# RISC-V based GPGPU on FPGA: A Competitive Approach for Scientific Computing ?

Eric Guthmuller

Jérôme Fereyre


*RISC-V Summit Europe, 2025-05-13*

# Motivation

- Scientific computing applications require 64b floating point computing precision
  - Sometimes, 64b is not even enough…

(see [1] E. Guthmuller, *et al.*, "*Xvpfloat*: RISC-V ISA Extension for Variable Extended Precision Floating Point Computation", (2024) IEEE Transactions on Computers)

| Rank | System | Cores | Rmax (PFlop/s) |
|---|---|---|---|
| 1 | **El Capitan** - HPE Cray EX255a, AMD 4th Gen EPYC 24C 1.8GHz, AMD Instinct MI300A, Slingshot-11, TOSS, HPE<br>DOE/NNSA/LLNL<br>United States | 11,039,616 | 1,742.00 |
| 2 | **Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE Cray OS, HPE<br>DOE/SC/Oak Ridge National Laboratory<br>United States | 9,066,176 | 1,353.00 |

2024-11 TOP500

- GPGPUs have enabled exa-FLOPs class performance in recent supercomputers
  - Codes have been adapted to GPGPU computing paradigm (costly)

- But AI market is exploding and is much bigger than scientific computing

⇒ GPGPUs are more and more optimized for low precision computing

⇒ **How long before 64b support is dropped or emulated ?**

# Motivation

- Scientific computing applications require 64b floating point computing precision
  - Some
  (see [1] G
  Extended
  Compute

- GPGP
  super
  - Cod

| Rank | System | Cores | Rmax (PFlop/s) |
|---|---|---|---|
| 1 | **El Capitan** - HPE Cray EX255a, AMD 4th Gen EPYC 24C 1.8GHz, AMD Instinct MI300A, Slingshot-11, TOSS, HPE<br>DOE/NNSA/LLNL<br>United States | 11,039,616 | 1,742.00 |
| | | 9,066,176 | 1,353.00 |

24-11 TOP500

t

Our objective:
Explore the feasibility/performance of a GPGPU implemented on FPGA with support for FP64 computation and targeting scientific computing use cases

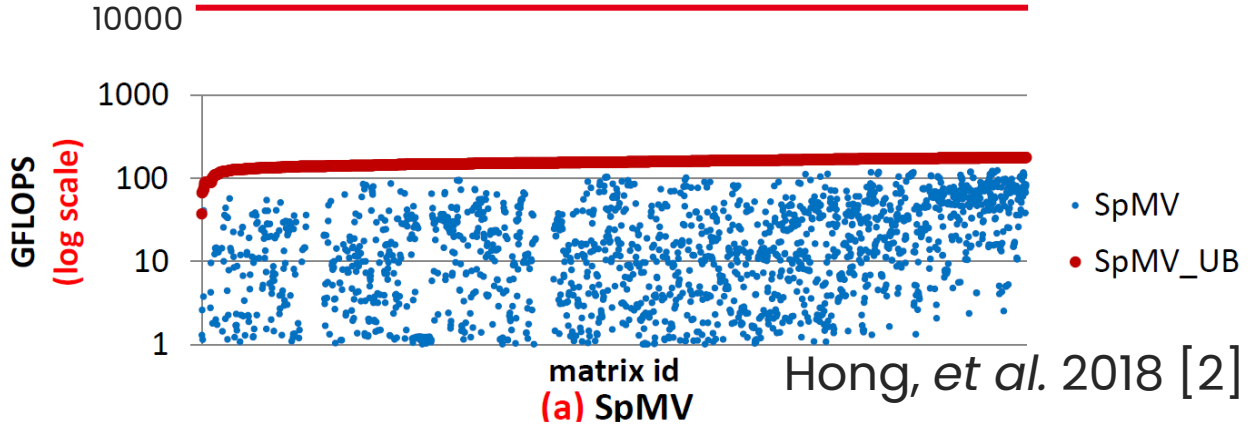
- But AI market is exploding and is much bigger than scientific computing

⇒ GPGPUs are more and more optimized for low precision computing

⇒ How long before 64b support is dropped or emulated ?

