Yun: An Open-Source, 64-bit RISC-V-Based Vector Processor with Multi-Precision Integer and Floating-Point Support in 65-nm CMOS

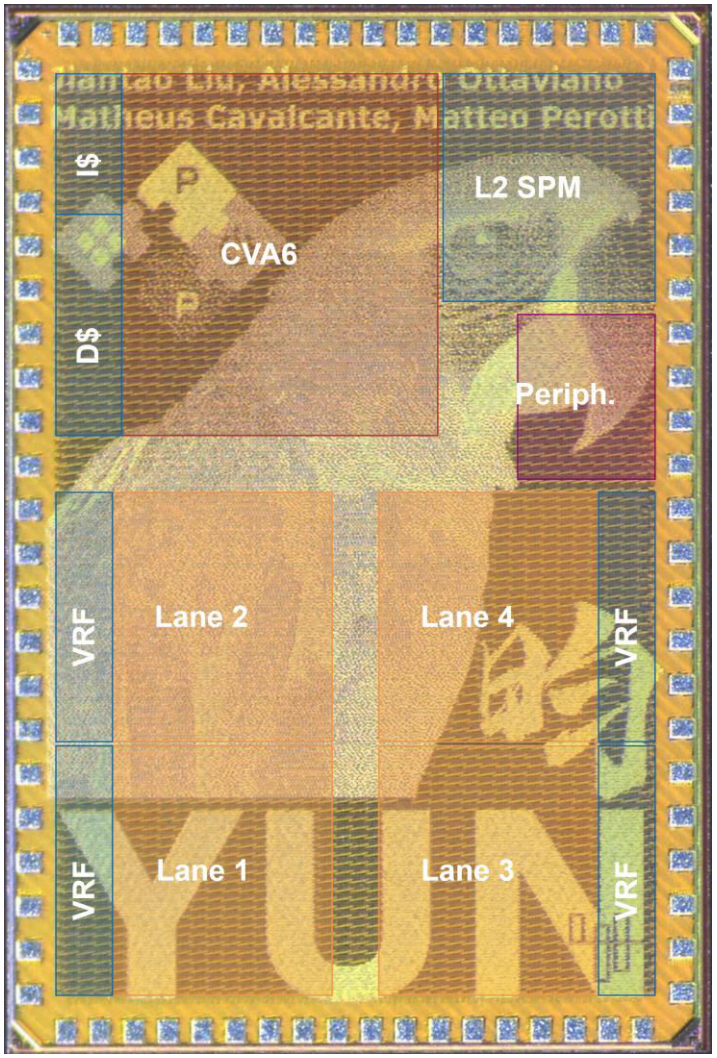
Matteo Perotti †, Matheus Cavalcante †, Alessandro Ottaviano †, Jiantao Liu ǂ, Luca Benini †‡

†ETH Zurich, ǂ University of California San Diego, ‡ University of Bologna

2023 IEEE International Symposium on Integrated Circuits and Systems

Outline

- Introduction
- Architecture
- Experiment setup
- Results
- Comparison with SoA
- Conclusion



Quest for Performance and Efficiency

- **End** of **Moore's Law** and **Dennard Scaling**
- **Quest** for **high performance** and **energy efficiency**
- From the **edge** to the **cloud**
- Need for **processing Data-Parallel applications**
- Promising approach – **Vector Processor Architecture**

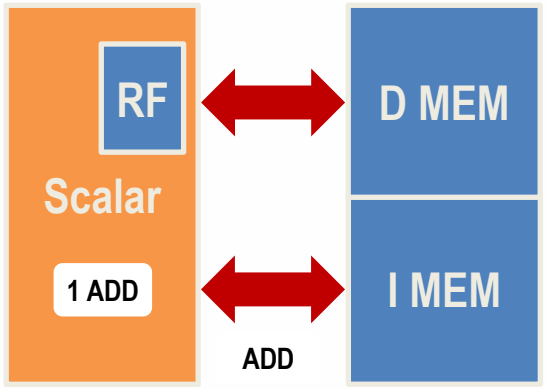
Vector Processor

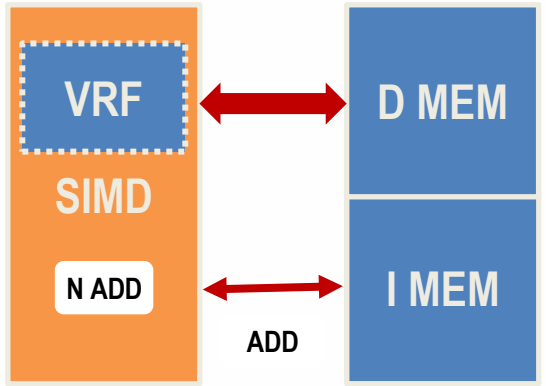
SCALAR CORE

- One instruction – One Operation
  ↑ **BW and Power on the I-MEM**
- RF size is usually fixed
  One data element per entry
  Too small for highly-intensive WL
  ↑ **BW and Power on the D-MEM**

VECTOR CORE

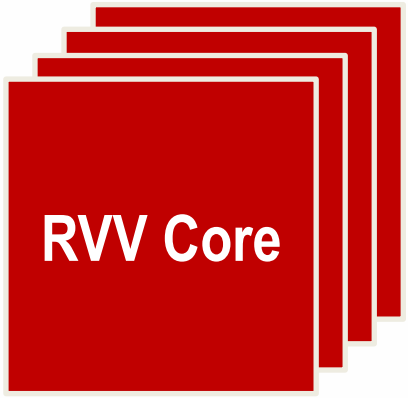
- One instruction – Many operations
  ↓ **BW and Power on the I-MEM**
- RF size is larger
  Multiple data elements per entry
  Better buffering, exploit locality
  ↓ **BW and Power on the D-MEM**



