The SPARKLE RISC-V Overlay Ecosystem for Datacenter FPGAs

Riadh Ben Abdelhamid[1], Dirk Koch[1]

[1] ZITI, Heidelberg University, Germany.

Hipeac 2025 WRC

Outline

1. Introduction
2. SPARKLE Ecosystem
3. BRISKI
4. SPARKLE
5. Physical Design
6. Programming Model
7. Evaluation
8. Conclusion
9. References

Computing with FPGAs (Field Programmable Gate Array)

Background

- Increasing use of FPGAs by major **cloud providers** for **hardware acceleration**.
- Large **datacenter FPGAs** have **underutilized** capacity for compute.
- Conventional FPGA flows are **time-consuming** and **complex**.

Challenges

- Need a **"software-like"** interface without sacrificing performance
- **Balancing** compute density, speed, and communication on limited FPGA space.
- **Compilation time** is **very long** (Place, Route,..) and often many iteration (timing closure, ..) and is a barrier for mainstream adoption.
- A **RISC-V overlay** can leverage open-source compiler/toolchain ecosystem and **reduce compilation time** on FPGAs.

RISC-V FPGA overlays

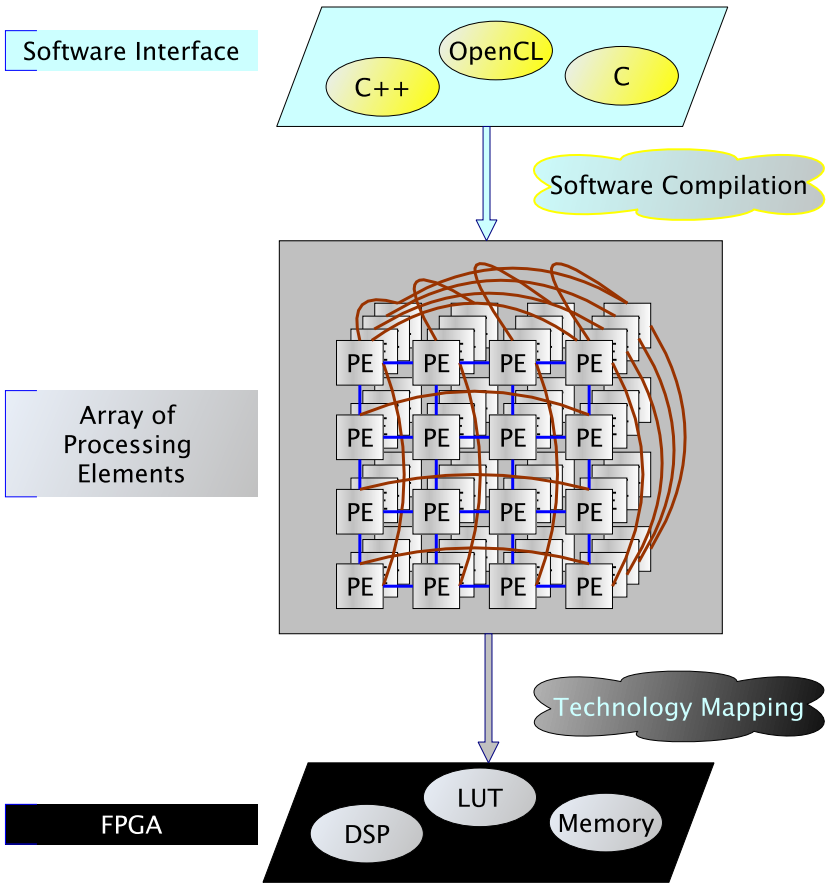


Figure – Typical layers of an overlay architecture

Why Overlays

- Provides a higher abstraction-level for FPGA hardware.
- Shorter design time and faster programming

Why RISC-V ?

- Open-source ISA, customizable instructions
- Vibrant ecosystem and community support
- Ideal for software-friendly FPGA acceleration
  $\Longrightarrow$ facilitate the creation of many soft cores with specialized features.

Challenges towards a Kilocore RISCV overlay

Main challenges

- Design a RISCV processor core with **high compute density (MIPS/LUT)** to maximize the #cores on chip and maximize overall throughput.
- Replicate the core in a hierarchy that **leverages existing resources in the most efficient way**.
- Define the **programming model** to be able to **optimize the implementation**.
- Deal with **physical design challenges** such as floorplanning, balancing resource utilization, extended pipeline around fixed primitives, etc.

The SPARKLE Ecosystem

BRISKI (Barrel RISC-V for Kilo-core Implementations)

BRISKI, a high compute density (**lightweight** and **fast**) **Barrel RISC-V** Processor core implementing (**RV32I + zalrsc + csrrs**) with configurable :

- Pipeline depth
- Amount of threads
- Barrel shifter
- Optional DSP-based ALU

SPARKLE (Scalable Parallel Architecture for RISC-V Kernel-Level Execution)

A scalable architecture with :

