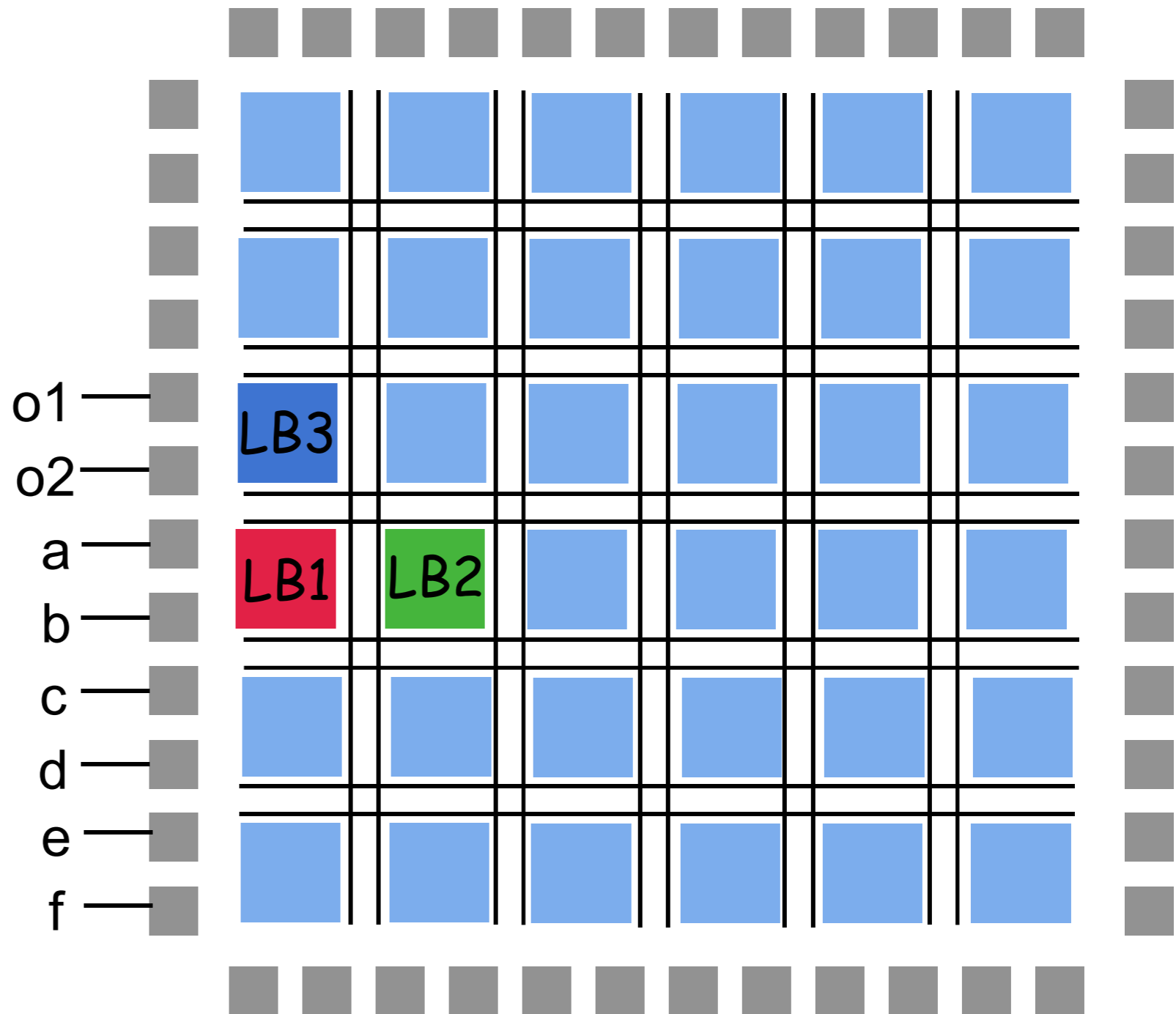
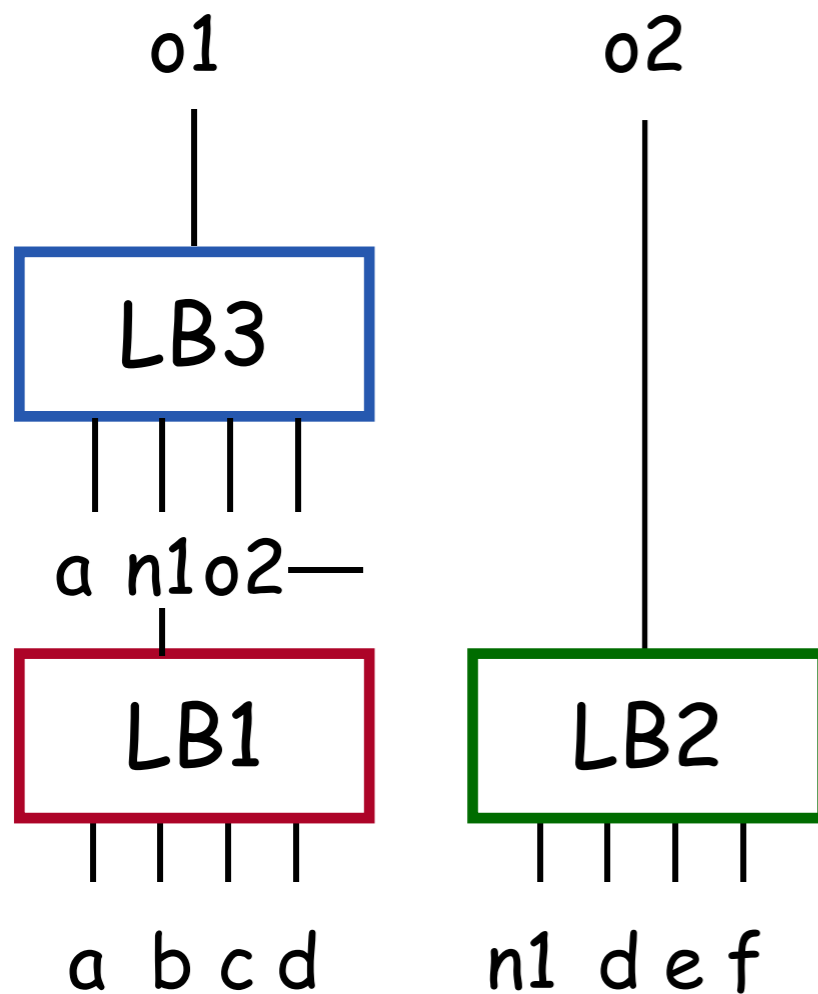
# FPGA mapping



