A RISC-V Vector Accelerator for Machine Learning Inference

Imad Al Assir, Mohamad El Iskandarani, Hadi Rayan Al-Sandid, Mazen A. R. Saghir

American University of Beirut, Lebanon

Why Arrow?

- Proliferation of data-parallel machine learning applications.
- Diminishing performance returns from ILP and TLP.
- Vector architectures and ISA extensions ideal for workloads with high levels of DLP.

Vector Processing

Example : Vector Addition

C Code

```
for (i=0; i <64; i++)
    C[i] = A[i] + B[i];
```

RISC-V (Scalar)

```
        li x4,64
loop:   lb x10,0(x1)
        lb x20,0(x2)
        add x30,x10,x20
        sb x30,0(x3)
        addi x1,x1,1
        addi x2,x2,1
        subi x4,x4,1
        bnez x4,loop
```

Execution time =
1+63*10+9+4 = 644 cycles

RISC-V (Scalar + Vector)

```
li x4,64
vsetvli t0, x4, e8, m1
vle8.v v1,(x1)
vle8.v v2,(x2)
vadd.vv v3,v1,v2
vse8.v v3,(x3)
```

Execution time =
1+1+64+64+64+64 + 4 = 262 cycles

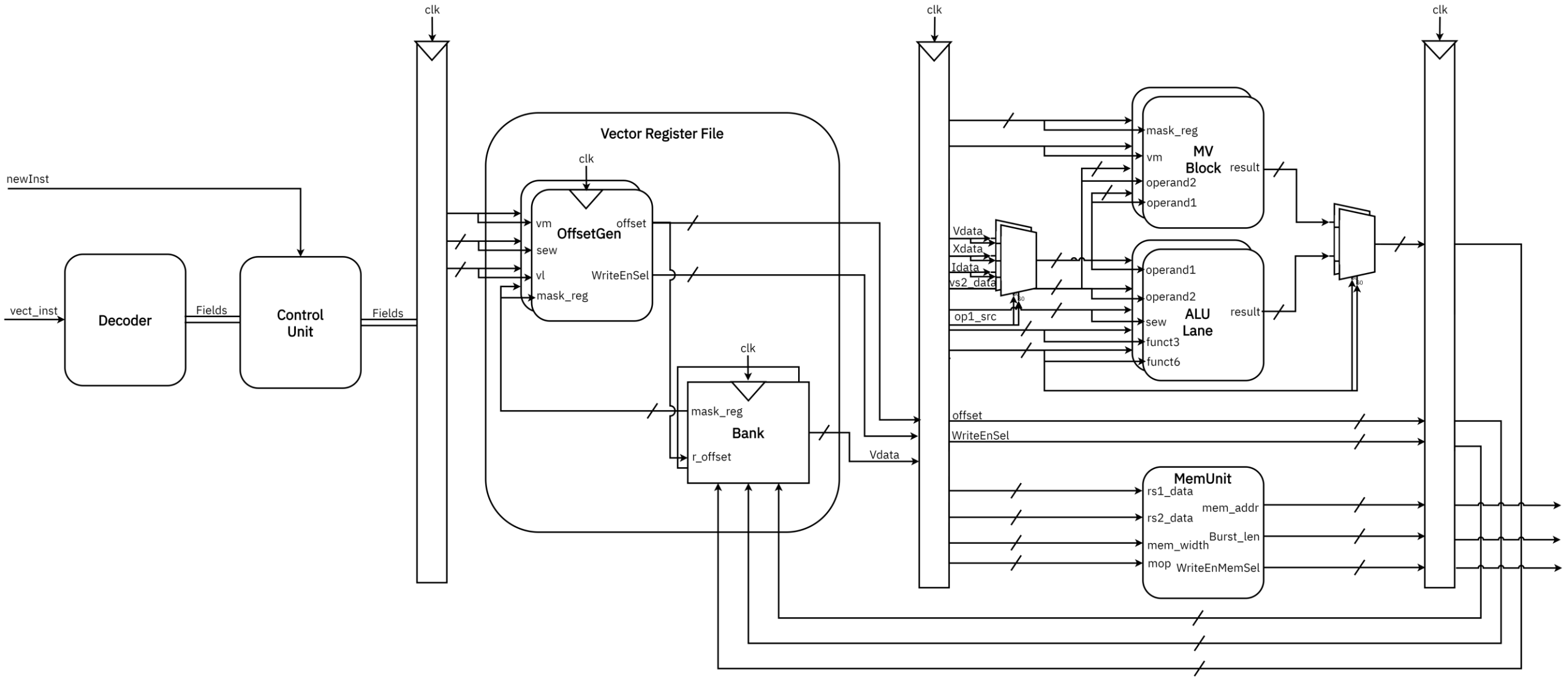
→ Speedup = 2.46x (without chaining)

What is Arrow?

