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

CS 110
Computer Architecture
Advanced Topics on Parallelism

Instructors:

Siting Liu & Chundong Wang

Course website: https://toast-lab.sist.shanghaitech.edu.cn/courses/CS110@ShanghaiTech/Spring-2024/index.html

School of Information Science and Technology (SIST)

ShanghaiTech University

2024/5/30

Administratives

- Lab 14 release soon, will be checked next week.
- The last HW8 to release soon.
- The last project 4 to release soon.
- The last discussion on DMA/IO (teaching center 301, this Friday and next Monday)

Outline

- Review on parallelism
- Multi-issue
  - Static multi-issue (VLIW)
  - Dynamic multi-issue (superscalar)
  - Design cases in modern computer systems
- Heterogenous computing
  - Modern SoCs (accelerator-level parallelism)
  - Introduction to FPGA

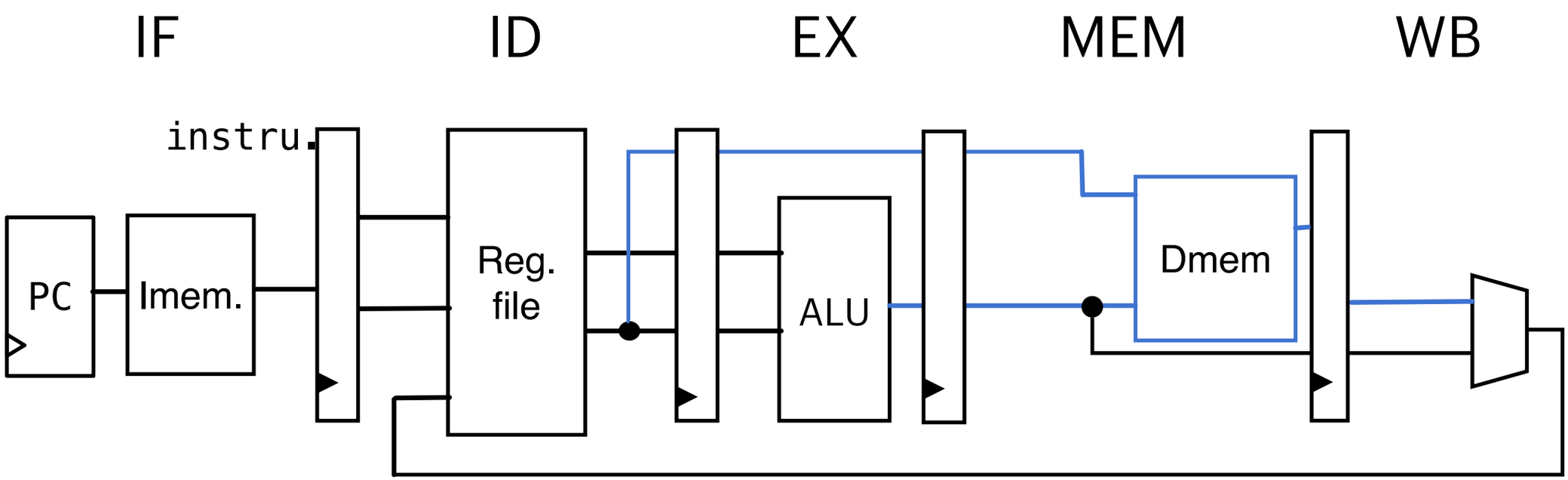
Review

- Review on parallelism
  - Instruction-level parallelism (ILP)
    - Pipeline, deeper for faster clock, but potentially more hazards
    - Today's lecture (Multi-issue)
  - Data-level parallelism (DLP)
    - SIMD
- Thread-level parallelism (TLP)
  - Multi-threading/Hardware hyper-threading

$$\frac{Time}{Program} = \frac{Instructions}{Program} \cdot \frac{Cycles}{Instruction} \cdot \frac{Time}{Cycle}$$

Single-issue

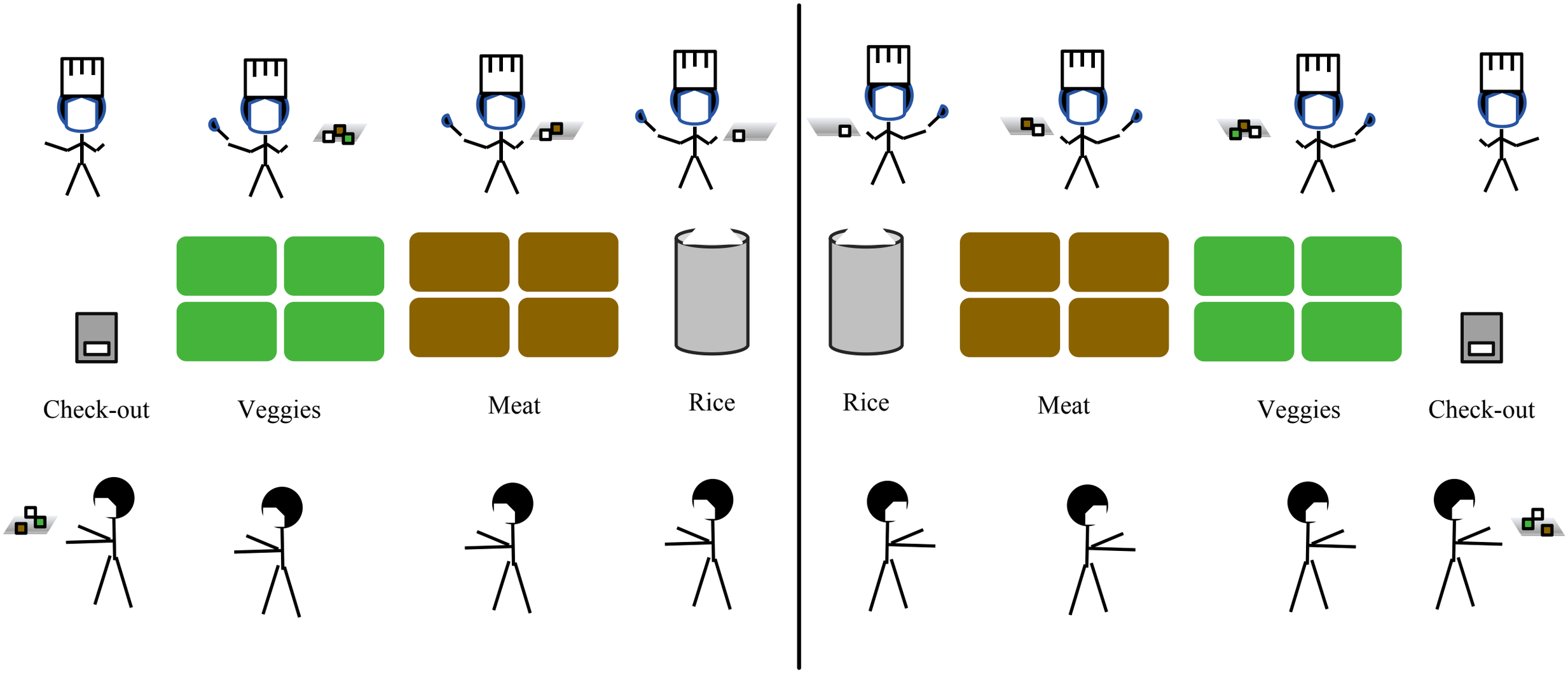
- Simplified 5-stage pipelined single-issue CPU datapath



- At most 1 instruction is “issued” to the datapath at 1 clock cycle

$$\text{average CPI} \geq 1$$

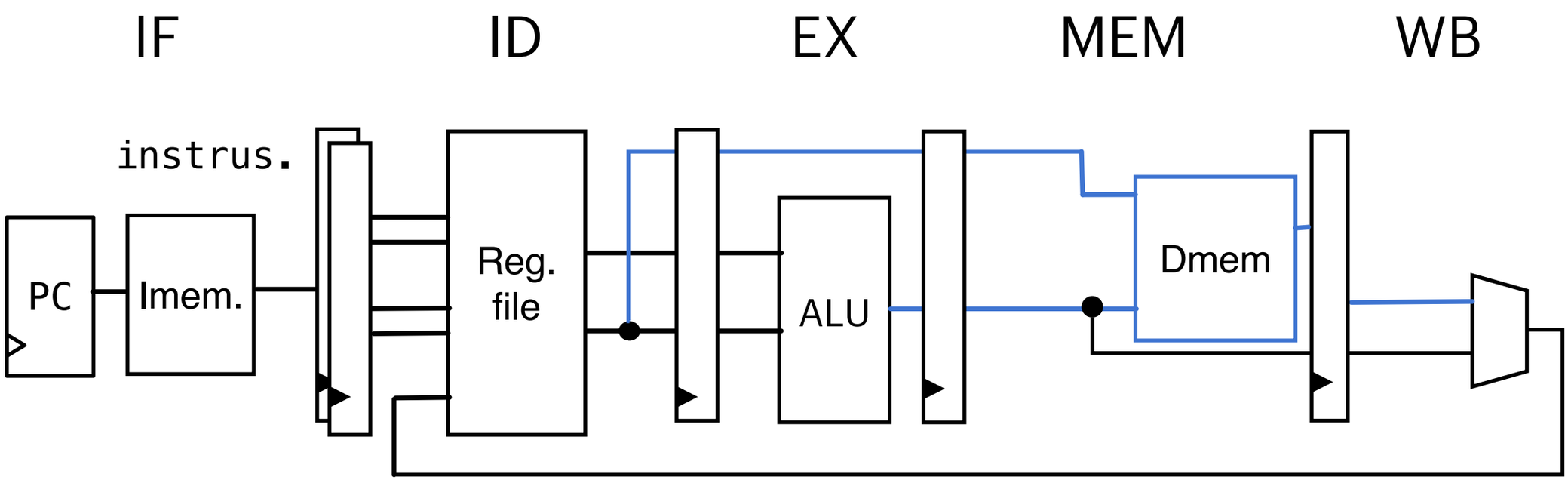
Multi-issue



Canteen analogy
(combined with pipelining)

Multi-issue (Hardware)

- Multi-issue CPU datapath

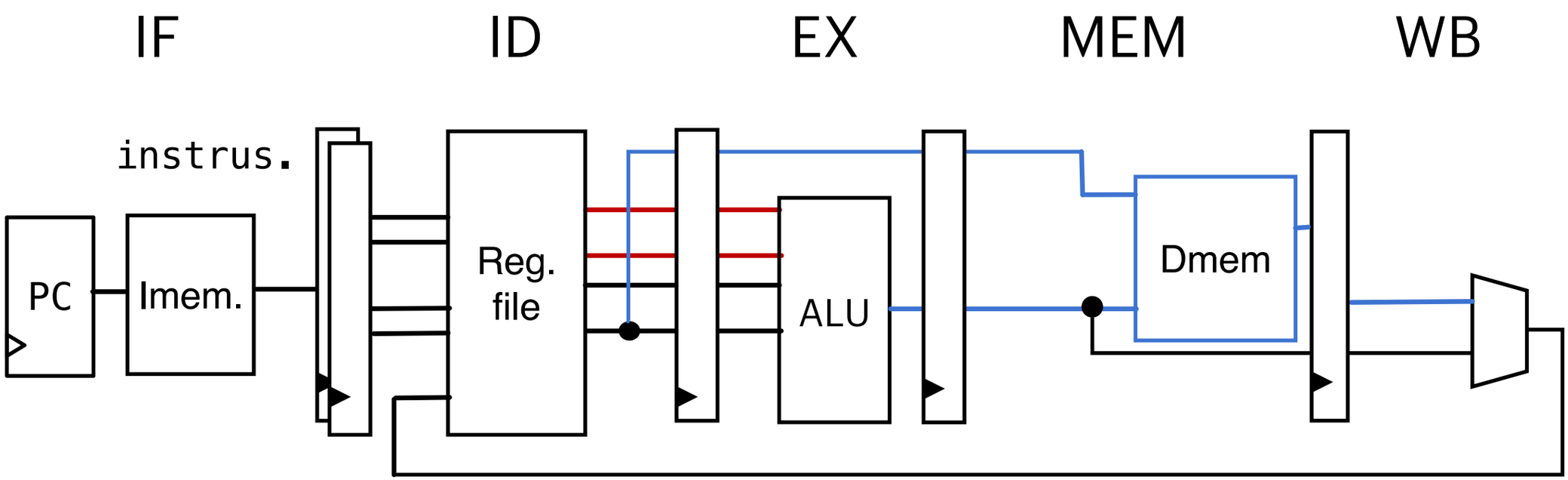


Hardware has to be adapted to avoid structural hazards

- Issue multiple instructions to the datapath in 1 clock cycle, average CPI can be smaller than 1.

Multi-issue (Hardware)

- Multi-issue CPU datapath

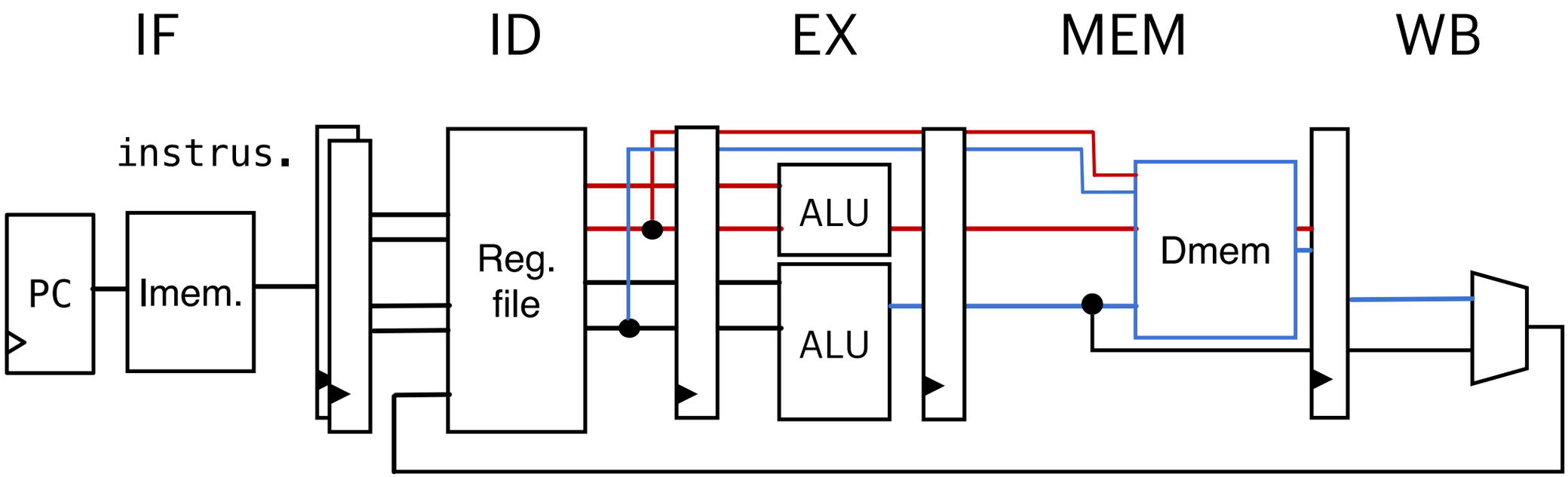


Hardware has to be adapted to avoid structural hazards

- Issue multiple instructions to the datapath in 1 clock cycle, average CPI may smaller than 1.