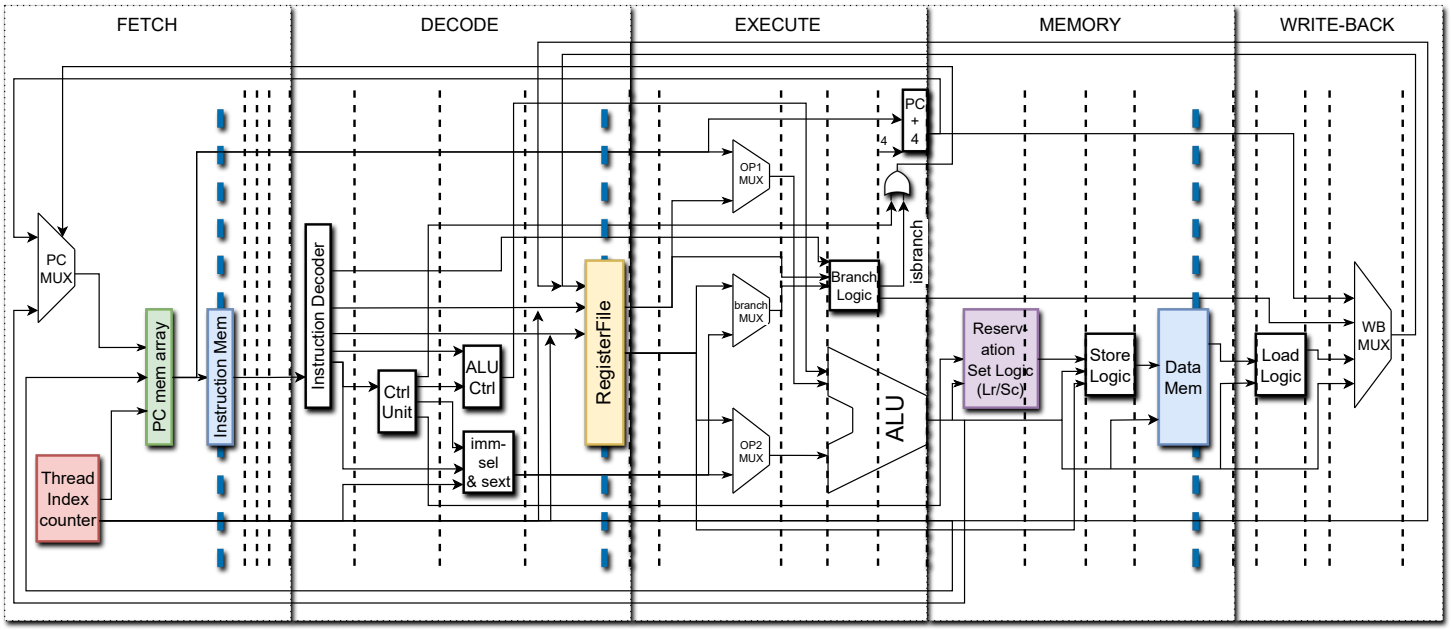
- **Flexible** programming model
- **1,024 RISC-V cores** / **16,384 Harts** on a single datacenter-class FPGA.
- Runs @ **500 MHz** for **0.5 RISC-V TOPS** (32-bit).

BRISKI : Barrel RISC-V for Kilo-core Implementations

Key attributes

- Based on RISC-V RV32I ISA (+ synchronization with LR.W/SC.W and CSRRS).
- Optimized for low area (**less than 700 LUTs** in most configurations) and high clock speeds (**up to 737 MHz**).
- 4-16 stage pipeline with extended stages around fixed primitives to manage routing delays and maintain speed when logic is spread far apart.

BRISKI is publicly available under : **https ://github.com/riadhbenabdelhamid/BRISKI**



What is Barrel Processing ? (Fine-grain multi-threading)

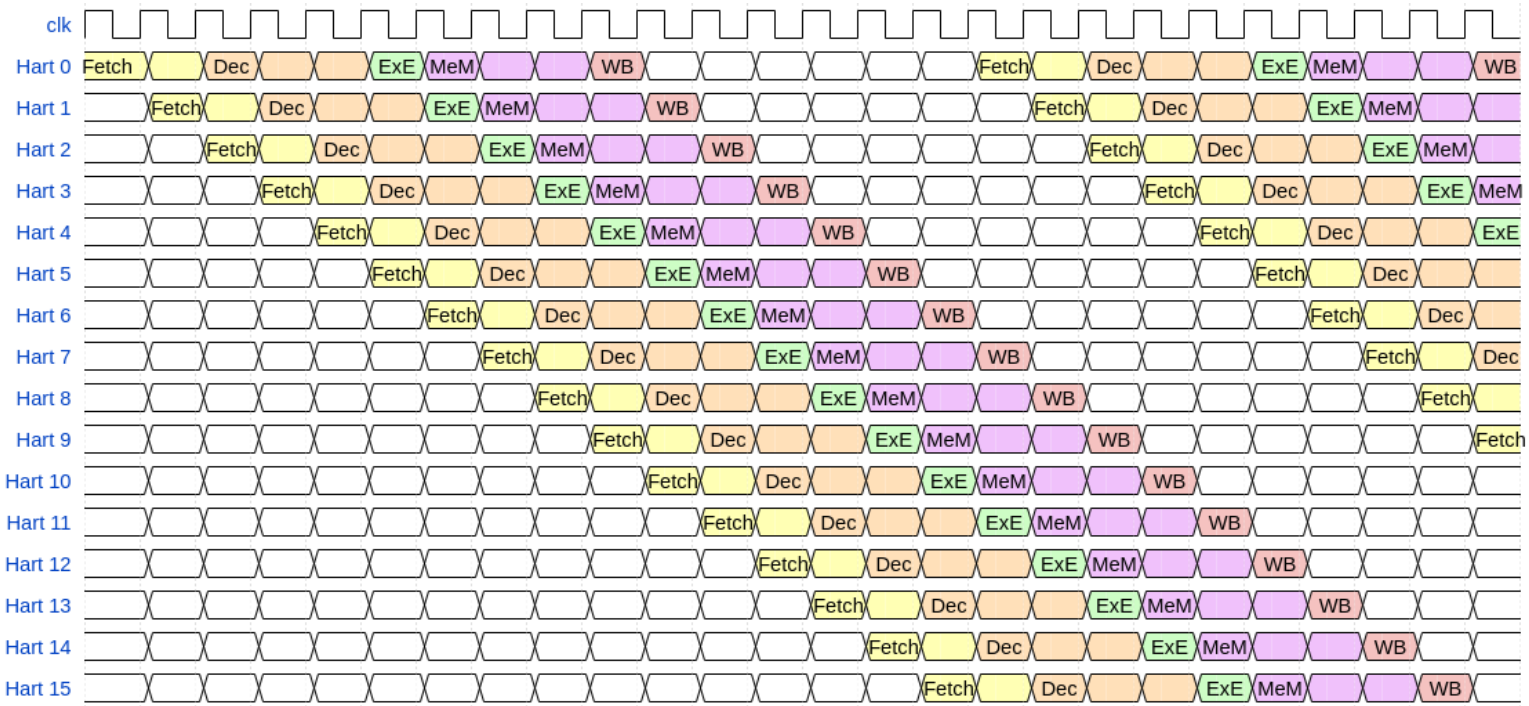


Figure – Interleaving 16 harts in the BRISKI barrel processor.