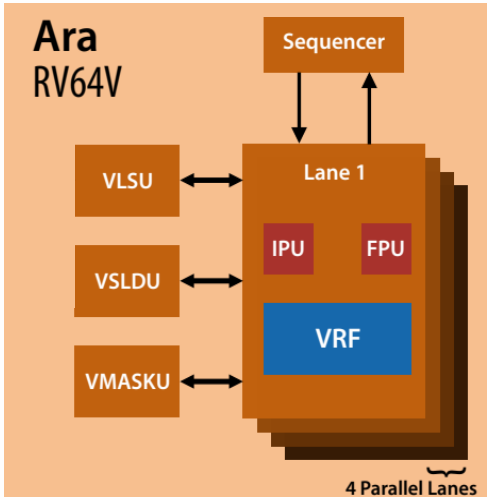


Vector Processors – Across the domains

Supercomputer **FUGAKU**
Arm SVE – Vector ISA
Scientific Computing



SiFive Intelligence **X280**
RISC-V V – Vector ISA
Machine Learning

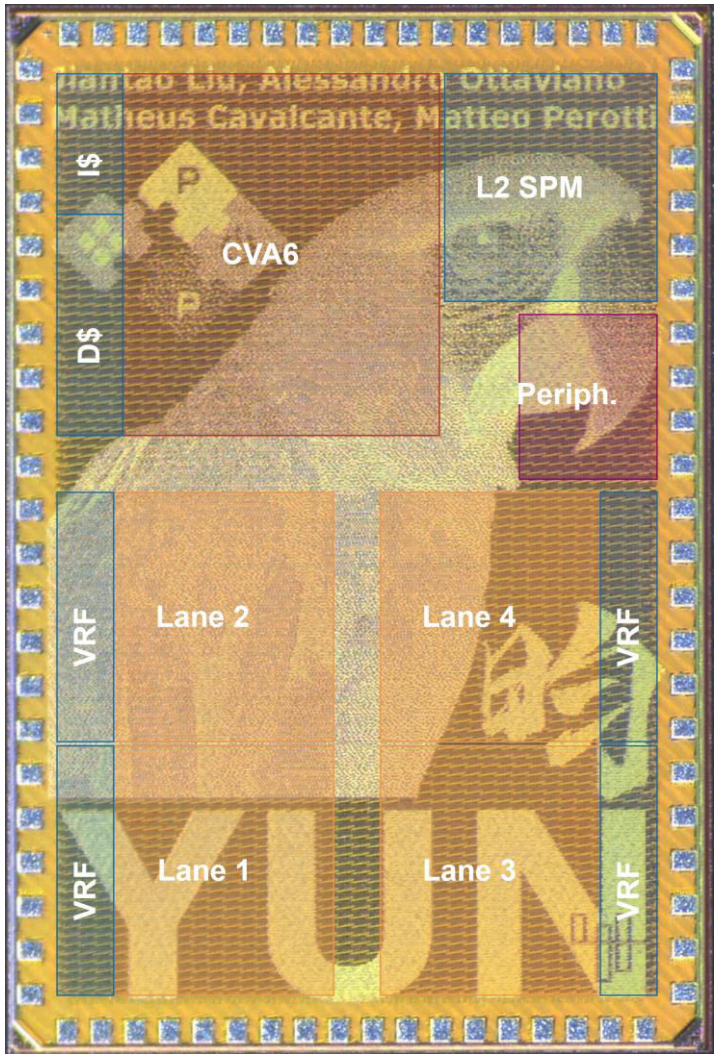


Ara (open-source)
RISC-V V – Vector ISA
Application Class

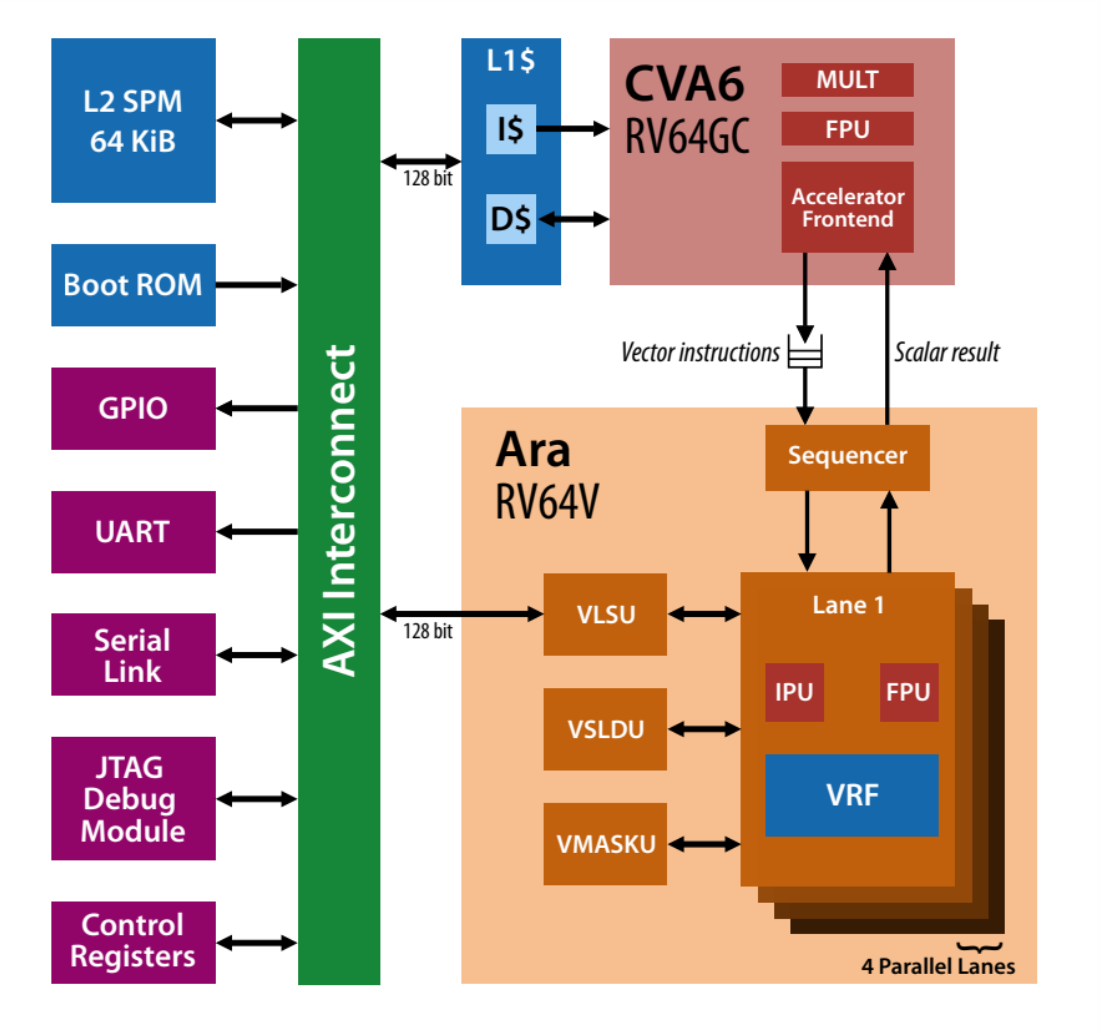
Author of the picture: Raysonho, https://en.wikipedia.org/wiki/Fugaku_(supercomputer)#/media/File:FugakuSupercomputerSC19.jpg