Multi-issue (Hardware)

- Multi-issue CPU datapath

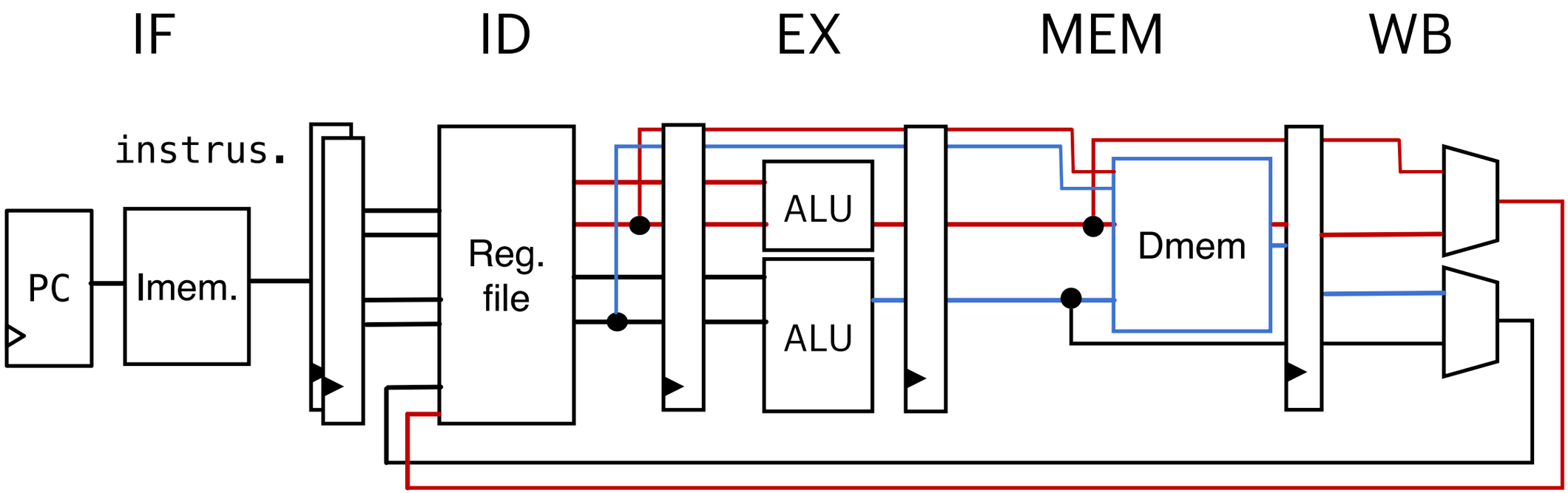


Hardware has to be adapted to avoid structural hazards

- Issue multiple instructions to the datapath in 1 clock cycle, average CPI may smaller than 1.

Multi-issue (Hardware)

- Multi-issue CPU datapath

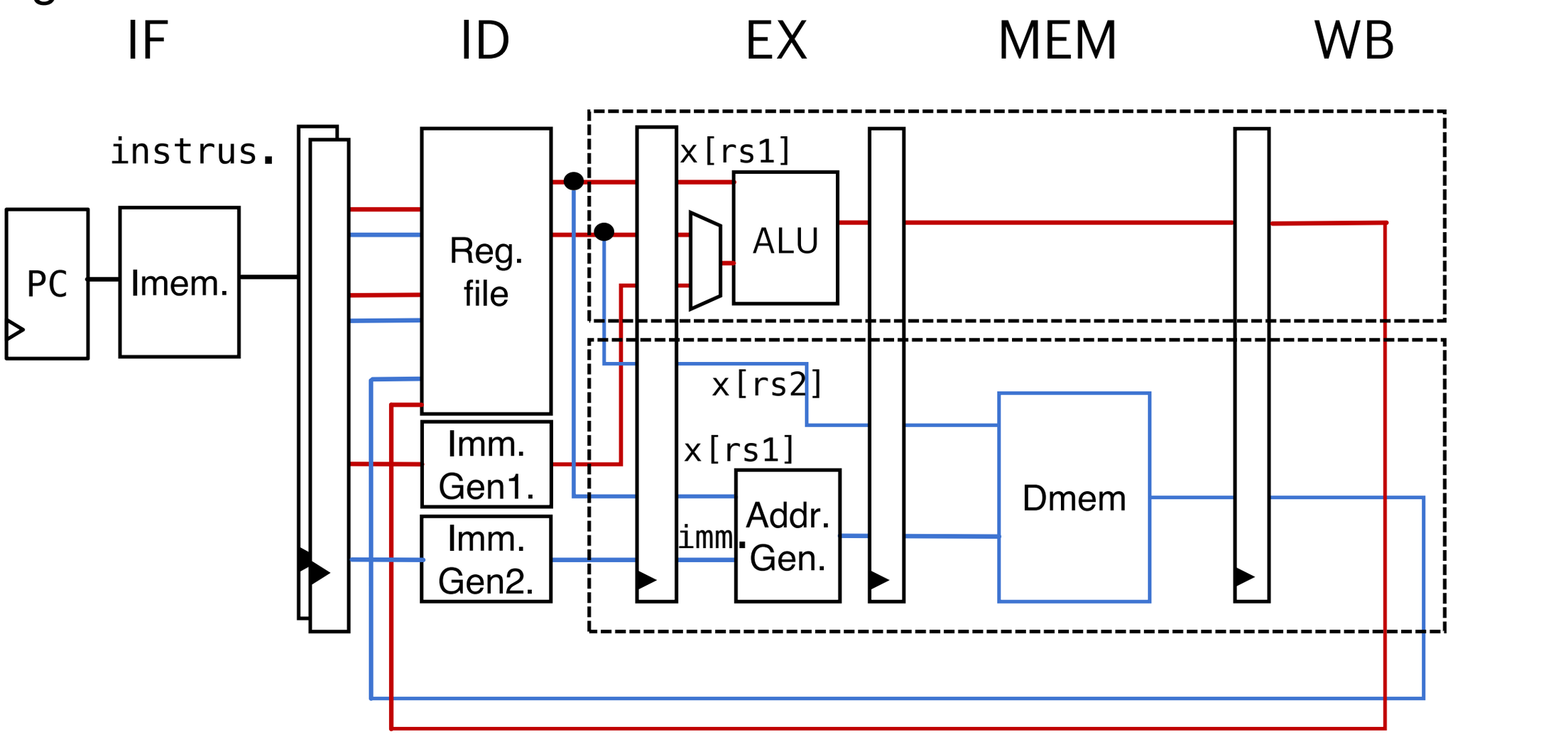


Hardware has to be adapted to avoid structural hazards

- Issue multiple instructions to the datapath in 1 clock cycle, average CPI may smaller than 1.

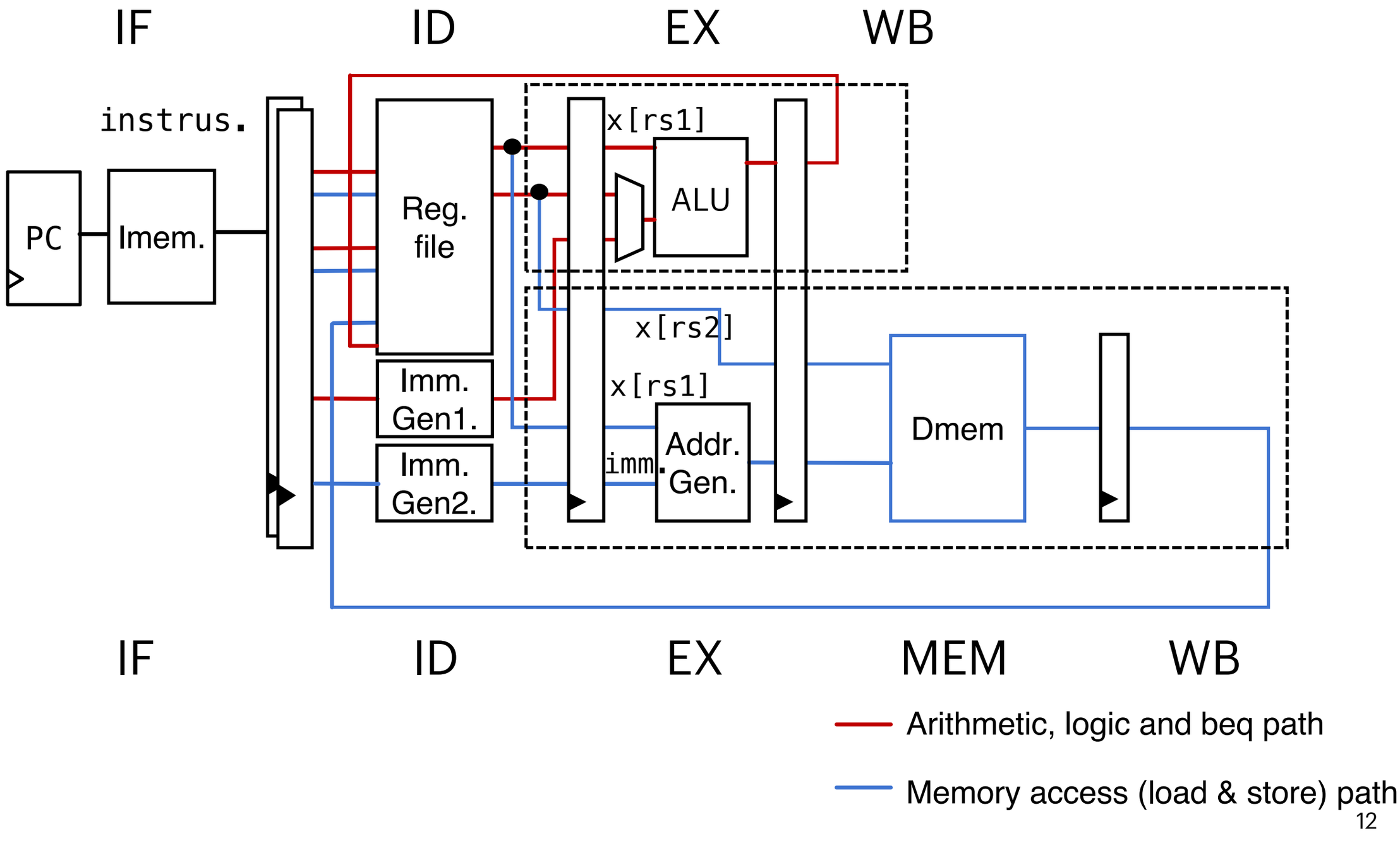
Multi-issue (Hardware)

- In practice, we build different datapaths for different types of instructions
- E.g.



Multi-issue (Hardware)

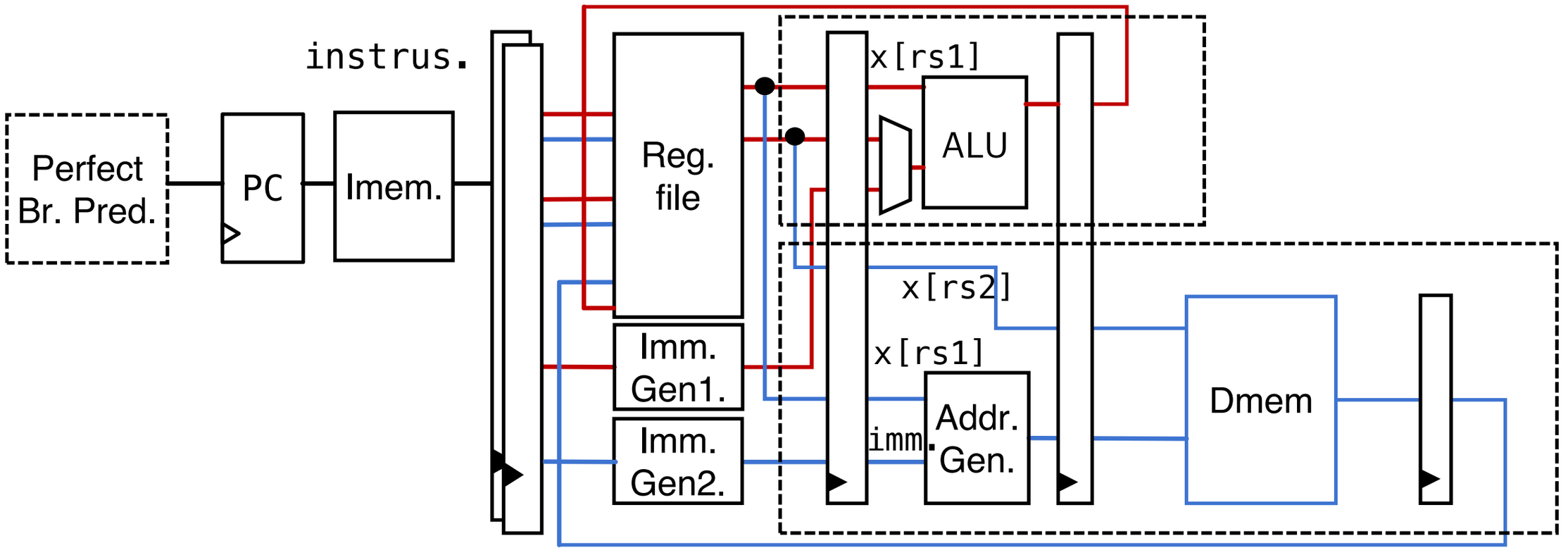
- In practice, we build different datapaths for different types of instructions
- E.g.



Multi-issue

- Ideally, issue two instructions with different type to ALU/mem datapaths

Instruciton type	cc1	cc2	cc3	cc4	cc5	cc6	cc7
ALU or branch	IF	ID	EX	WB			
Load or store	IF	ID	EX	MEM	WB		
ALU or branch		IF	ID	EX	WB		
Load or store		IF	ID	EX	MEM	WB	
ALU or branch			IF	ID	EX	WB	
Load or store			IF	ID	EX	MEM	WB



Multi-issue

```
for (int i=1000; i>0; i=i-1)
     x[i] = x[i] + s;
```

```
1.      addi s0,x0,s      #initialize s0
2.Loop:lw   t3,0(t1)      #load array element
3.      add  t3,t3,s0     #add s to $t3
4.      sw   t3,0(t1)     #store result
5.      addi t1,t1,-4     #t1 =t1-4
6.      bne  t1,t2,Loop   #repeat loop if t1!=t2
```

Instruciton	cc1	cc2	cc3	cc4	cc5	cc6	cc7	cc8	cc9
1.addi	IF	ID	EX	WB					
2.lw	IF	ID	EX	MEM	WB				
3.add		IF	ID	EX	WB				
4.sw		IF	ID	EX	MEM	WB			
5.addi			IF	ID	EX	WB			
nop			nop	nop	nop	nop	nop		
6.bne			IF	ID	EX	WB			
2.lw			IF	ID	EX	MEM	WB		
3.add				IF	ID	EX	WB		
4.sw				IF	ID	EX	MEM	WB	
5.addi					IF	ID	EX	WB	
nop					nop	nop	nop	nop	nop

Hazards!

