

# Accelerator Design at High Level of Programming Abstraction

Prashant Ravi, Abhishek Kumar Jain, Suhaib A. Fahmy, Douglas L. Maskell  
prashant014@ntu.edu.sg, abhishek013@ntu.edu.sg, s.fahmy@warwick.ac.uk, asdouglas@ntu.edu.sg

## Background and Motivation

Accelerator Design Productivity Issues:

- Difficulty of accelerator design at low level
- Complex Host-Accelerator interfacing
- Lack of software like abstractions
- Long compilation times (Place and route)

One solution: High Level Synthesis (HLS)

- Accelerator design in a high level language
- Automated host-accelerator interfacing
- RTL generation from C/C++/OpenCL
- Benefits of software like abstractions
- SDSoC, **AOCL**, SDAccel
- Long compilation time still an issue

Coarse-Grained FPGA Overlays:

- Fast compilation and development cycles
- Easy to use even by novice programmers
- Access of FPGA for software developers
- **MXP**, GRVI-Phalanx, DeCO, DySER

## Contributions

- Analysis of Overlay architectures like Vectorblox MXP and Altera OpenCL SDK as alternatives to pure RTL design flow. Highlighting the ease of use and fast learning curves of these methods.
- Benchmarking and comparison of timing performance as well as operations per cycle for the Vectorblox MXP processor and the Altera OpenCL Implementation with arm CPU implementation.

## Observations

The Vectorblox MXP soft vector processor:

- Extremely short learning curve
- Easy to use MXP C/C++ APIs
- Compilation and debugging time equal to that of traditional C/C++ debugging

The Altera OpenCL SDK:

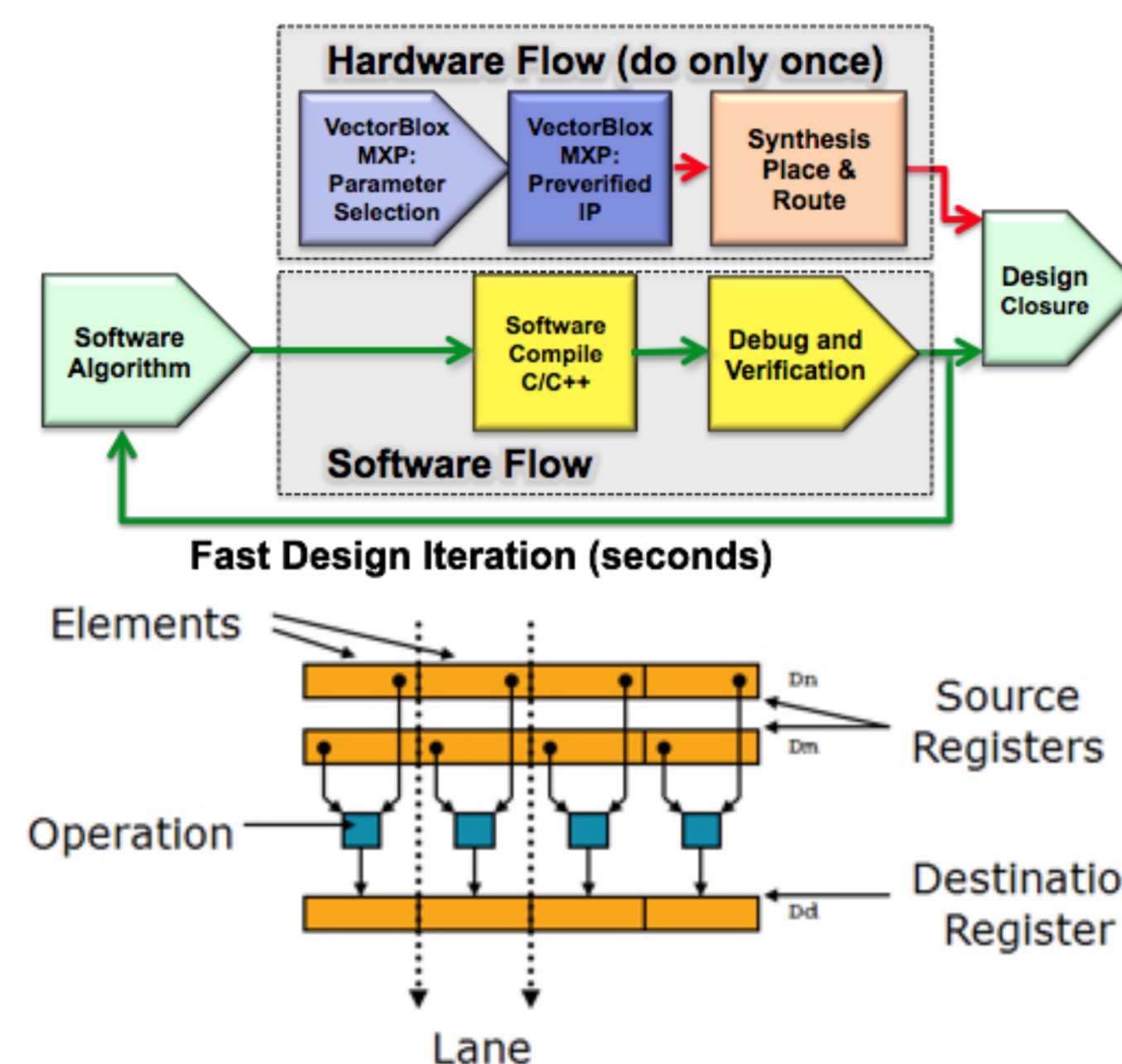
- Minimal modification of traditional OpenCL code required
- Kernel optimizations like vectorization, hardware replication for parallelism
- Host code portability across all devices