Contributions

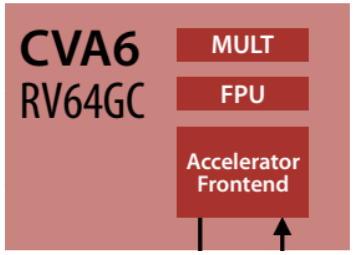
- **Yun** System-on-Chip (**SoC**)
- **First** tape-out of **open-source RISC-V V** vector **processor!**
- **Multi-precision** capabilities (int**64** → int**8**, fp**64** → fp**32**)
- **90% max. perf.** on all data types
- **Performance** and **Power** Characterization
- Leading-Edge **Area Efficiency**



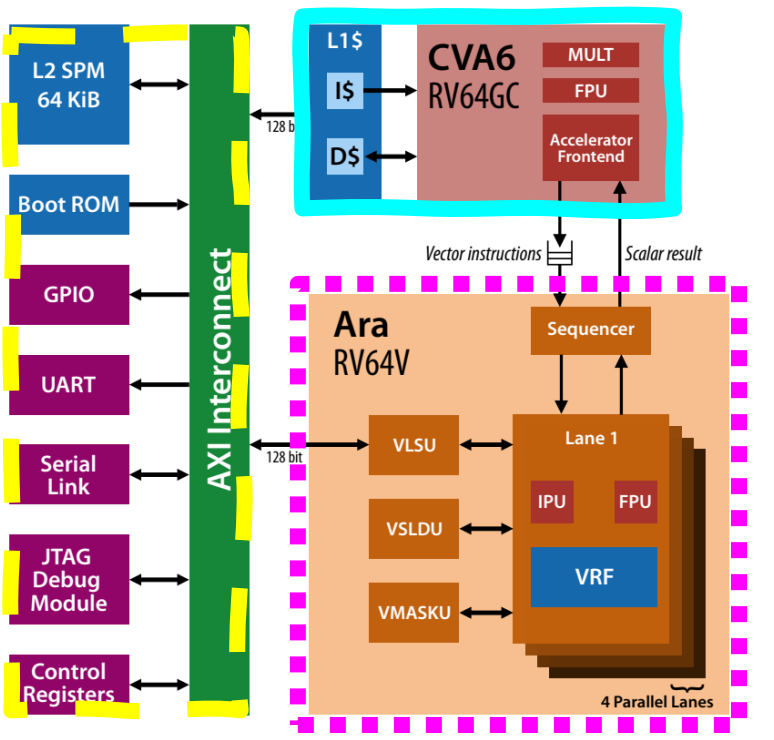
Yun SoC

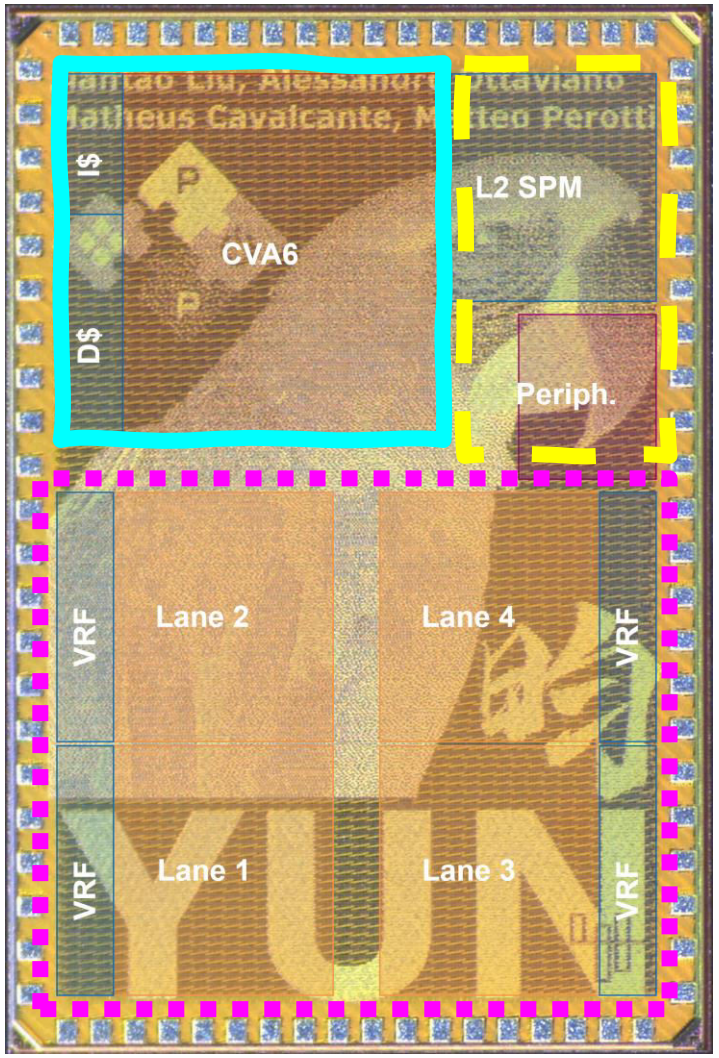

L2 SPM
64 KiB
Boot ROM
GPIO
UART
Serial
Link
JTAG
Debug
Module
Control
Registers
AXI Interconnect
128 bit
L1$
I$
D$
CVA6
RV64GC
MULT
FPU
Accelerator
Frontend
Vector instructions
Scalar result
Ara
RV64V
Sequencer
128 bit
VLSU
VSLDU
VMASKU
Lane 1
IPU
FPU
VRF
4 Parallel Lanes