Multi-issue

```
for (int i=1000; i>0; i=i-1)
     x[i] = x[i] + s;
```

```
1.      addi s0,x0,s      #initialize s0
2.Loop:lw   t3,0(t1)      #load array element
3.      add  t3,t3,s0     #add s to $t3
4.      sw   t3,0(t1)     #store result
5.      addi t1,t1,-4     #t1 =t1-4
6.      bne  t1,t2,Loop   #repeat loop if t1!=t2
```

Instruciton	cc1	cc2	cc3	cc4	cc5	cc6	cc7	cc8	cc9
1.addi	IF	ID	EX	WB					
2.lw	IF	ID	EX	MEM	WB				
nop		nop	nop	nop	nop				
nop		nop	nop	nop	nop	nop			
3.add			IF	ID	EX	WB			
4.sw			IF	ID	EX	MEM	WB		
5.addi				IF	ID	EX	WB		
nop				nop	nop	nop	nop	nop	
6.bne					IF	ID	EX	WB	
2.lw					IF	ID	EX	MEM	WB
... ...									
... ...									

Forwarding

Forwarding

Forwarding

Multi-issue

- Loop unrolling & register renaming to optimize (previously used in SIMD)

```
1.   addi    s0,x0,s
2.L:lw       t3,0(t1)
3.   lw      t4,-4(t1)
4.   lw      t5,-8(t1)
5.   lw      t6,-12(t1)
6.   add     t3,t3,s0
7.   add     t4,t4,s0
8.   add     t5,t5,s0
9.   add     t6,t6,s0
10.  sw      t3,0(t1)
11.  sw      t4,-4(t1)
12.  sw      t5,-8(t1)
13.  sw      t6,-12(t1)
14.  addi    t1,t1,-16
15.  bne     t1,t2,L
```

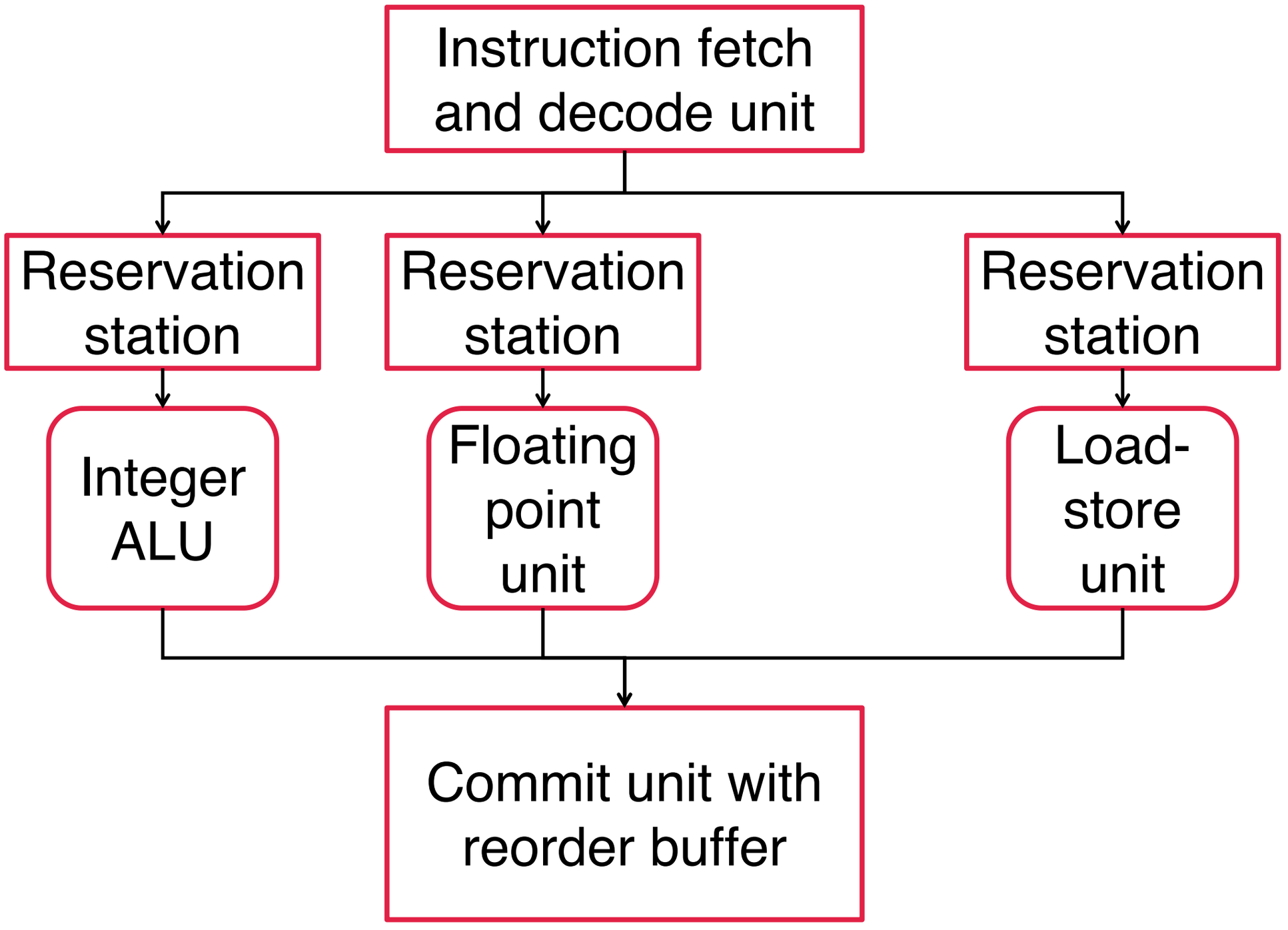
Instruciton	cc1	cc2	cc3	cc4	cc5	cc6	cc7	cc8	cc9
1.addi	IF	ID	EX	WB					
2.lw t3	IF	ID	EX	MEM	WB				
nop		nop	nop	nop	nop				
3.lw t4		IF	ID	EX	MEM	WB			
6.add t3			IF	ID	EX	WB			
4.lw t5		IF	ID	EX	MEM	WB			
7.add t4			IF	ID	EX	WB			
5.lw t6			IF	ID	EX	MEM	WB		
8.add t5				IF	ID	EX	WB		
10.sw t3				IF	ID	EX	MEM	WB	
9.add t6					IF	ID	EX	WB	
11.sw t4					IF	ID	EX	MEM	
14.addi						IF	ID	EX	
12.sw t5						IF	ID	EX	
15.bne							IF	ID	
13.sw t6							IF	ID	

Static vs. Dynamic multi-issue

- Static multi-issue
  - Package instructions into issue slots and detect hazards statically (at compile time mostly)
  - Hardware may also detect/resolve hazards
  - Also called VLIW (very long instruction word)
- Dynamic multi-issue
  - Package instructions into issue slots and detect hazards dynamically (during execution by hardware mostly)
  - Compiler may also help avoiding hazards
  - Also called superscalar

Instruciton type	cc1	cc2	cc3	cc4	cc5	cc6	cc7
ALU or branch	IF	ID	EX	WB			
Load or store	IF	ID	EX	MEM	WB		
ALU or branch		IF	ID	EX	WB		
Load or store		IF	ID	EX	MEM	WB	
ALU or branch			IF	ID	EX	WB	
Load or store			IF	ID	EX	MEM	WB

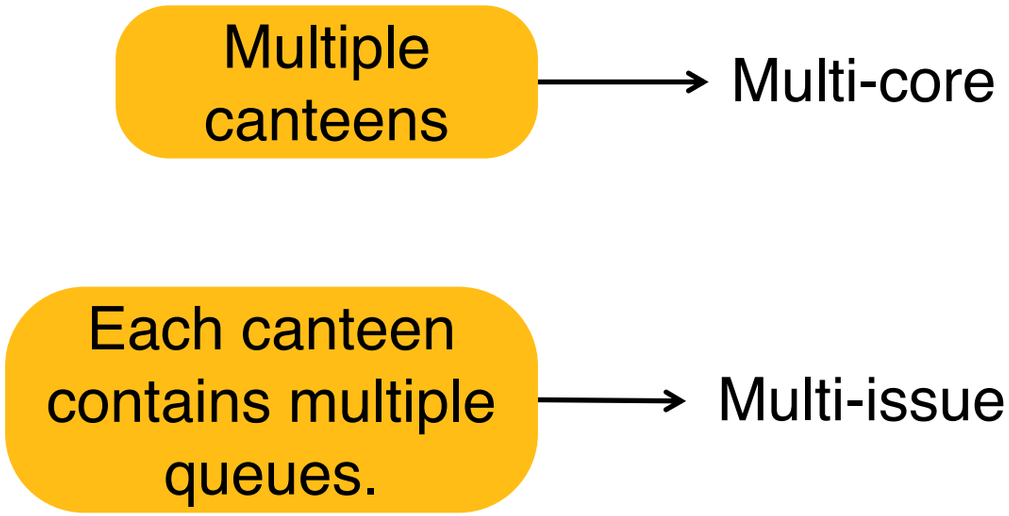
Hardware implementation of superscalar



More in CS211 CAII

Multi-issue

- Multi-issue is not multi-core;
- Multi-issue is not SIMD;
- Multi-issue can be combined with pipelining, SIMD, multi/hyper-threading, etc. to improve the performance of the processor;



Multi-issue

- Multi-issue is not multi-core;
- Multi-issue is not SIMD;
- Multi-issue can be combined with pipelining, SIMD, multi/hyper-threading, etc. to improve the performance of the processor;

One streaming multi-processor inside an H100 GPU

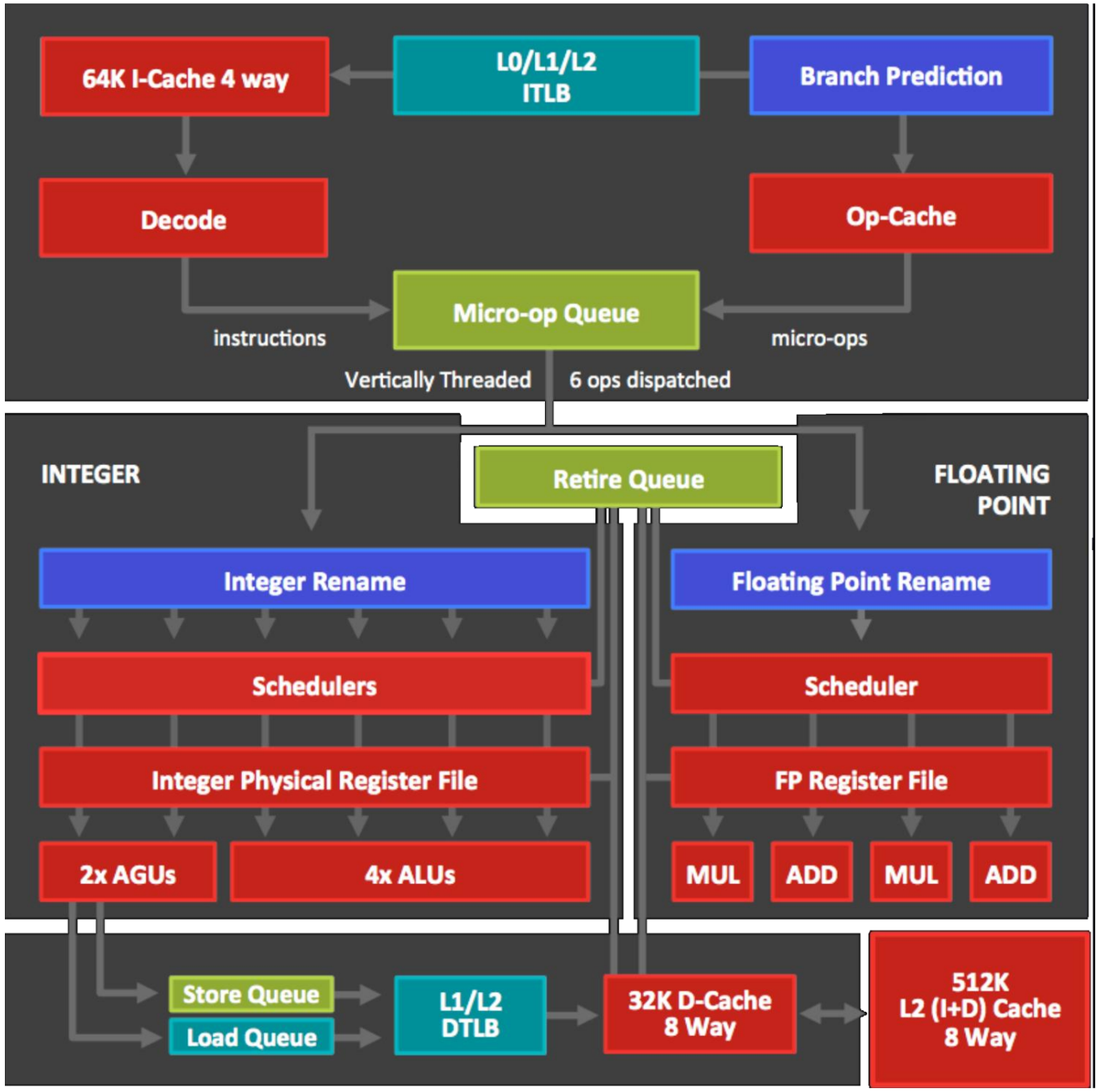
Credit to Nvidia

More in EE219 AI computing systems

Multi-issue

- Multi-issue is not multi-core;
- Multi-issue is not SIMD;
- Multi-issue can be combined with pipelining, SIMD, multi/hyper-threading, etc. to improve the performance of the processor;