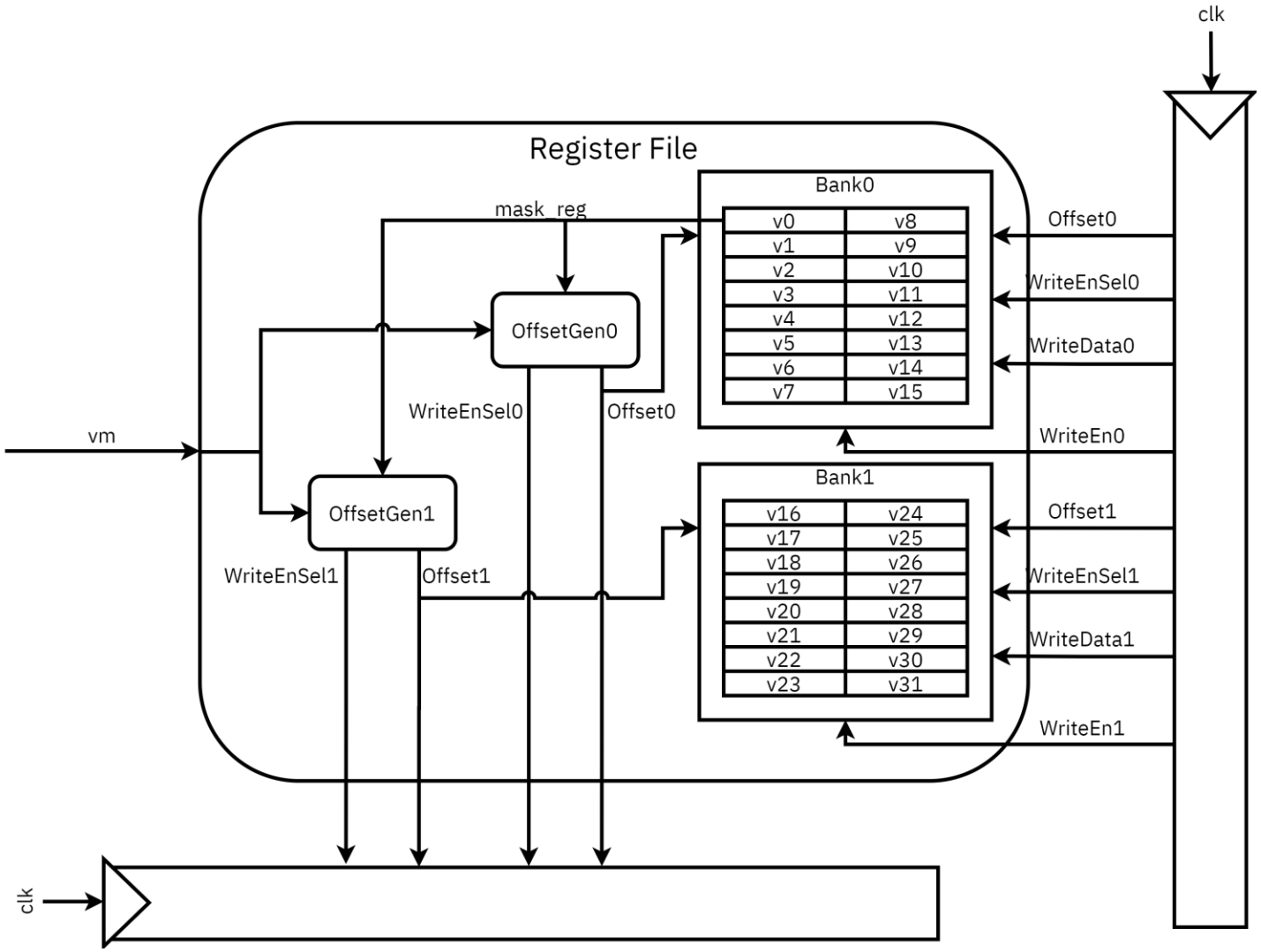
A RISC-V Vector Accelerator for Machine Learning Inference

- Implements integer subset of RISC-V Vector Specs 0.9.
- Implemented on a Xilinx NEXYS Video FPGA board (XC7A200T-1SBG484C).
- Designed to be customizable using VHDL.

