Embedded Intelligent System and Novel Computer Architecture

Lecture 06 – FPGA (Field Programmable Gate Array) and CGRA (Coarse-Grained Reconfigurable Architectures)

Pengju Ren
Institute of Artificial Intelligence and Robotics
Xi'an Jiaotong University

http://gr.xjtu.edu.cn/web/pengjuren

Three Important Trends

Requirement： Big Data
Advances in ML, Data Analytics
(Challenges: Data-driven discovery, Search, Analyze data in real time)

Increasing compute & memory requirements

Limitations： Moore' Law
Dennard Scaling
(Power\Memory\Utilization Wall)

Specialization: Domain Specific Architecture
Performance / Watt is key

Challenges： Algorithms change rapidly
High NRE Costs with ASICs

Flexible Hardware

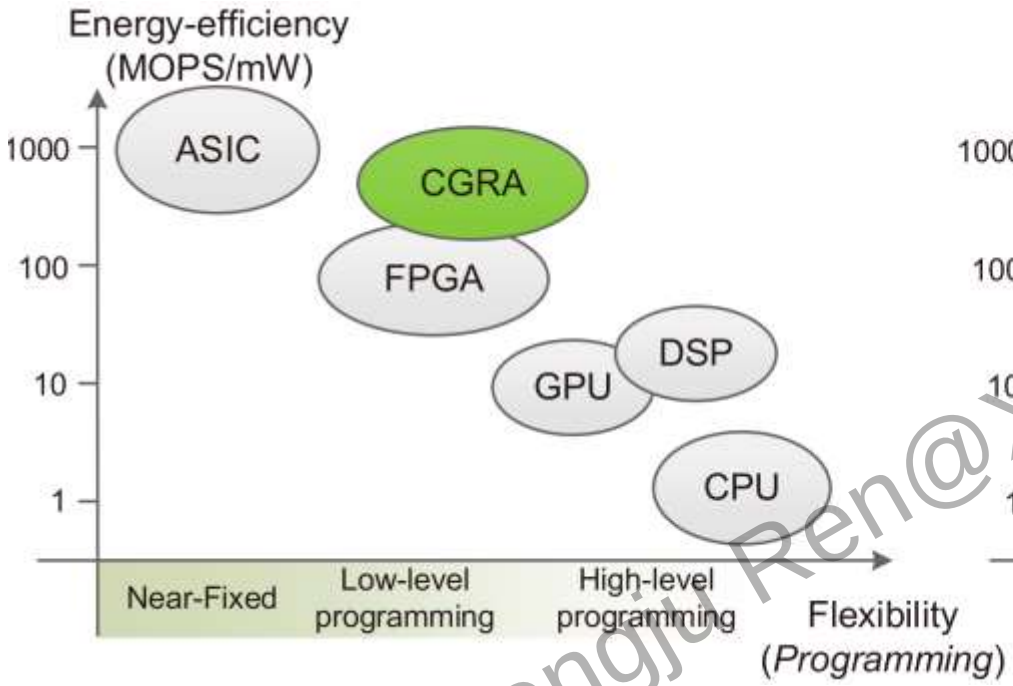
Flexibility: Instructions

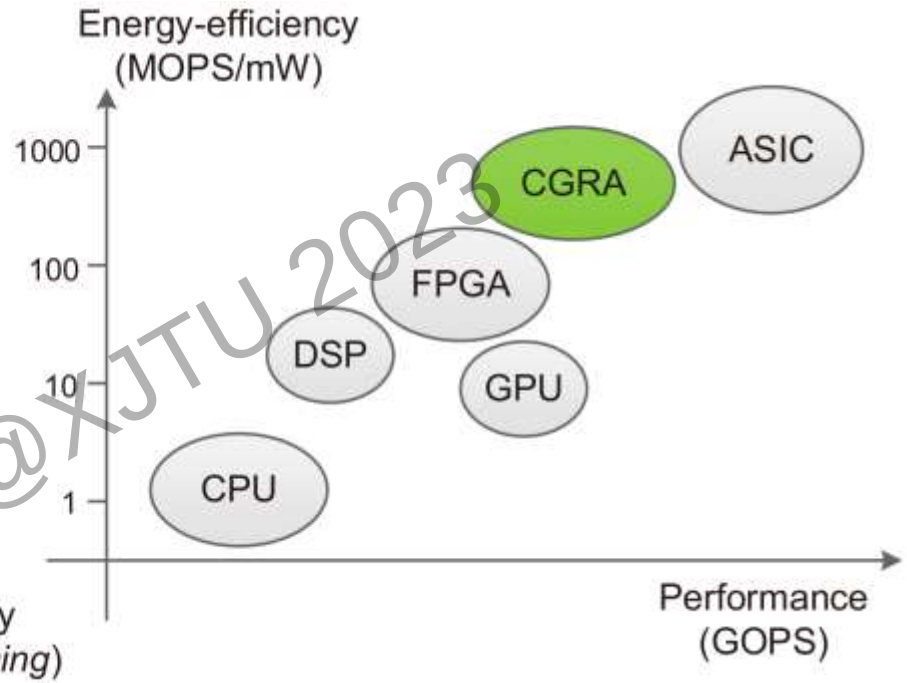
Instructions add overheads:

- **Instruction fetch, decode, register file**
- **Roughly 40% of datapath energy on CPU[2]**
- **Roughly 30% of dynamic power on GPU[3]**

[1] Mark Horowitz, Computing's Energy Problem (and what we can do about it), ISSCC 2014
[2] Hameed et al, Understanding Sources of Inefficiency in General-purpose Chips, ISCA 2010
[3] Leng et al, GPUWattch: Enabling Energy Optimizations in GPGPUs, ISCA 2013

Flexibility: Reconfigurable Hardware





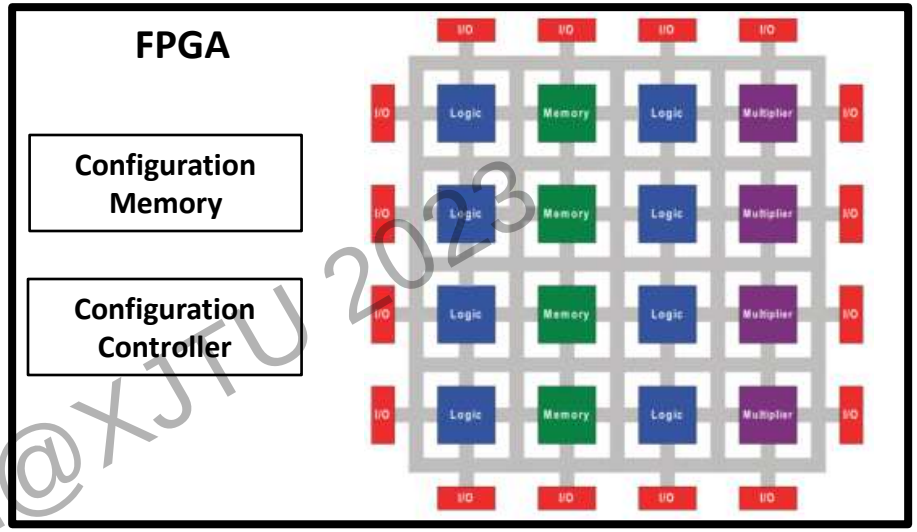
- **FPGAs, CGRAs are pre-fabricated silicon devices that can be reprogrammed using a stream of configuration bits.**
  - **Sea of reconfigurable elements**
  - **Programmable interconnect**
  - **Statically programmed**
  - **No instruction overheads**

