A Case for MVPs:

Mixed-Precision Vector Processors

Albert Ou Quan Nguyen Yunsup Lee Krste Asanović

Department of Electrical Engineering and Computer Sciences
University of California, Berkeley

14 June 2014

Accelerators

Specialized hardware $\longleftrightarrow$ General-purpose hardware

Efficient $\longleftrightarrow$ Flexible

Efficiency advantages in fixed-function accelerators:

- Datapath widths tailored to application
- Diminished overhead of instruction delivery
- Specialized memory systems (scratchpads)
- Data forwarding (bypassing register file)

Accelerators

Specialized hardware $\longleftrightarrow$ General-purpose hardware
Efficient $\longleftrightarrow$ Flexible

Efficiency advantages in fixed-function accelerators:

- Datapath widths tailored to application
- Diminished overhead of instruction delivery
- Specialized memory systems (scratchpads)
- Data forwarding (bypassing register file)

This particular work focuses on improving these two aspects in **programmable accelerators**.

Reduced Precision

- Less energy and higher performance with fewer bits computed/communicated
- Fixed-function hardware exploits minimal precisions in custom datapaths.

- One possibility with programmable accelerators:
  Configure the entire datapath to use one reduced precision at time.
- However, using one reduced precision globally is not always optimal.

Mixed Precision

Applications exhibit a broad range of precisions simultaneously.

- Coexistence of different widths:
  e.g., integer data versus addresses
- Explicit mixed-precision operations:
  e.g., widening fused multiply-add (FMA)
  Accumulate product of n-bit values into $2n$-bit sum

Overarching theme

How mixed-precision computation enables greater energy efficiency and performance while retaining ease of programmability

Data-Level Parallelism

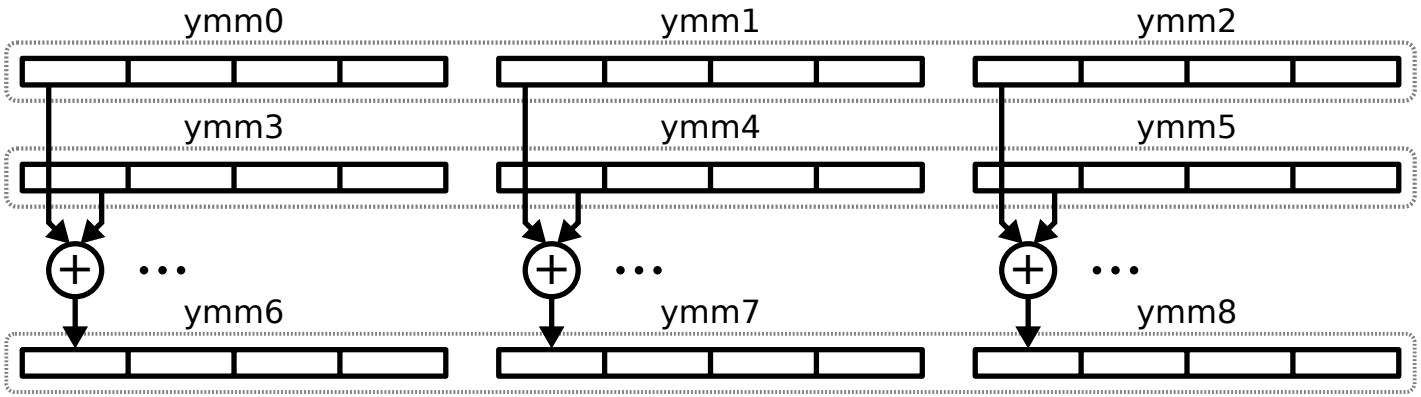
Prevalent in a variety of intensive application domains:

multimedia, signal processing, computer vision, machine learning, physics simulation

- Apply a homogenous operation to multiple data elements
- Opportunity for more computation with fewer instructions
- Potential for simplified control logic
- Amortize instruction fetch, decode, and sequencing