## Conclusions and Future Work

- More popularity of overlays and high level synthesis tools
- Place and route efficiency improvements in OpenCL to hardware
- Overall generated hardware efficiency improvement
- More awareness in the software world and amongst software developers

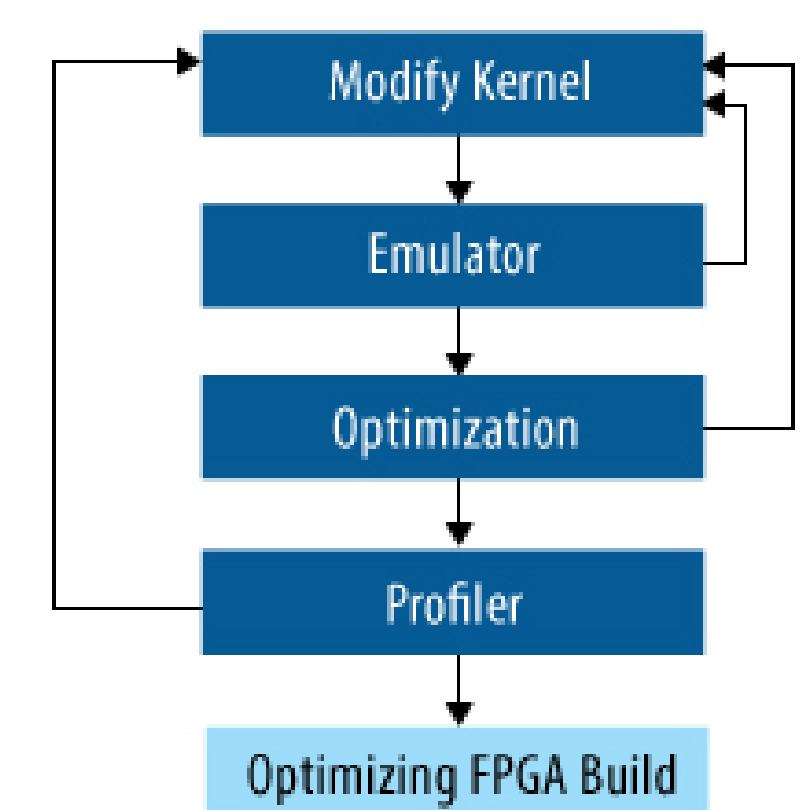
## The Vectorblox MXP Processor

- Accelerating Bare metal C/C++ applications on Zynq using MXP
- Easy to use software APIs for embedded vector processing
- 16 vector lanes each containing an ALU
- Good for integer operations, lacks floating point