Fine-grained v.s Coarse-grained RA

Fine-Grained RA

Configure at bits-level with connected flexibly

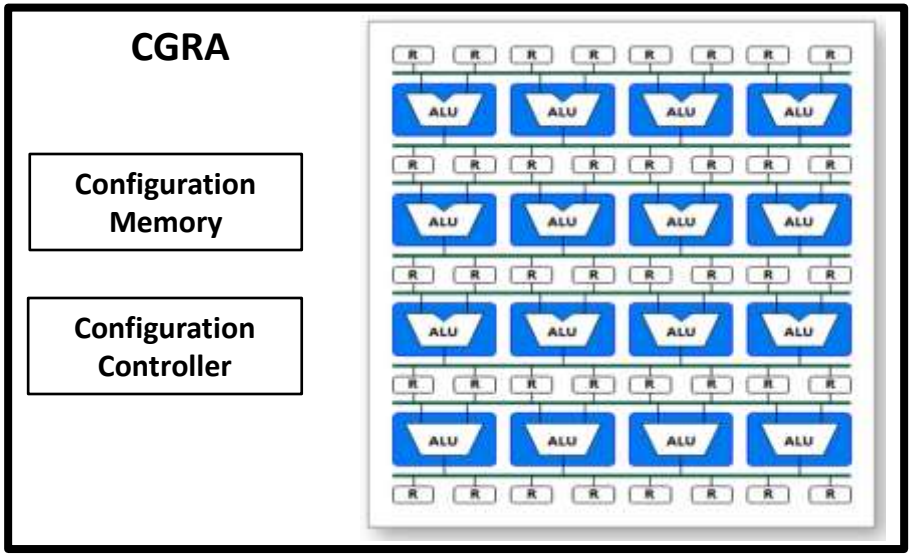
Example: FPGA



Coarse-Grained RA

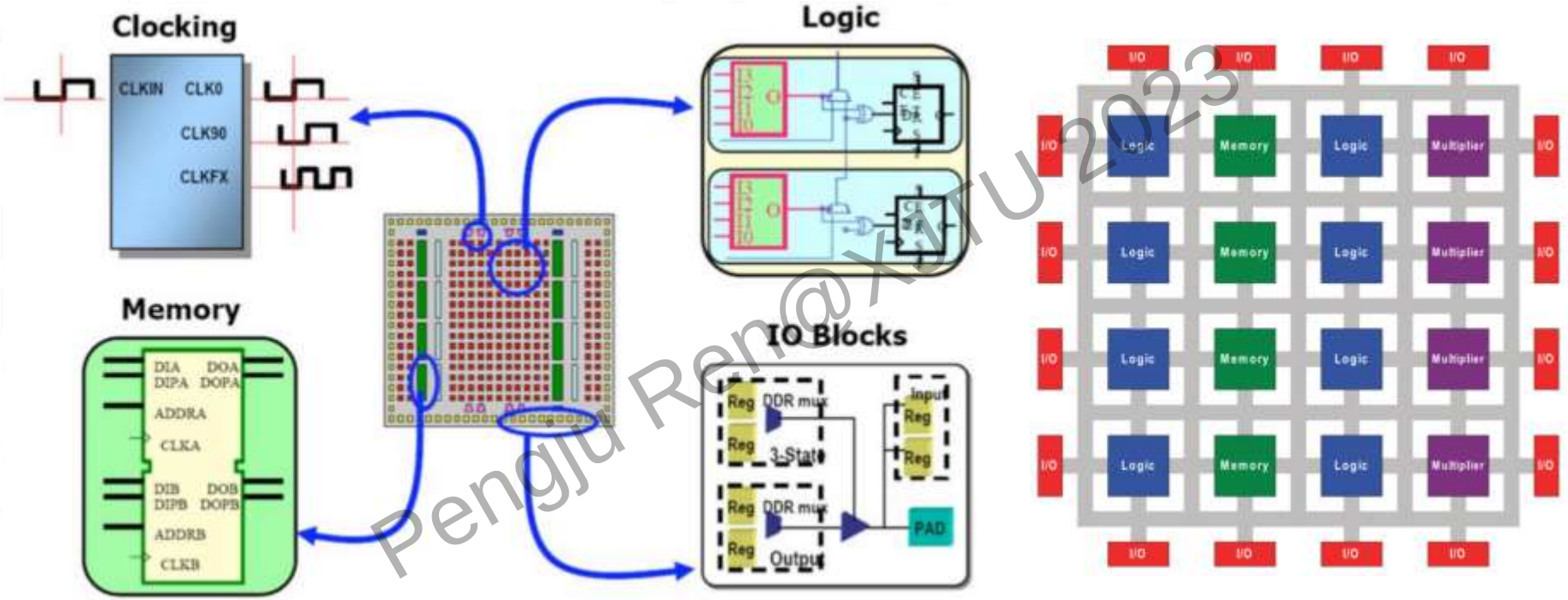
Configure at word-level(16/32 INT/FP) with local connectivity

Example: CGRA



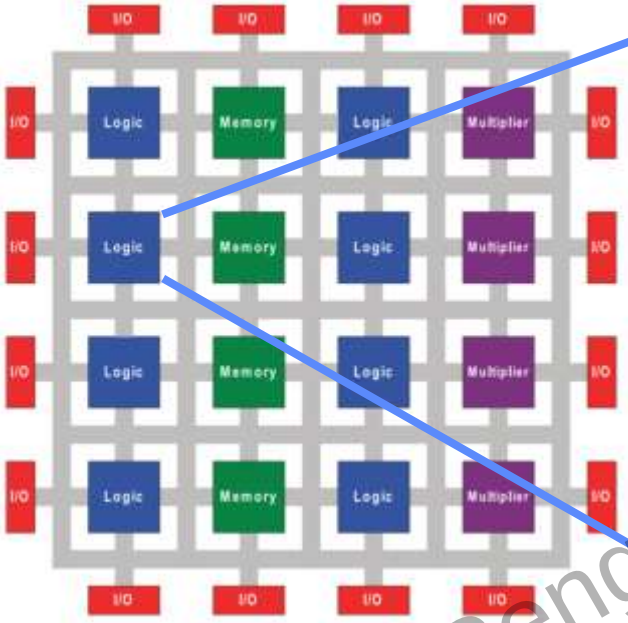
What is an FPGA ?

FPGA = Field Programmable Gate Array

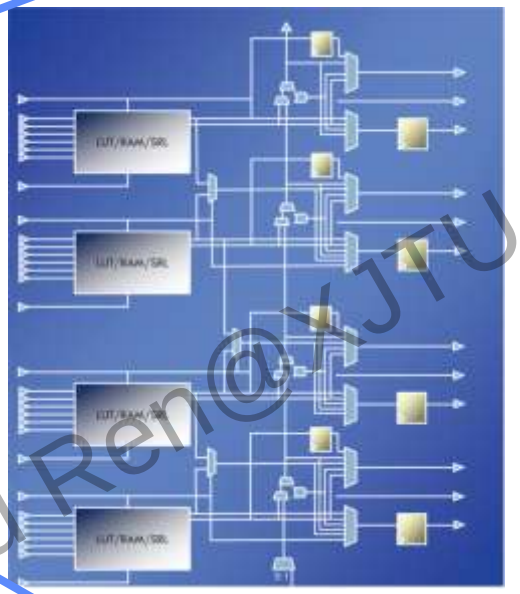


Programmable logic, Clock, Interconnections and Routing
Programmable in System (ISP)
Dedicated Blocks: Memory\Clock Control\DSP blocks\Embedded Processor\I/O blocks

What is an FPGA ? – Configurable Logic Block



Basic FPGA structure



Configurable Logic Block(CLB)

Or Logic Element(LE)

Look-up Tables(LUTs)

Flip-flop

Outputs:

- Combinational
- Registered

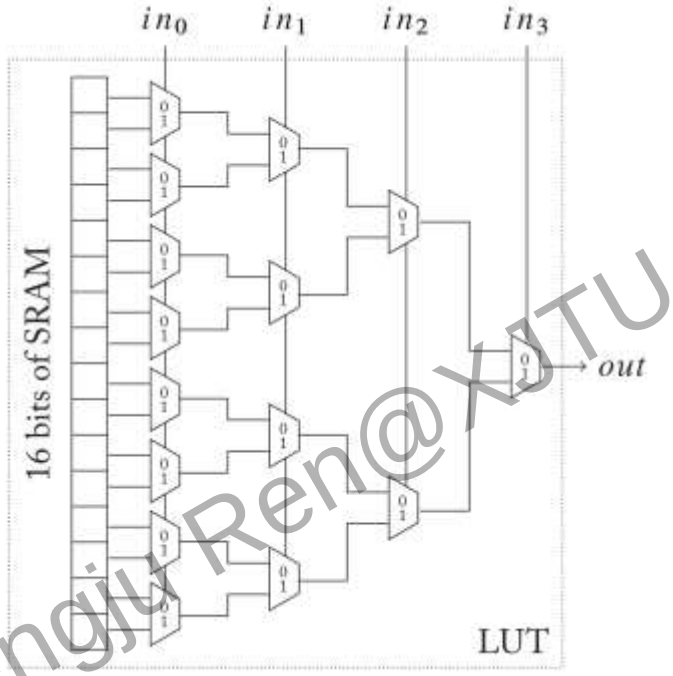
I/O carry chain

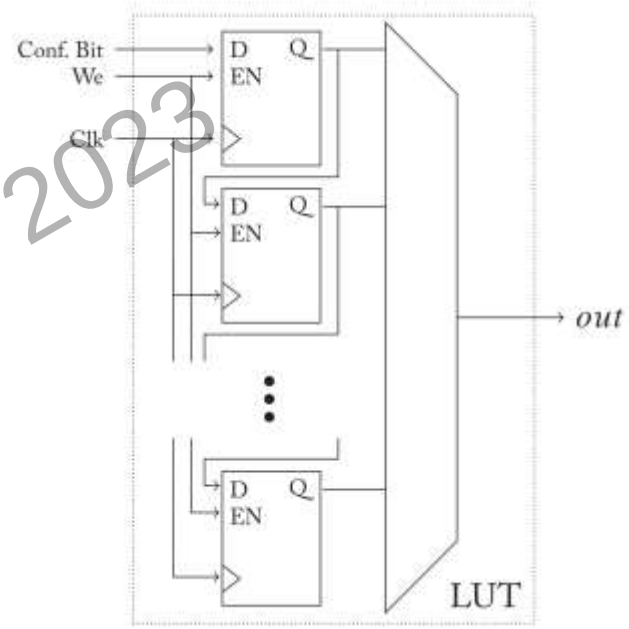
…

CLBs contains： LUTs for creating arbitrary combinatorial logic functions
flip-flops for clocked storage elements,
multiplexers to route the logic within the block and to and from external resources
The muxes also allow polarity selection and reset and clear input selection.

Look-Up Table——the Key to re-programmability

Inputs				Output
In0	In1	In2	In3	out
0	0	0	0	#
0	0	0	1	#
0	0	1	0	#
0	0	1	1	#
0	1	0	0	#
0	1	0	1	#
0	1	1	0	#
0	1	1	1	#
1	0	0	0	#
1	0	0	1	#
1	0	1	0	#
1	0	1	1	#
1	1	0	0	#
1	1	0	1	#
1	1	1	0	#
1	1	1	1	#

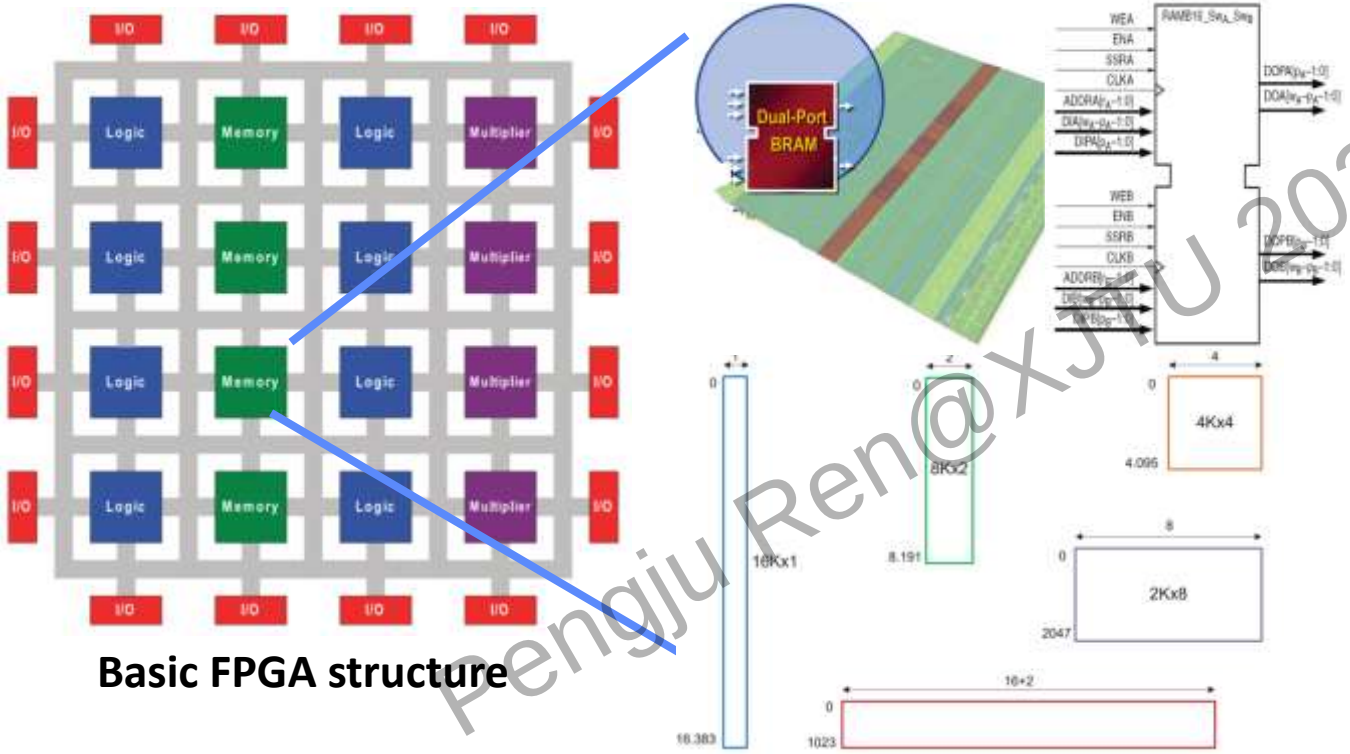




Reading(left) and writing(left) of 4-input LUT

- An **n-input LUT** can be used to implement an arbitrary Boolean-valued function with up to n Boolean arguments
- LUTs can also be used as ***memory elements***, small FIFO, Shift Registers