Yun SoC



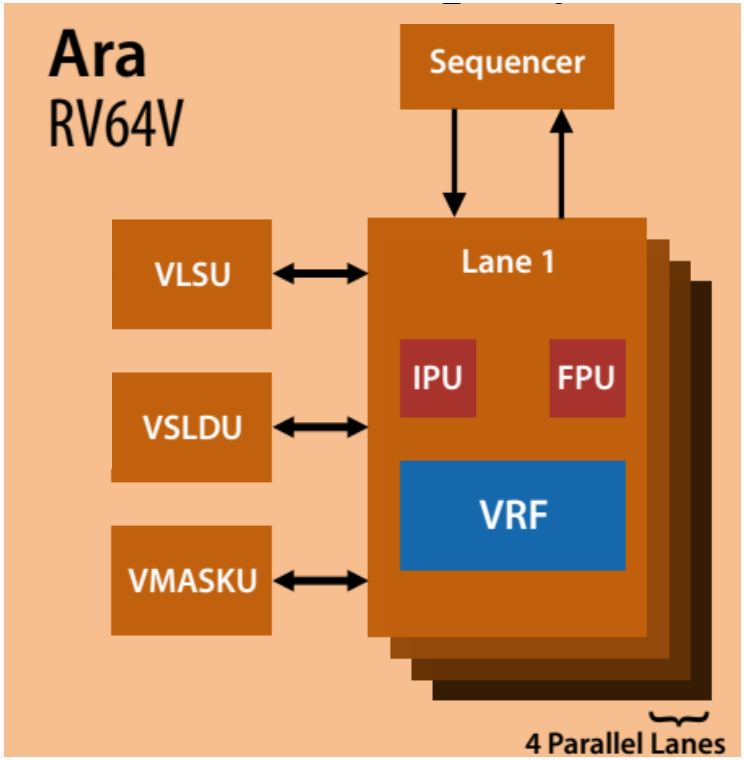
Yun SoC


L2 SPM
64 KiB
Boot ROM
GPIO
UART
Serial Link
JTAG Debug Module
Control Registers
AXI Interconnect
128 b
L1$
I$
D$
CVA6
RV64GC
MULT
FPU
Accelerator Frontend
Vector instructions
Scalar result
Ara
RV64V
Sequencer
128 bit
VLSU
VSLDU
VMASKU
Lane 1
IPU
FPU
VRF
4 Parallel Lanes


I$
D$
CVA6
L2 SPM
Periph.
VRF
Lane 2
Lane 4
VRF
VRF
Lane 1
Lane 3
VRF

Ara Architecture

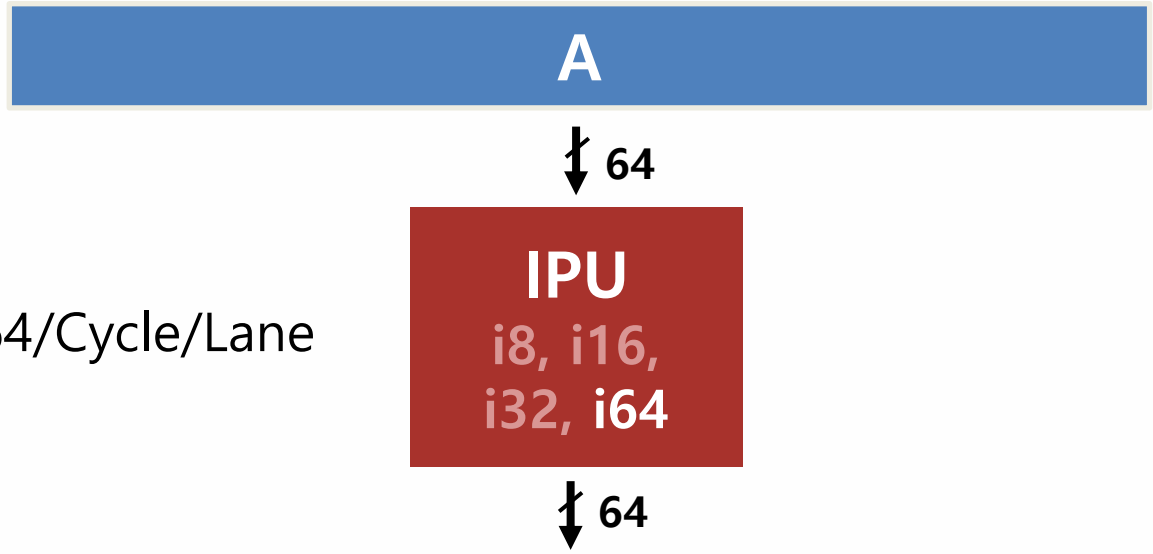
- **RISC-V V**-based vector **processor**
- Fully **open-source**!
- **4** parallel **vector lanes** (Yun)
- Lane:
  - **IPU (Integer Processing Unit)**
  - **FPU (Floating-Point Unit)**
  - **VRF (Vector Register File)**



github.com/pulp-platform/ara

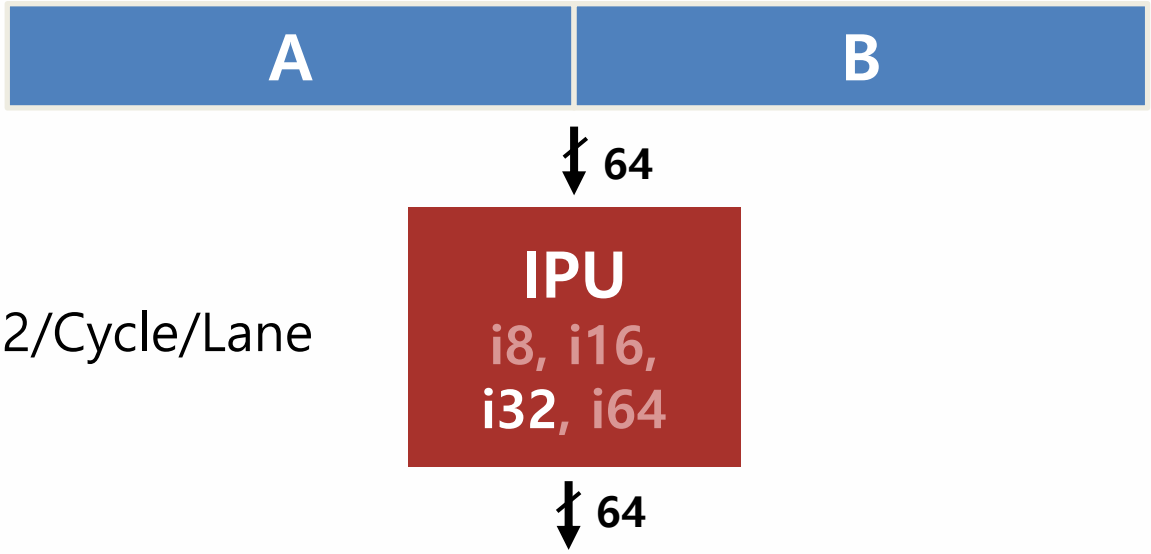
Ara – Multi-Precision Ready!