NOTE

- $N_{\textbf{Hardware Threads}} \geq N_{\textbf{physical pipeline stages}}$.
- **Time-sliced** (interleaved) multithreading : A new Hart is fetched each clock cycle.
- By the time the same Hart is fetched again, all branches and data hazards are resolved $\Longrightarrow$ No need for branch prediction or Register forwarding $\Longrightarrow$ Better MIPS/LUT and higher number of cores is possible

Barrel processor against other Soft-core approaches

Barrel processor against other Soft-core approaches.

	OoO	bitserial	in-order	Barrel (others)	Barrel (Ours)
Ref	[1]	[2]	[3]	[4]	BRISKI(New)
Year	2019	-	2016	2021	2025
FPGA	XC7Z020	Artix7	VU9P	KU040	VU9P
LUT	∼15k	**125**	320	2411	468
BRAM	6	-	0.5-1**	15	1
ISA	RV32IM	RV32I	RV32I+lr/sc*	RV32I	RV32I+lr/sc
Fmax(MHz)	95.3	220	375	250	**717 (Up to 737)**
CPI	-	>32	1.6	**1**	**1**
NativeMIPS (Fmax/CPI)	194	∼7	∼234	250	**717**
(MIPS/LUT)	0.012	0.055	0.73	0.103	**1.53**

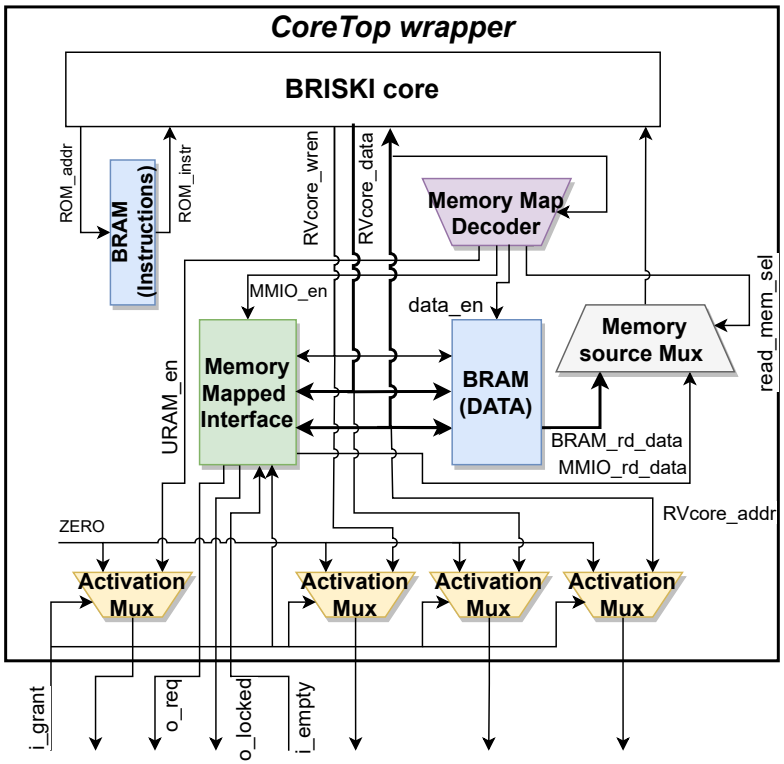
∗∗ The work [3] reports a BRAM utilization of 4 to 8 in a cluster of 8 cores leading to 0.5 to 1 BRAM for a single core.

∗ The work [3] reports that Multiply-shift and load/store byte-align sign-extension logic are implemented but shared by core pairs in a cluster.

BRISKI : Barrel RISC-V for Kilo-core Implementations

Interfacing BRISKI in the Kilo-core design

- **Memory Mapped Interface** to translate between LD/ST and control signals.
- **BRAM (4KB)** to store both private data and instructions .
- **URAM (32KB) shared** by many cores to collect results and stream them in packets to the host through PCIe.



SPARKLE System Hierarchy : General view

Hierarchical Design

- **MiniRow, Row, QuadRow clusters** ensure efficient scaling and facilitate physical mapping.
- Each QuadRow streams data to the backplane, managed by Global Arbiter.
- Asynchronous FIFOs and URAM for streaming computation results back to the host.

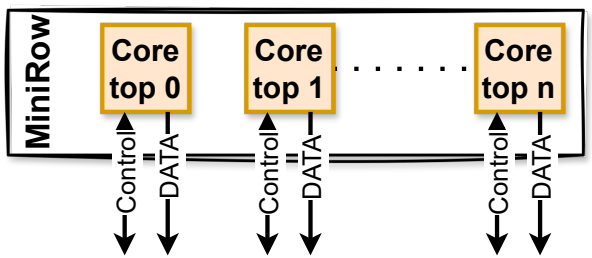


Figure – Structure of a MiniRow Cluster.

