# A Dataflow Overlay for Monte Carlo Multi-Asset Option Pricing on AMD Versal AI Engines


**Mohamed Bouaziz**[1], Michael Samet[2], Suhaib A. Fahmy[1]

[1]: *KAUST, Saudi Arabia,* [2]: *RWTH Aachen University, Germany*

# Multi–Asset Option Pricing

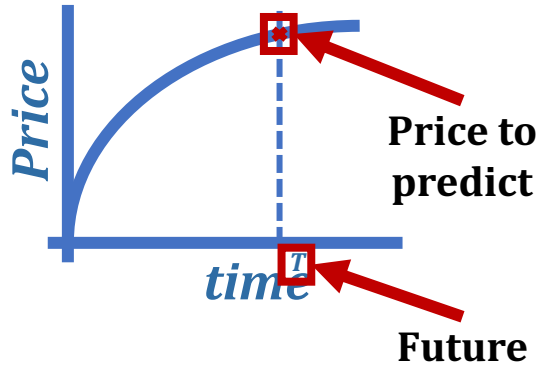



**Fig. : Asset price over time.**

*Correlated variates of the option's multiple assets*

**Price evoluation (GBM model)**

*Multiple Assets*

$$S_0(T) = S_0(0)\exp\left(\left[r - \frac{1}{2}\sigma_0^2\right]T + \sigma_0\sqrt{T}\sum_{1\le j\le d} L_{0,j}Z_j\right)$$

$$S_1(T) = S_1(0)\exp\left(\left[r - \frac{1}{2}\sigma_1^2\right]T + \sigma_1\sqrt{T}\sum_{1\le j\le d} L_{1,j}Z_j\right)$$

$$\vdots$$

$$S_d(T) = S_d(0)\exp\left(\left[r - \frac{1}{2}\sigma_d^2\right]T + \sigma_d\sqrt{T}\sum_{1\le j\le d} L_{d,j}Z_j\right)$$

# Multi–Asset Option Pricing

*Brownian motion (source of randomness)*

**Price evoluation (GBM model)**

$$S_i(T) = S_i(0)\exp\left(\left[r - \frac{1}{2}\sigma_i^2\right]T + \sigma_i\sqrt{T}\sum_{1\le j\le d} L_{i,j}Z_j\right)$$