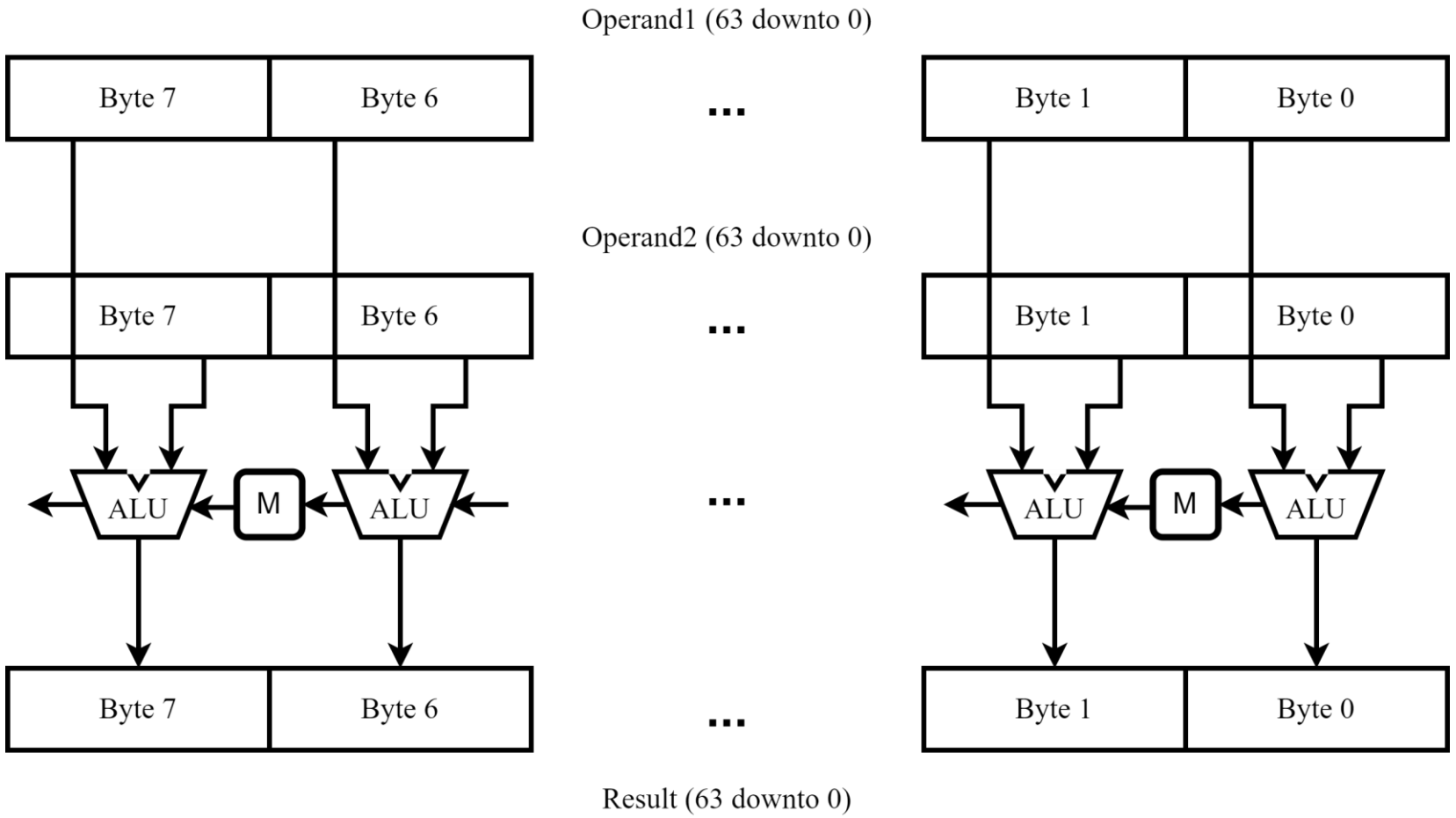
Arrow's Architecture



Vector Register File Design



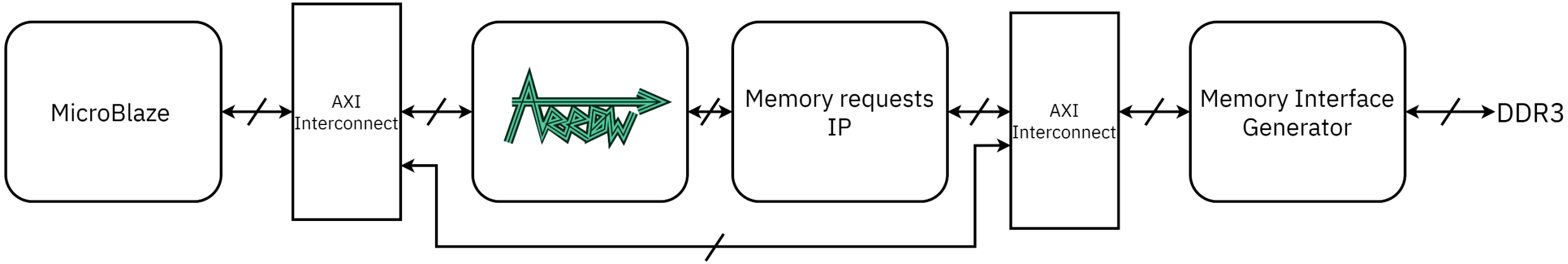
SIMD ALU


Operand1 (63 downto 0)
Byte 7
Byte 6
...
Byte 1
Byte 0
Operand2 (63 downto 0)
Byte 7
Byte 6
...
Byte 1
Byte 0
ALU
M
ALU
...
ALU
M
ALU
Byte 7
Byte 6
...
Byte 1
Byte 0
Result (63 downto 0)

FPGA Implementation

- Implemented on a Xilinx XC7A200T-1SBG484C FPGA used in the Nexys Video board.
- Using the Xilinx Vivado 2019.1 Design Suite.
- Arrow datapath packaged as an AXI IP.
- Xilinx MicroBlaze v11.0 as scalar processor.
- Generic and bus-independent.

FPGA Implementation



FPGA Synthesis Results

- Area Overhead:

	MicroBlaze		MicroBlaze + Arrow	
Resource	Utilization	Utilization %	Utilization	Utilization %
LUT	2241	1.67	2715	2.03
FF	1495	0.56	2268	0.85

FPGA Synthesis Results

- Power Consumption:

System	Power Consumption
MicroBlaze	0.270W
MicroBlaze + Arrow	0.297W

- Clock Frequency: **100** MHz

Code Development Tools

- **Requirement** : Toolchain to cross-compile C/C++ code to RISC-V binaries.
- **Solution** : LLVM/Clang fork offered by the EPI project. Offers support for RISC-V vector instructions v0.9, which can be called using inline assembly or intrinsic functions.

- **Requirement** : Testing functional validity of our cross-compiled code / RISC-V binaries, containing vector instructions.
- **Solution** : SPIKE RISC-V ISA simulator for functional validation of our benchmarks. Offers support for the RISC-V "V" extension v0.9