- Assume we use 4-input LUT, each followed by a 1-bit register/bypass path



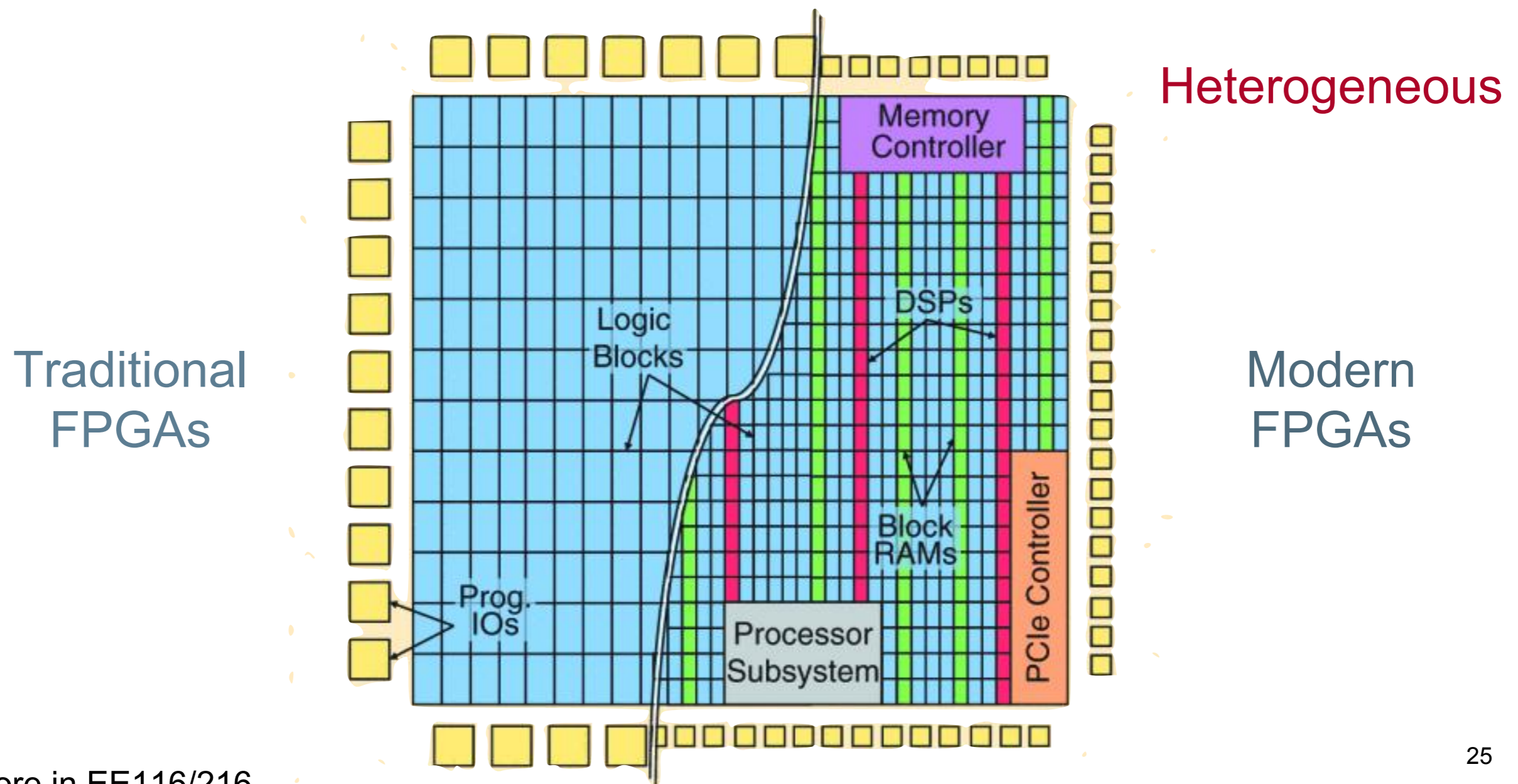
# FPGA placement & routing



After P&R, generate bitstream file to configure the SRAM cells

# Modern FPGAs

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



# Question

- **(True or False)** Given enough resources (Logic blocks, connections and RAMs), an FPGA can implement a RISC-V CPU (e.g., RV32I).