- **Multi-Precision SIMD** IPUs and FPUs
- SIMD **IPU**: from **8**-bit to **64**-bit elements
- SIMD **FPU**: from **32**-bit to **64**-bit elements



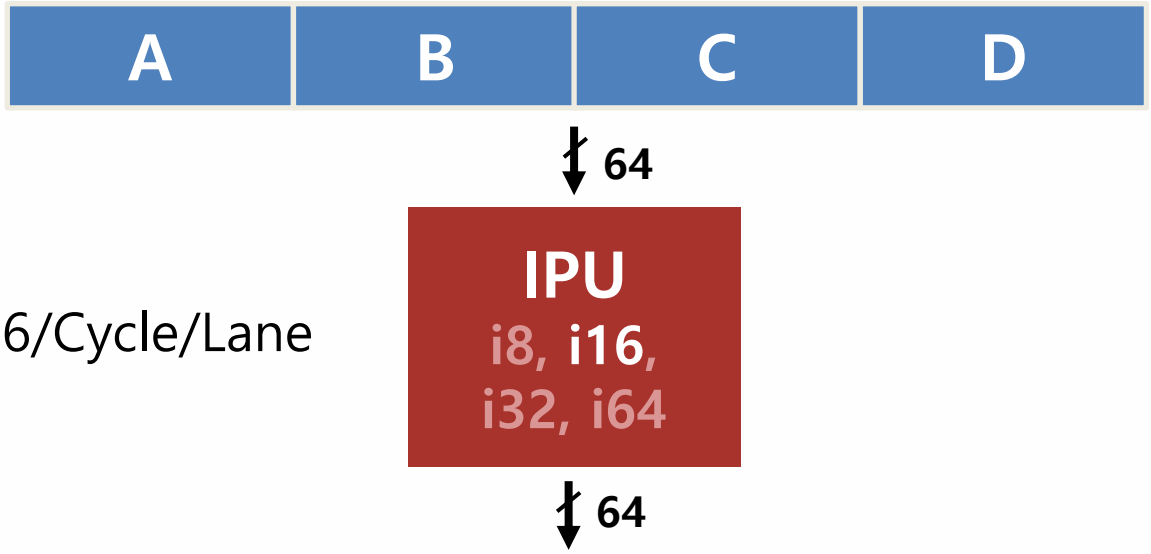
Ara – Multi-Precision Ready!

- **Multi-Precision SIMD** IPUs and FPUs
- SIMD **IPU**: from **8**-bit to **64**-bit elements
- SIMD **FPU**: from **32**-bit to **64**-bit elements



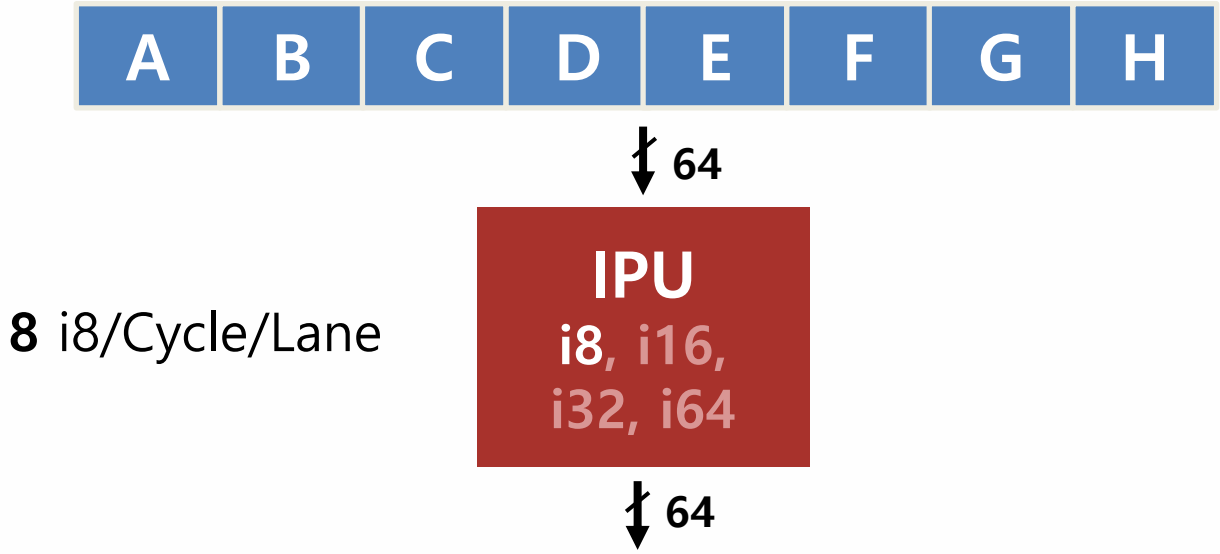
Ara – Multi-Precision Ready!

- **Multi-Precision SIMD** IPUs and FPUs
- SIMD **IPU**: from **8**-bit to **64**-bit elements
- SIMD **FPU**: from **32**-bit to **64**-bit elements



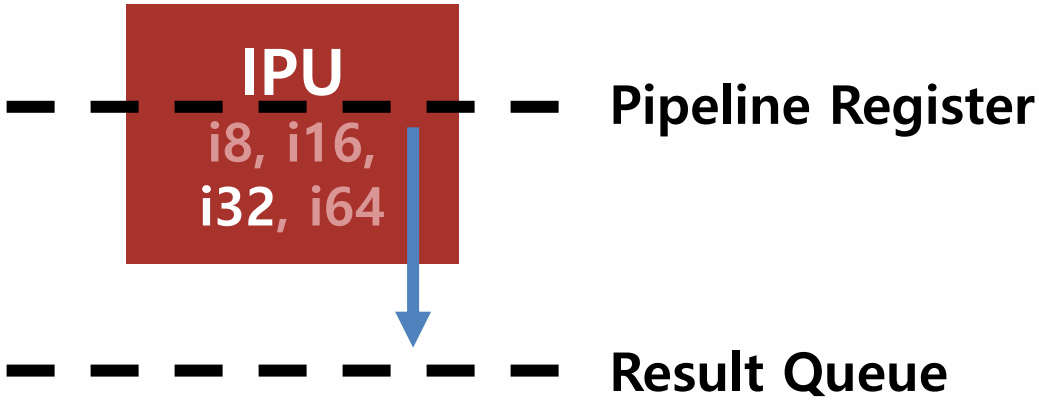
Ara – Multi-Precision Ready!