## DE0-NANO-SoC with OpenCL

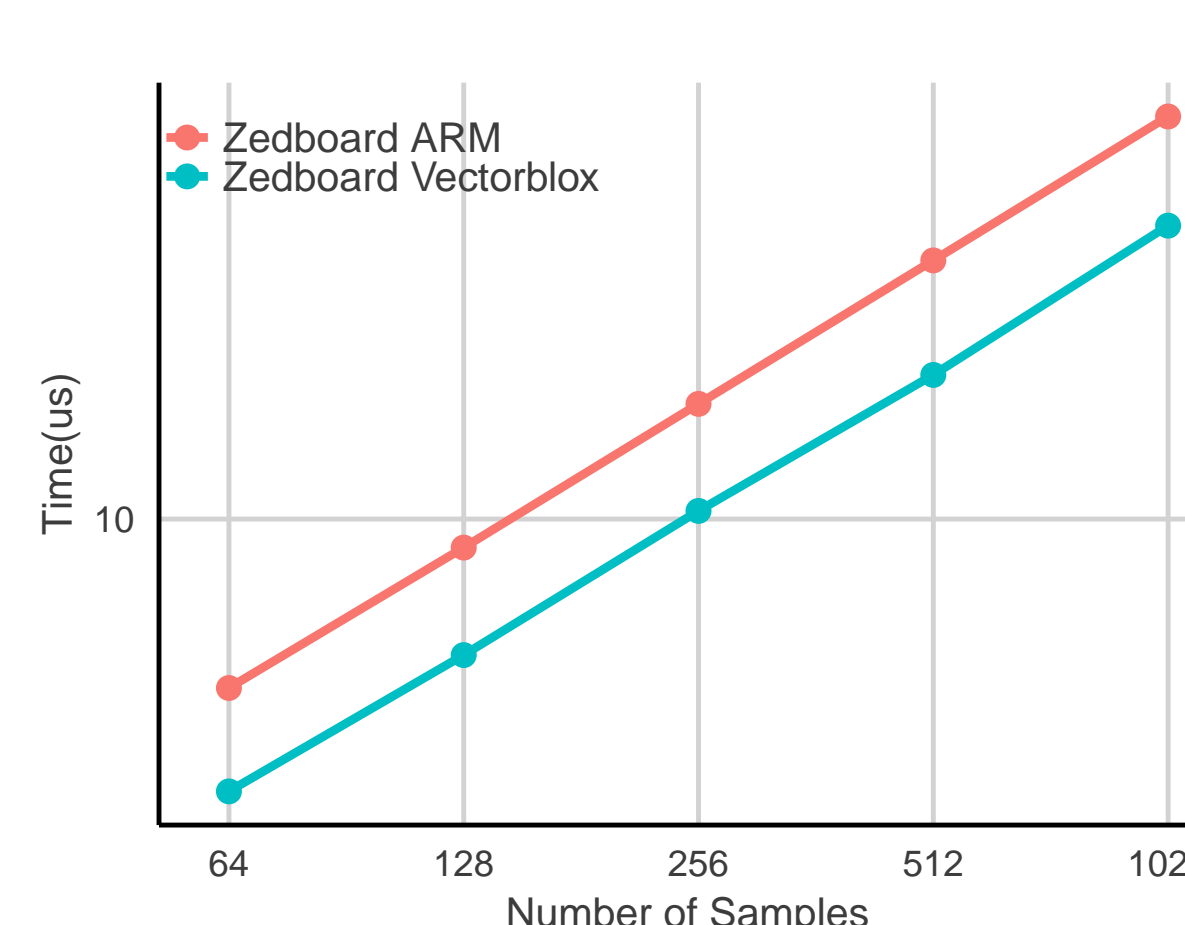
- Creation of a semi-customized pipelined datapath using OpenCL Kernels.
- Full control of hardware replication at the kernel level with loop unrolling.
- Functional C emulator for functional verification without generating hardware.
- Special AOCL optimizations for floating point to reduce area.
- Dedicated AOCL memory channels for data transfer to and from accelerator.