AMD CPU (Zen architecture) that supports simultaneous multithreading



Credit to AMD

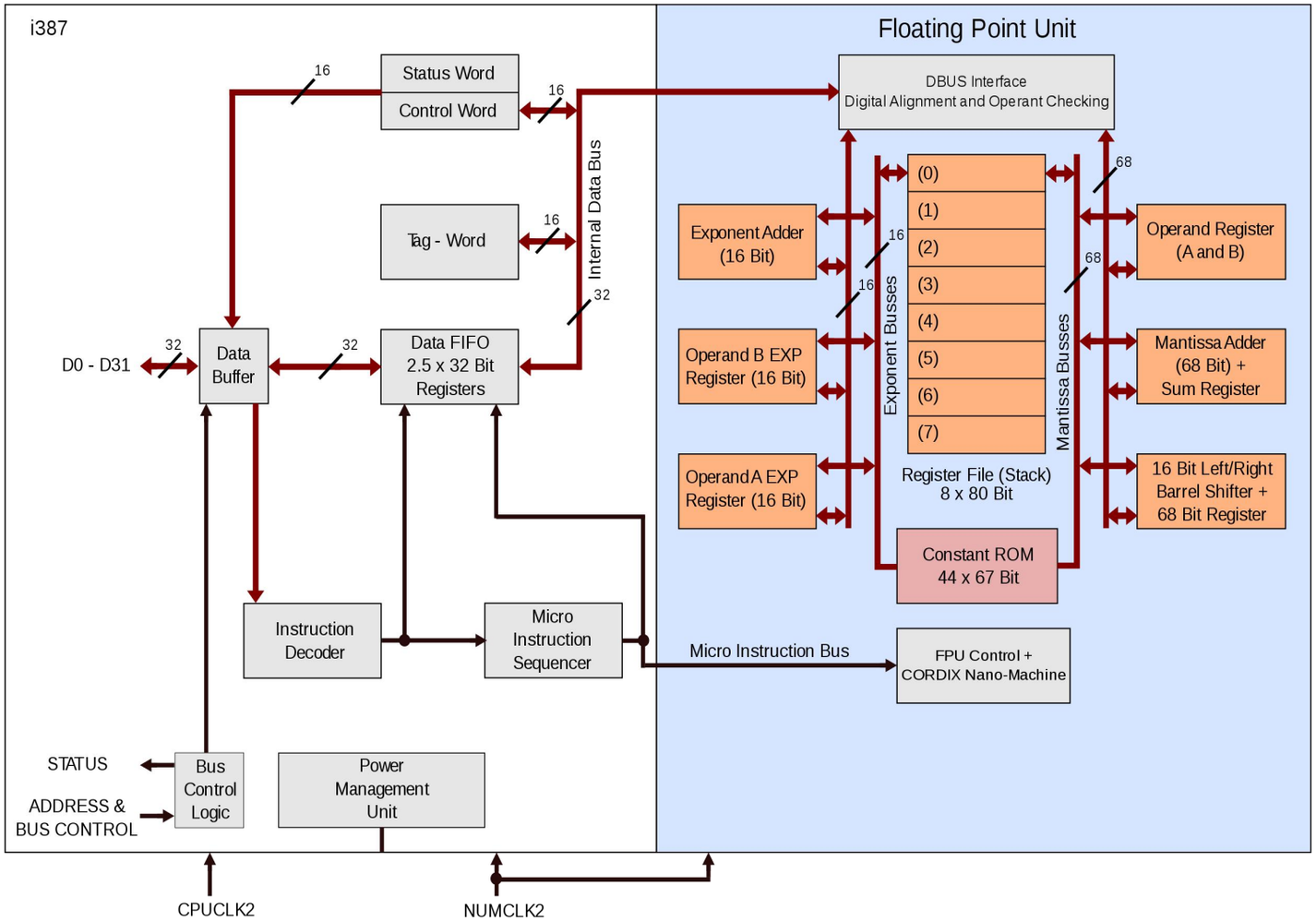
Outline

- Review on parallelism
- Multi-issue
  - Static multi-issue (VLIW)
  - Dynamic multi-issue (superscalar)
  - Design cases in modern computer systems
- Heterogenous computing
  - Modern SoCs (accelerator-level parallelism)
  - Introduction to FPGA

Heterogenous 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.

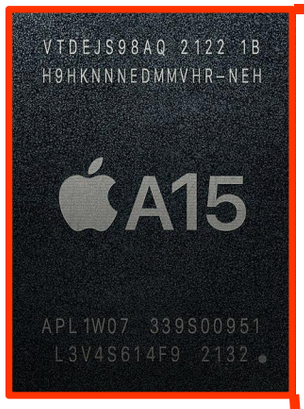
- Historically, Intel uses math-coprocessor to accelerate floating-point arithmetics.
- Otherwise, floating-point arithmetics are performed as in Proj. 1.1.
- More coprocessors with different microarchitecture emerges targeting certain domain (a.k.a. domain-specific architectures)



Intel 80387 microarchitecture from Wikipedia.

Heterogenous computing (SoC)

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



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

Apple A15 SoC 107.68 mm²

Huawei Kirin 990

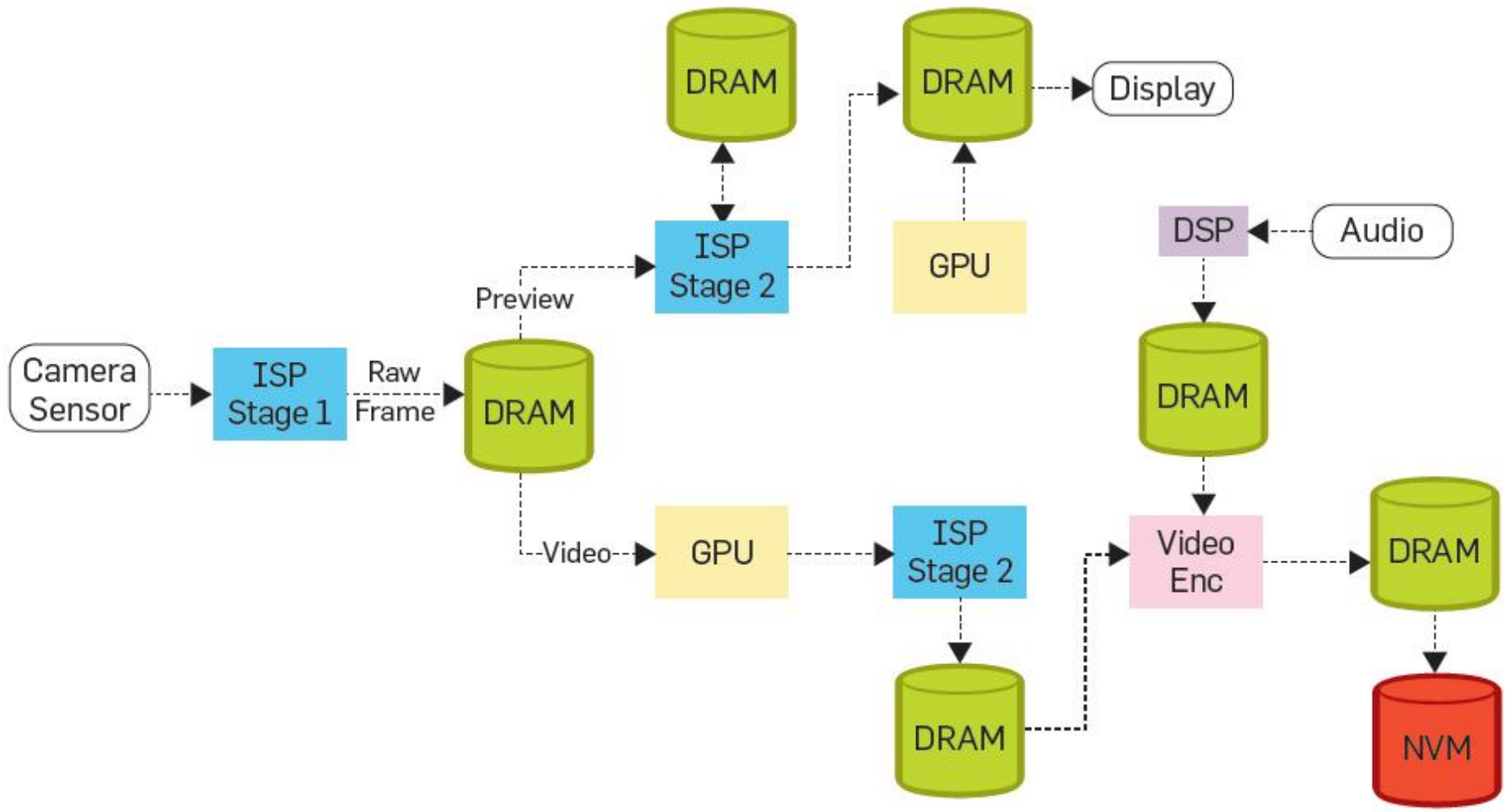
https://www.semianalysis.com/p/apple-m2-die-shot-and-architecture

From Techinsights & AnandTech

Heterogenous computing (SoC)

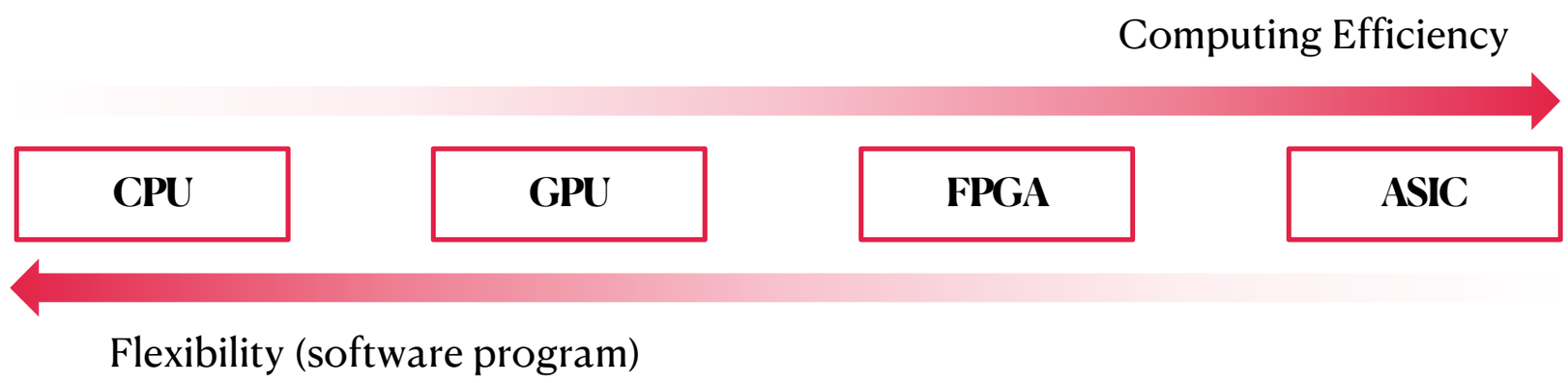
- Accelerator-level parallelism example

A smartphone performing video recording
The usage of the SoC



Heterogenous 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 prgrammable 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

```
assign sum=a+b;
```

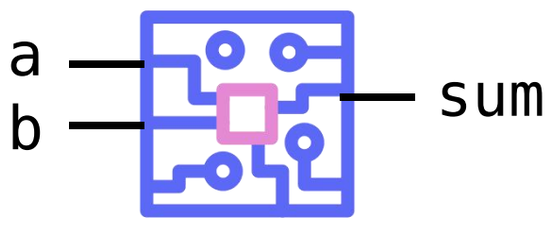
Synthesize ↓

Gate-level netlist

Logic gates and their connections

Place and route ↓

Actual hardware adder circuit



- Instruction-based serial execution of programs

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

Compiler ↓

```
add s0,t1,t0
```

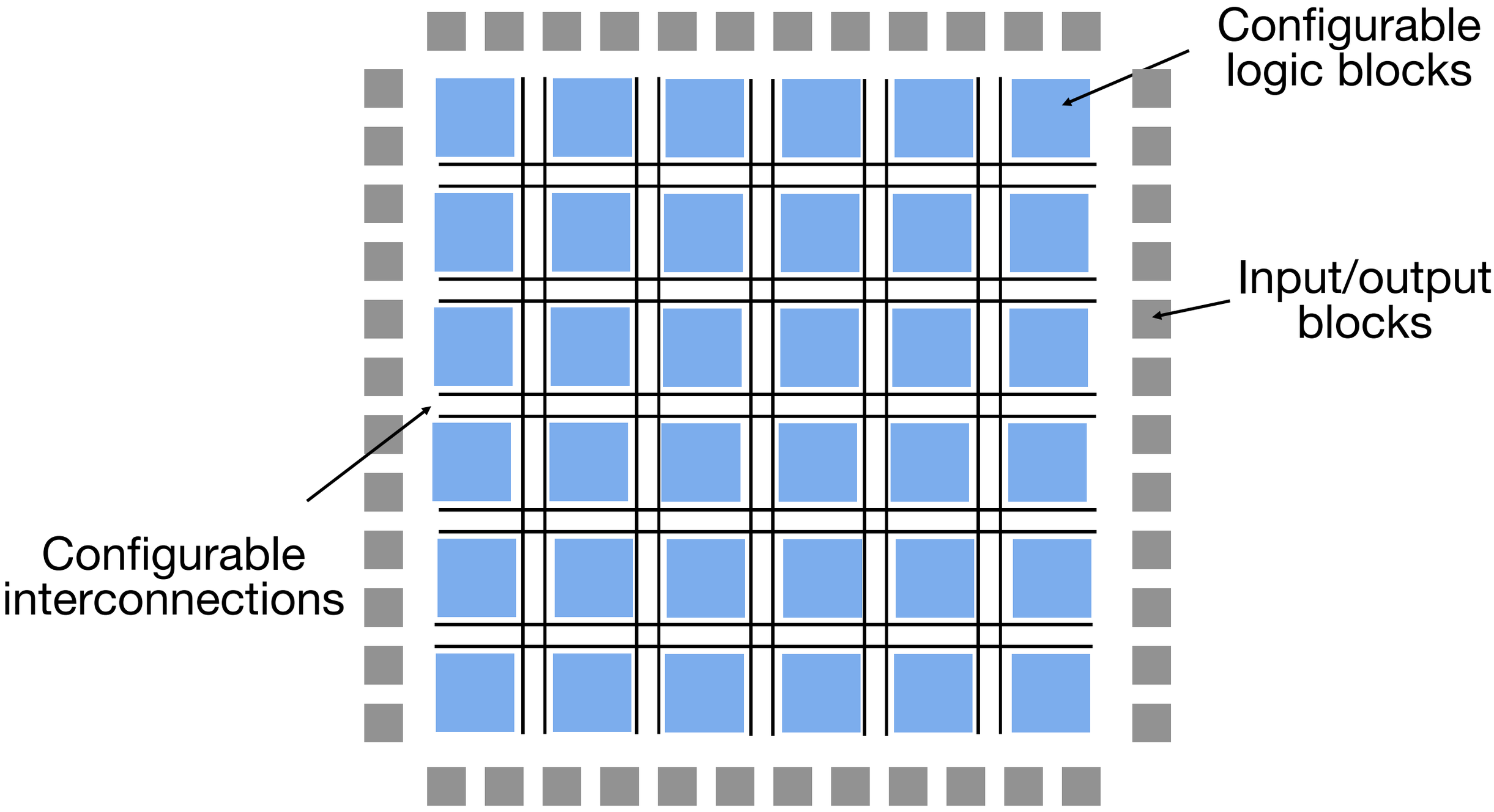
Assembler ↓

```
0x00530433
```

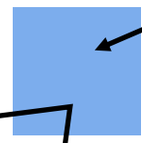
↓

Activate “add” datapath and change the state

FPGA implements any logics
(given enough hardware resources)



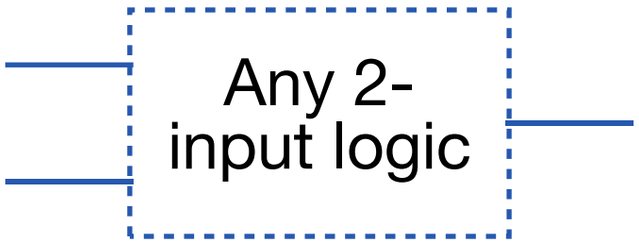
LUTs inside configurable logic blocks



Configurable logic blocks (CLBs)