# Brief intro to scientific computing kernels

- Scientific computing apps rely on well optimized linear algebra frameworks, e.g. *BLAS*
  - Mostly vector-vector (lvl1) and matrix-vector (lvl2)




Hong, *et al.* 2018 [2]

- These kernels have low arithmetic intensity

⇒ Limited by memory bandwidth

  - For example Sparse Matrix-Vector (SpMV) is running at ~1% peak FLOPS of Nvidia P100

```
function solve_CG(A::Matrix{Float64},
    b::Vector{Float64},
    tol::Float64=1e-12; max_iter::Int=Int(1e5)
    )
n = length(b)
x = zeros(n)  # Initial guess
r = b - A * x  # Initial residual
p = copy(r)  # Initial search direction
r_norm = norm(r)

for iter in 1:max_iter
    # matrix-vector multiplication
    Ap = A * p
    # vector-vector operations
    alpha = dot(r, r) / dot(p, Ap)
    x += alpha * p
    r -= alpha * Ap
    new_r_norm = norm(r)

    if new_r_norm < tol
        return x, iter
    end

    beta = (new_r_norm / r_norm)^2
    p = r + beta * p
    r_norm = new_r_norm
end
return x, max_iter
end
```

Roofline model for HBM2 (2 stacks, 820 GB/s)

| Kernel | Operation | Max GFLOPS |
|---|---|---|
| DOT | $\propto = x \cdot y$ | 205 |
| AXPY | $y = \propto x + y$ | 68 |
| SCAL | $x = \propto x$ | 102 |
| SpMV | $y = Ax$ | 135 |

# Brief intro to scientific computing kernels

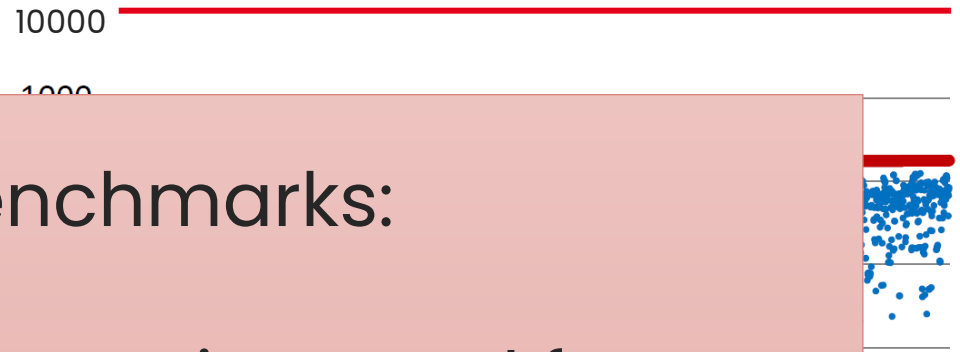
- Scientific computing apps rely on well
  algeb
  - Mos
    mat
- These
- ⇒ Limite
  - For e
    ~1% p

Targeted benchmarks:

1/ LINPACK: arithmetic intensive, used for TOP500 ranking
2/ HPCG [3]: memory & IO intensive; although pessimistic, more representative of real use cases

Roofline model for HBM2 (2 stacks, 820 GB/s)

| Kernel | Operation | Max GFLOPS |
|---|---|---|
| DOT | $\propto = x \cdot y$ | 205 |
| AXPY | $y = \propto x + y$ | 68 |
| SCAL | $x = \propto x$ | 102 |
| SpMV | $y = Ax$ | 135 |




*et al.* 2018 [2]

```
Matrix{Float64},
2; max_iter::Int=Int(1e5))

l guess
al residual
 search direction

ultiplication

operations
, r) / dot(p, Ap)
x += alpha * p
r -= alpha * Ap
new_r_norm = norm(r)

if new_r_norm < tol
    return x, iter
end

beta = (new_r_norm / r_norm)^2
p = r + beta * p
r_norm = new_r_norm
end
return x, max_iter
end
```