Two ISA approaches: SIMD extensions and vector processing

SIMD Extensions

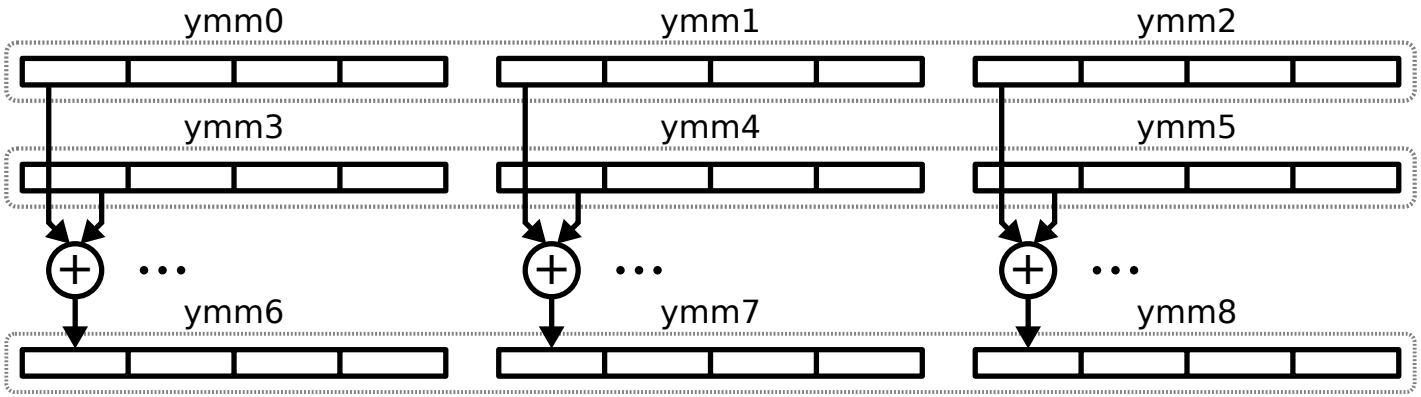


- Subword-packed operations
- Opcode-encoded fixed bit-width

```
vaddpd %ymm0, %ymm3, %ymm6
vaddpd %ymm1, %ymm4, %ymm7
vaddpd %ymm2, %ymm5, %ymm8
```

- Popular in commercial ISAs: Intel AVX, ARM NEON

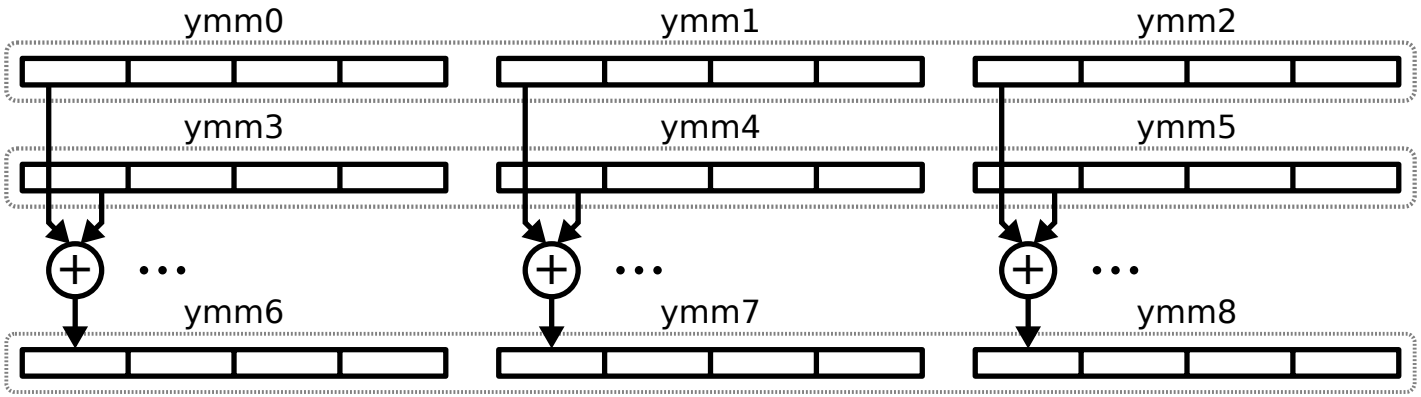
SIMD Extensions



"Vector" registers tend to be relatively short:

- Originally ad-hoc repurposing of scalar datapath/control
- Reliance on superscalar issue to saturate functional units
  $\Longrightarrow$ greater control complexity
- May require splitting dataset across multiple architectural registers
  $\Longrightarrow$ more instructions
- Counteracts efficiency gains from DLP