→ **Monte Carlo Simulation** ✓

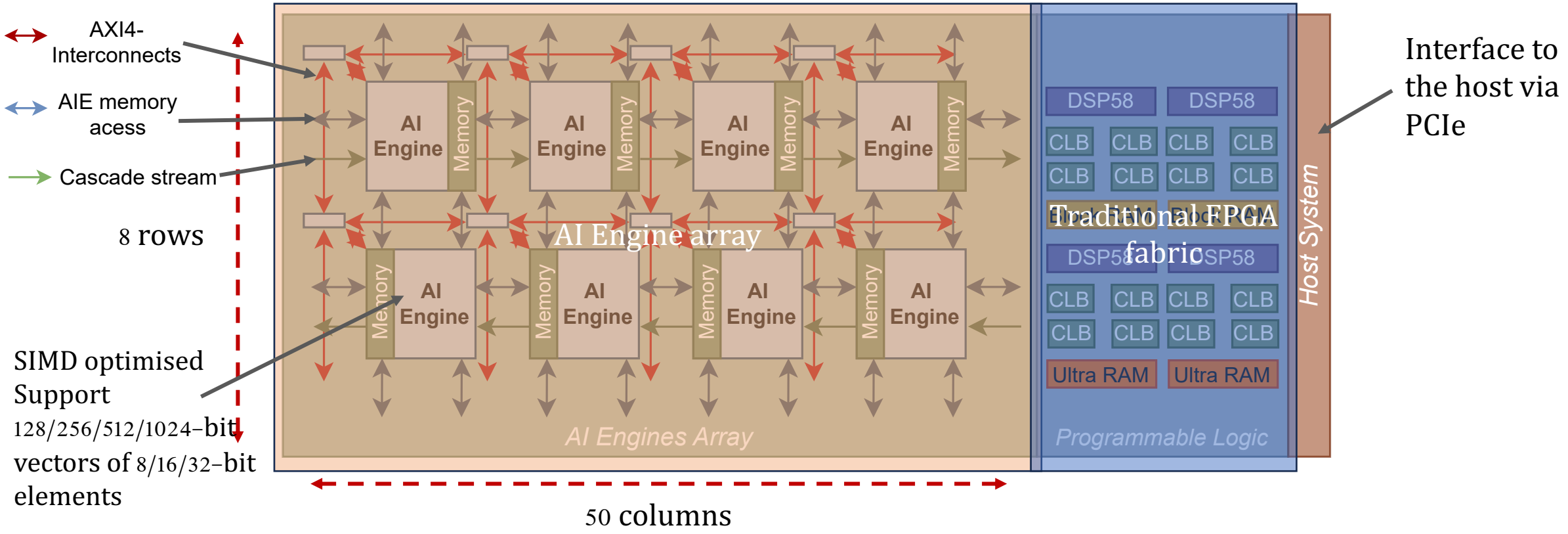
***Compute intensive*** ✗

Existing work:

- Mostly focuses on pricing many single–asset options in parallel.
- Lacks a proper, scalable dataflow design.