Many LUTs inside a CLB

Basic principle of LUTs

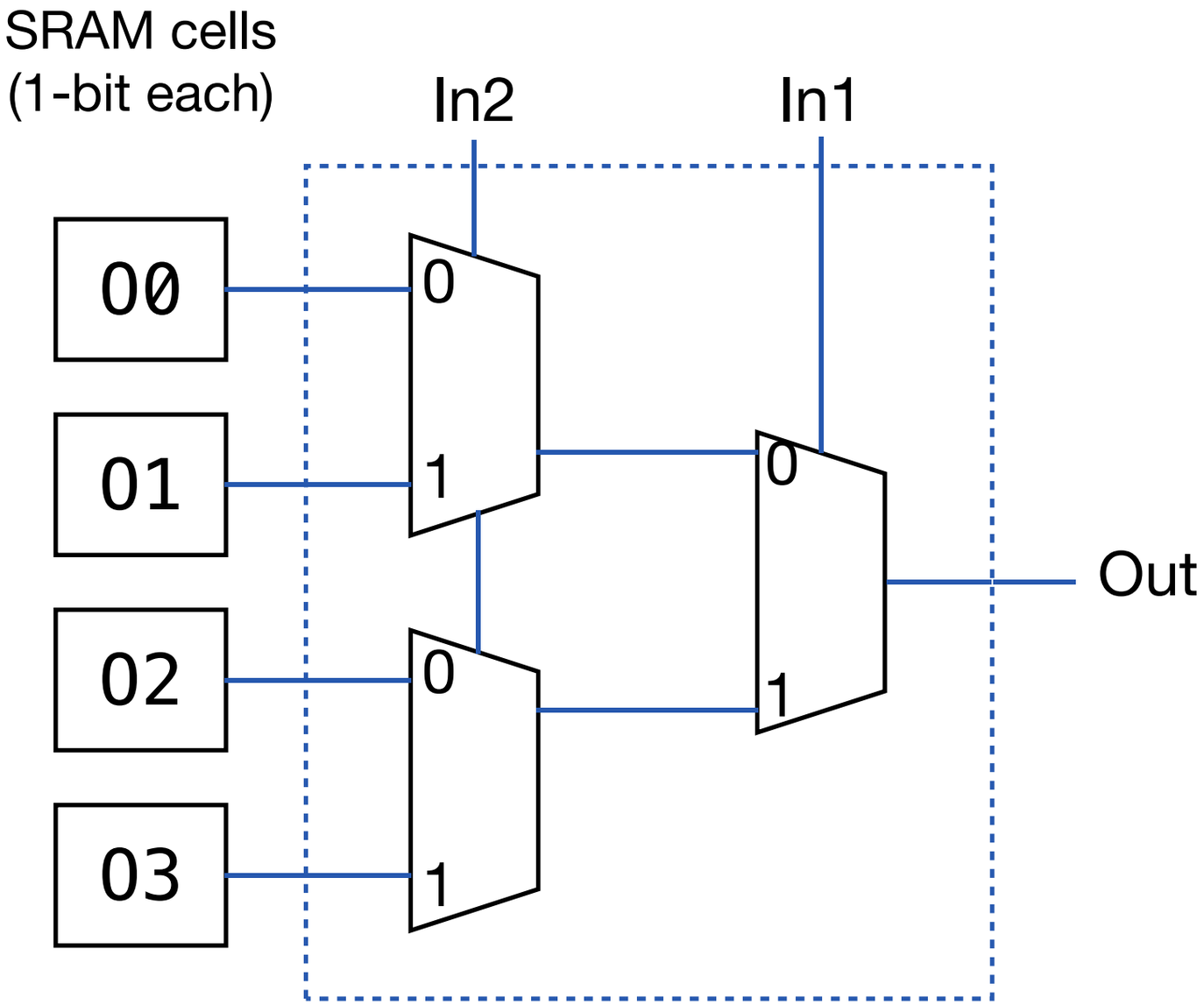


In1	In2	Out
0	0	00
0	1	01
1	0	02
1	1	03

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

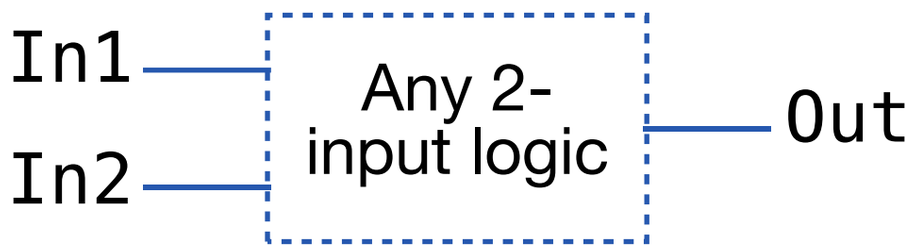
LUTs inside configurable logic blocks

- Look-up table (LUT)



In1	In2	Out
0	0	00
0	1	01
1	0	02
1	1	03

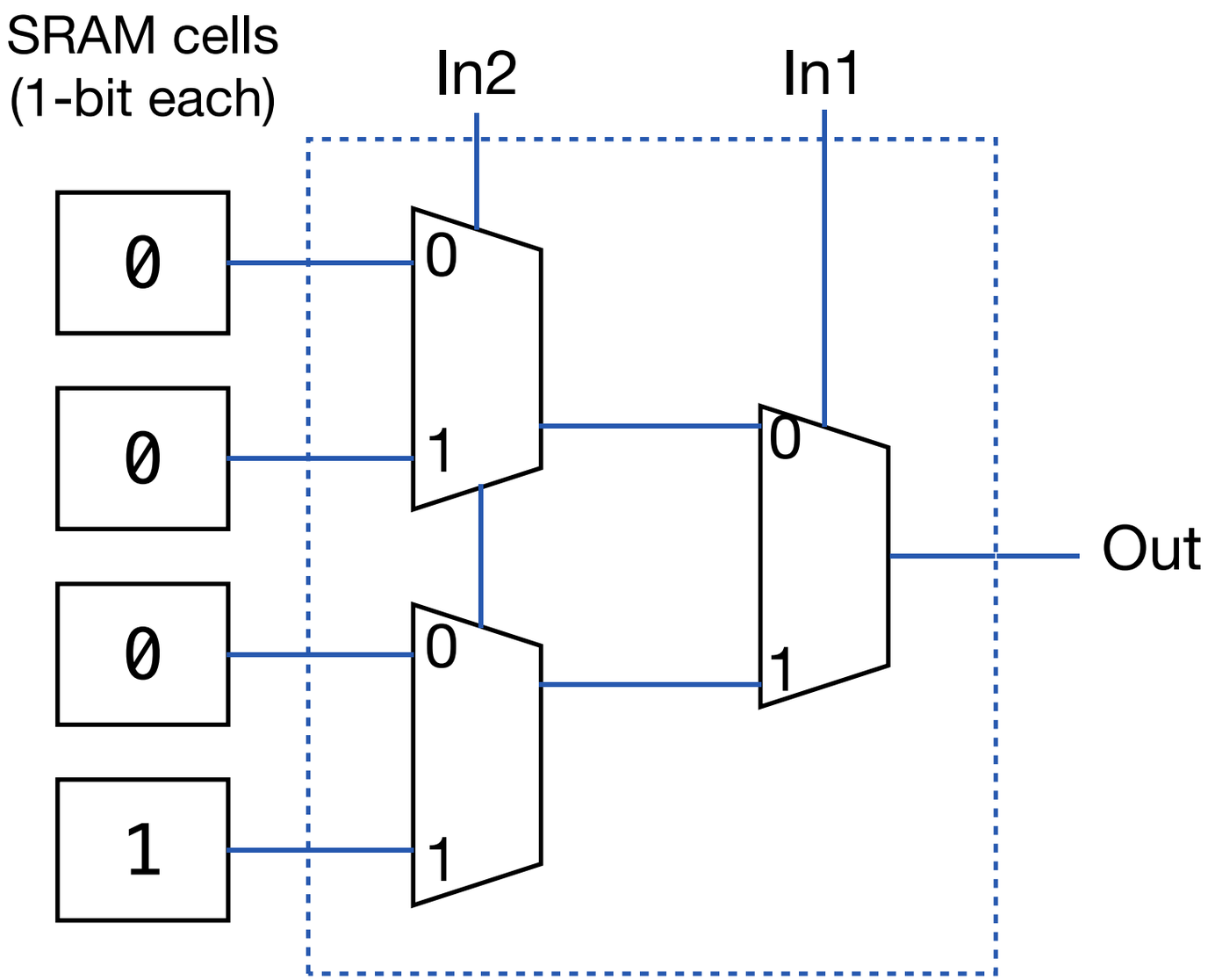
Equivalent to



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

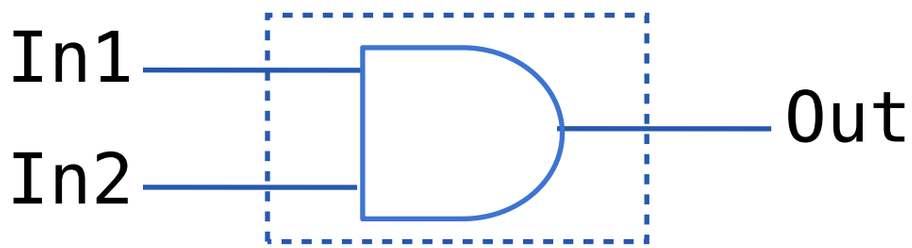
LUTs inside configurable logic blocks

- Example, 2-input LUT implements an AND gate



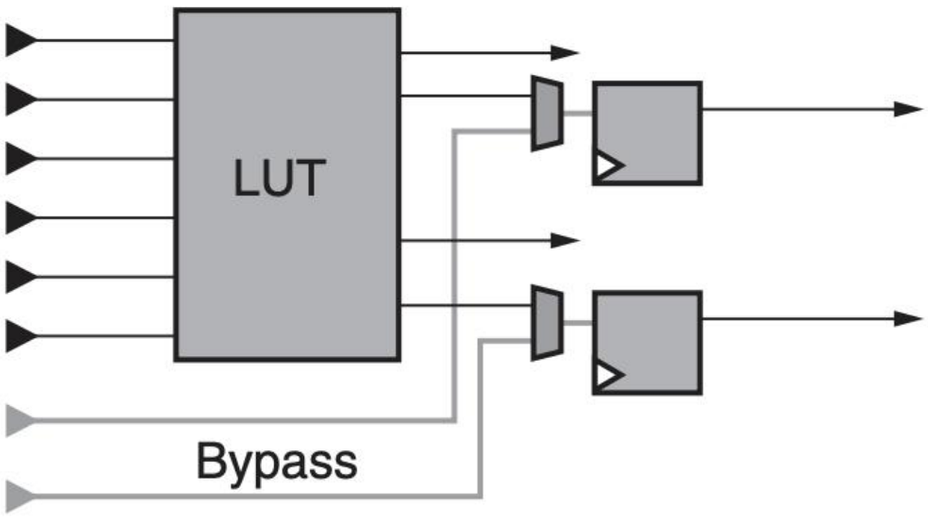
In1	In2	Out
0	0	0
0	1	0
1	0	0
1	1	1

Equivalent to



LUTs inside configurable logic blocks

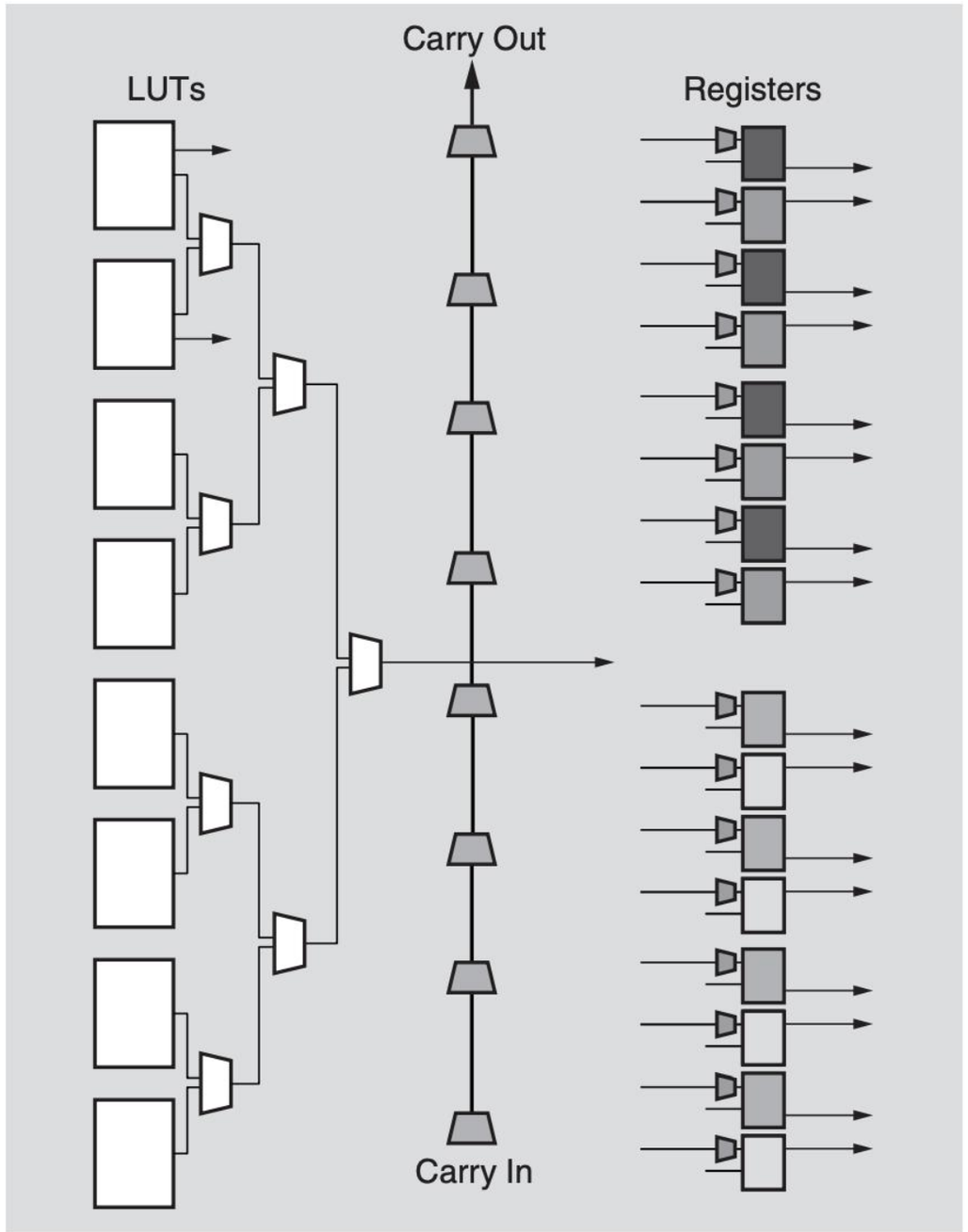
- In reality: LUT with larger number of inputs are more capable