SIMD Extensions

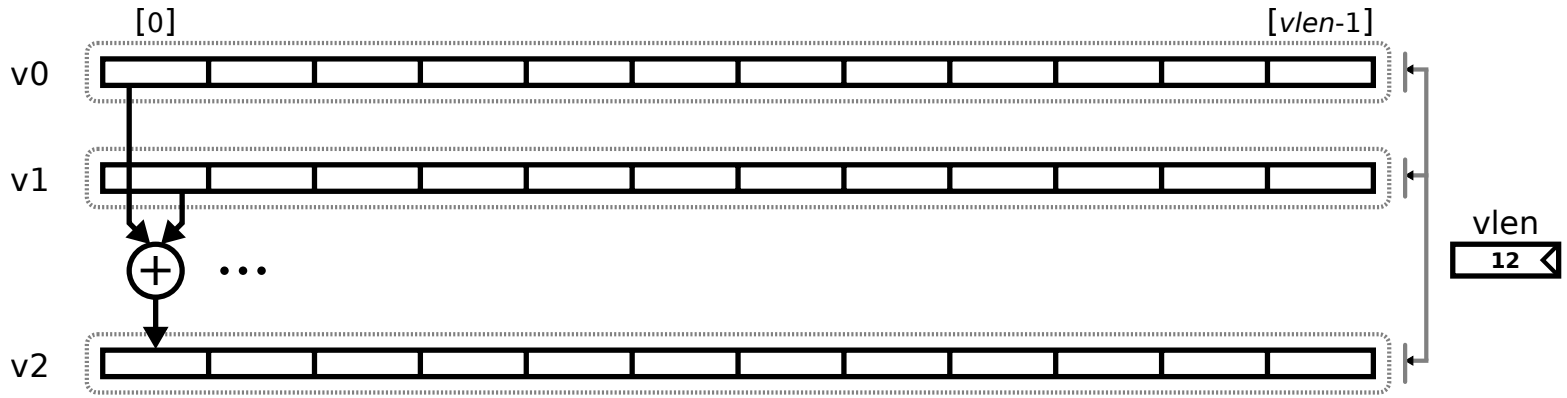


Gradual transition to wider SIMD:

64 bit MMX → 128 bit SSE / SSE2 / SSE3 / SSSE3 / SSE4 → 256 bit AVX / AVX2 → 512 bit AVX-512 → 1024 bit ?

Proliferation of redundant instructions and legacy:
e.g., addpd (`66 0F 58 /r`), vaddpd (`VEX.128`, `VEX.256`, `EVEX.512`)

Vector Processors

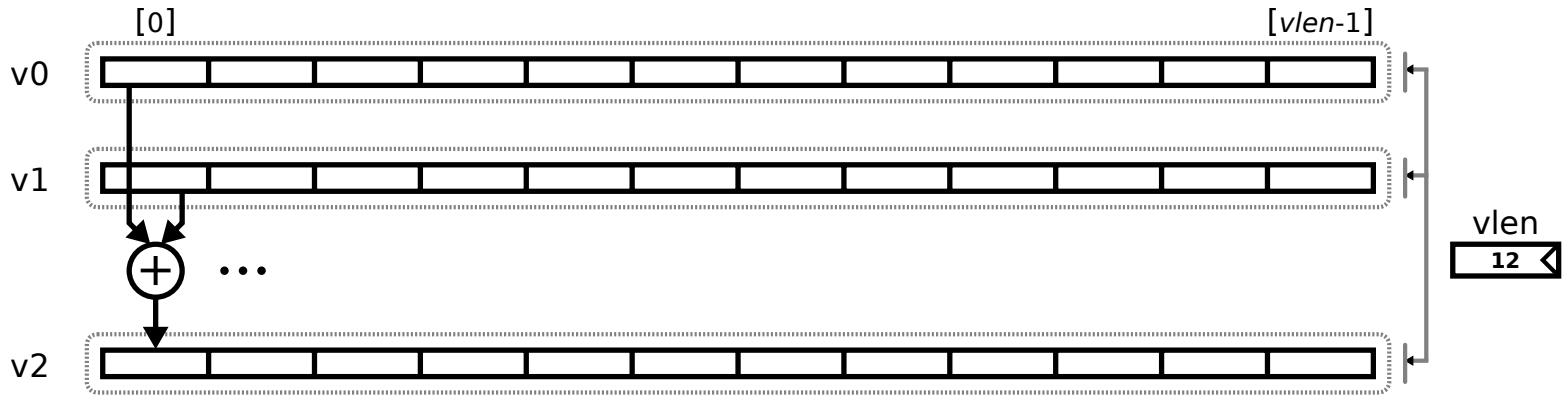


- Heritage in supercomputers: Cray, NEC SX series
  (but also well-suited to compact mobile implementations)