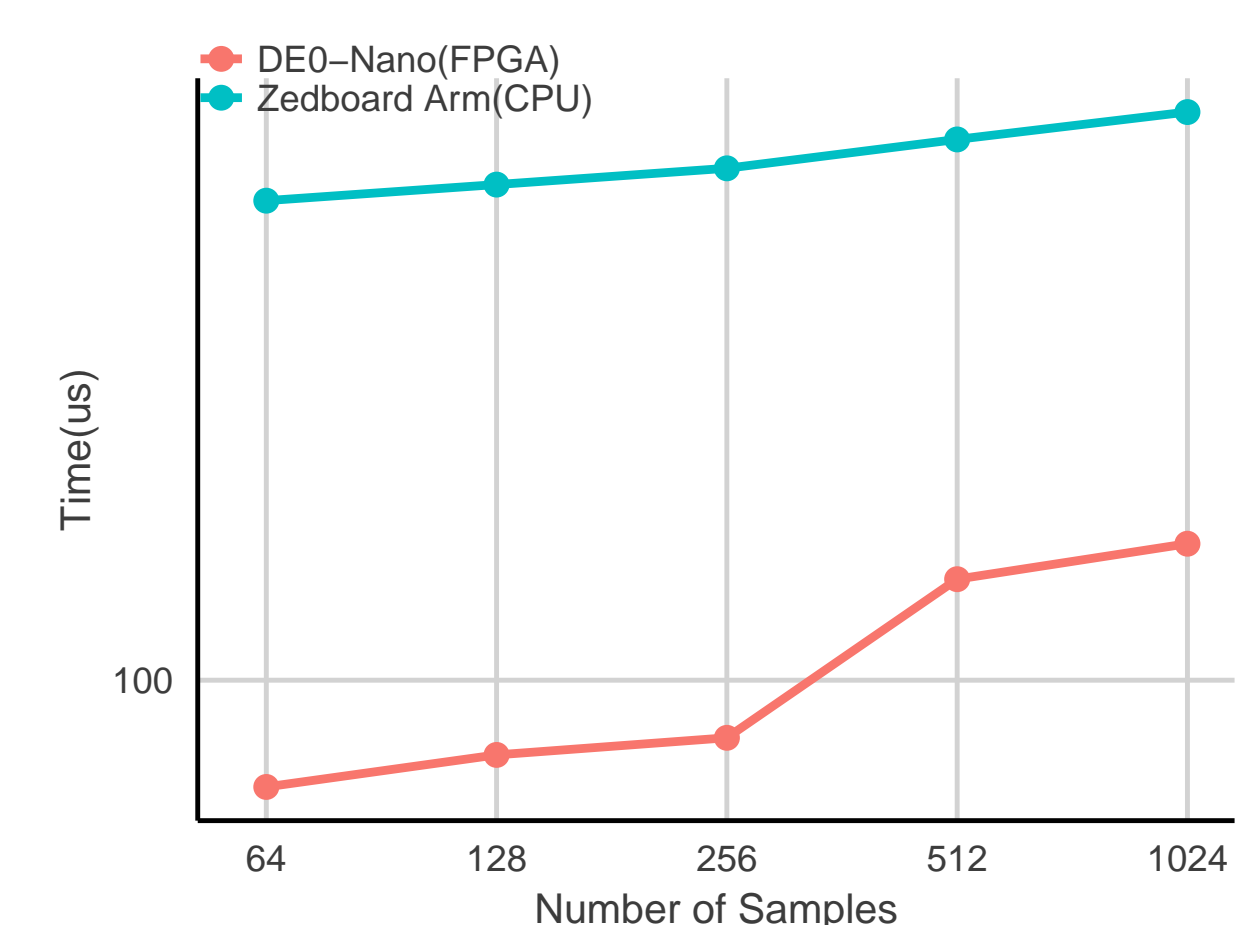
## Quick Easy Mapping of Compute Kernels