→ ***Need for a highly parallel dataflow design of the MC–based pricer***

# AMD Versal SoC




- SIMD optimised
- Support 128/256/512/1024-bit vectors of 8/16/32-bit elements

# Dataflow Overlay Architecture

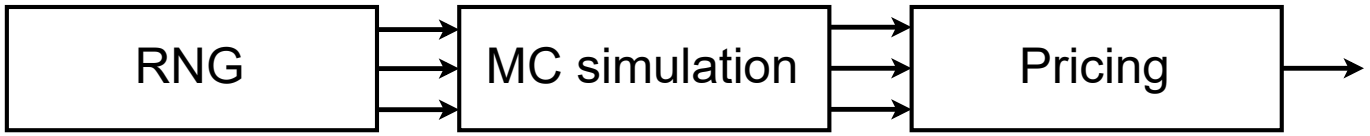



**Fig. : Pricer flow.**

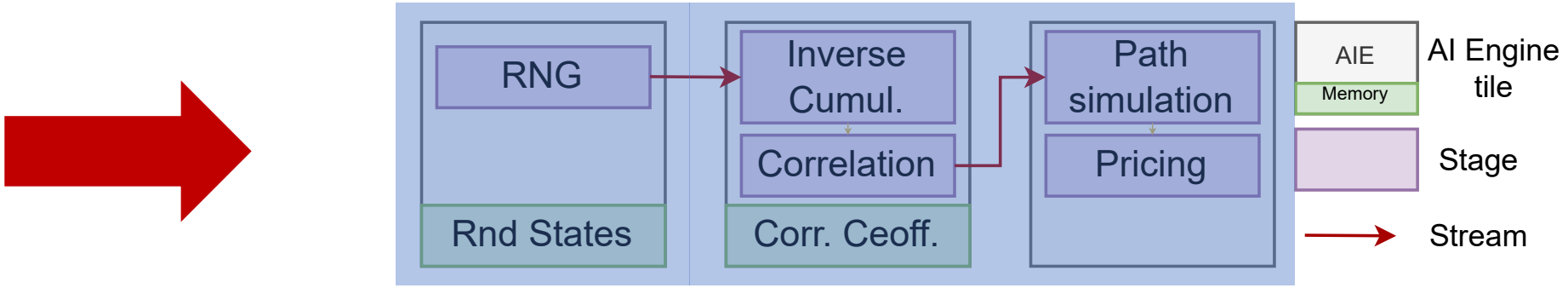



**Fig. : Pricer flow mapping to single CU.**

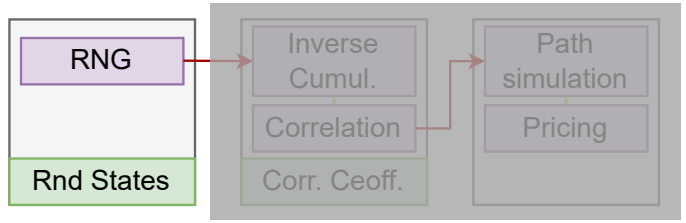
# Single Compute Unit Design

RNG is needed for MC simulation

SIMD compliant

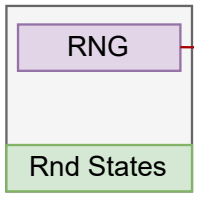
Suitable for MC

SIMD Fast Mersenne Twister (SFMT) *MCQMC'06*




- 128–bit (4 x 32–bit) based SIMD operations.
- Long period generator.

# Single Compute Unit Design




128–bit (4 x 32–bit) based SIMD operations:

- Bitwise logical operations.
- Per–lane shifting operations.
- Vector shifting operations.

*These are not natively supported!*

*Emulating these operations by composing supported ones.*

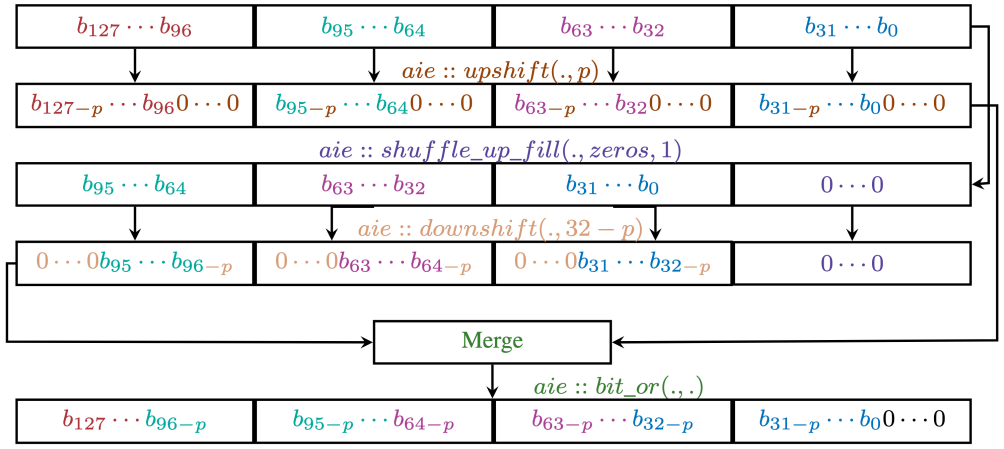



**Fig. : 128–bit vector shift left operation**

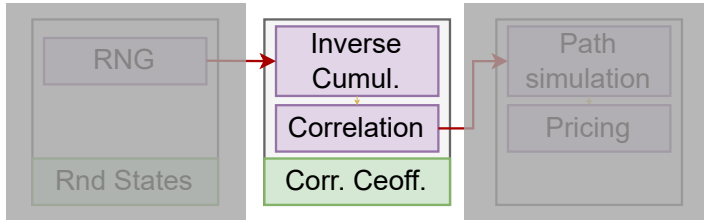
For more PRNG exploration on AI Engines: *M. Bouaziz, S. A. Fahmy: "**PRNGine**: Massively Parallel Pseudo-Random Number Generation and Probability Distribution Approximations on AMD AI Engines", IPDPSW'25*

# Single Compute Unit Design

**Price evoluation (GBM model)**

*Correlation operation* *Brownian motion*

$$S_i(T) = S_i(0)\exp\left(\left[r - \frac{1}{2}\sigma_i^2\right]T + \sigma_i\sqrt{T}\sum_{1\le j\le d} L_{i,j}Z_j\right)$$




➡ Need to generate normally distributed numbers

➡ Apply the inverse cumulative normal distribution function to the uniformly distributed numbers.