# Summary

- New school computer architecture



# Summary

- New school computer architecture



# Summary

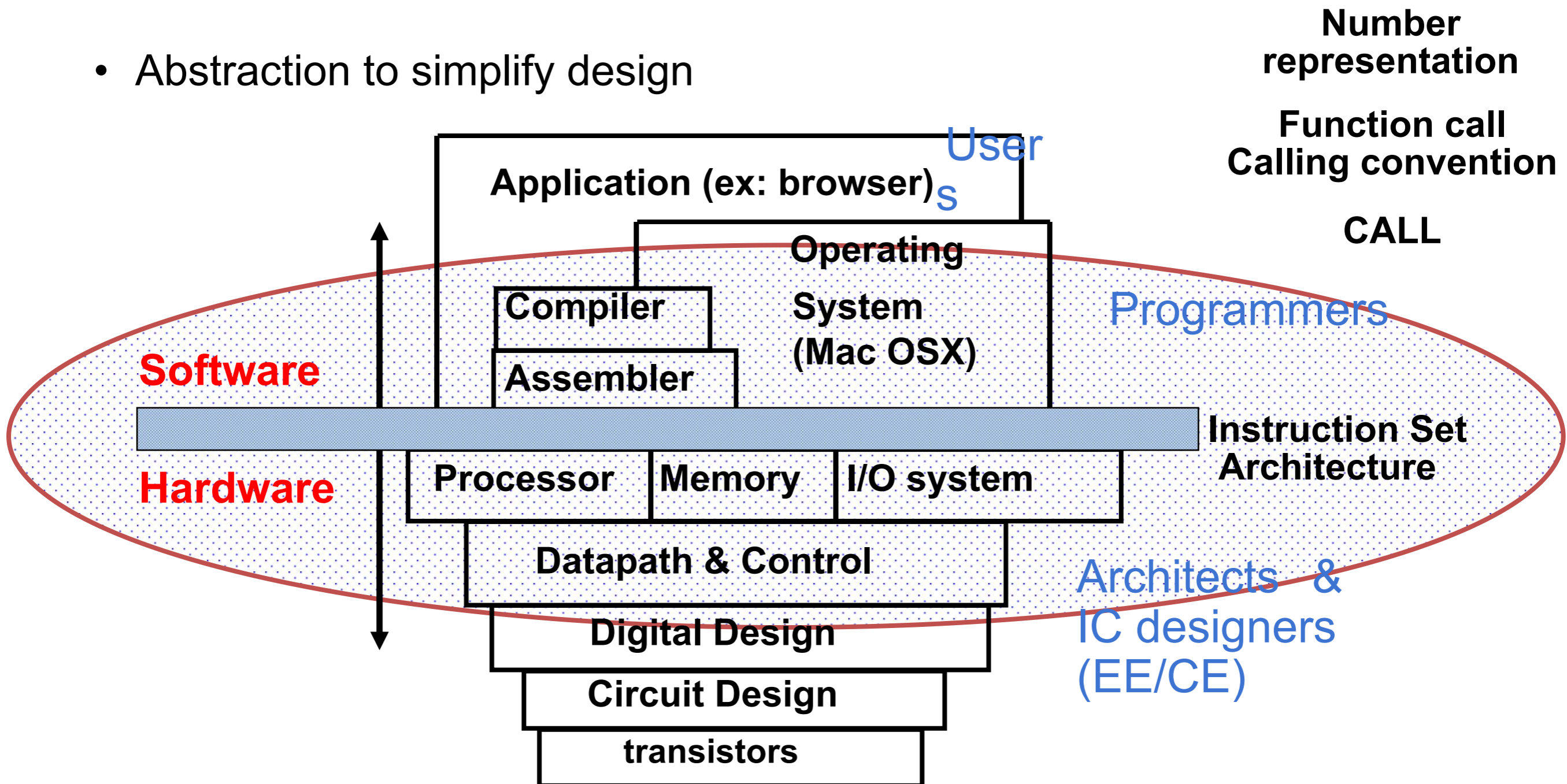
- New school computer architecture
  - Small, embedded, edge devices
  - Ultra large-scale clusters
  - Heterogeneous
  - Alternative computing platform: FPGA, DSA, etc.
- Still relies on old-school computer architecture knowledge