- **Multi-Precision SIMD** IPUs and FPUs
- SIMD **IPU**: from **8**-bit to **64**-bit elements
- SIMD **FPU**: from **32**-bit to **64**-bit elements



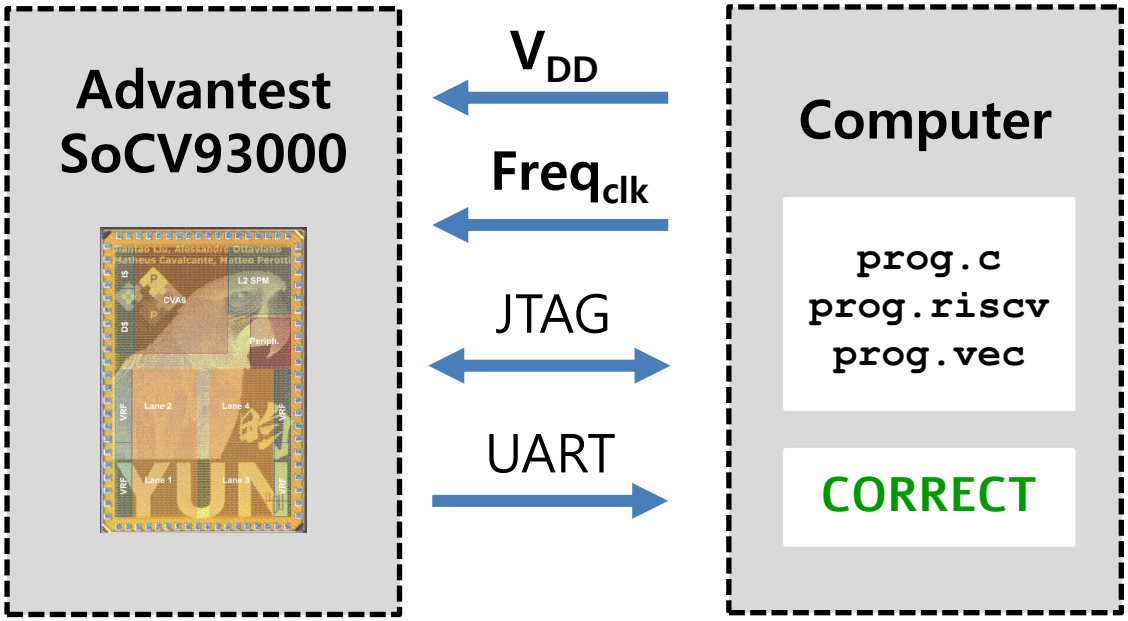
Ara – Critical Path

- Ara's **Critical Path**: SIMD **Multiplier** in **i32-IPU**
- **~30 FO4 Inv.** (from 65-nm technology implementation)



Experiment Setup

- Preload **benchmark** in **L2** Memory via **JTAG**
- Sweep $\mathbf{V_{DD}}$ and clock frequency ($\mathbf{Freq_{clk}}$)
- **Check** results via **UART** and **JTAG**
- Measure **max** $\mathbf{Freq_{clk}(V_{DD})}$ and **Power**