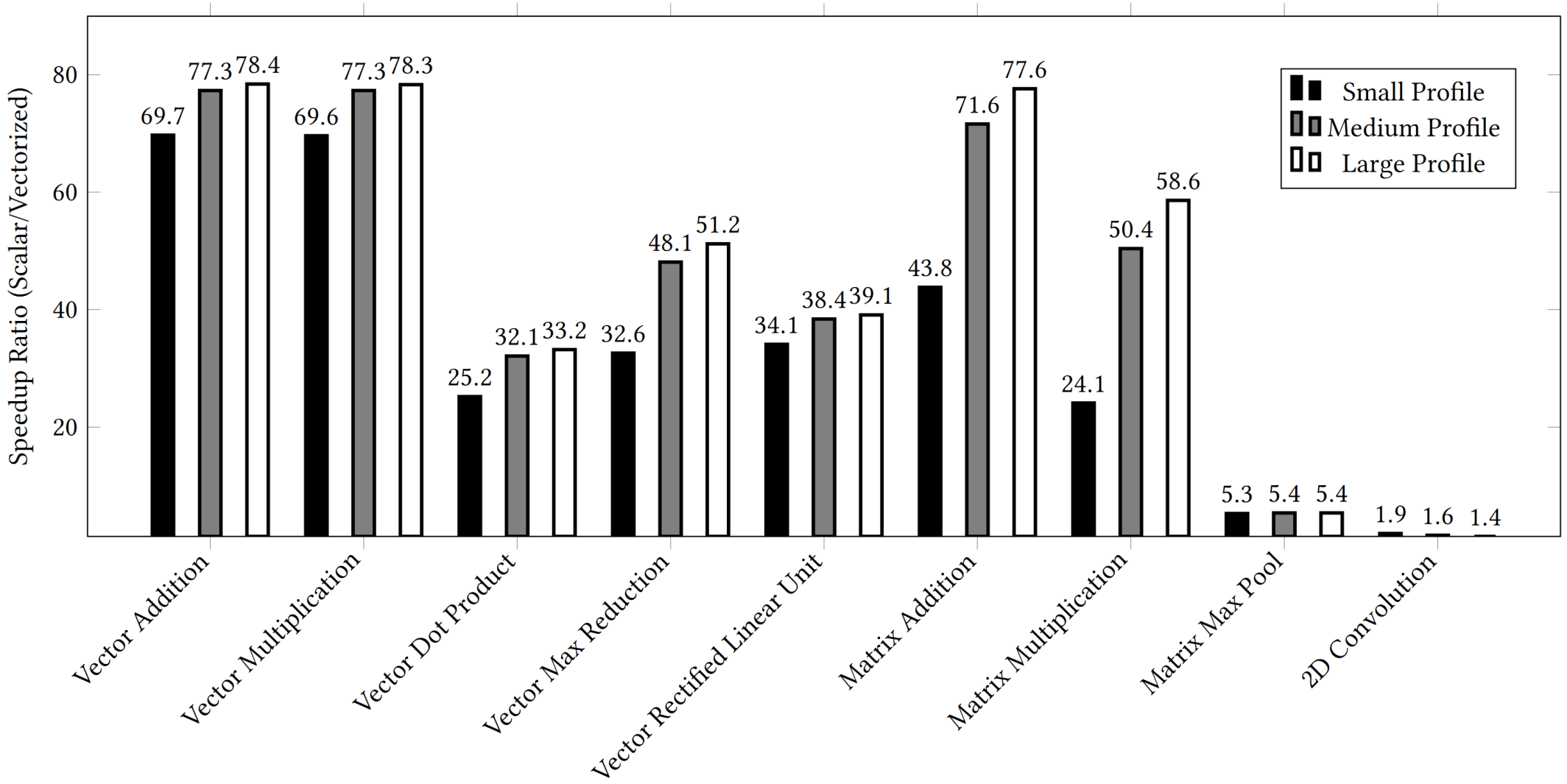
Benchmark Setup

- Basic Vector and Matrix operations from the University of Southampton GitHub:
  - Vector Addition, Multiplication, Dot Product, Max Reduction, ReLu
  - Matrix Addition, Multiplication, Max Pool
  - 2D Convolution
- Developed a cycle count model for scalar and vector operations. Scalar cycle counts within 8% of Spike cycle counts.

Performance Model

- **Execution Time** computed by multiplying cycle count by Arrow clock cycle time.
- **Energy Consumption** computed by multiplying power consumption (from synthesis report) by execution time.

Benchmark Results



Benchmark Results

- Energy reduced by:
  - 96% to 99% for vector benchmarks.
  - 80% to 99% for matrix operations.
  - 20% to 43% for 2D convolution.

Conclusion

Due to its high performance and low energy consumption,

Arrow is well suited for edge machine learning applications.

Future Work

- Integrating Arrow as a tightly coupled accelerator in a RISC-V datapath (e.g. WD SweRV EH1).
- Developing a gem5 model of the accelerator for more thorough performance study using MLPerf and TinyMLPerf benchmark suites.
- Supporting ML data types (e.g. bfloat16 and posits) and ML-related instructions.

Thank you for listening.
Questions?