This should be done in SIMD fashion ➡ Inner part of Acklam's approximation: rational fraction

# Single Compute Unit Design

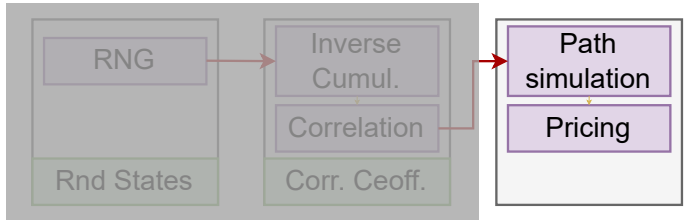
**Price evoluation (GBM model)**

*Constant: Calculated once*

*Calculated with every path*

$$S_i(T) = S_i(0)\exp\left(\left[r - \frac{1}{2}\sigma_i^2\right]T + \sigma_i\sqrt{T}\sum_{1\le j\le d} L_{i,j}Z_j\right)$$

*Emulated by the programming API*




➡ The exponential operation is expensive

BUT! $Payoff = \max\{0, \min(S_i(T)) - K\}$ ➡ There is need for $S_i(T)$ iff $\min(S_i(T)) > K$

iff $log(\min(S_i(T))) > \log(K)$

iff $min(\log(S_i(T))) > \log(K)$ ∗

➡ There is a need to calculate the expensive exponential only when the condition ∗ holds

# The Overlay Architecture

- 3–AIE – based compute units are replicated throughout the entire array
- Simulation parameters and the results are streamed in and out.

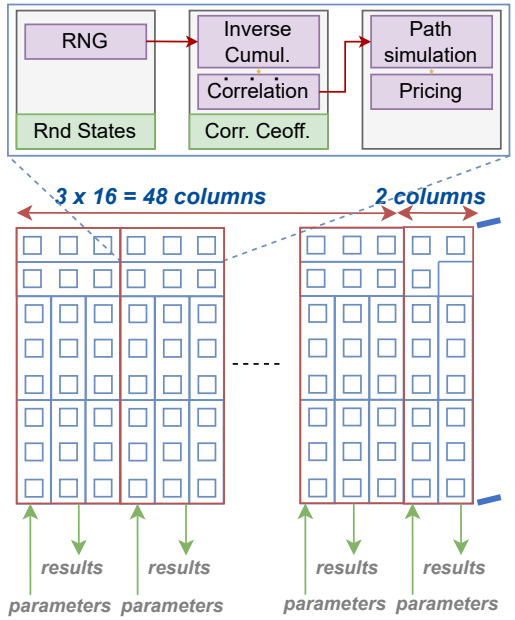



Fig.: Mapping of 133 CUs to AIE array.