# Why GPGPU/Vortex on FPGA ?

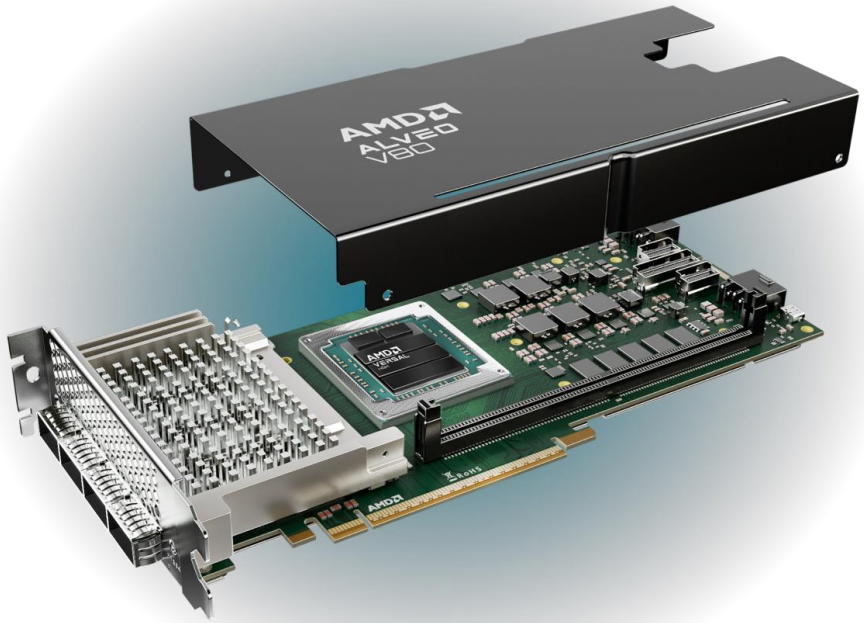
- CGRA architectures certainly map better on FPGA
  - Very high resource usage possible (>90%) at high frequency (>400MHz)

- But programming model is a prime decision factor
  - CEA sci comp users: “Scientific computing is not limited to BLAS calls”
  - Already familiar with GPGPU frameworks, e.g. OpenCL

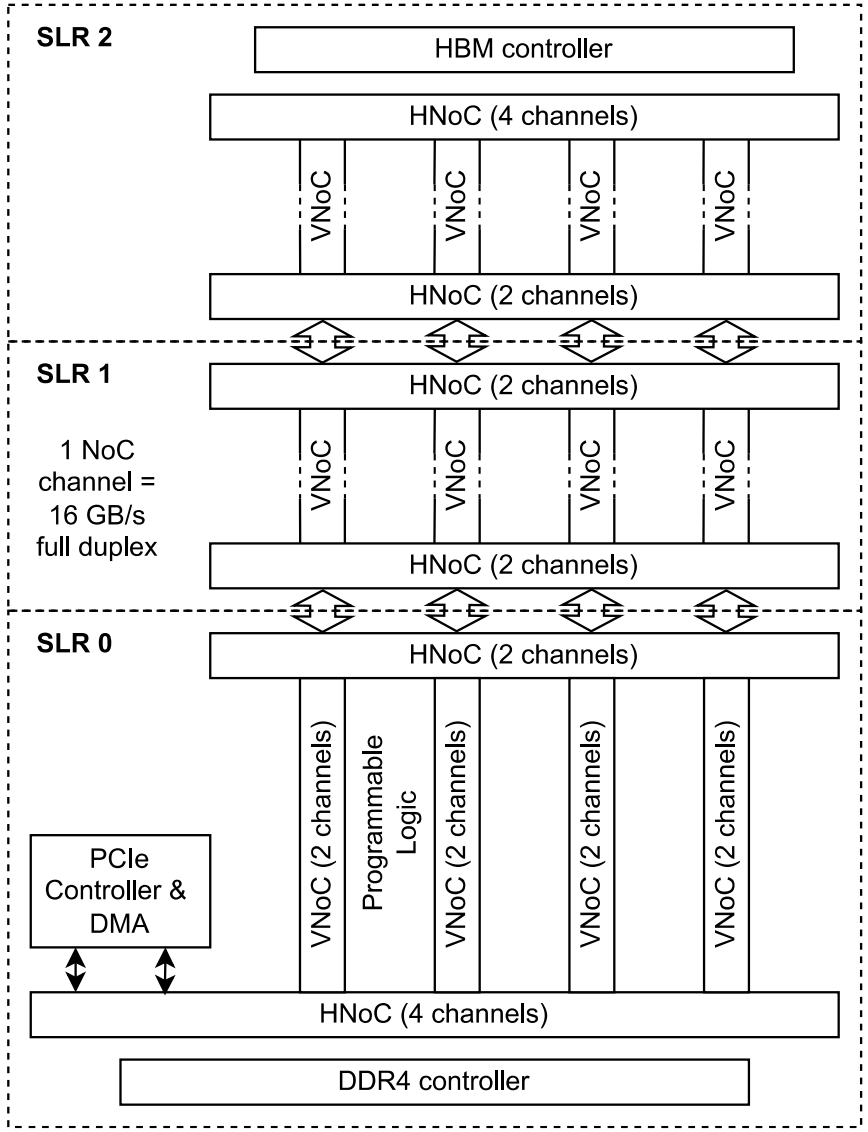
- Previous works on RISC-V HW & SW
  - CEA member of OpenHW Group & contributing to open HW