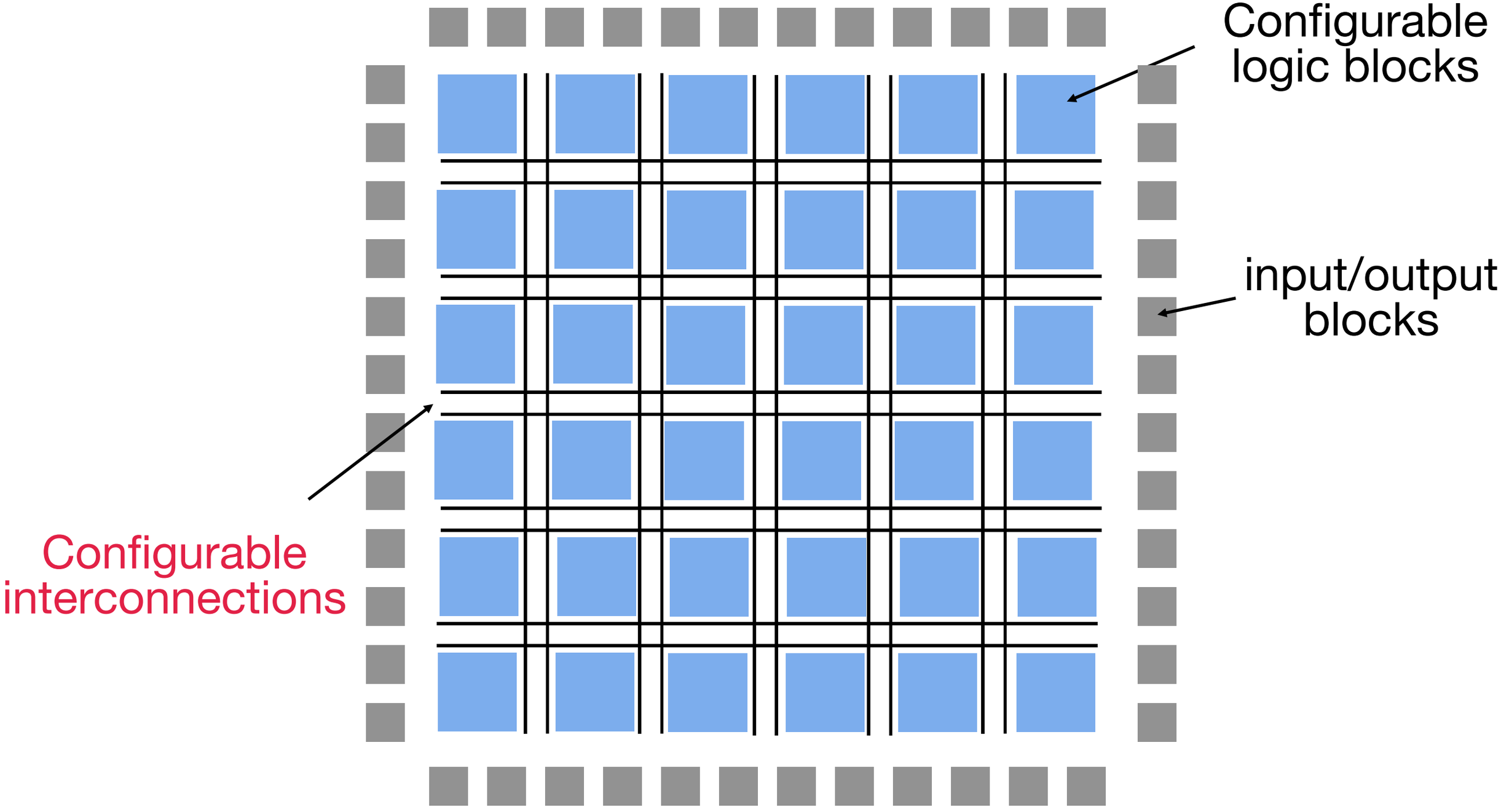
6-input LUT with bypass path & registers

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

Images from AMD



Configurable interconnections



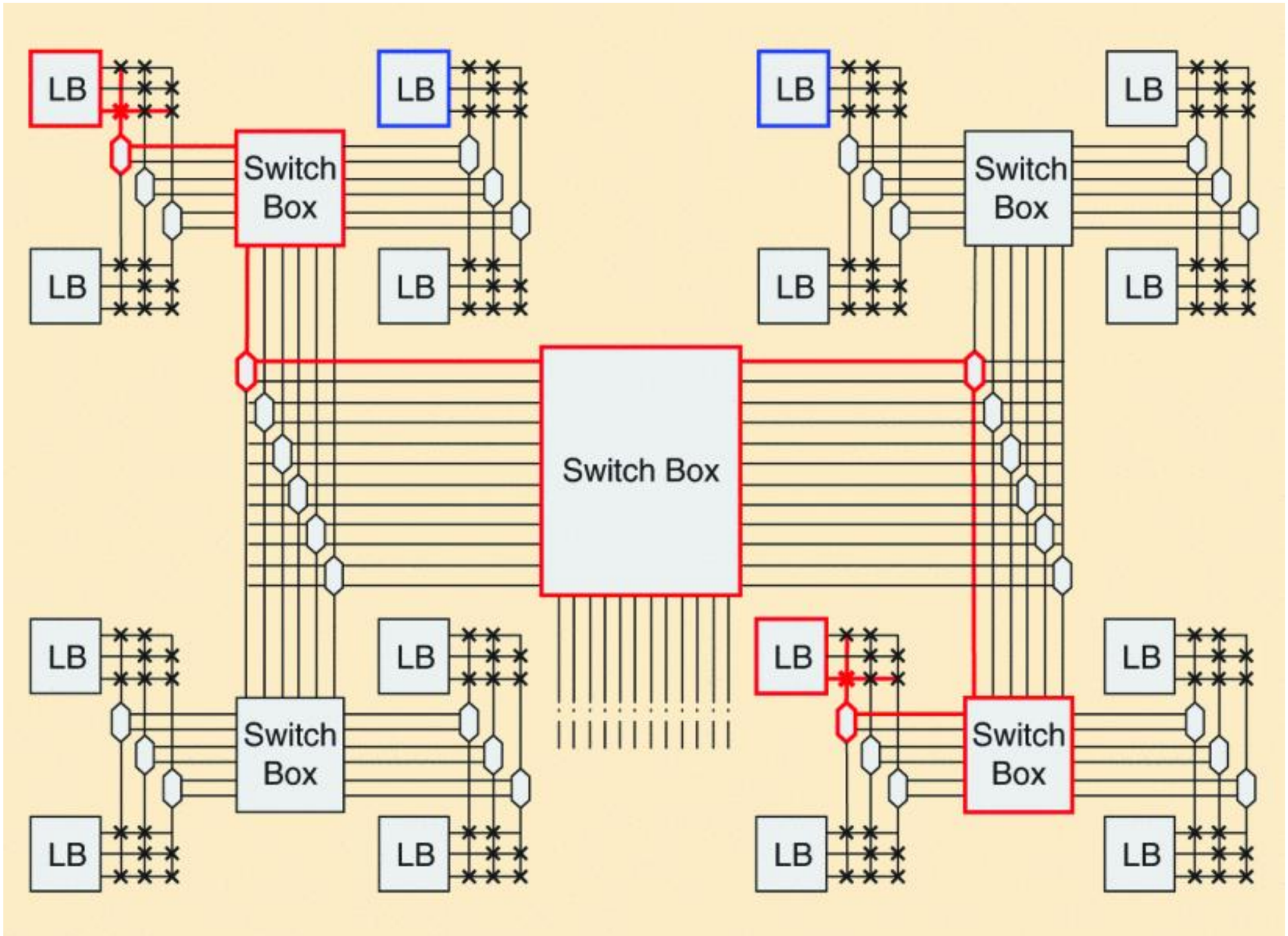
Configurable interconnections

- 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

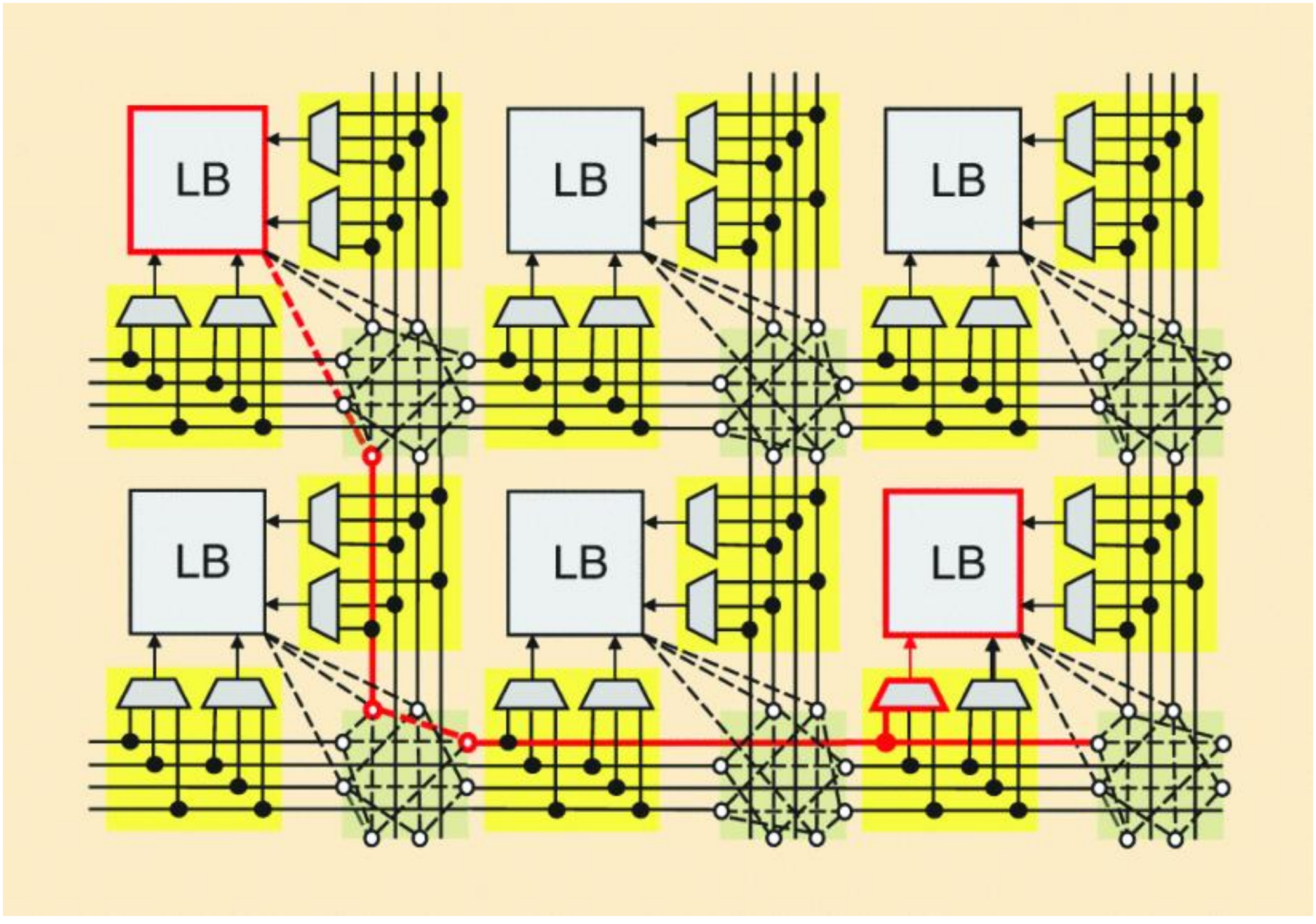
- Routing, a.k.a., interconnecting
  - Through programmable wires and switches
  - Between logic blocks (CLBs), and between I/O blocks and logic blocks
- Routing is a challenging problem
  - Routing technique used in an FPGA largely decides the amount of area used by wire segments and programmable switches, 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



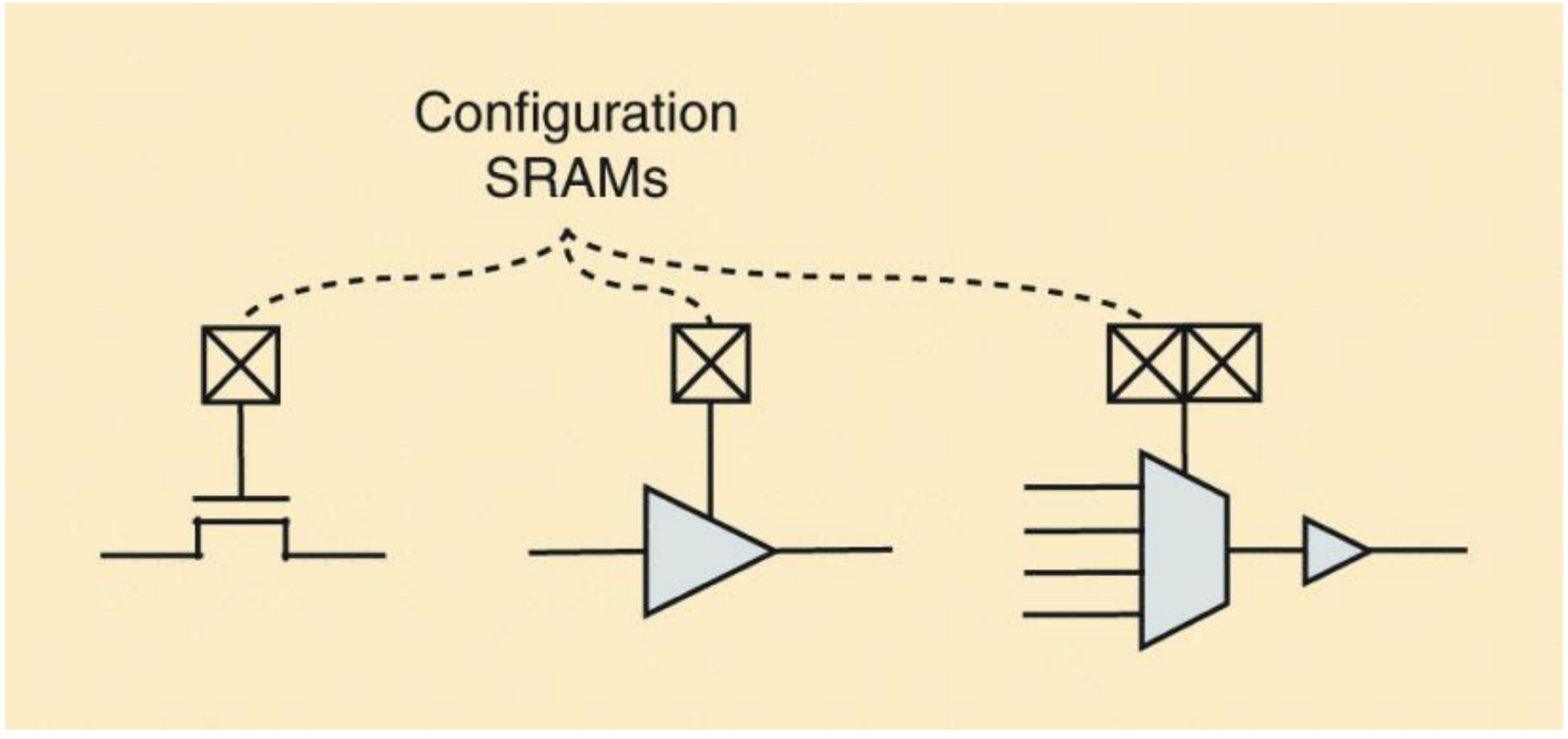
A. Boutros and V. Betz, "FPGA Architecture: Principles and Progression," in *IEEE Circuits and Systems Magazine*, vol. 21, no. 2, pp. 4-29, Secondquarter 2021.