# Great Ideas in Computer Architecture

- Abstraction to simplify design
- Design for Moore's Law
- Make the common case fast
- Memory hierarchy
- Performance via parallelism/pipelining/prediction
- Performance measurement & improvement
- Dependability via redundancy

# Great Ideas in Computer Architecture

- Abstraction to simplify design



# Great Ideas in Computer Architecture

- Abstraction to simplify design
  - I/O management (polling, interrupt, exception)
    - Network, DMA (parallelism)

# Great Ideas in Computer Architecture

- Abstraction to simplify design
  - Finite-state machine (FSM): model everything
    - ISA, hardware, program
    - Dynamic branch prediction, cache coherence, cache controller, etc.

# Great Ideas in Computer Architecture

- Abstraction to simplify design
  - Abstraction introduces extra overhead, modern DSA seeks for cross-layer optimization opportunities, a.k.a. hardware-software co-design, while keep less layers; programmers/architects with both hardware-software knowledge are more likely to survive;
- Design for Moore's Law
- Make the common case fast
- Memory hierarchy
- Performance via parallelism/pipelining/prediction
- Performance measurement & improvement
- Dependability via redundancy

# Great Ideas in Computer Architecture

- Abstraction to simplify design
- Design for Moore's Law
  - Hardware foundation for everything in a computer
- Make the common case fast
- Memory hierarchy
- Performance via parallelism/pipelining/prediction
- Performance measurement & improvement
- Dependability via redundancy

# Great Ideas in Computer Architecture

- Design for Moore's Law
  - Nearly everything built upon the transistors
    - E.g., pipeline, multi-issue, multicore, SIMD, virtual memory, TLB, dynamic branch predictor, heterogeneous computing, FPGA, etc.

# Great Ideas in Computer Architecture

- Abstraction to simplify design
- Design for Moore's Law
  - The frequency of the processors barely increase due to **power wall**, while the number of transistors can continue to scale; Exploit computer architecture knowledge to optimize hardware designs, e.g., use heterogeneous computing (new architectures such as DSA) to offload CPU burden;
- Make the common case fast
- Memory hierarchy
- Performance via parallelism/pipelining/prediction
- Performance measurement & improvement
- Dependability via redundancy

# Great Ideas in Computer Architecture

- Abstraction to simplify design
- Design for Moore's Law
- **Make the common case fast**
  - You can always rely on this to optimize
- Memory hierarchy
- Performance via parallelism/pipelining/prediction
- Performance measurement & improvement
- Dependability via redundancy