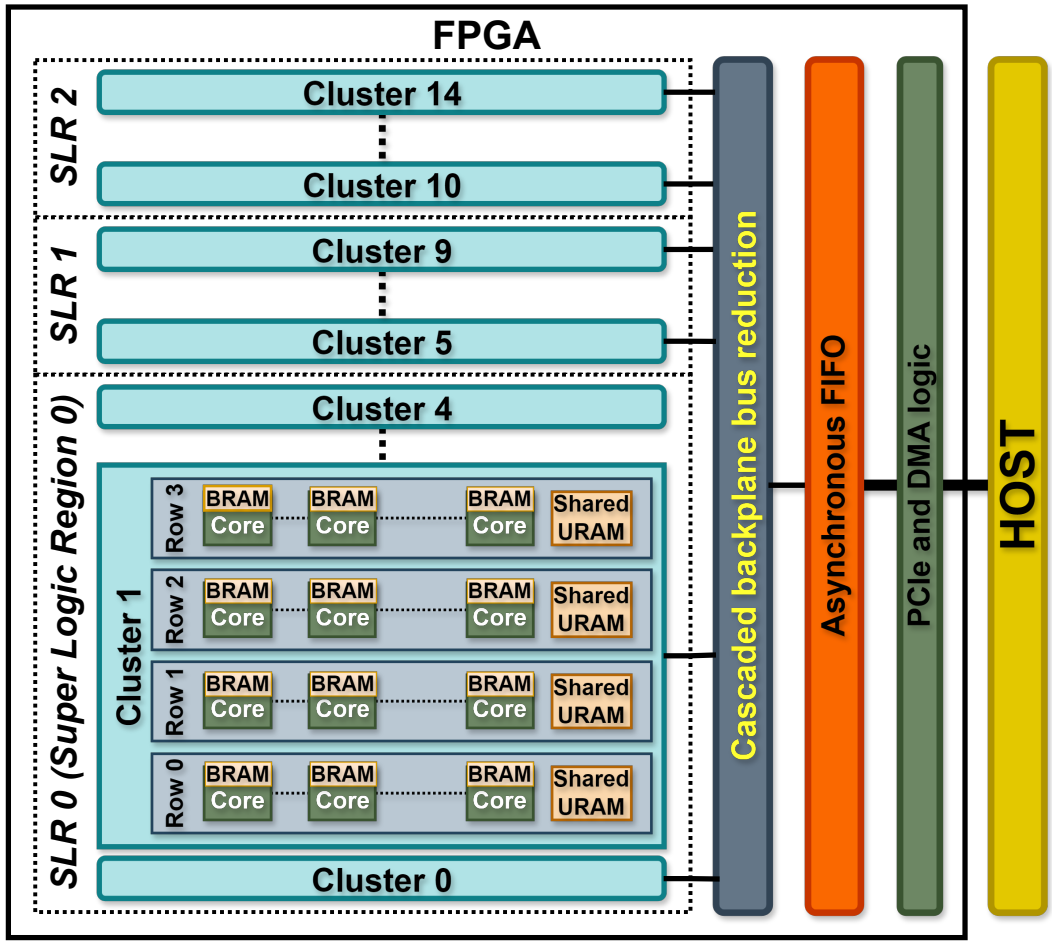


Figure – A bird's eye view on the SPARKLE Kilo-core system.

SPARKLE System Hierarchy : Row cluster

Hierarchical Design

- MiniRow clusters facilitate physical mapping.
- Shared memory access per Row, managed by Row Arbiter and Row synchronizer.
- Pipelined Cascaded reduction of processor streams to avoid large multiplexers towards the Row's shared URAM memory and maintain high clock speed.

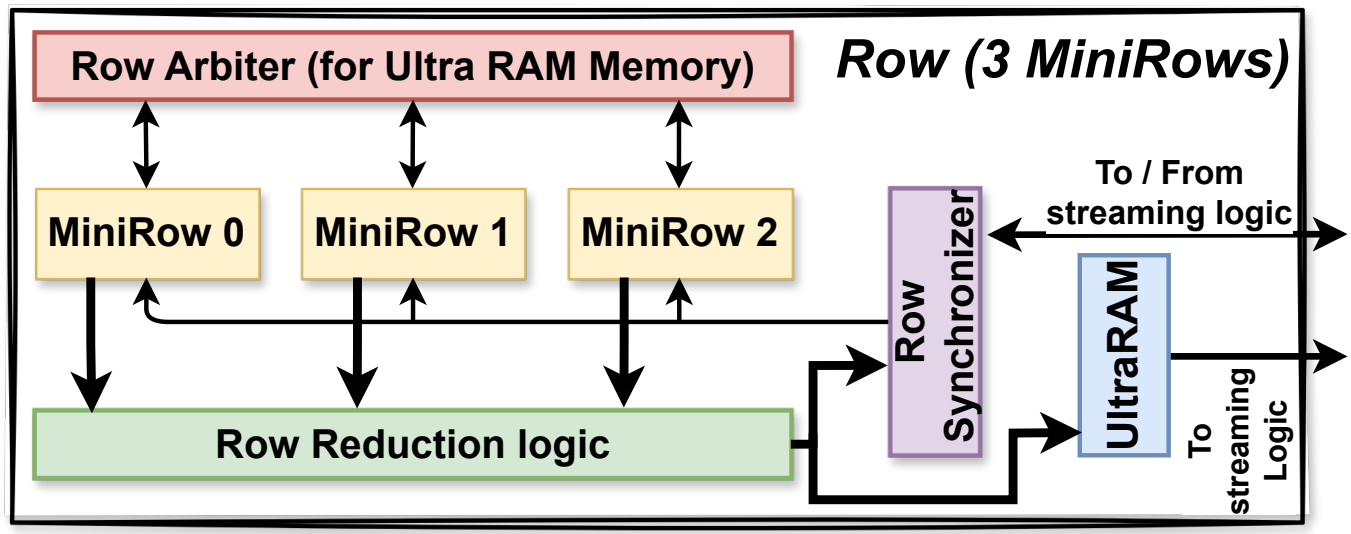


Figure – Architecture of a Row Cluster.

SPARKLE System Hierarchy : QuadRow cluster

Hierarchical Design

- Each QuadRow consists of 4 Row clusters.
- Streaming logic facilitates **overlap of data transfers and computations**.
- Control flow and data are cascaded in the upper and lower directions and pipelined using skid buffers.