What is an FPGA ? – Distributed Memories



Basic FPGA structure

Different Sizes

Multiple configuration:

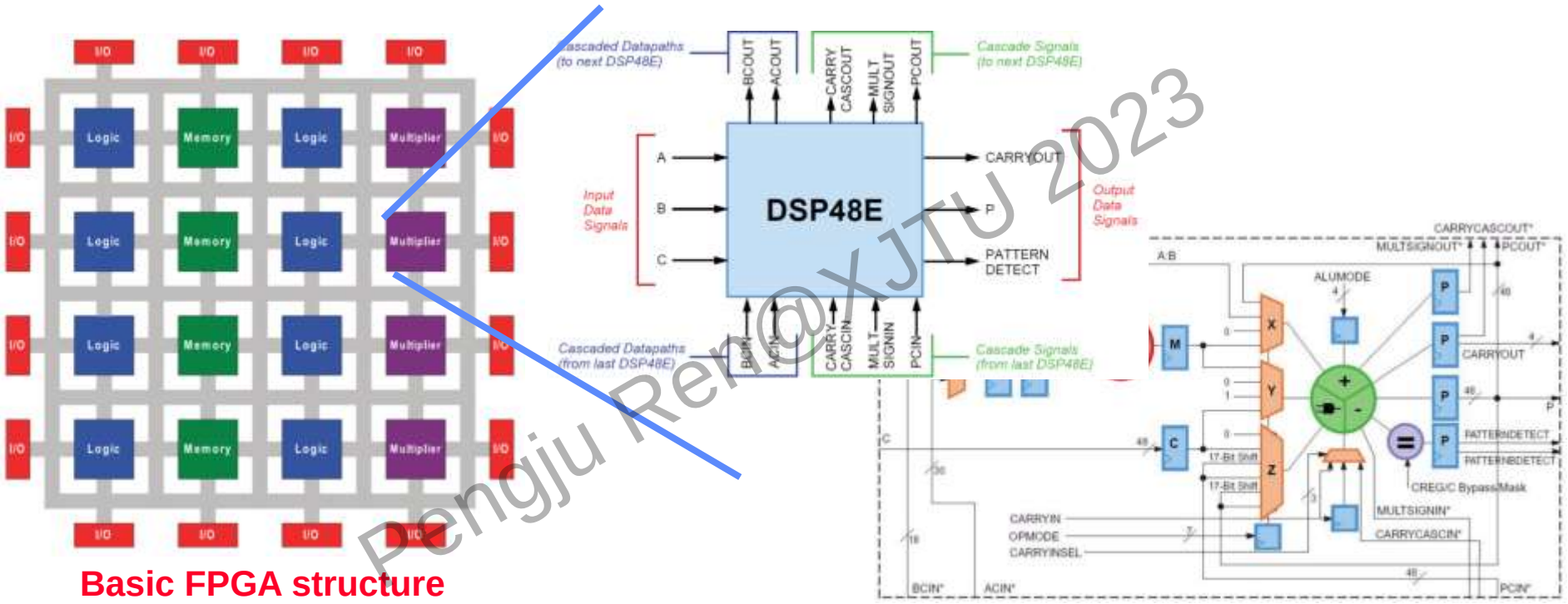
Dual-port (Async or Sync)

Different formats:

Word-wide, Depth, …

- **The FPGA fabric includes embedded memory elements that can be used as random-access memory (RAM), read-only memory (ROM), or shift registers. A single BRAM block can hold a few kilobytes of data (e.g., 4 KiB), a few hundred BRAMs can be accessed in parallel.**
- **BRAMs can be used for clock domain crossing and bus width conversion in an elegant way.**

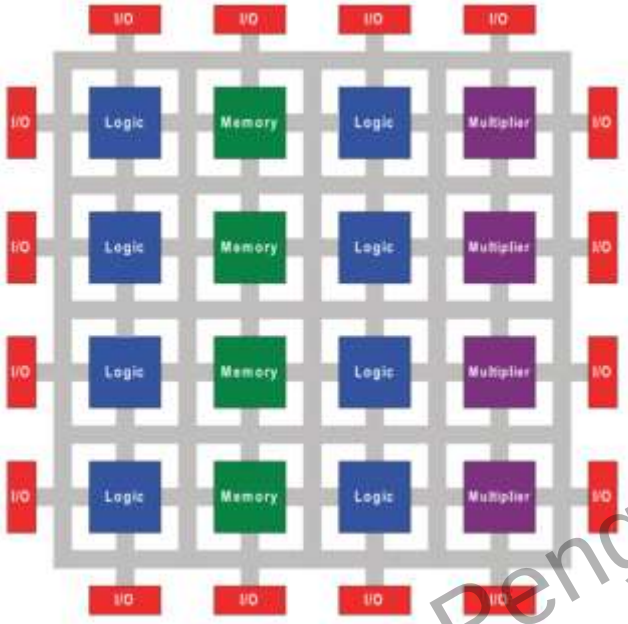
What is an FPGA ? - DSP



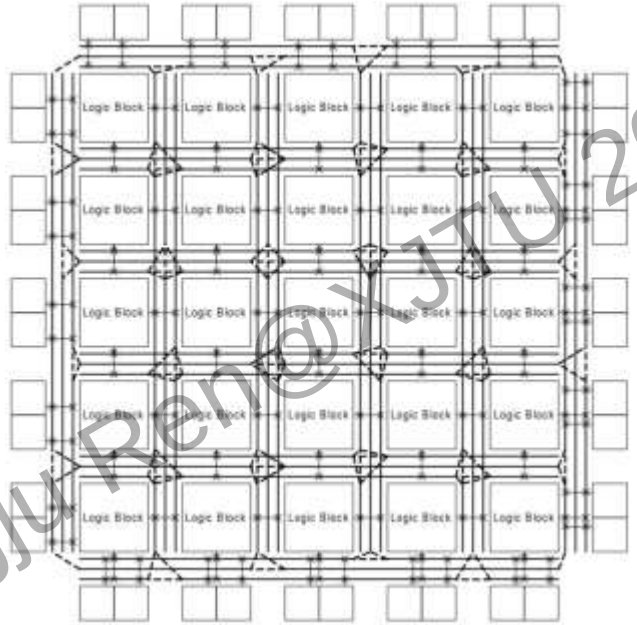
Basic FPGA structure

- Xilinx DSP48E slice has three input ports (which are 25 bits, 18 bits, 48 bits wide) and provides a 25x18-bit multiplier in combination with a pipelined second stage that can be programmed as 48-bit substractor or adder with optional accumulation feedback.
- DSP units can be used in a variety of modes, and perform operations such as multiply, multiply-and-accumulate, multiply-and-add/subtract, three input addition, wide bus multiplexing, barrel shifting, etc.

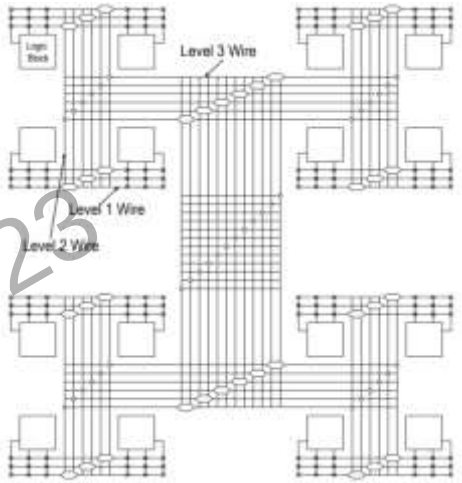
What is an FPGA ? – Configurable Connections



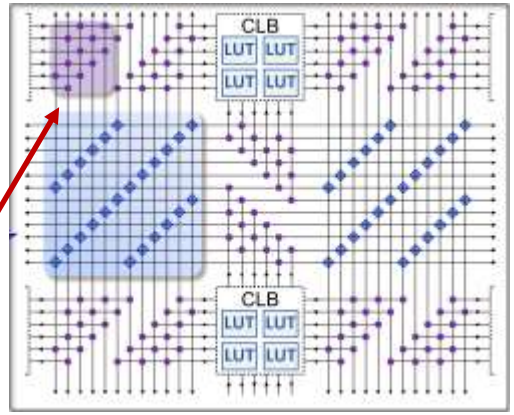
Basic FPGA structure



Island-style(Distributed)