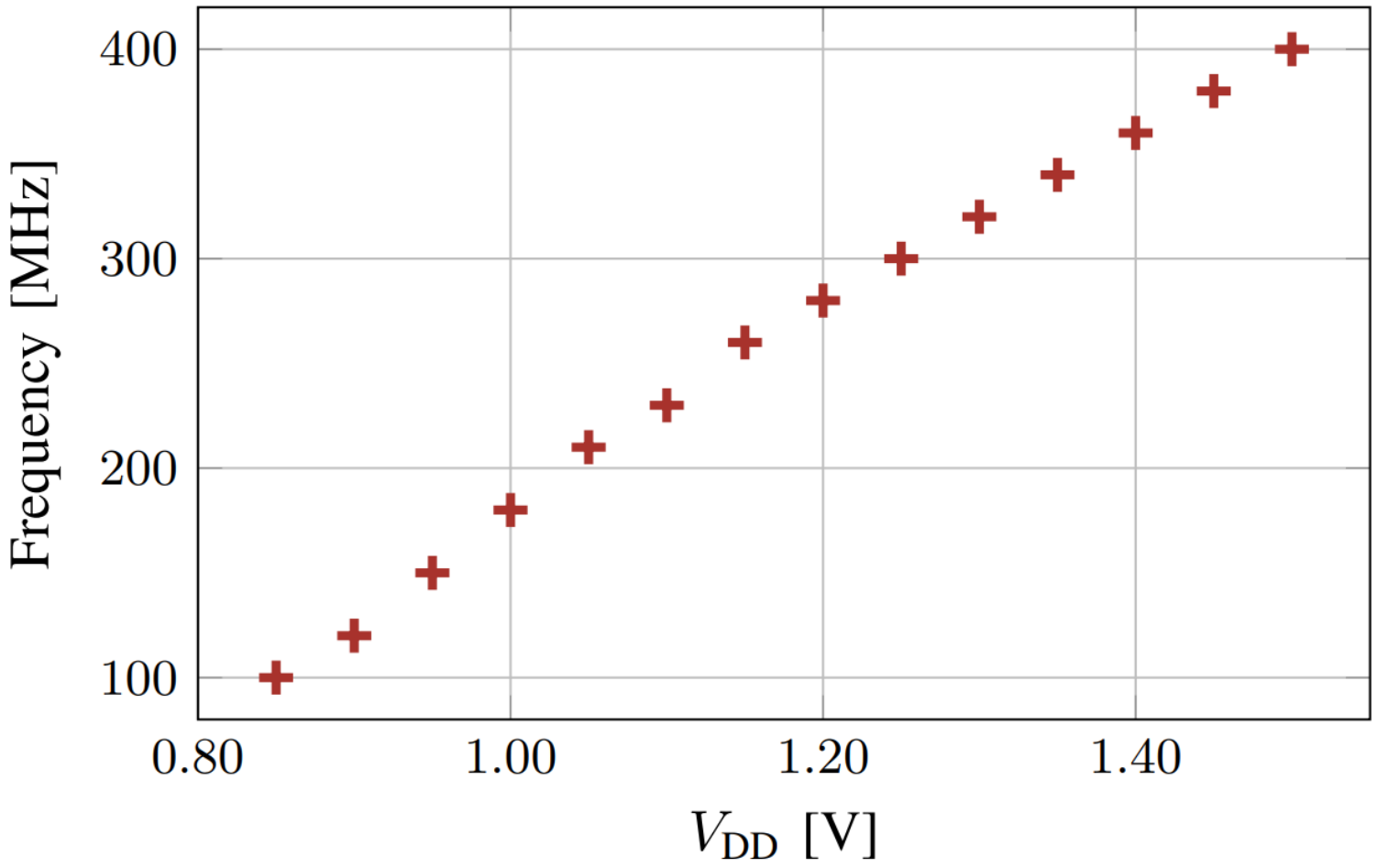
Benchmarks

Matrix Multiplication: $C^{M\times N} \leftarrow A^{M\times K} B^{K\times N}$

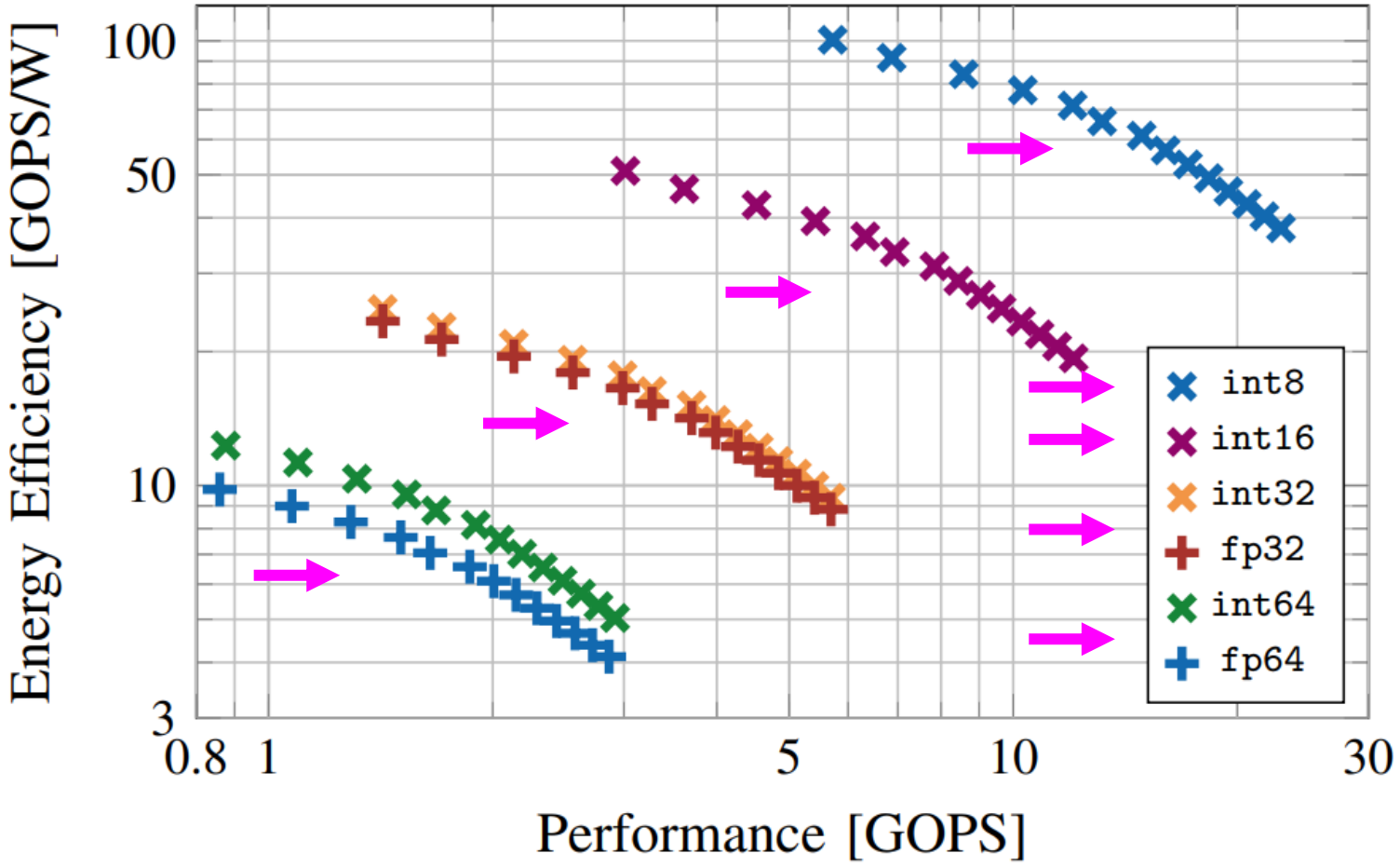
Precision	$M \times N \times K$	Perf (OP/cycle)
int64	$16 \times 128 \times 32$	7.3
int32	$32 \times 128 \times 32$	14.2
int16	$64 \times 256 \times 32$	30.1
int8	$16 \times 512 \times 32$	57.2
fp64	$16 \times 128 \times 32$	7.2
fp32	$32 \times 128 \times 32$	14.2

- **~90% maximum** theoretical **performance**
- Main **performance driver**: **N** dimension
- Higher N → Longer Vector

Maximum Frequency vs. V_{DD}

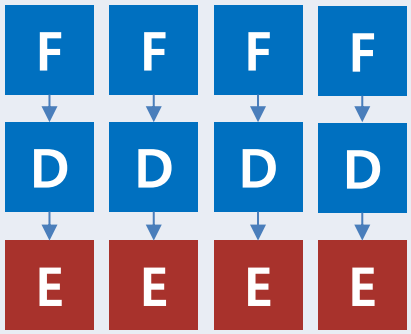


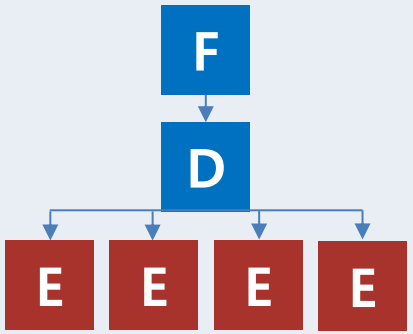
Energy Efficiency vs. Performance



Comparison with SoA

	[21]	[22]	[23]	[19]	[20]	Yun
Name	C0	C1	C2	V0	V1	Yun
Year	2021	2021	2022	2014	2022	2023
Tech	22 nm	65 nm	65 nm	45 nm	16 nm	65 nm





- **C0**, **C1**, **C2**: Cluster-based with **independent** instr. **fetch pipelines**
- **V0**, **V1**: Hwacha **Vector** Processors
- **Yun**: RISC-V V **Vector** Processor

SoA – Performance

- Comparison **issues**:
  - **C0**, **V1**, **V2**: more advanced technologies
  - **C1**, **C2**: integers focus on **lower precisions** (sub-byte)