# Great Ideas in Computer Architecture

- Abstraction to simplify design
- Design for Moore's Law
- Make the common case fast
- **Memory hierarchy**
  - **Cache & virtual memory**
- Performance via parallelism/pipelining/prediction
- Performance measurement & improvement
- Dependability via redundancy

# Great Ideas in Computer Architecture

- Memory hierarchy
  - Cache
    - Cache basics: hit & miss (types of misses)
    - All the policies (placement, replacement, write)
  - Virtual memory
    - Virtual address -> physical address, through page tables
    - Multi-level page table, TLB to optimize
    - OS uses page tables to create the illusion
    - Hardware registers record page table addresses (part of memory management unit, may also include page table walker/TLB)
    - Context switch
- Cache coherence
- Synchronization

# Great Ideas in Computer Architecture

- Abstraction to simplify design
- Design for Moore's Law
- Make the common case fast
- **Memory hierarchy**
  - An implementation of the previous idea on memory; while emerging DSAs largely rely on scratchpad memory, memory access is still the bottleneck for many applications; emerging architectures such as computing-in-memory (CIM) and memory based on new materials are the recent research trends
- Performance via parallelism/pipelining/prediction
- Performance measurement & improvement
- Dependability via redundancy

# Great Ideas in Computer Architecture

- Abstraction to simplify design
- Design for Moore's Law
- Make the common case fast
- Memory hierarchy
- Performance via parallelism/pipelining/prediction
  - Instruction-level parallelism
    - Pipeline & multi-issue (static vs. dynamic)
  - Data-level parallelism
    - SIMD & mapreduce model
  - Thread-level parallelism
    - SMT, multi-core, time-multiplexing, openMP
  - Request-level parallelism
    - Map-reduce model
- Performance measurement & improvement
- Dependability via redundancy

# Great Ideas in Computer Architecture

- Performance via parallelism/pipelining/prediction
  - Instruction-level parallelism
    - Pipeline
      - Implementation & hazards (hardware or software to resolve)
    - Multi-issue (static vs. dynamic)
      - VLIW (commonly used in AI chip/NN accelerator) vs. superscalar (commonly used for general purpose CPU)
  - Data-level parallelism
    - SIMD (no data dependency, same operation on multiple data)
      - Intel Intrinsic, vector machine hardware

# Great Ideas in Computer Architecture

- Performance via parallelism/pipelining/prediction
  - Thread-level parallelism
    - OS uses time-multiplexing to give the illusion that multiple threads execute simultaneously;
    - OpenMP to implement TLP, there are also other tools
    - Race condition and lock/sync. (atomic instructions & lr/sc) to resolve
    - SMT (still single core)
    - Multi-core
      - Cache coherence and snooping protocol

# Great Ideas in Computer Architecture

- Abstraction to simplify design
- Design for Moore's Law
- Make the common case fast
- Memory hierarchy
- Performance via parallelism/pipelining/prediction
  - Prediction
    - Static branch prediction
    - Dynamic branch prediction (using history information to predict)
    - Prefetch to accelerate
  - May lead to security issues
- Performance measurement & improvement
- Dependability via redundancy

# Great Ideas in Computer Architecture

- Abstraction to simplify design
- Design for Moore's Law
- Make the common case fast
- Memory hierarchy
- Performance via parallelism/pipelining/prediction
  - Major schemes to achieve higher performance
    - Instruction/Data/Thread-level parallelism
  - New-school computer architectures employ more opportunities of parallelism (request-/accelerator-level parallelism)
- Performance measurement & improvement
- Dependability via redundancy

# Great Ideas in Computer Architecture

- Abstraction to simplify design
- Design for Moore's Law
- Make the common case fast
- Memory hierarchy
- Performance via parallelism/pipelining/prediction
- Performance measurement & improvement
  - CPU (execution) time/wall clock time
  - AMAT to profile performance of cache
  - Amdahl's law to compute the acceleration factor (strong/weak scaling)
  - Timing analysis of digital circuit to obtain max frequency
- Dependability via redundancy

# Great Ideas in Computer Architecture

- Abstraction to simplify design
- Design for Moore's Law
- Make the common case fast
- Memory hierarchy
- Performance via parallelism/pipelining/prediction
- Performance measurement & improvement
- Dependability via redundancy
  - MTTF/MTBF/availability/AFR...
  - Hamming ECC code (information redundancy)
  - RAID (hardware redundancy, balance performance and reliability)

All the best!  
Good luck!

Happy Dragon  
Boat Festival!