⇒ **VORTEX [4] is the best candidate for our project**

# Targeted FPGA platform: AMD Alveo V80



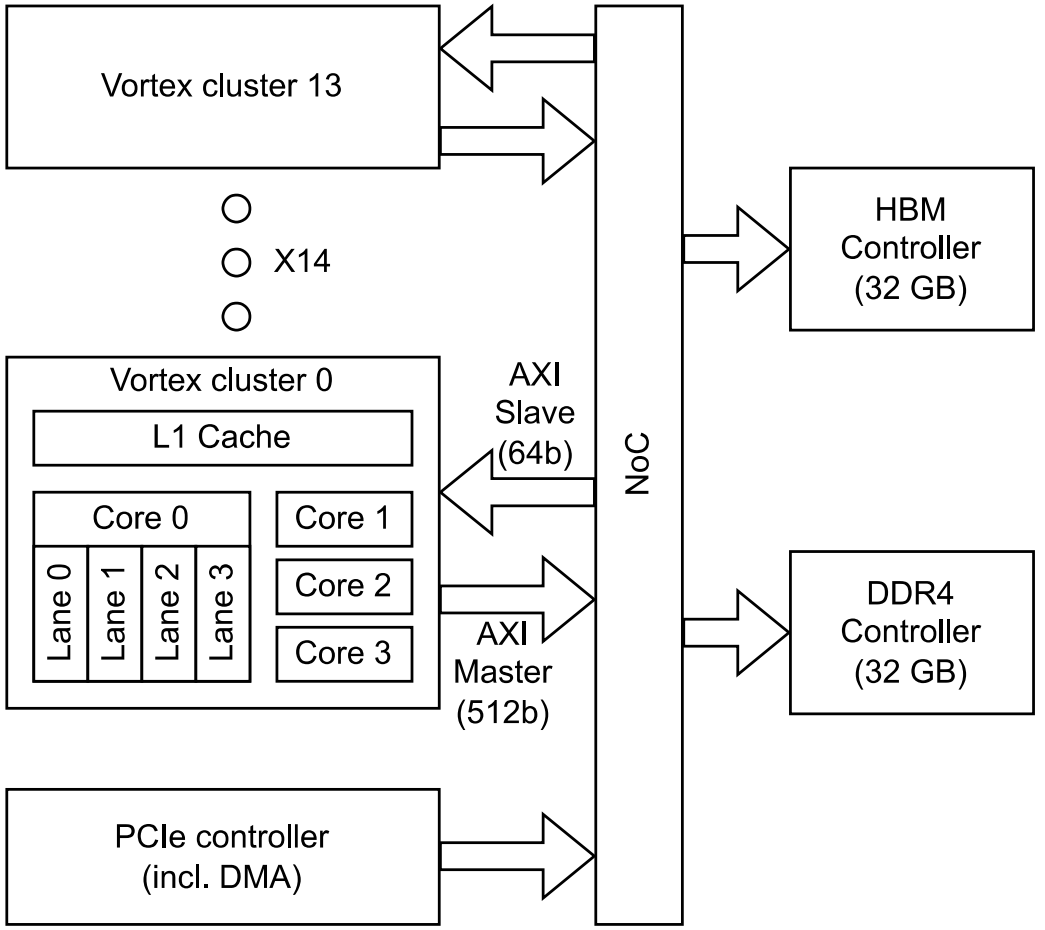

32 GB of HBM, 820 GB/s
32 GB of DDR4, 26 GB/s
PCIe 5.0 8x, 64 GB/s
128 GB/s VNoC
256 GB/s HNoC




- Targeting biggest AMD FPGA that includes HBM
- Alveo V80 offers a GPU form factor designed for server blade integration