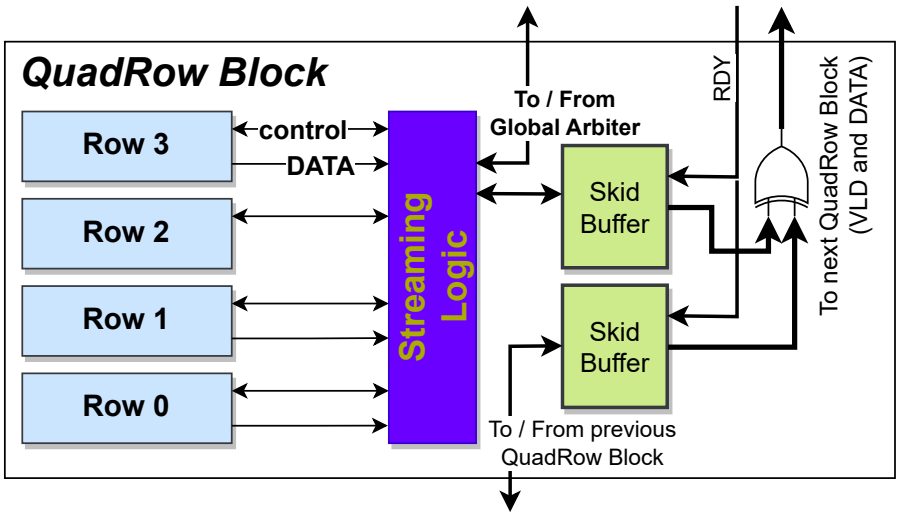


Figure – Architecture of a QuadRow Block.

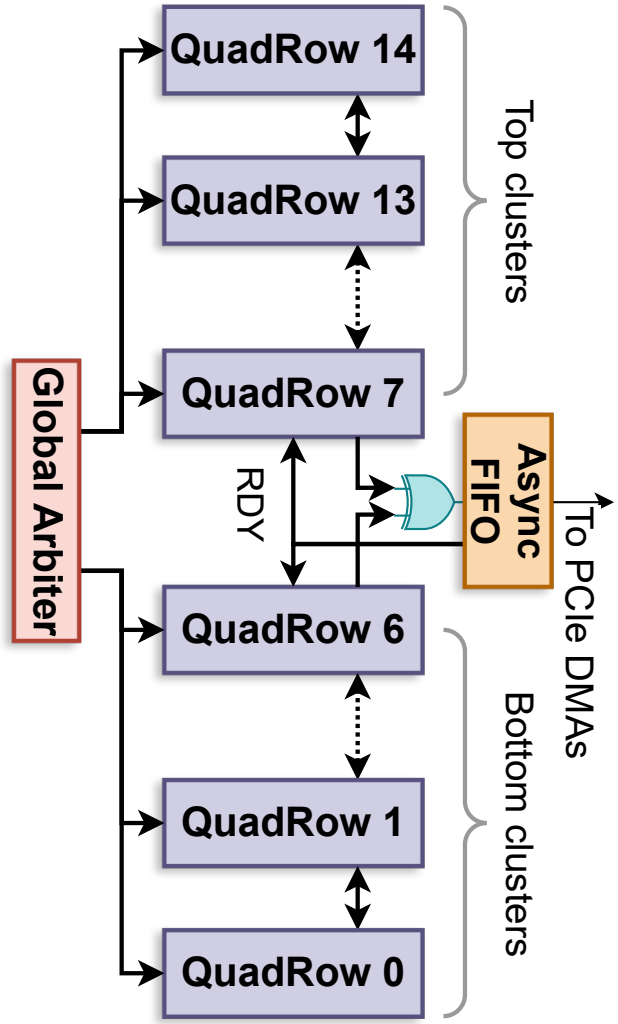


Figure – Architecture of the 1024-core Wrapper.

Target FPGA Resource Layout

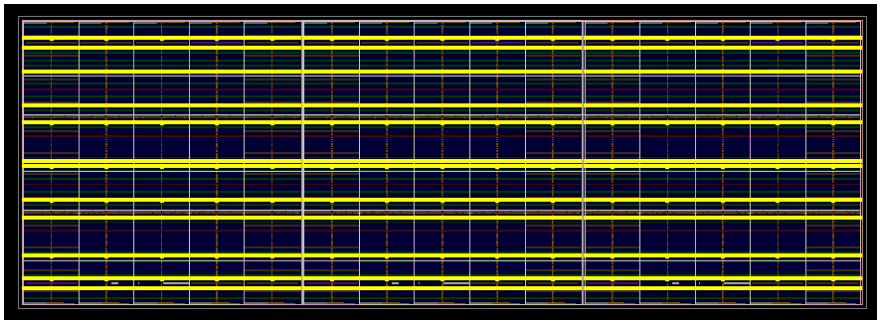


Figure – 12 Columns containing BlockRAM resources (180 BRAMs/Col for a total of 2160 BRAM (2160 RAMB36 or 4320 RAMB18)).

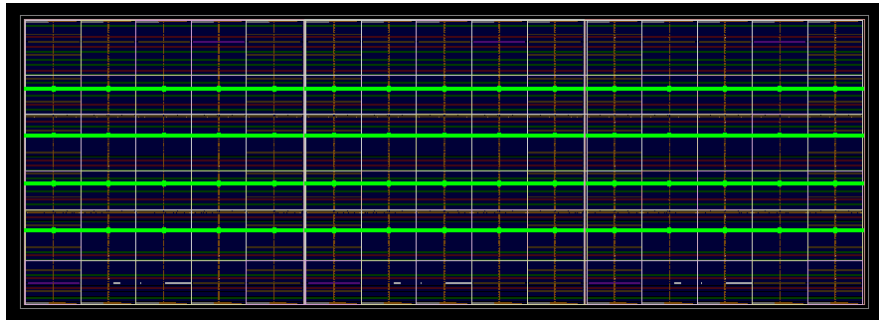


Figure – 4 Columns containing UltraRAM resources (240 URAMs/Col for a total of 960 UltraRAMs).