- Use **technology-independent metrics** for **floating-point**

	C0	C2	V0	V1	Yun
32b Util.	55%	44%	-	-	**89%**
64b Util.	-	-	78%	-	**89%**
32b Perf.	4.4	3.6	-	-	**14.2**
64b Perf.	-	-	3.1	-	**7.1**

- ***Utilization*** [%] → $Performance / Max\_Performance$
- ***Performance*** [FLOP/Cycle] → $Throughput\ [FLOPS] / Frequency\ [Hz]$

SoA – Area

- Comparison **issues**:
  - **Different technologies**

- **Scale** the area **by** each technology's **gate equivalent** area

Area [MGE]	C0	C2	V0	V1	Yun
Chip	60.3	9.4	17.0	173.9	4.7
Core, Mem	35.2	4.8	7.2	84.8	4.7
Area Eff.	0.12	0.75	0.86	-	**3.00**

- ***Area Efficiency*** [$FLOP_{SP}$/Cycle/MGE] → $Performance\,[FLOPSP/cycle] / Area\,[MGE]$

SoA – Energy Efficiency

- Comparison **issues**:
  - **Different technologies** and **voltages** (use **efficiency** at **1V**)

	C0	C2	V0	V1	Yun
Tech	22 nm	65 nm	45 nm	16 nm	65 nm
32b Eff.$_{1V}$	19.8	6.8	-	**27.9**	16.9
64b Eff.$_{1V}$	-	-	7.3	**17.1**	7.8

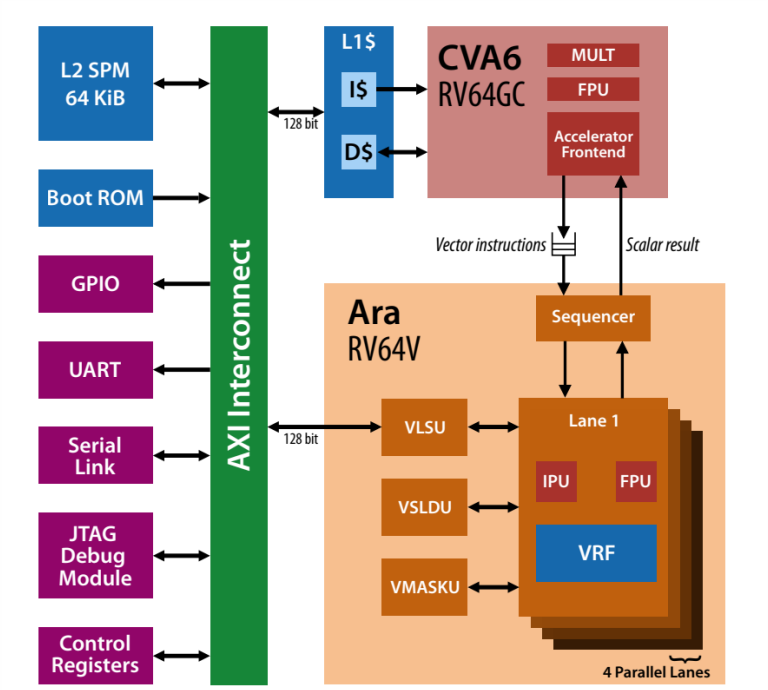
- ***Efficiency @1V*** [GFLOPS/W]
  - When possible → report paper's value
  - Otherwise → $PeakEfficiency(@VDD) \times VDD^2$

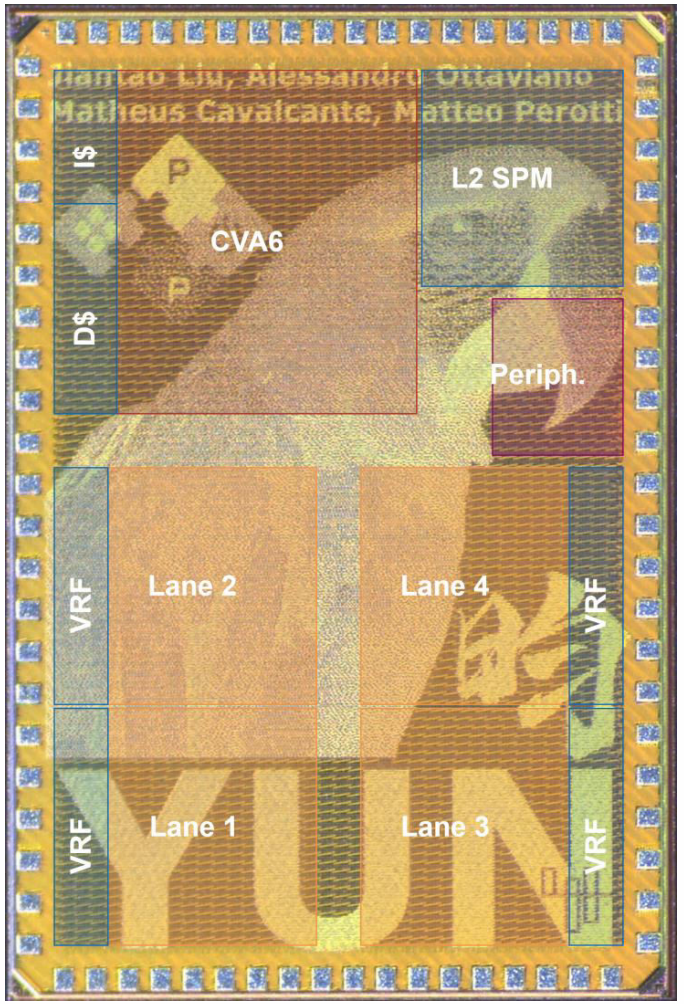
- **+7%** from **V0**
- **-15%** from **C0, despite** additional **64-bit** support
- **Hard** to **compare** with **V1**, very **different tech** nodes

Conclusions

- **Yun – First RISC-V V Open-Source Vector Processor** on Silicon
- **Base for** future **comparisons** and **architectural analyses**
- Multi-precision – **int8 → int64**, **fp32 → fp64**
- Peak FP **Perf**. – **5.7 GFLOPS$_{SP}$** (1.5 V, 400 MHz)
- Peak FP **Efficiency** – **23.4 GFLOPS$_{SP}$/W** (0.85 V, 100 MHz)
- Leading-edge **Area Efficiency** – **3 FLOP$_{SP}$/Cycle/MGE**

Thank you





Matteo Perotti, Ph.D. Student
ETH Zurich, Switzerland
mperotti@iis.ee.ethz.ch