Island-style FPGA



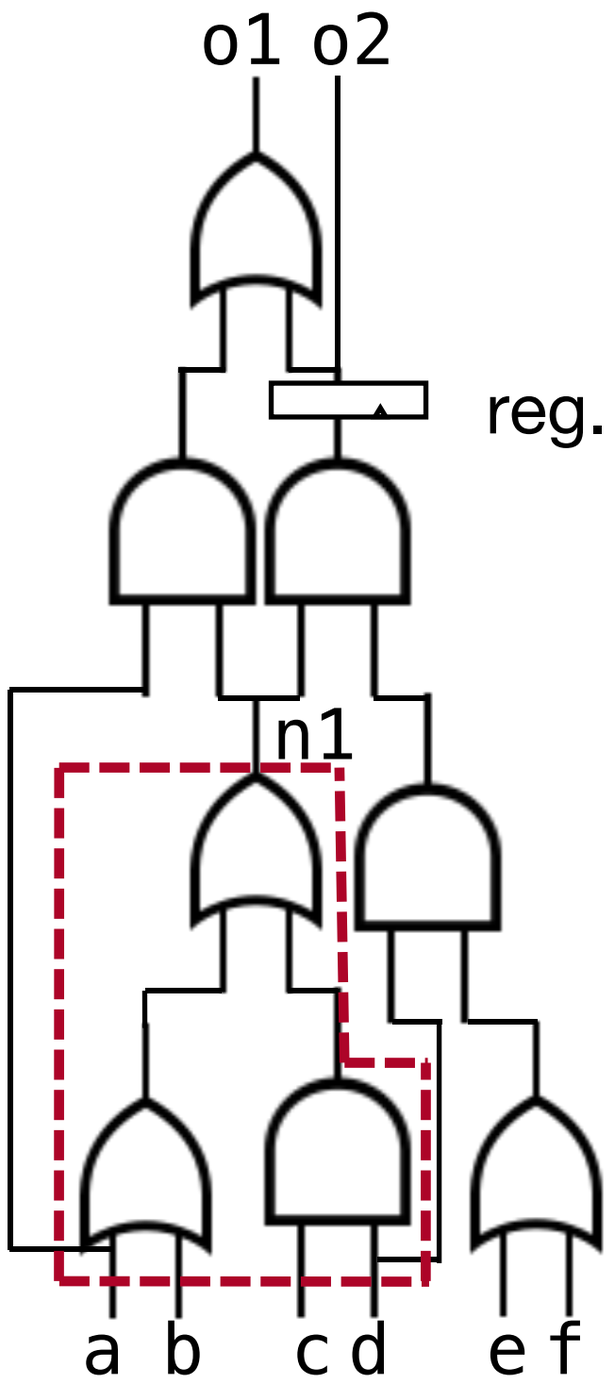
A. Boutros and V. Betz, "FPGA Architecture: Principles and Progression," in *IEEE Circuits and Systems Magazine*, vol. 21, no. 2, pp. 4-29, Secondquarter 2021.

Programmable switches

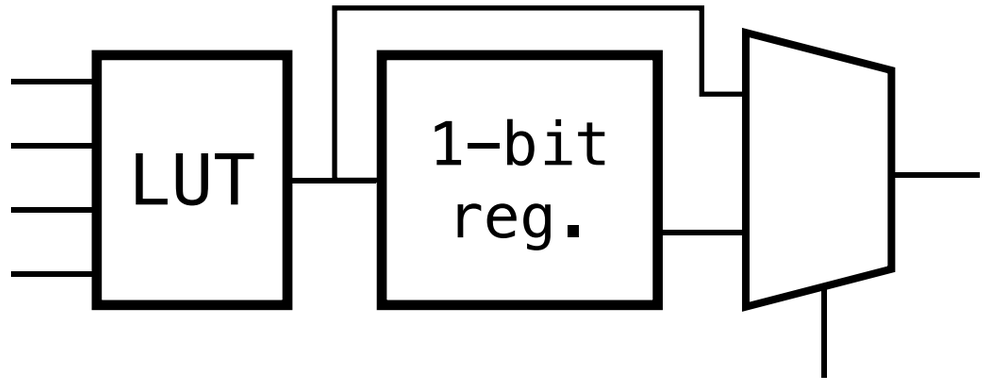


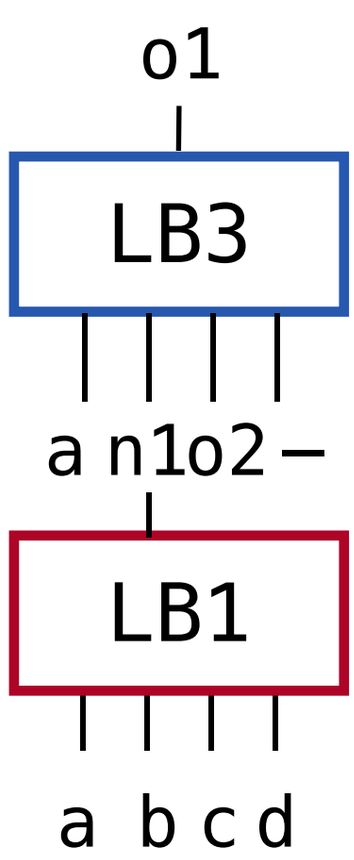
A. Boutros and V. Betz, "FPGA Architecture: Principles and Progression," in *IEEE Circuits and Systems Magazine*, vol. 21, no. 2, pp. 4-29, Secondquarter 2021.

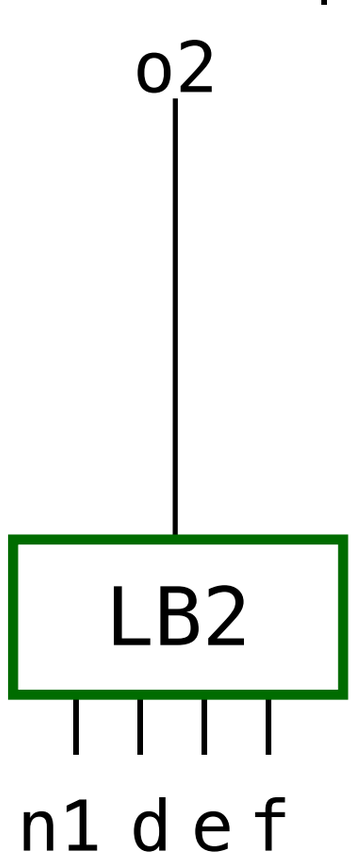
FPGA mapping



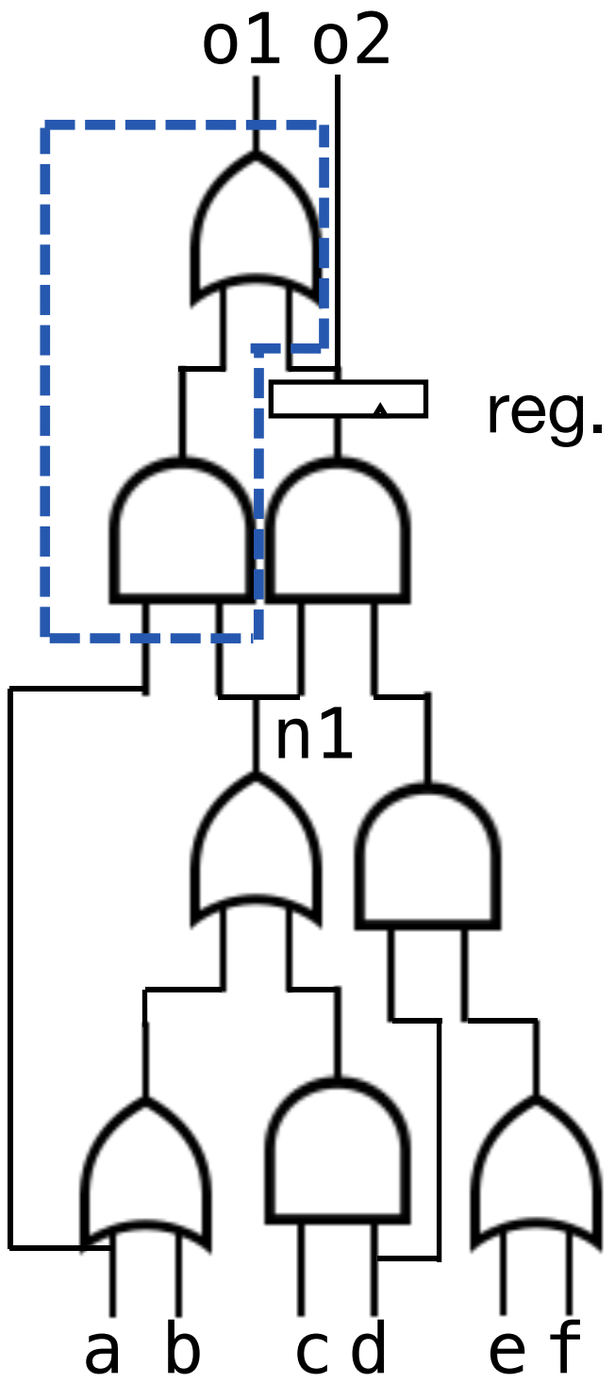
- Assume we use 4-input LUT, each followed by a 1-bit register/bypass path