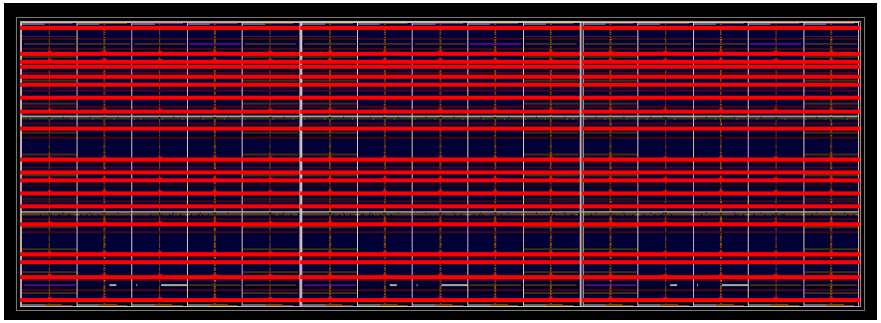


Figure – 19 Columns containing DSP resources (360 DSPs/Col for a total of 6840 DSPs).

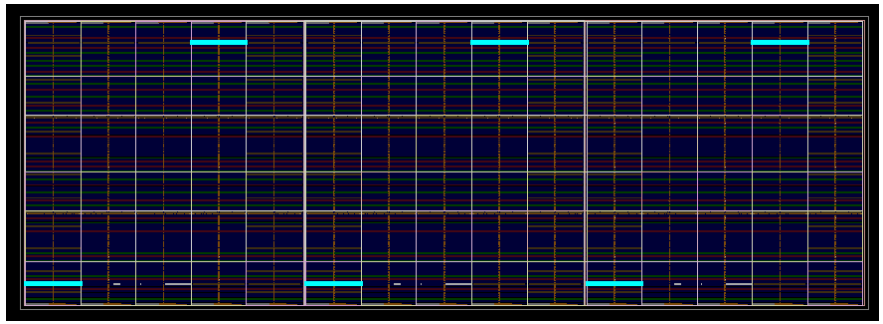


Figure – Columns containing PCIe resources

Finding an atomic partition (1)

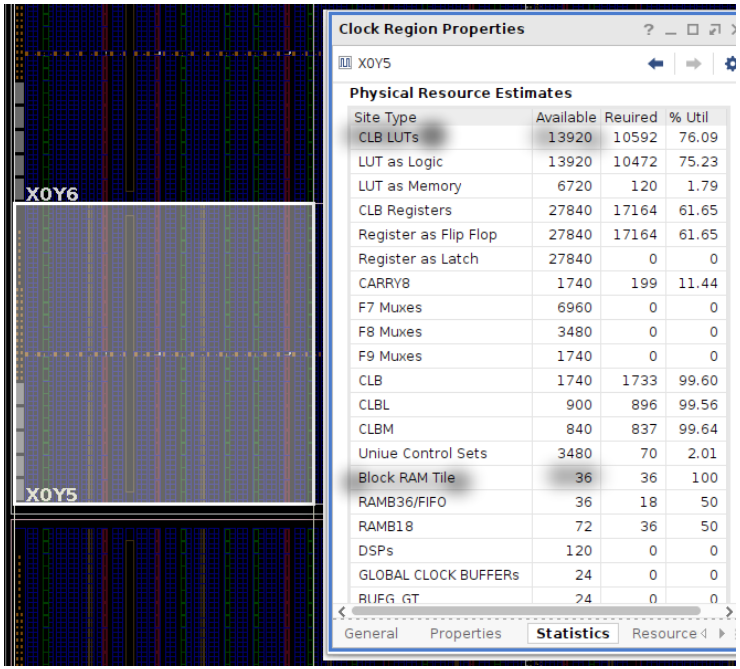


Figure – Example resources within a **wide** Clock Region (X0Y5).

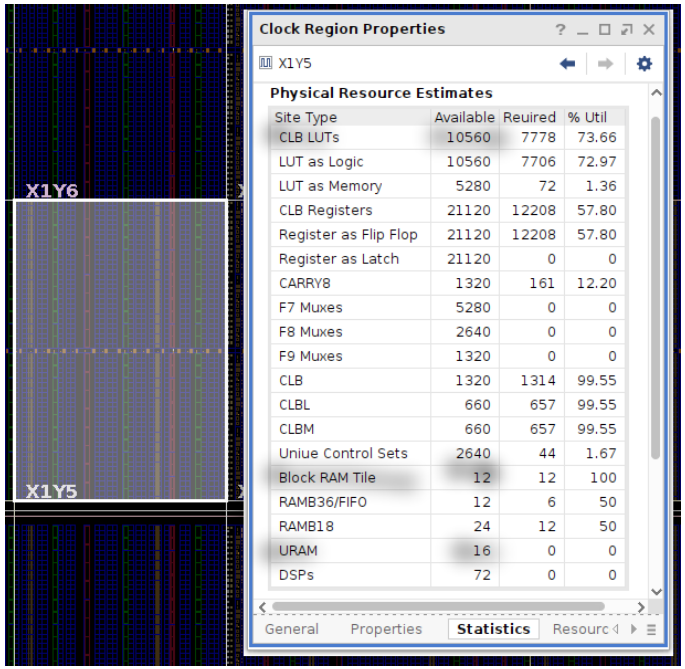


Figure – Example resources within a **narrow** Clock Region (X1Y5).

Available resources in Clock Region X0Y5

- 13920 LUTs
- 36 BlockRAMs (36 RAMB32 or 72 RAMB18).
- 0 UltraRAMs

Available resources in Clock Region X1Y5

- 10560 LUTs
- 12 BlockRAMs (12 RAMB32 or 24 RAMB18).
- 16 UltraRAMs

Finding an atomic partition (2)

Available resources in Clock Region X0Y5

- 13920 LUTs
- 36 BlockRAMs (36 RAMB32 or 72 RAMB18).
- 0 UltraRAMs

Available resources in Clock Region X1Y5

- 10560 LUTs
- 12 BlockRAMs (12 RAMB32 or 24 RAMB18).
- 16 UltraRAMs

Combined available resources in both Clock Region X0Y5 and X1Y5

- 24480 LUTs $\Longrightarrow$ can host **24 cores** with 1000 LUTs/core
- 48 BlockRAMs (48 RAMB32 or 96 RAMB18). $\Longrightarrow$ can host **24 cores** with 2 BlockRAMs/core (two RAMB18 (one BRAM) for RegisterFile and one BRAM for data/instructions).
- 16 UltraRAMs. $\Longrightarrow$ for a 24-core partition, each 6 cores can share 4 UltraRAMs, **however we will have a different use case for those UltraRAMs**.