Hierarchical Connection

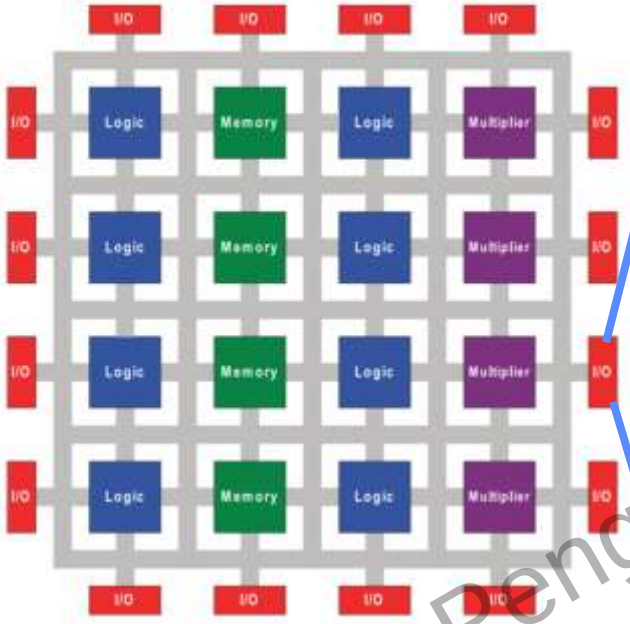


Programmable switches

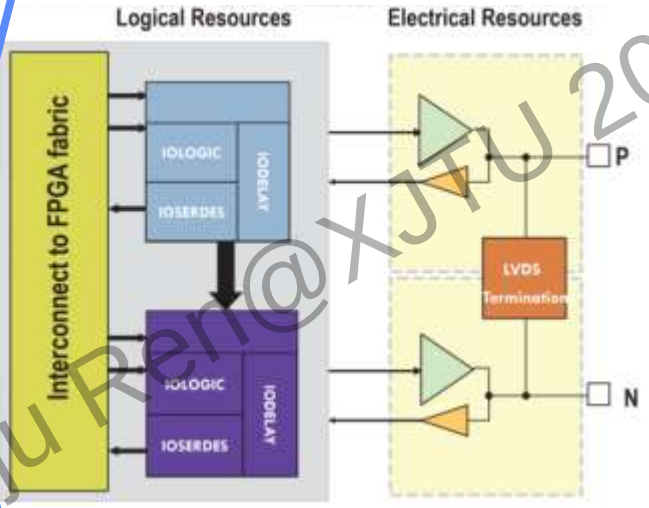
Distributed Connection

The programmable routing in an FPGA provides connections among logic blocks and I/O blocks to complete a user-designed circuit. It consists of wires and programmable switches

What is an FPGA ? – Configurable I/O



Basic FPGA structure



Configurable I/O Block(IOB)

IO-Serdes:

- **Parallel to serial**
- **Serial to Parallel**

IO-Delay

IO-Standard:

- **LVCMOS (3.3, 2.5, 1.8.1.2V)**
- **SSTL**
- **LVDS**

…

The I/O block (IOB) is used to drive signals to the pins of the CPLD device at the appropriate voltage levels with the appropriate current. Two main classes of I/O standards being single-ended (used, e.g., in PCI) and for higher performance differential (used, e.g., in PCI Express, SATA, 10G Ethernet, etc.)

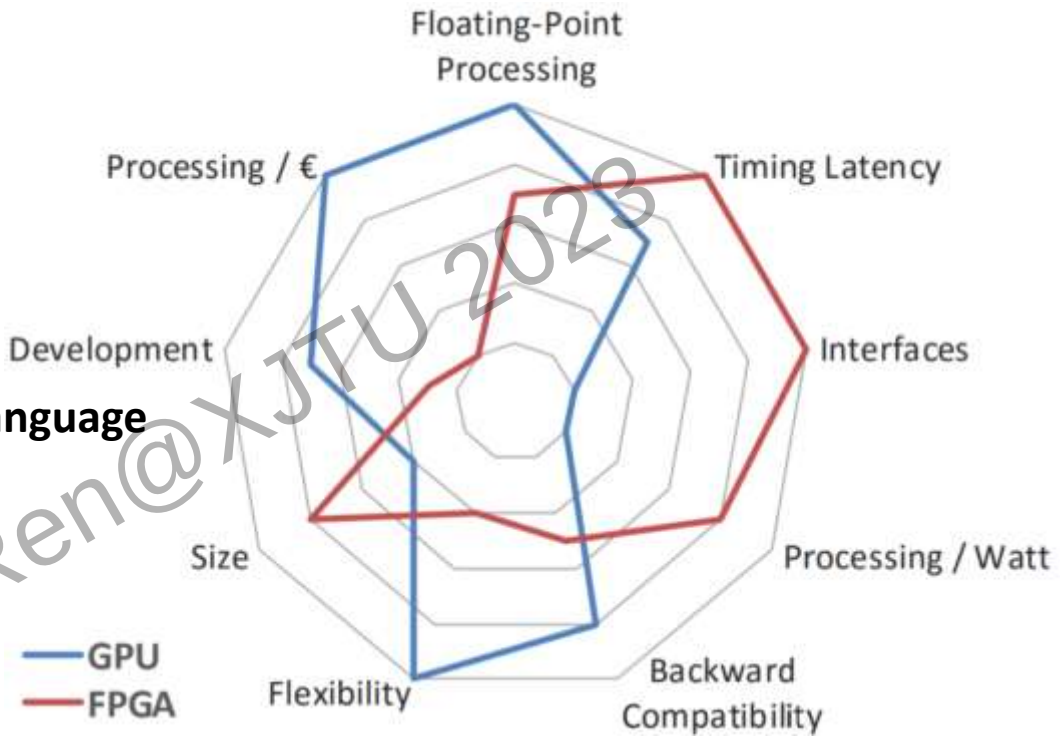
FPGA v.s GPU

GPU (Instruction-based)

- **Ideal for SIMD (SIMT)**
- **Suited for Floating Point**
- **Thousands of concurrent threads**
- **Develop using Parallel Processing Language**

FPGA (Could be data-driven)

- **HW-like Performance**
- **Deterministic**
- **Highly Parallelizable**
- **Limited Resources**
- **Develop using HDL(Hardware Description Language)**



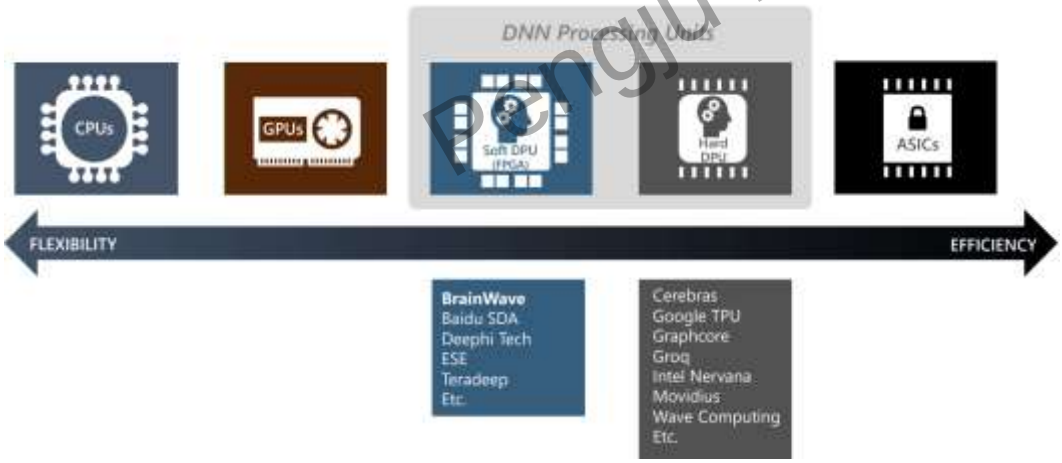
FPGA v.s CPU and GPU

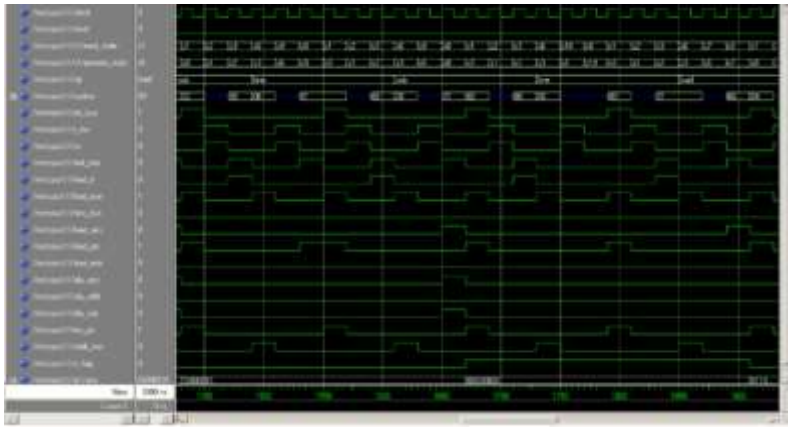
A processor is programmed with instructions (CPU and GPU)

FPGA contains configurable blocks with logics and configurable connection lines between these blocks, it is programmed with a circuit description.

Pros and Cons:

- **Low-level hardware Control and Data movement Operations (Low Programming Efficiency)**
- **Flexibility comes at the cost of large compile (place/route) and debug times**
- **FPGA Engineers are hard to hire**





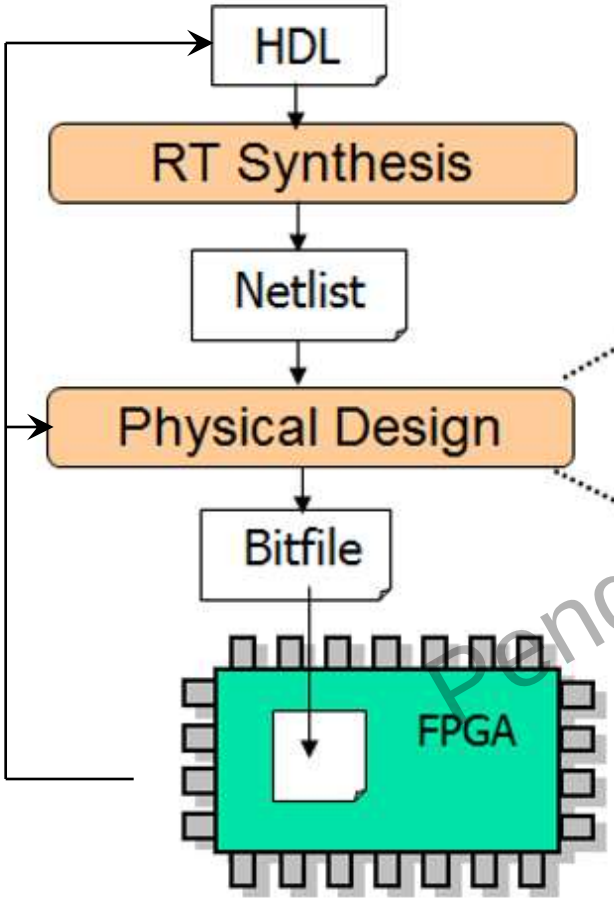
FPGA v.s CPU and GPU

Feature	Analysis	Winner
Floating-point Processing	The total floating-point operations per second of the best GPUs are higher than the FPGAs' with the maximum DSP capabilities.	GPU
Timing Latency	Algorithms implemented into FPGA provide deterministic timing, with latencies one order of magnitude less than GPUs.	FPGA
Processing / Watt	Measuring GFLOPS per watt, FPGAs are 3-4 times better. Although still far away, latest GPU products are dramatically improving the power burning.	FPGA
Interfaces	GPUs interface via PCIe, while FPGA flexibility allows connection to any other device via -almost- any physical standard or custom interface.	FPGA
Backward Compatibility	Software developed for older GPUs will work in the new devices. FPGA HDL can be moved to newer platforms, but with some reworking.	GPU
Flexibility	FPGA lacks flexibility to modify the hardware implementation of the synthesized code, being a no-problem issue for GPUs developers.	GPU
Size	FPGA's lower power consumption requires less thermal dissipation countermeasures, implementing the solution in smaller dimensions.	FPGA
Development	Many algorithms are designed directly for GPUs, and FPGA developers are difficult and expensive to hire.	GPU