# Early results

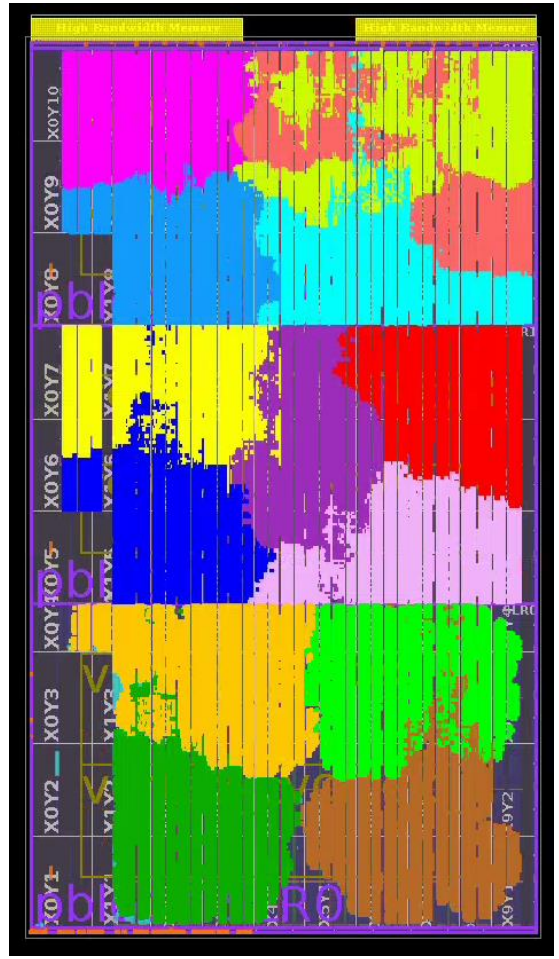
- Up to 14 clusters of 4 cores (4W4T)
  - Connected to VNoC, distributed to maximize bandwidth
  - SpMV difficult to vectorize → narrower cores ?
  - FP32 for now (FP64 not available on FPGA in vanilla Vortex)
- Each cluster has its own configuration interface *(WIP)*
  - Goal: avoid SLR crossing
- Using Xilinx PCIe included DMA (called QDMA) for host↔device communication
  - Goal: avoid Xilinx XRT software stack
  - Successfully ran OpenCL kernels on smaller configuration using AMD VCK190 board



# Early results

- Up to 126 GFLOPS peak at 56 cores
  - Frequency quite stable, even with quite high utilization ratio (70%)
  - No SLR crossing

- Only L1 caches → no big RAM (URAM) used
  - Memory hierarchy WIP



| Config | LUTs | FFs | RAM small | RAM big | DSP | Freq | Peak FP32 |
|---|---|---|---|---|---|---|---|
| 4 cores | 5% | 3% | 4% | 0% | <1% | 300 MHz | 10 GFLOPS |
| 56 cores | 70% | 39% | 39% | 0% | 4% | 282 MHz | 126 GFLOPS |
| AMD Versal HBM XCV80 | 2.5 M | - | 132 Mb | 541 Mb | 10.8 K | ~800 MHz | 17.5 TFLOPS |

# Conclusion/Challenges Ahead

- Beginning of our journey towards FP64 GPGPU on FPGA
  - First implementation trials promising: high utilization and high frequency
  - First kernels already running over AMD Versal PCIe QDMA
- Next tasks:
  - Porting OpenCL BLAS & benchmarks: Linpack and HPCG
  - Use FP64 soft macros from AMD/Xilinx to implement FP64
- Main challenges: memory hiearchy
  - AMD's FPGA NoC is limited to 128 GB/s of vertical bandwidth, falling short of the 819 GB/s of HBM bandwidth → Need for another vertical interconnect, maybe also horizontal ?
  - Read latency could top one hundred cycles, much higher than what could be masked using the 4 hardware threads of our current Vortex configuration → More wavefronts and streamers ?



# Thanks !

Contact: eric.guthmuller@cea.fr

# References


[1] E. Guthmuller, *et al.*, "*Xvpfloat*: RISC-V ISA Extension for Variable Extended Precision Floating Point Computation", (2024) IEEE Transactions on Computers

[2] C. Hong, *et al.*, "Efficient sparse-matrix multi-vector product on gpus," in Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing, ser. HPDC '18. New York, NY, USA: Association for Computing Machinery, 2018. doi: 10.1145/3208040.3208062. ISBN 9781450357852 p. 66–79.

[3] J. Dongarra, M. A. Heroux, and P. Luszczek, "A new metric for ranking high-performance computing systems," National Science Review, vol. 3, no. 1, pp. 30–35, 01 2016. doi:10.1093/nsr/nwv084. ISSN 2095-5138

[4] B. Tine, K. P. Yalamarthy, F. Elsabbagh, and K. Hyesoon, "Vortex: Extending the RISC-V ISA for GPGPU and 3D-Graphics," in MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture, ser. MICRO '21. New York, NY, USA: Association for Computing Machinery, 2021. doi: 10.1145/3466752.3480128. ISBN 9781450385572 p. 754–766.