Floorplanning and Resource Management

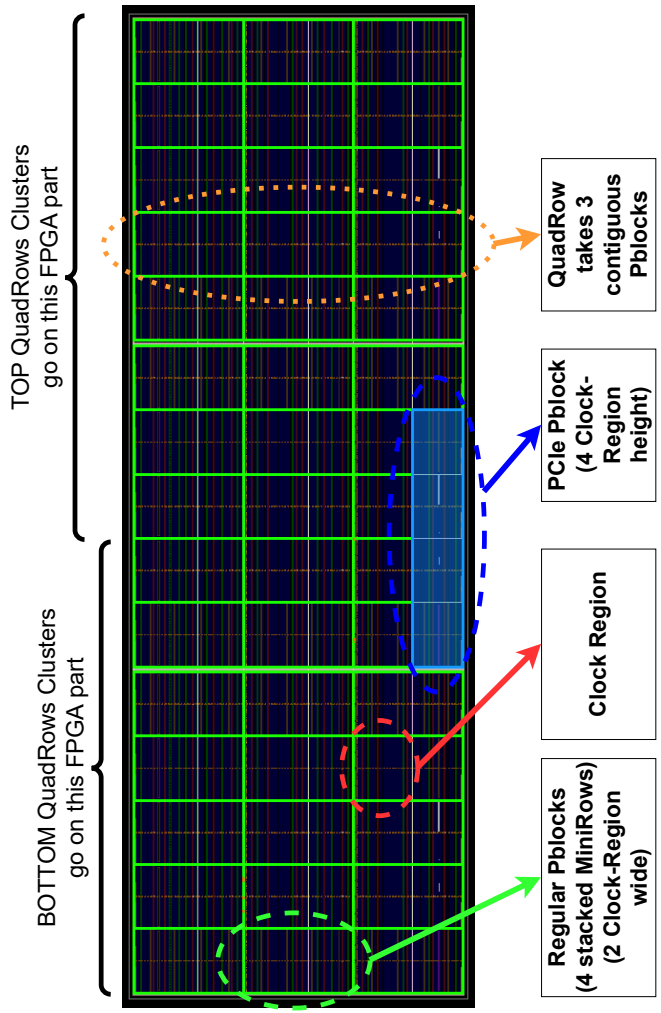


Figure – Floorplan of SPARKLE on the VU9P FPGA.

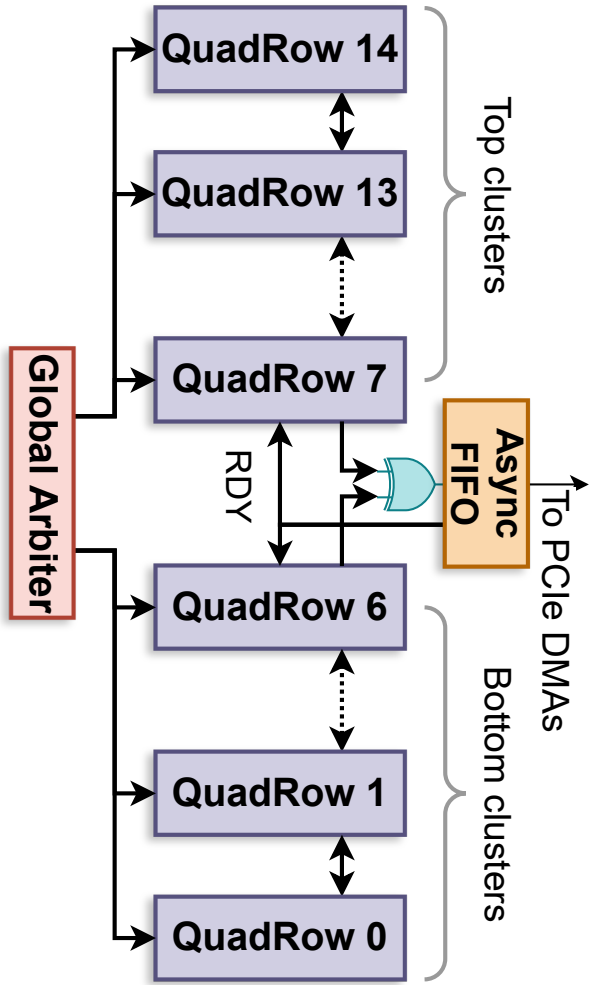


Figure – Architecture of the 1024-core Wrapper.

Programming Model and Data Management

Programming Paradigms

- Support for SIMD, MIMD, and SPMD execution models.

Memory Management

- Each Hart operates on isolated memory regions for parallel execution.

Synchronization

- Hardware and software barriers for multi-level orchestration.

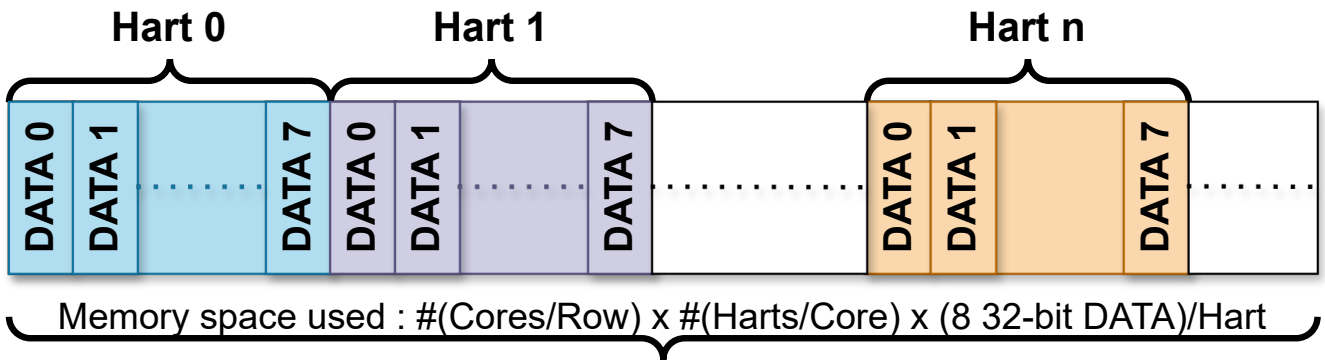


Figure – Memory Space partitions between Harts of a given Row and relative mapping on the Row's URAM memory.