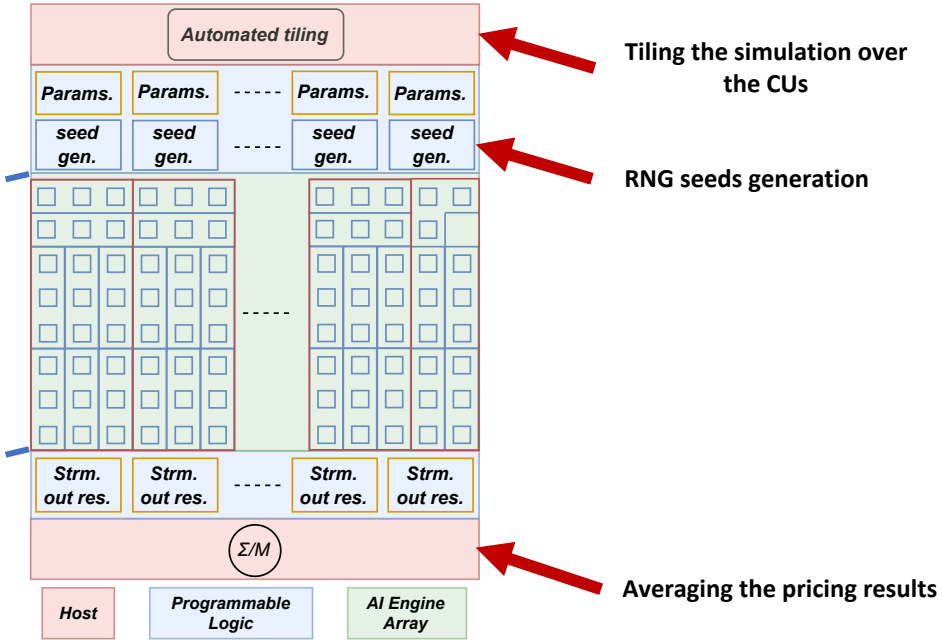
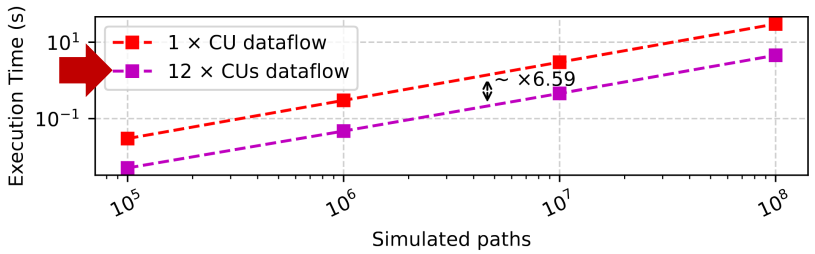
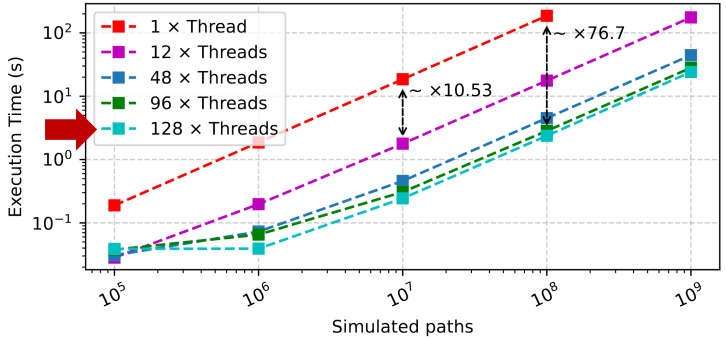



Fig.: Tasks allocation on Versal SoC.

# Experimental Setup

| FPGA | CPU | GPU |
|---|---|---|
| • Dataflow design using the AMD Vitis quant. finance library.<br>• 1—12 CUs dataflow. | • C++ based multi-threaded implementation using OpenMP.<br>• 1—128 parallel threads on AMD EPYC 7763. | • CUDA-based implementation on Nvidia RTX A6000 GPU. |