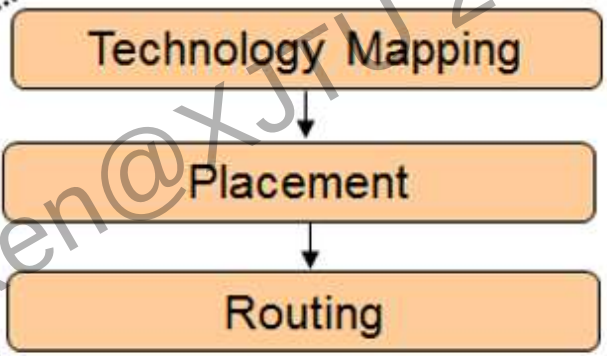
Table 1. Evaluation of FPGA and GPUs characteristics

FPGA Design Flow

Refining your design

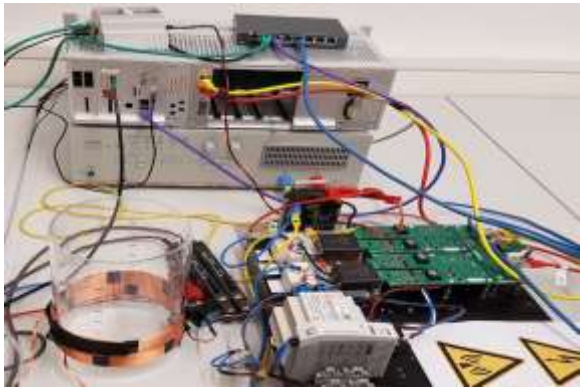


Using Hardware Description Language (Verilog RTL, VHDL, HLS) to implement your design following the SPEC



Divide whole design into sub elements

Complete physical implementation and optimize it with constraints



FPGA verification and debugging

Anatomy of a basic CGRA

- **Hardware Building blocks:** Compute, Memory, Interconnect
  - Compute: ALUs of varying capability
  - Memory: Programmer-managed scratchpads, caches
  - Interconnect: Statically programmed paths vs. dynamically routed data
- **Hardware Organization:** Topology
  - Data path hierarchy: ALUs vs. clusters of ALUs
  - Communication granularity: bit-level vs. word-level
  - Interconnect topology: Mesh, Torus, …
- **Software:** Programming Model
  - Software abstraction: Threads, VLIW, spatially configurable ALUs, …
  - Compiler technology to map high-level applications to CGRAs

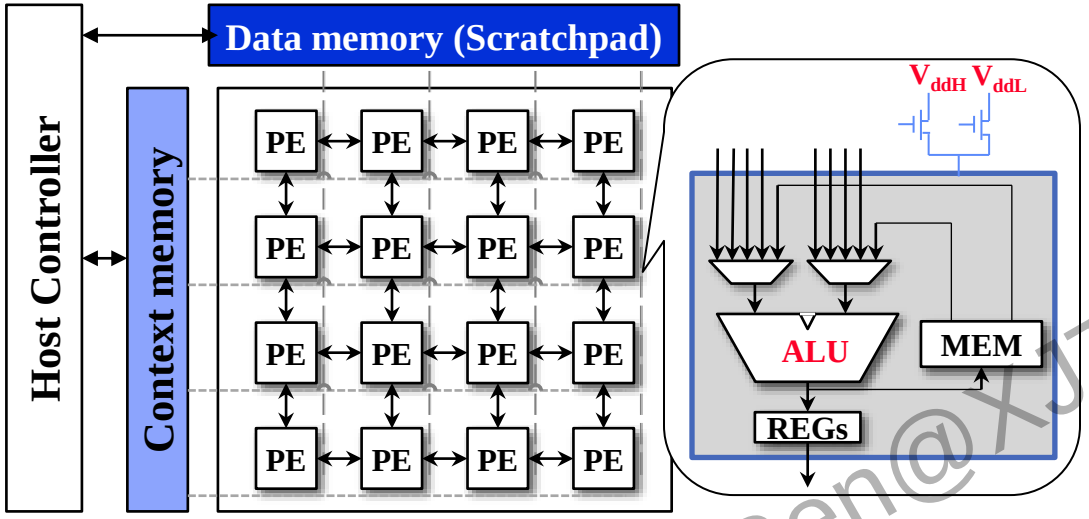
CGRA v.s FPGA

- **Fine-grained reconfigurability introduces overheads**
  - Much higher area, delay and power vs. standard cell ASIC
  - Introduce coarse-grained building blocks, much less interconnect
- **Programmability**
  - Fine grained architecture leads to long place and route times (> 2 hours)
  - Not a good computing substrate target for a compiler
- **CGRA architecture**
  - New reconfigurable architectures and new compiler technology

Top-Down Design of CGRA

- Observation: We can abstract key software constructs that are amenable to hardware acceleration
  - Nested data and pipeline parallelism
  - Data locality
- Parallel Patterns: Software abstractions that capture parallelism and locality
  - Loops with special properties
  - Expressive over wide range of domains (ML, SQL, Graph analytics, etc)
  - Enables building optimizing compilers with aggressive compiler optimizations
- Design a CGRA to accelerate parallel patterns

CGRA: Basic Structure and Configuration



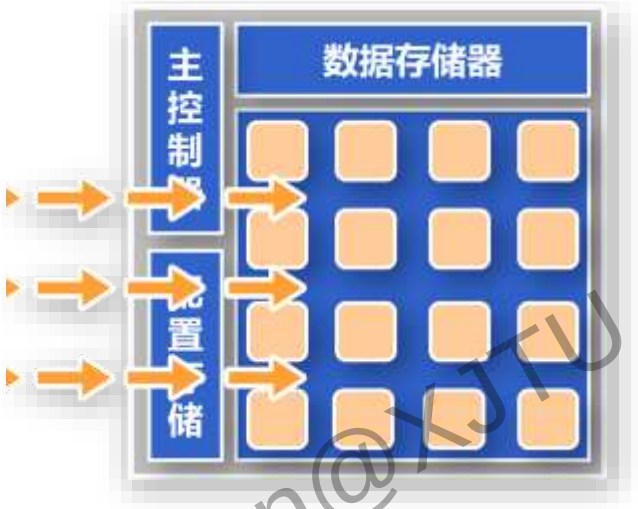
A + B	A >> B	A > B	(A+B)>>C
A - B	A + (B>>C)	A == B	(A+B)<<C
A & B	A	A < B	(A-B)>>C
A | B	A + (B<<C)	A >= B	(A+B)<<C
A ^ B	A - (B<<C)	A <= B	A×B_H
A ~^ B	|A-B|	A != B	Clip(A, -B, B)
~A	(A>>C)-B	A - (B>>C)	Clip(A, 0, B)
A << B	A×B_L	(A<<C)-B	C?A:B

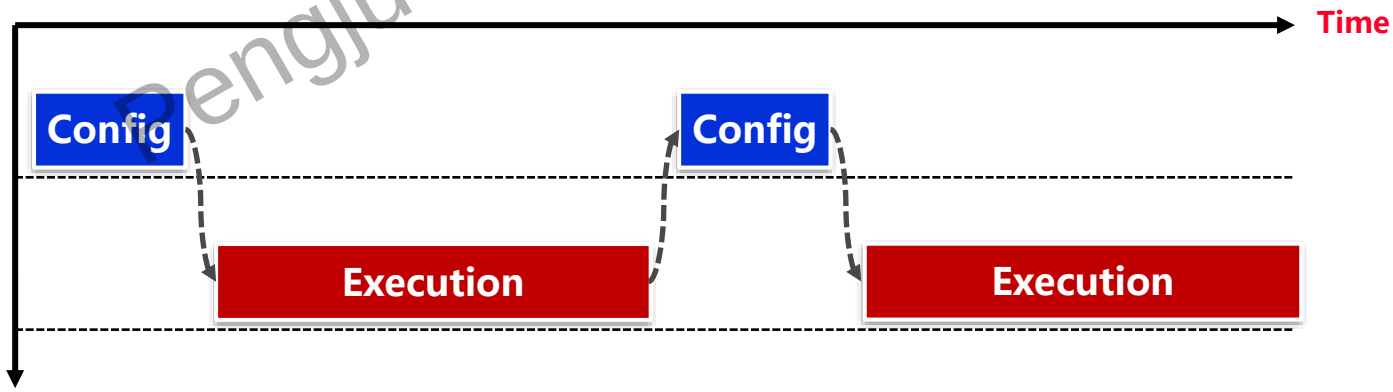
Functions of ALU

Reconfigurable hardware

- Implement hardware structures dynamically
- Objective: high performance but low power
- Key Tech: spatial parallel computing like 2D-PE array
- Connection is programmable → routing