Evaluation Summary

Results

- **1,024 cores** and **16,384 hardware threads** at **500 MHz**.
- **630K LUTs — 1,024 DSPs —- 2,048 BRAMs —— 60 URAMs**
- **Thousands of DSPs** and nearly **half the FPGA LUTs** available for **accelerators** and **custom instructions**.

Scalability

- Designed to fit **larger or smaller** FPGA configurations.
- Demonstrates flexibility in speed and throughput optimization.

Key Insights

- Outperforms similar architectures in overall compute density (**0.81 MIPS/LUT**).
- Combines high thread count and full RISC-V ISA compatibility.

Scalability

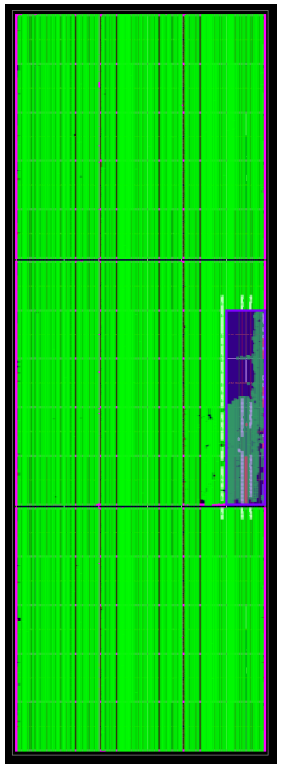


Figure – 1,024 cores (16,384 threads) @400 MHz
on a VU9P FPGA with a PCIe backplane.

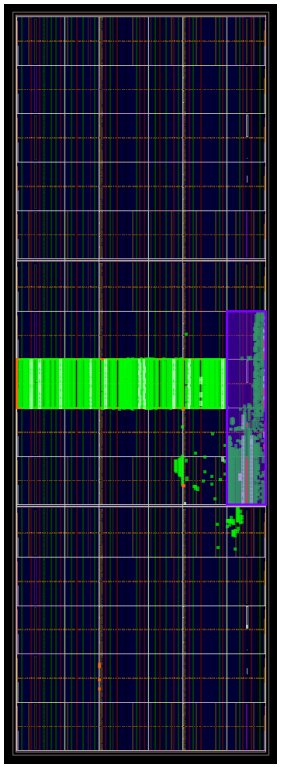


Figure – 56 cores (896 hardware threads) @500 MHz
on a VU9P FPGA with a PCIe backplane.

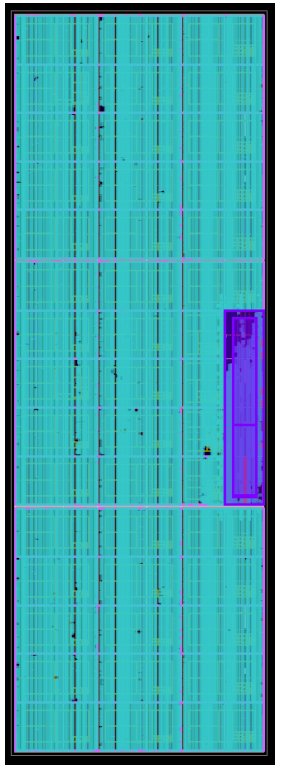


Figure – 1,024 cores (16,384 threads) @500 MHz
on a VU9P FPGA with a PCIe backplane.

Demonstrated many-core implementations

- 56-core design at 500 MHz.
- 1,024-core design at 400 MHz (**new design pushed to 500 MHz**)

Conclusion and Future Work

Conclusion

SPARKLE sets a benchmark in scalable, high-speed RISC-V computing on FPGAs with **16,384 threads clocking @ 500 MHz**.

Applications

- Sorting benchmarks **tested on the card**.
- Many other kernels in active development (**Raymarching, Mandelbrot, DNA alignment, text search, hashing, etc.**).

Future Directions

- **Enhance** communication/synchronization mechanisms.
- **Extend** ISA support with custom instructions/accelerators.
- **Build** a full software stack and compute kernels.
- **Port** to other FPGAs targets.

References

Reference [1]

S. Mashimo, A. Fujita, R. Matsuo, S. Akaki, A. Fukuda, T. Koizumi, J. Kadomoto, H. Irie, M. Goshima, K. Inoue, and R. Shioya, “An open source fpga-optimized out-of-order risc-v soft processor,” in 2019 International Conference on Field-Programmable Technology (ICFPT), 2019, pp. 63–71.

Reference [2]

O. Kindgren. bit-serial risc-v. [Online]. Available : https ://github.com/ olofk/serv

Reference [3]

J. Gray. (2017) GRVI Phalanx : A Massively Parallel RISC- V FPGA Accelerator Framework A 1680-core, 26 MB Parallel Processor Overlay for Xilinx UltraScale+ VU9P. [Online]. Available : https ://carrv.github.io/2017/papers/gray-phalanx-carrv2017.pdf

Reference [4]

M. AskariHemmat, O. Bilaniuk, S. Wagner, Y. Savaria, and J.-P. David, “RISC-V Barrel Processor for Deep Neural Network Accelera- tion,” in 2021 IEEE International Symposium on Circuits and Systems (ISCAS), 2021, pp. 1–5.