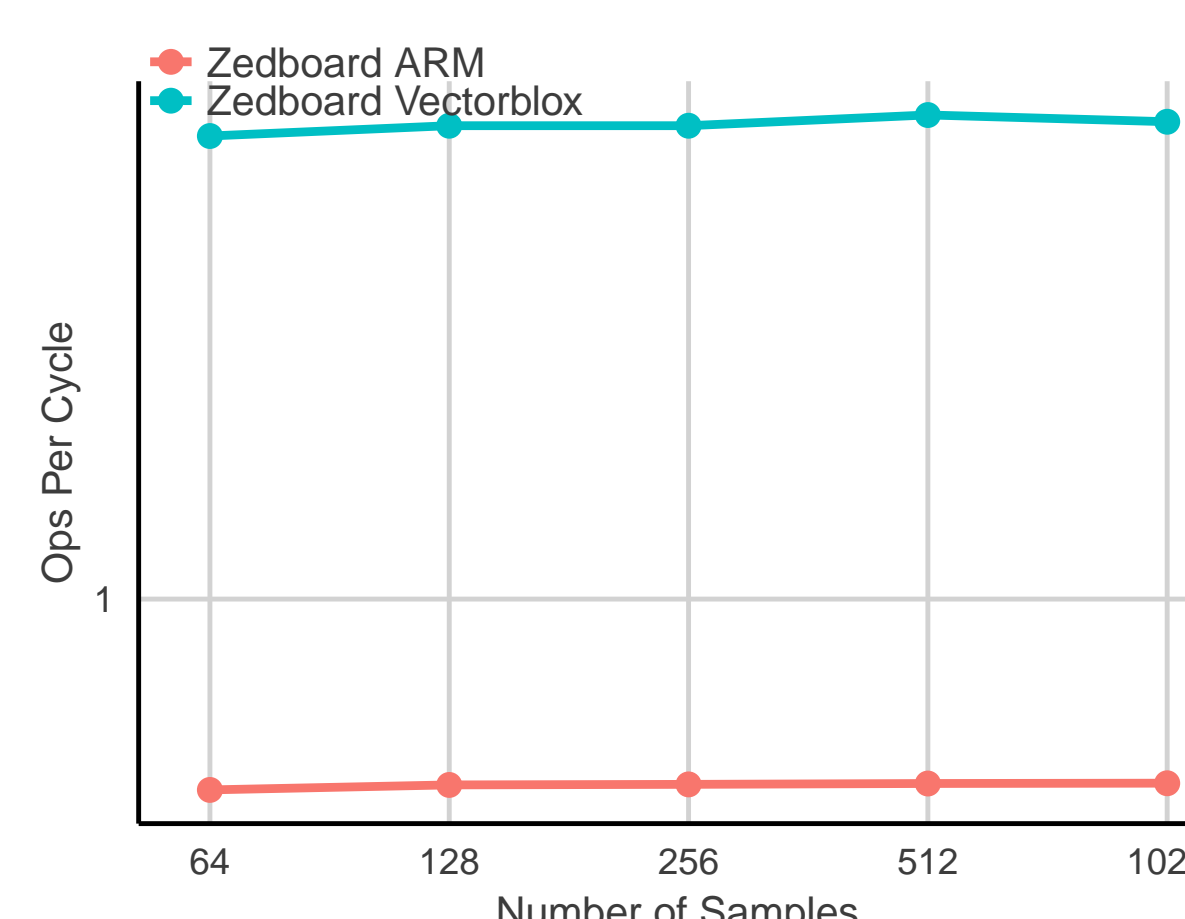
## Experimental Evaluation



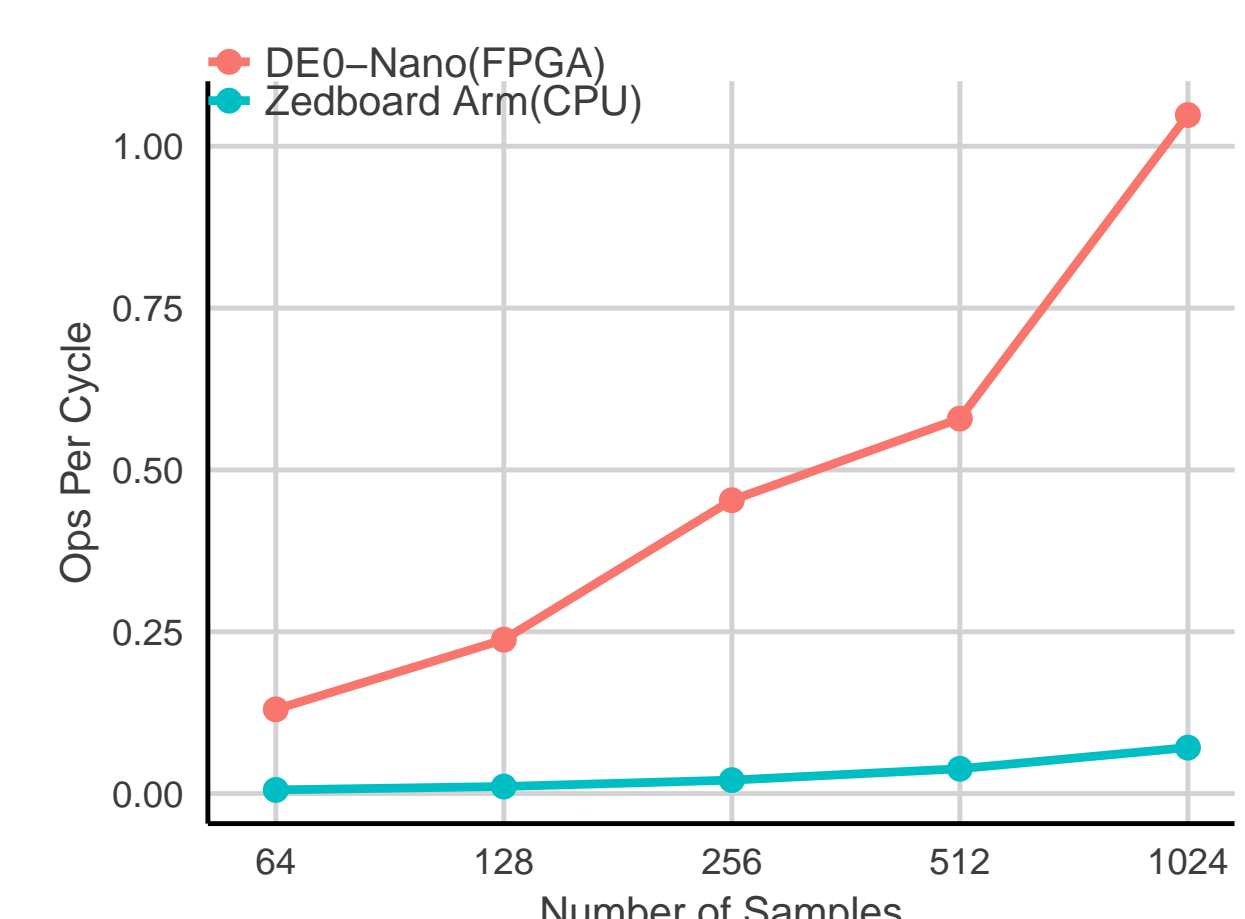
Vectorblox C API Implementation Timing Comparison with naive C Implementation on ARM CPU.



OpenCL Implementation Timing Comparison between ARM CPU and AOCL FPGA Implementation.



Vectorblox C API Implementation Operations Per Cycle Comparison with naive C Implementation on ARM CPU.



OpenCL Implementation Operations Per Cycle Comparison between ARM CPU and AOCL FPGA Implementation.

- The MXP gave a **1.6x Operations Per Cycle** increase over ARM and a **1.5x Speedup**.