CGRA: Basic Structure and Configuration



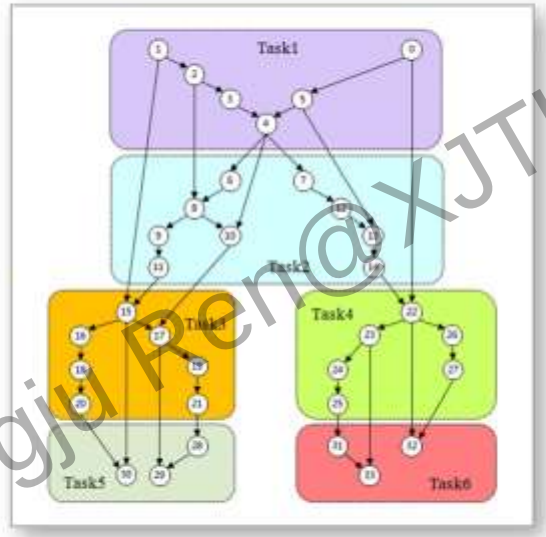


CGRA: Spacial Execution, Data Driven

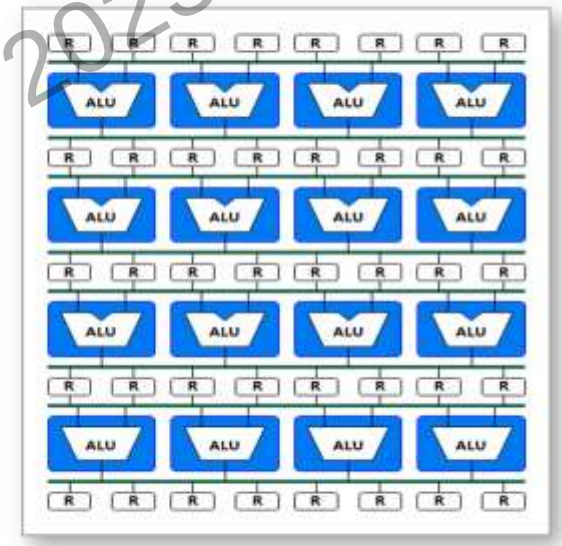
软件程序

```
IF (condition = 1) THEN
   r1 = b * b;
   r2 = a * c;
   r2 = r2 * 4;
   r1 = r1 - r2;
   IF (r1 < 0) THEN
      state = '1';
   ELSE
      state = '0';
      r3 = r1 * 4;
      r3 = r3 + 1;
      r3 = r3 * 0.2;
      FOR i = 1 to 3 LOOP
         r4 = r1 / r3;
         r3 = r3 + r4;
         r3 = r3 / 2;
      END LOOP;
      r1 = 0 - b;
      r2 = a + a;
      r4 = r1 + r3;
      x1 = r4 / r2;
      r5 = r1 - r3;
      x2 = r5 / r2;
   END IF;
END IF;
```

数据流图

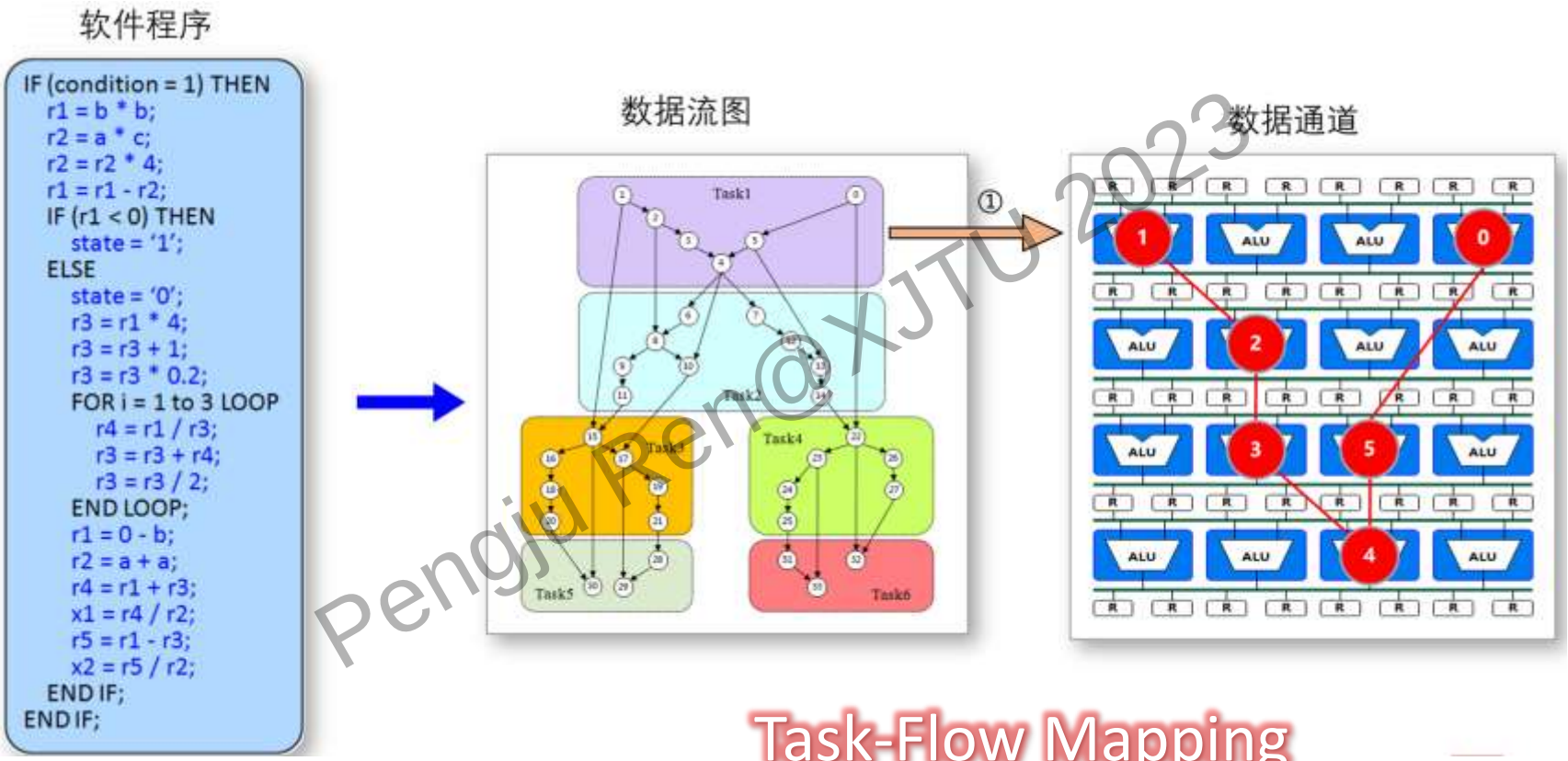


数据通道



- **Compilation tool flow is critical to CGRAs**
- **Mapping & Scheduling: From DFG to Spatial/Temporal deployment**

CGRA: Spacial Execution, Data Driven



Task-Flow Mapping

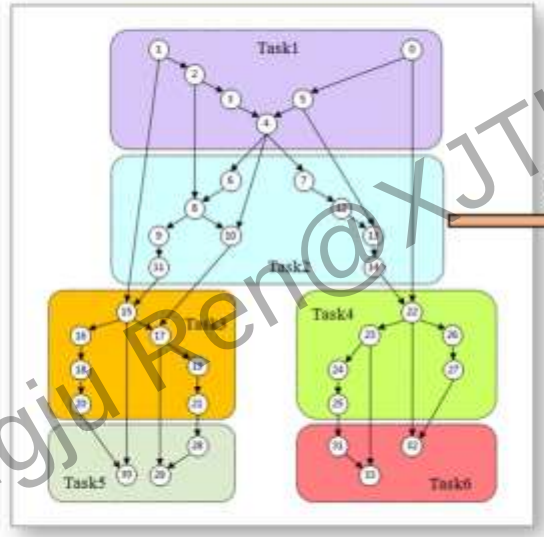
- Operator Mapping
- Memory Mapping
- Interconnection configuration

CGRA: Spacial Execution, Data Driven

软件程序

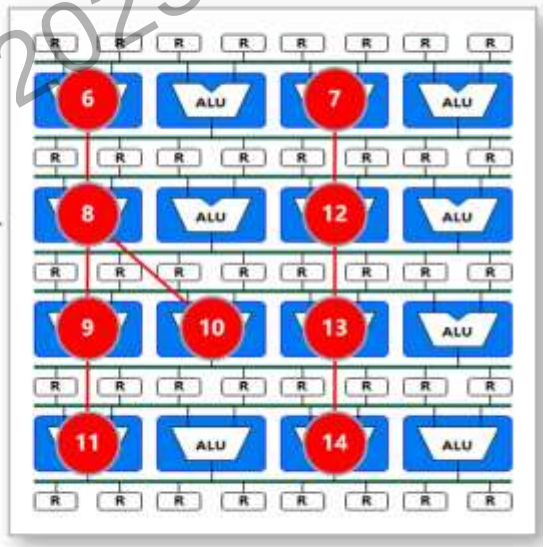
```
IF (condition = 1) THEN
   r1 = b * b;
   r2 = a * c;
   r2 = r2 * 4;
   r1 = r1 - r2;
   IF (r1 < 0) THEN
      state = '1';
   ELSE
      state = '0';
      r3 = r1 * 4;
      r3 = r3 + 1;
      r3 = r3 * 0.2;
      FOR i = 1 to 3 LOOP
         r4 = r1 / r3;
         r3 = r3 + r4;
         r3 = r3 / 2;
      END LOOP;
      r1 = 0 - b;
      r2 = a + a;
      r4 = r1 + r3;
      x1 = r4 / r2;
      r5 = r1 - r3;
      x2 = r5 / r2;
   END IF;
END IF;
```