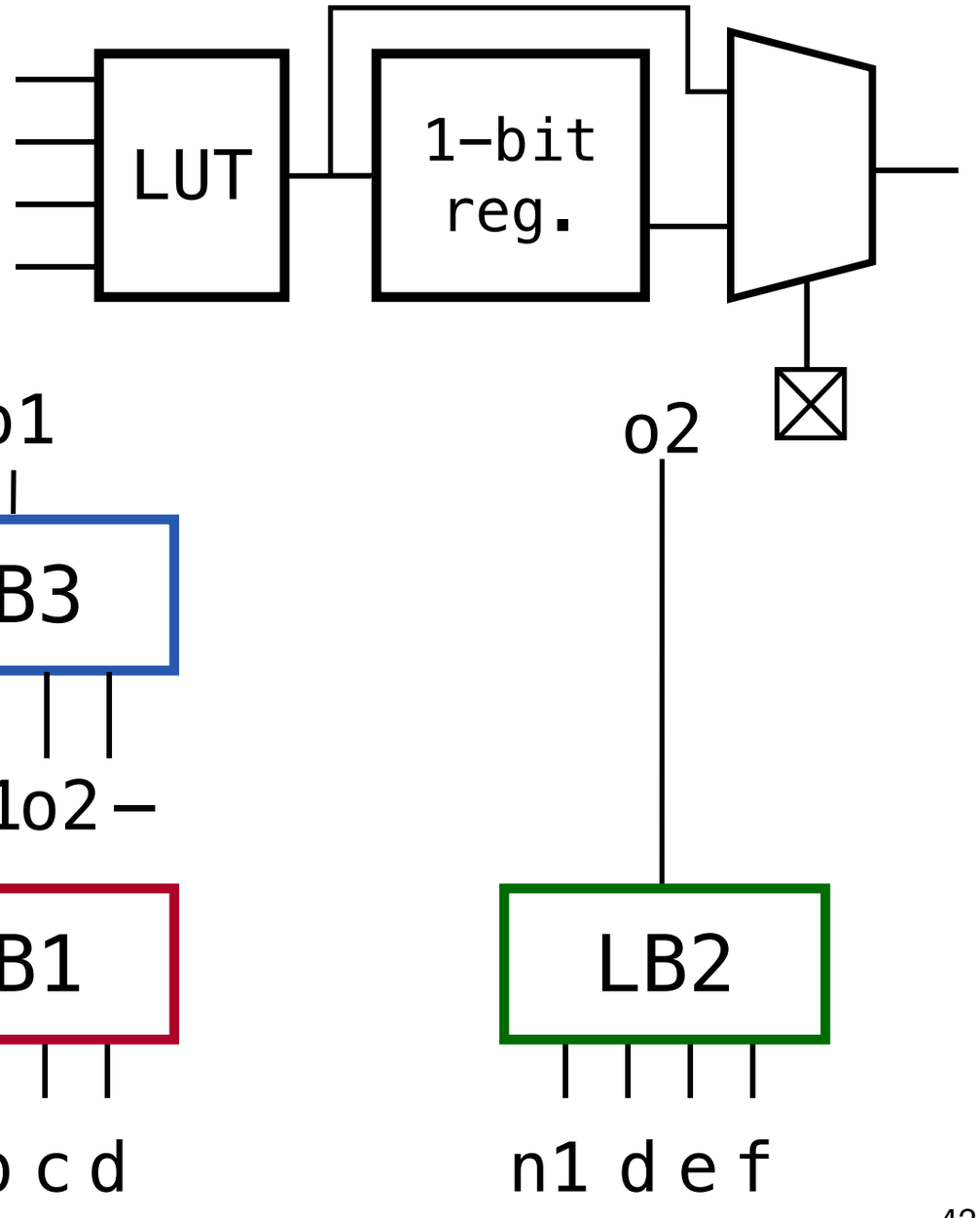




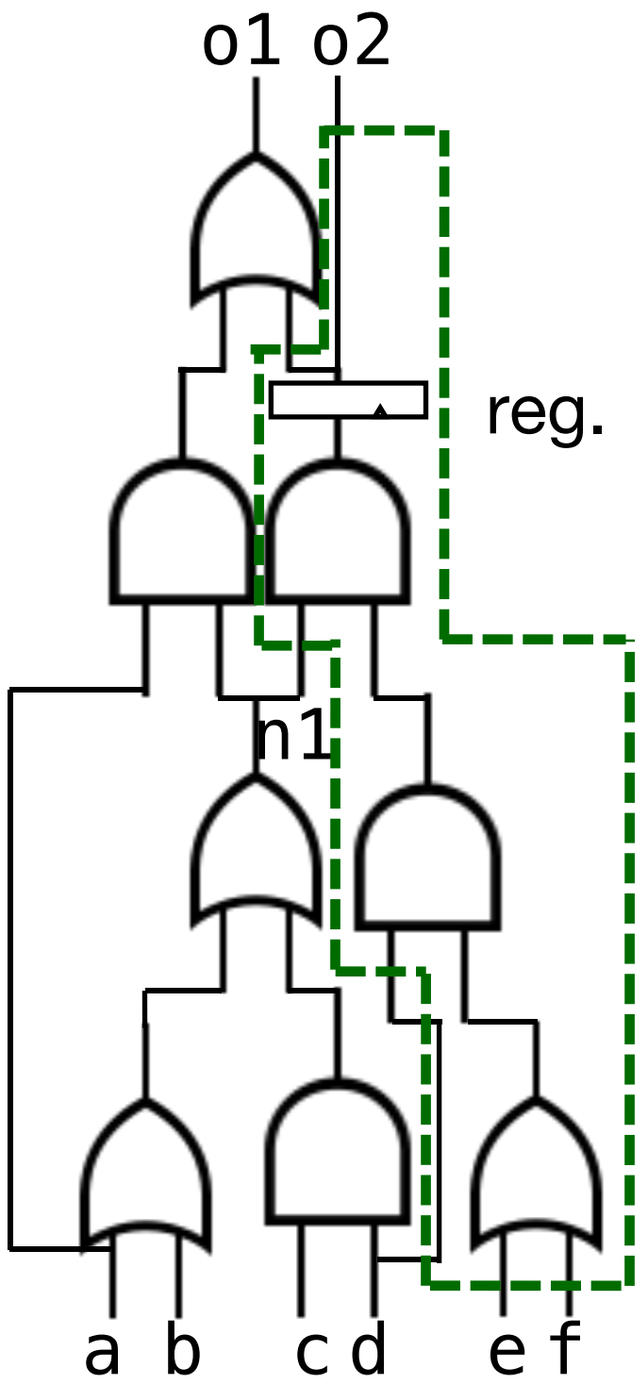
FPGA mapping



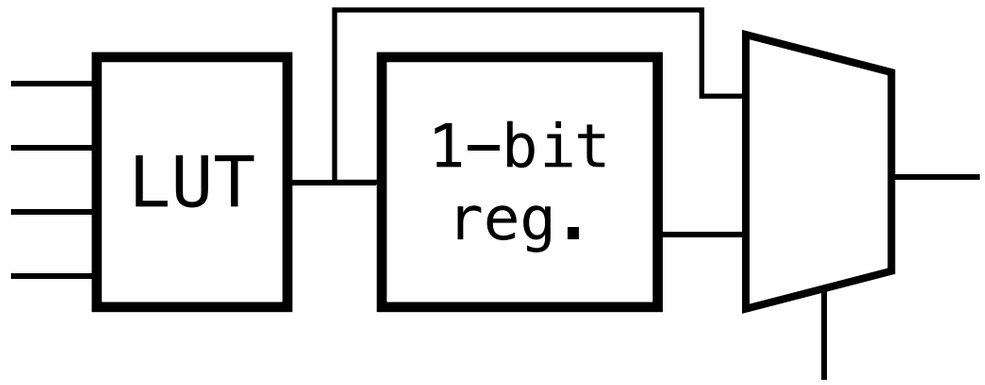
- Assume we use 4-input LUT, each followed by a 1-bit register/bypass path

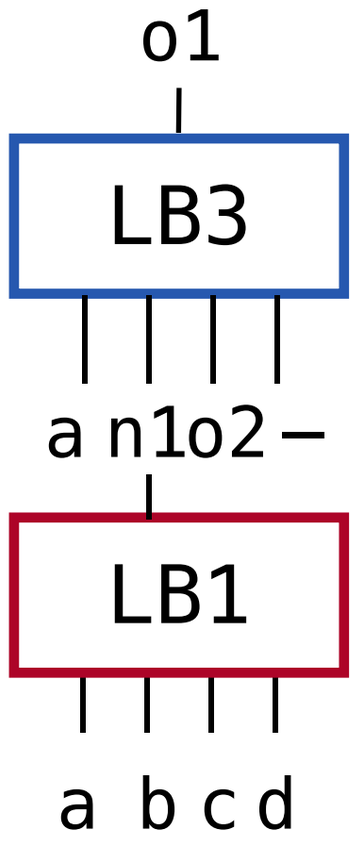


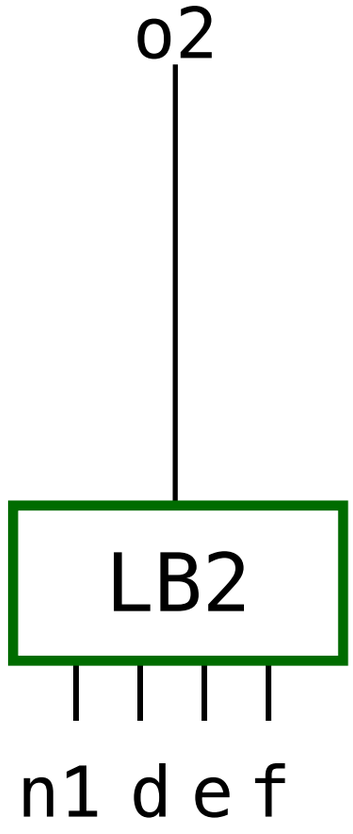
FPGA mapping



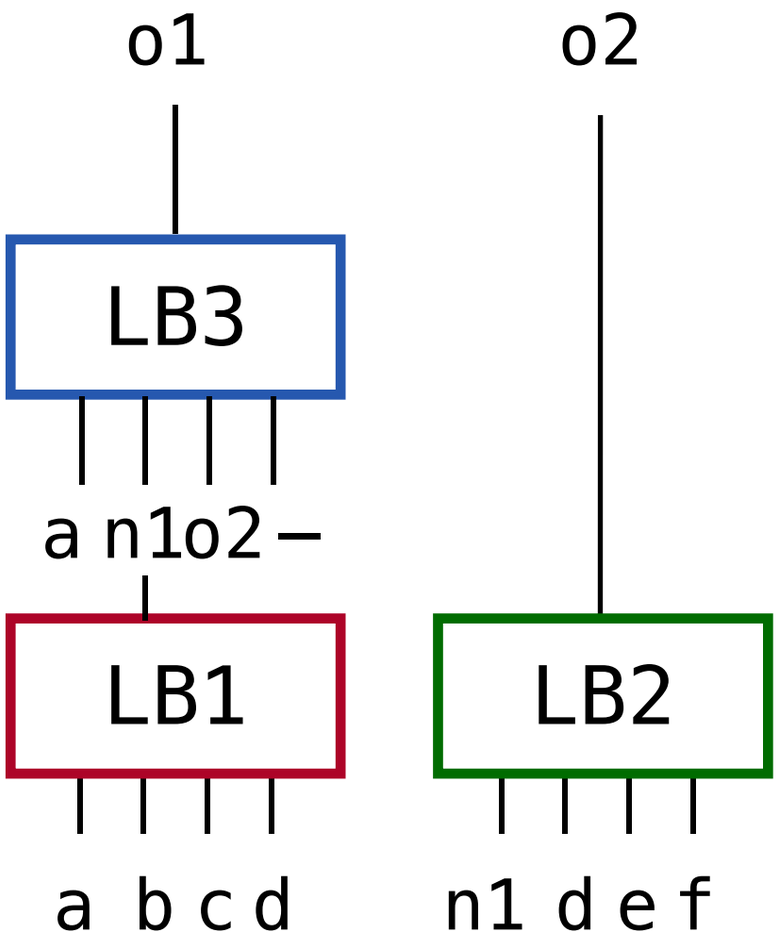
- Assume we use 4-input LUT, each followed by a 1-bit register/bypass path

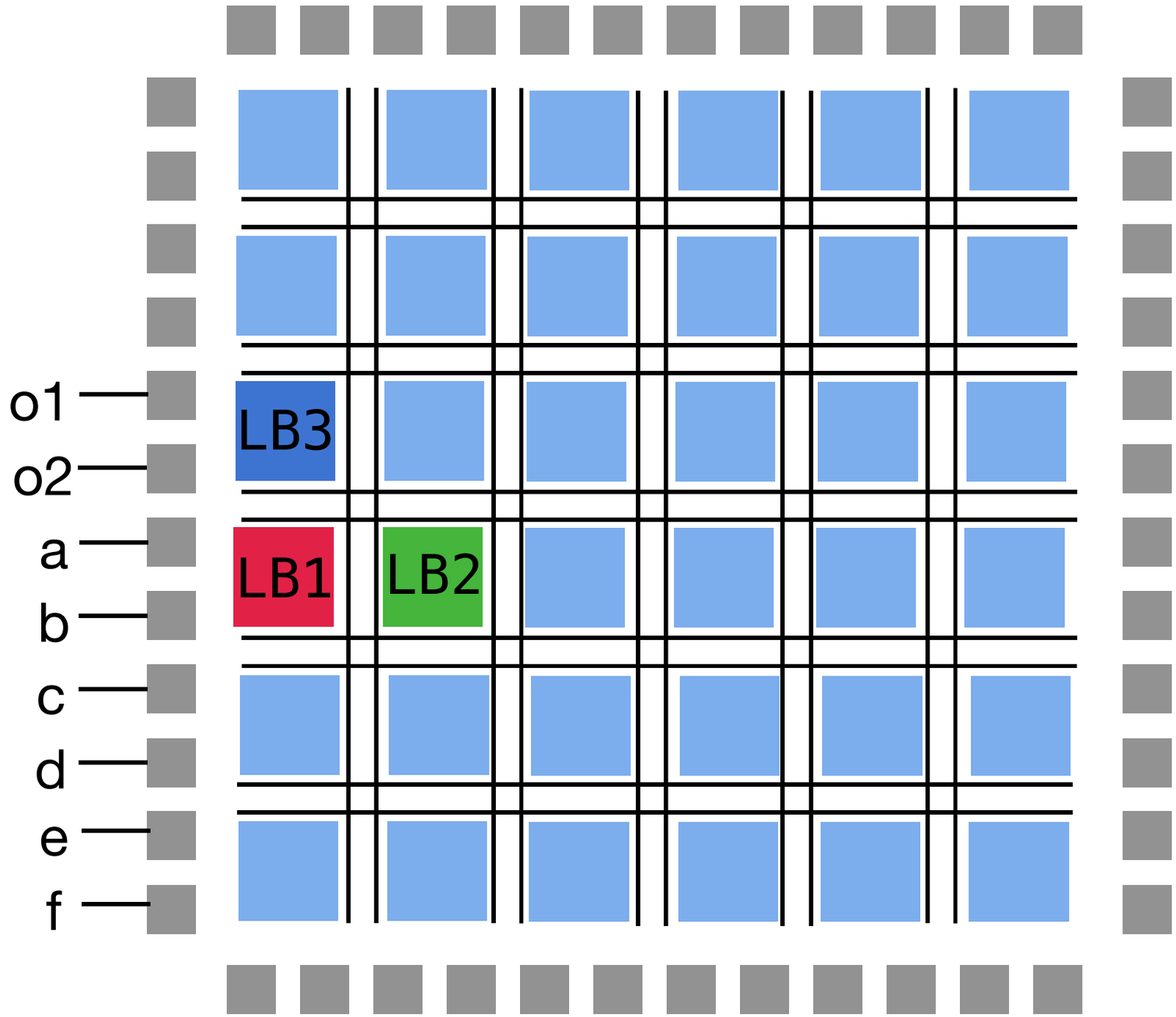






FPGA placement & routing



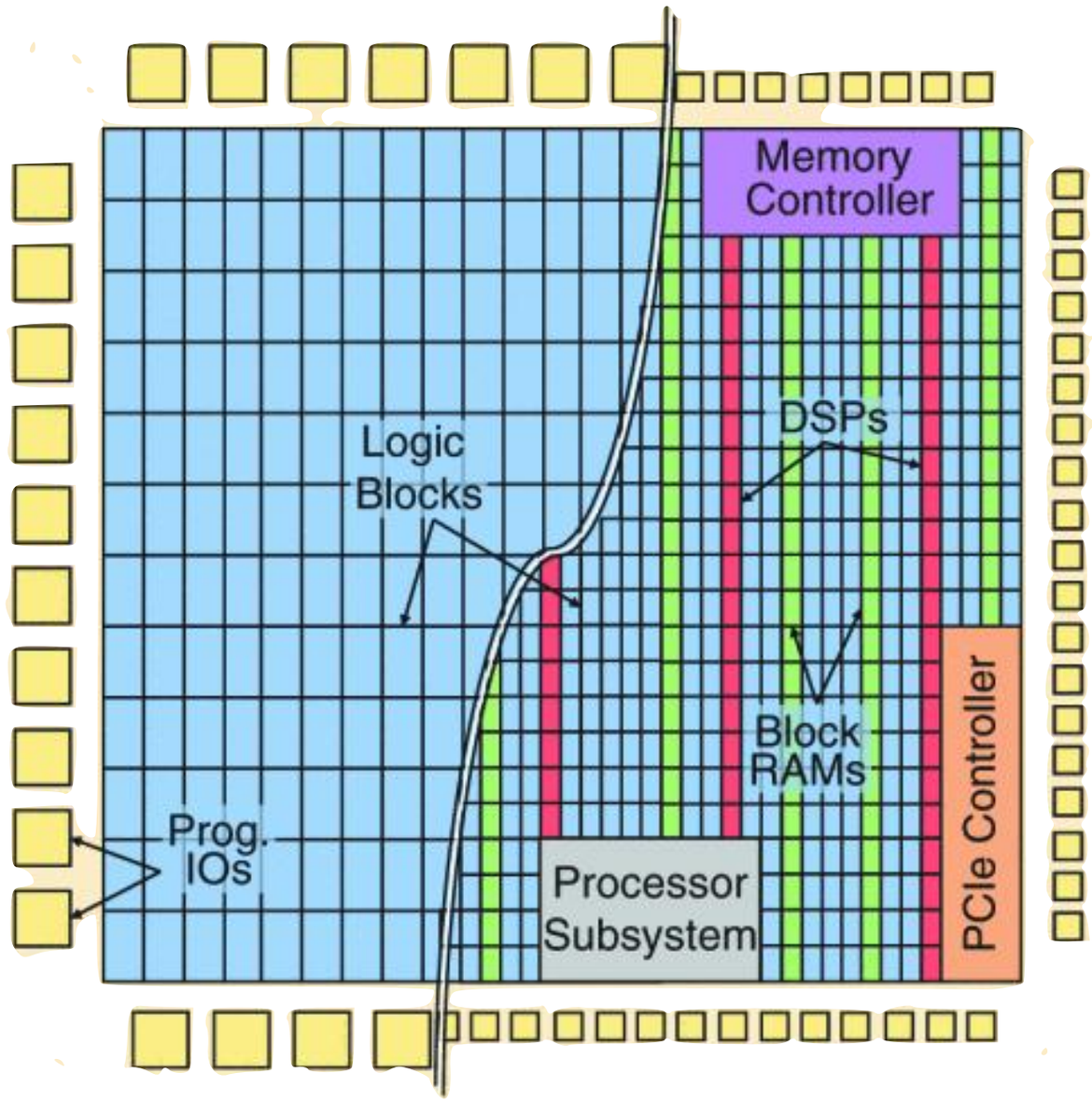


After P&R, generate bitstream file to configure the SRAM cells

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

Traditional FPGAs



Heterogeneous

Modern FPGAs

More in EE116/216

Question

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