# Performance result

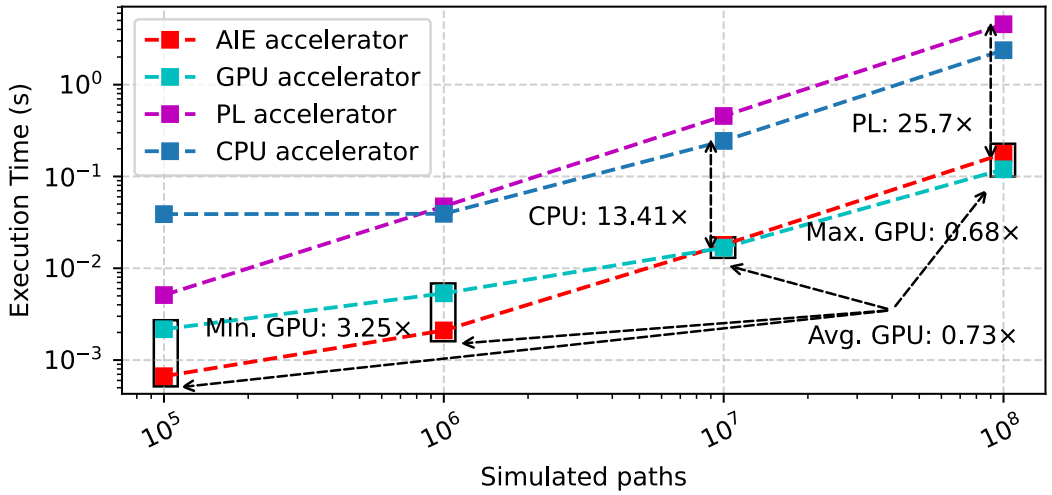



Fig. : Execution time over multiple paths.

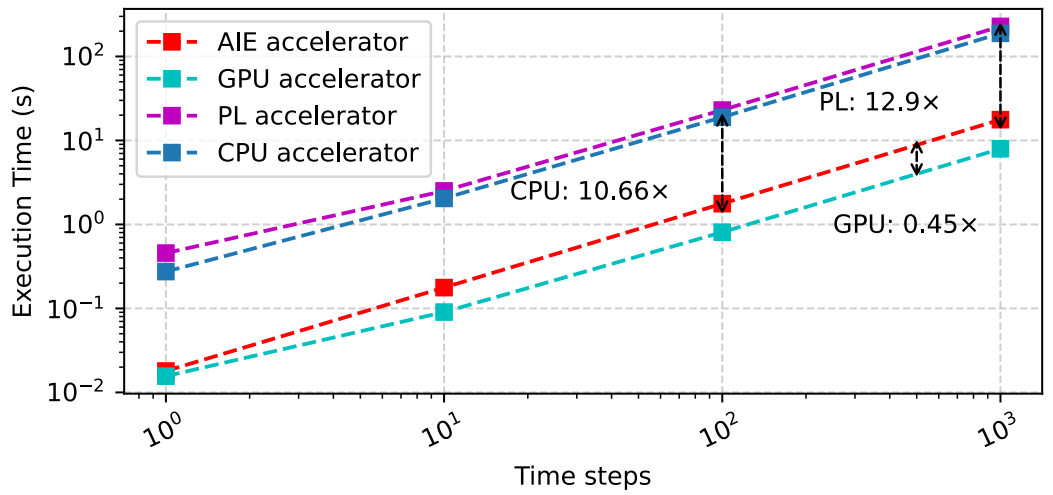



Fig. : Execution time over multiple time steps.

12. 9—25. 7x faster than FPGA, 10. 66—13. 41x faster multi-threaded CPU,
0. 45—0. 73x as fast as GPU but 1. 82x more energy efficient.

# Performance projection

For a comparable-sized device, what would performance look like?

$$\Rightarrow \quad [maximum\ speedup]_{ij} = speedup_{ij} \times \frac{para\_cu(design_i)}{para\_cu(design_j)}$$

$para\_cu(AIE) = 133$

- $para\_cu(CPU) = 128 \Rightarrow maximum\ speedup = 10.26 - 12.9\text{x}$
- $para\_cu(FPGA) = 12 \Rightarrow maximum\ speedup = 1.16 - 2.32\text{x}$

$para\_cu(AIE) = 3192$ $\quad para\_cu(GPU) = 10752 \Rightarrow maximum\ speedup = 1.52 - 2.29\text{x}$

# Conclusion

- We leverage the network of high–performance stream interconnects and the SIMD capabilities of the AMD AI Engines to compose a dataflow of overlay for Monte Carlo multi–asset option pricing.
- We propose a design that efficiently makes use of the heterogeneous components (AIEs, FPGA) to run the accelerator at high performance.
- We effectively outperform FPGA and CPU in speedup and GPU in energy efficiency.

# A Dataflow Overlay for Monte Carlo Multi-Asset Option Pricing on AMD Versal AI Engines


Github Repository

Reach out via email

Connect on LinkedIn