数据流图



②

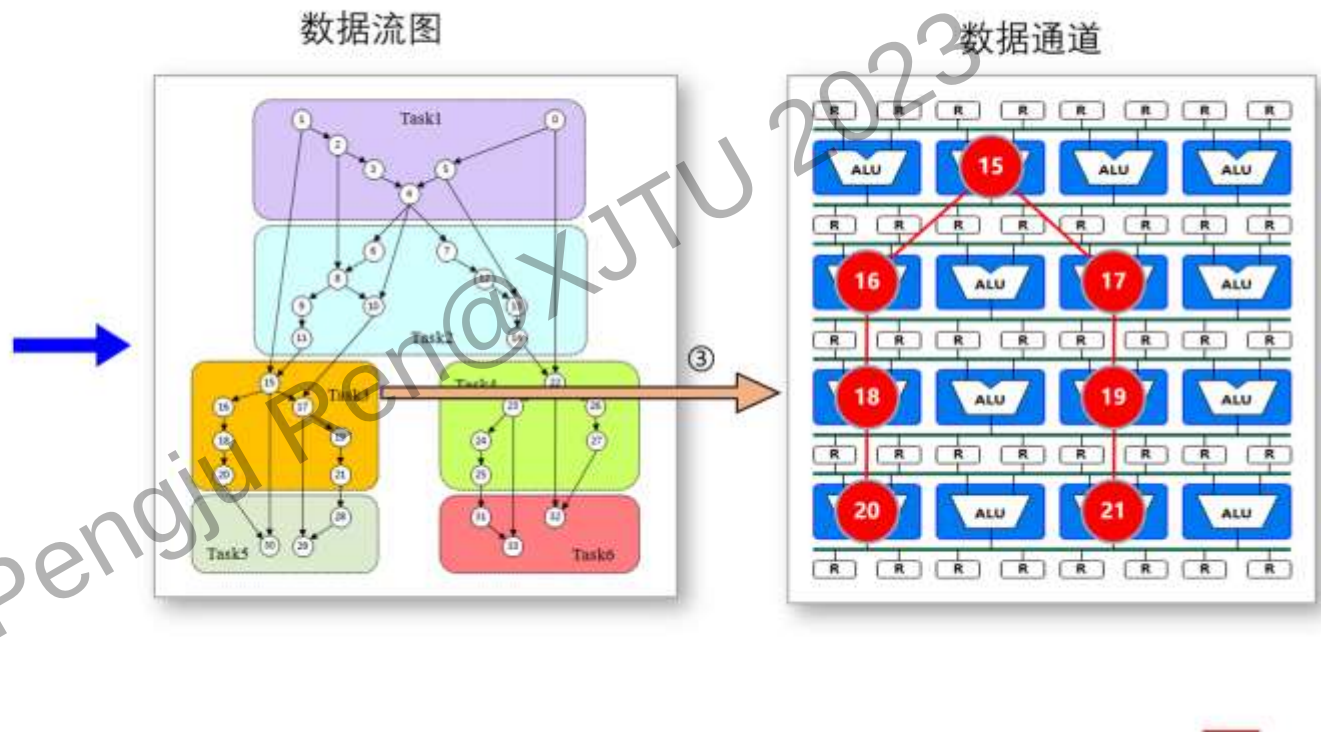
数据通道



CGRA: Spacial Execution, Data Driven

软件程序

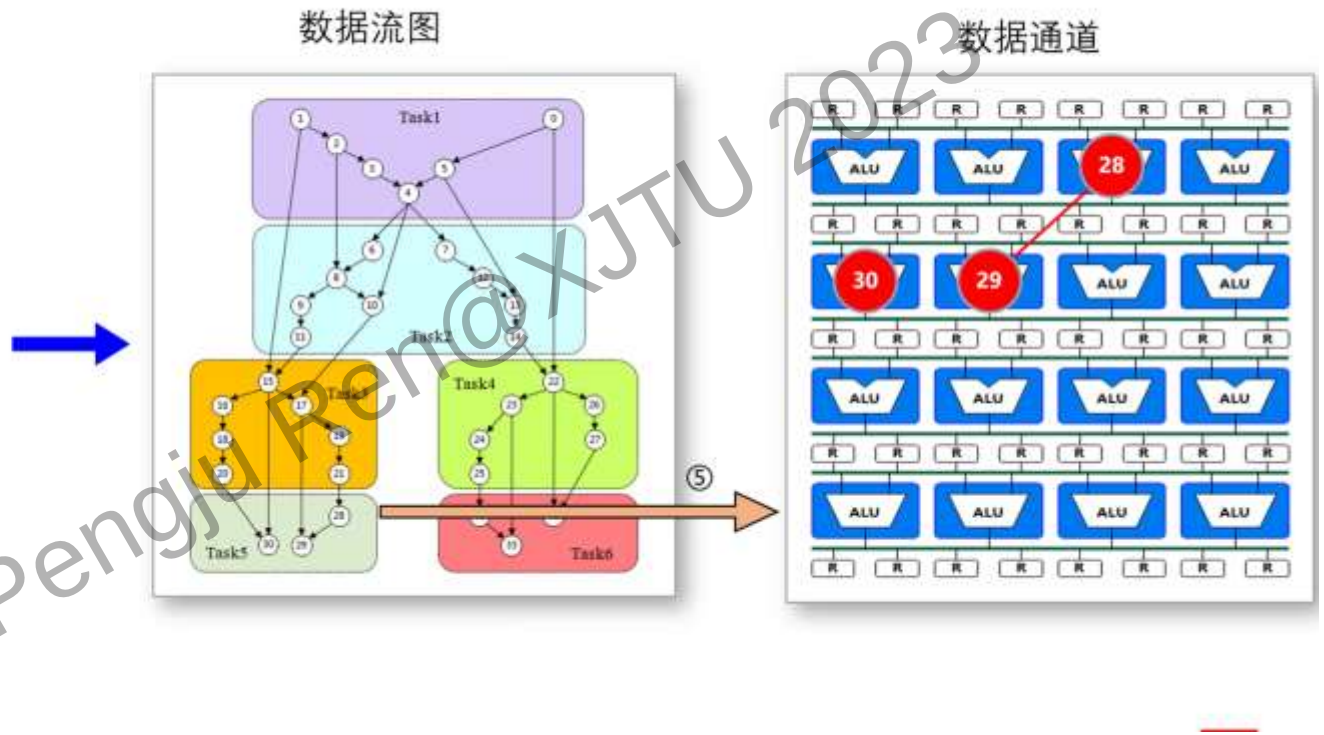
```
IF (condition = 1) THEN
   r1 = b * b;
   r2 = a * c;
   r2 = r2 * 4;
   r1 = r1 - r2;
   IF (r1 < 0) THEN
      state = '1';
   ELSE
      state = '0';
      r3 = r1 * 4;
      r3 = r3 + 1;
      r3 = r3 * 0.2;
      FOR i = 1 to 3 LOOP
         r4 = r1 / r3;
         r3 = r3 + r4;
         r3 = r3 / 2;
      END LOOP;
      r1 = 0 - b;
      r2 = a + a;
      r4 = r1 + r3;
      x1 = r4 / r2;
      r5 = r1 - r3;
      x2 = r5 / r2;
   END IF;
END IF;
```



CGRA: Spacial Execution, Data Driven

软件程序

```
IF (condition = 1) THEN
   r1 = b * b;
   r2 = a * c;
   r2 = r2 * 4;
   r1 = r1 - r2;
   IF (r1 < 0) THEN
      state = '1';
   ELSE
      state = '0';
      r3 = r1 * 4;
      r3 = r3 + 1;
      r3 = r3 * 0.2;
      FOR i = 1 to 3 LOOP
         r4 = r1 / r3;
         r3 = r3 + r4;
         r3 = r3 / 2;
      END LOOP;
      r1 = 0 - b;
      r2 = a + a;
      r4 = r1 + r3;
      x1 = r4 / r2;
      r5 = r1 - r3;
      x2 = r5 / r2;
   END IF;
END IF;
```

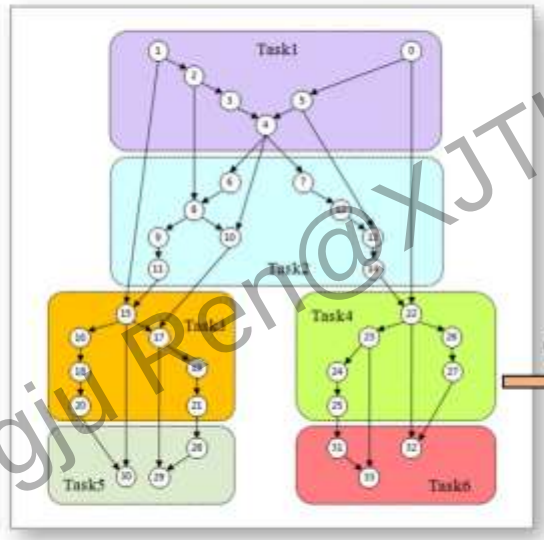


CGRA: Spacial Execution, Data Driven

软件程序

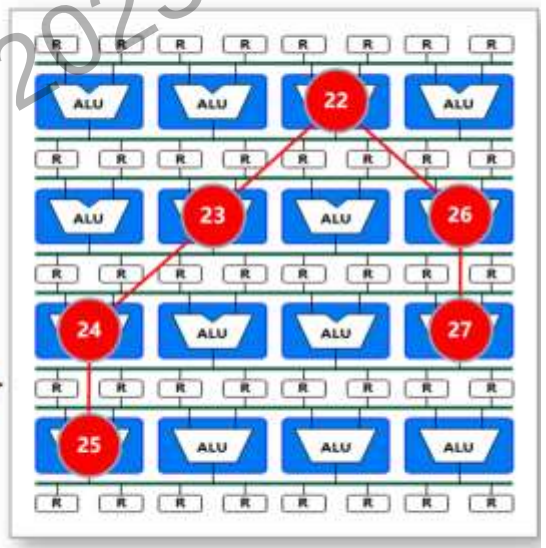
```
IF (condition = 1) THEN
   r1 = b * b;
   r2 = a * c;
   r2 = r2 * 4;
   r1 = r1 - r2;
   IF (r1 < 0) THEN
      state = '1';
   ELSE
      state = '0';
      r3 = r1 * 4;
      r3 = r3 + 1;
      r3 = r3 * 0.2;
      FOR i = 1 to 3 LOOP
         r4 = r1 / r3;
         r3 = r3 + r4;
         r3 = r3 / 2;
      END LOOP;
      r1 = 0 - b;
      r2 = a + a;
      r4 = r1 + r3;
      x1 = r4 / r2;
      r5 = r1 - r3;
      x2 = r5 / r2;
   END IF;
END IF;
```

数据流图



④

数据通道

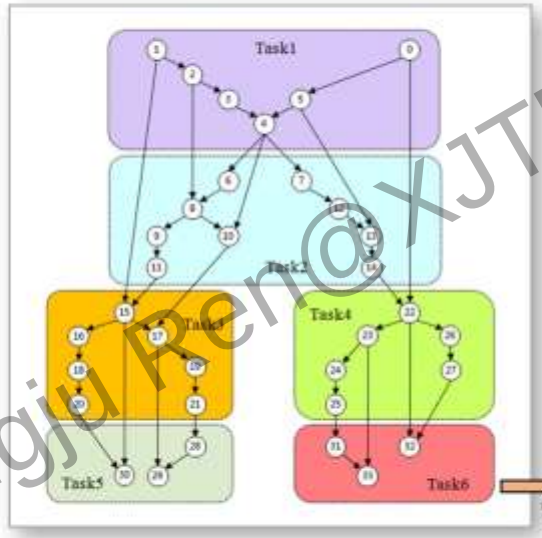


CGRA: Spacial Execution, Data Driven

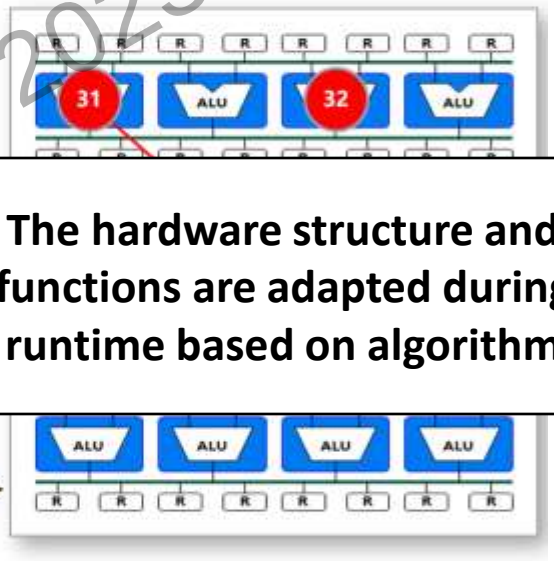
软件程序

```
IF (condition = 1) THEN
   r1 = b * b;
   r2 = a * c;
   r2 = r2 * 4;
   r1 = r1 - r2;
   IF (r1 < 0) THEN
      state = '1';
   ELSE
      state = '0';
      r3 = r1 * 4;
      r3 = r3 + 1;
      r3 = r3 * 0.2;
      FOR i = 1 to 3 LOOP
         r4 = r1 / r3;
         r3 = r3 + r4;
         r3 = r3 / 2;
      END LOOP;
      r1 = 0 - b;
      r2 = a + a;
      r4 = r1 + r3;
      x1 = r4 / r2;
      r5 = r1 - r3;
      x2 = r5 / r2;
   END IF;
END IF;
```

数据流图

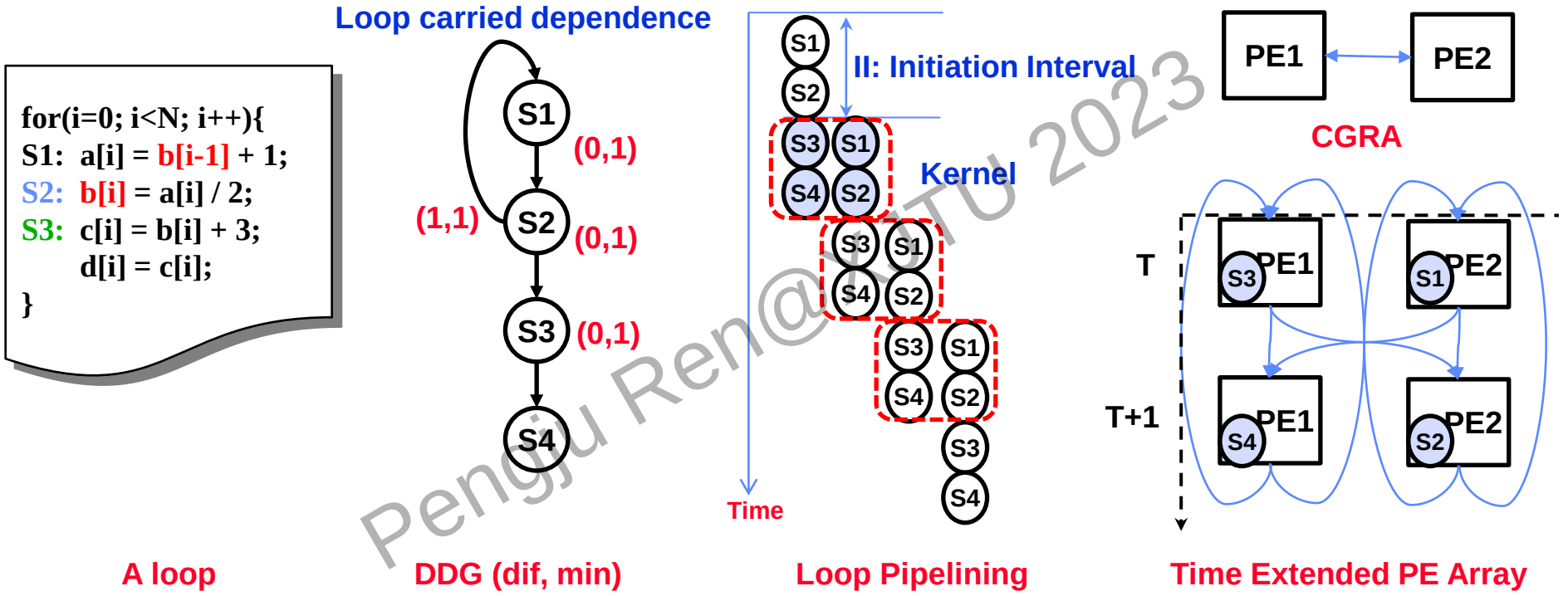


数据通道



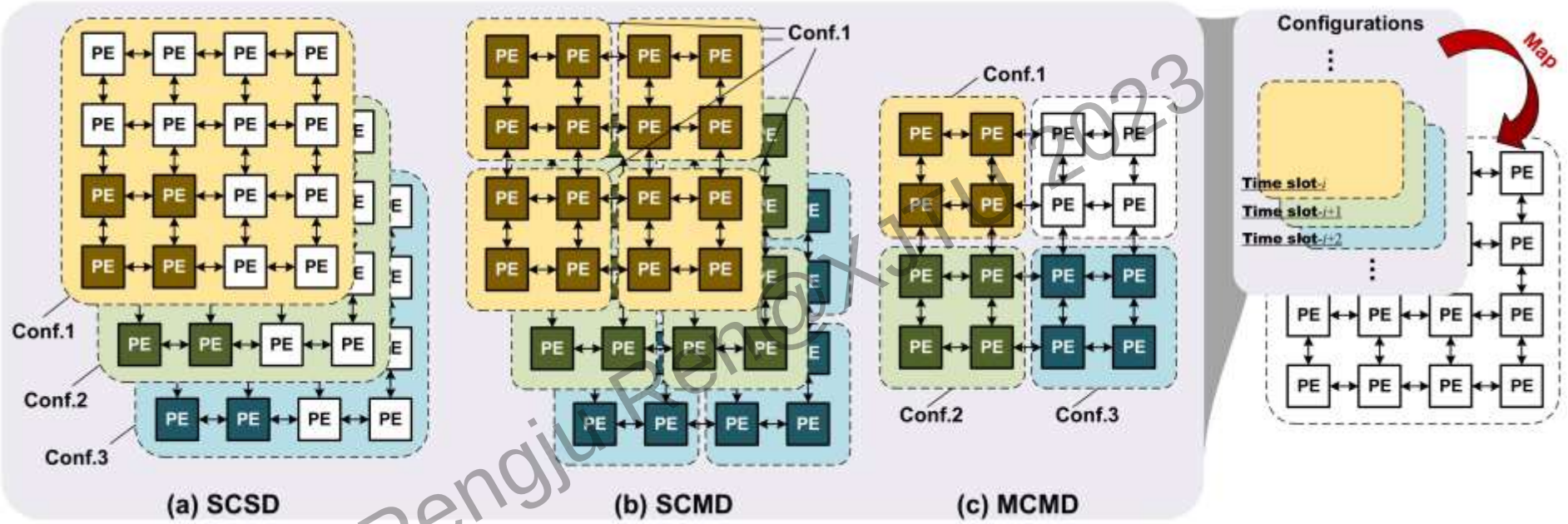
The hardware structure and functions are adapted during runtime based on algorithm

CGRA: orchestrate dataflow and hardware



- Exploiting operator level parallelism
- Finding better OP-PE binding according to CGRA's arch features

Computation models of CGRAs

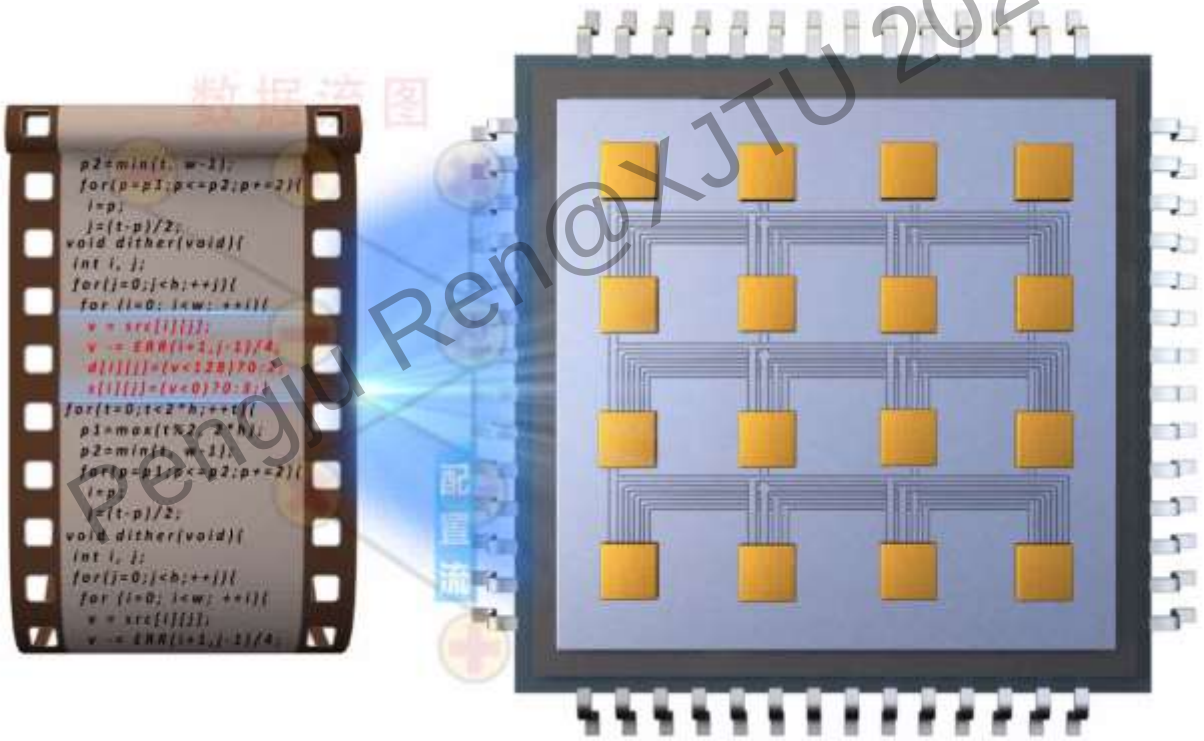


Single/Multiple Configuration Single/Multiple Data (SCSD、SCMD、MCMD)

Configuration-1 through configuration-3 are independent and asynchronous; rectangles with different colors represent different configurations, and blank ones represent idle

可重构计算处理器

- 可重构计算芯片以运算单元阵列为核心部件，由**配置流和数据流共同驱动（不使用指令）**，从而实现软硬件双编程。
- 应用任务中的每个运算**都可以在运算阵列中找到对应的单元，单元间的互连与任务一一对应**，由此获得比拟专用电路的能量效率。



高级语言程序， 如C/C++

（软件编程）

功能、互连等可按需动态改变的阵列

（硬件编程）

Next Lecture：DNN Acceleractor & HiPU Arch

(Given by Prof. Wenzhe Zhao)