- Longer, genuine vectors (typically 64+ elements)

```
vsetvl vlen, 12
vadd v2, v0, v1
```

Vector Processors

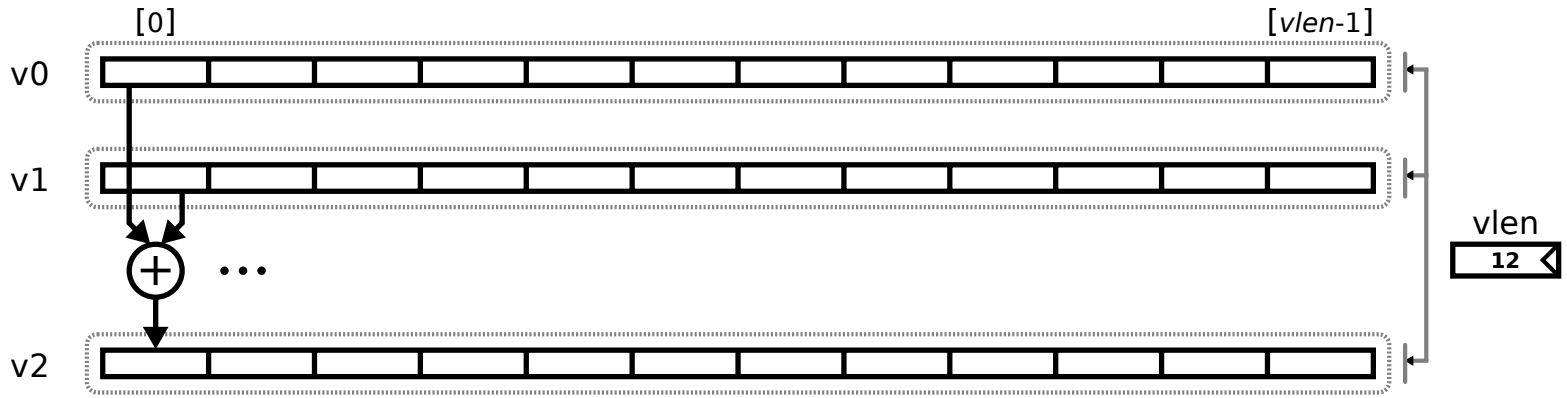


Vector length control

- Count in terms of elements instead of bits
- Hardware vector length is flexible
- Adjustable through a *vector length register*

  e.g., `vsetvl vlen, 12`

Vector Processors

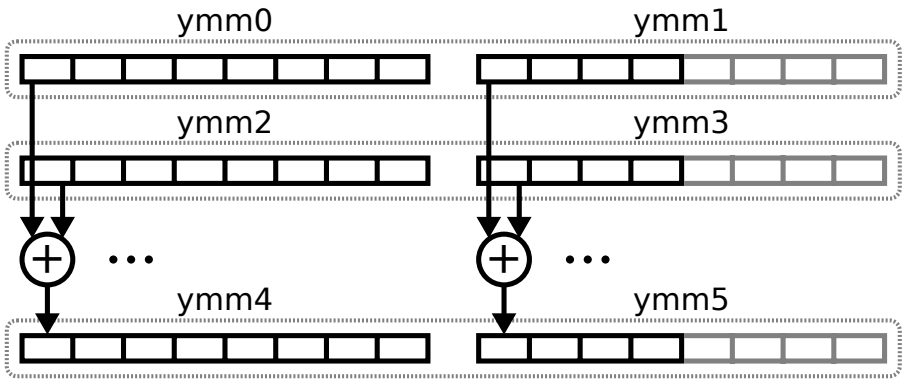


Vector length control

- Read from `vlen` register to *stripmine* arbitrary application vectors
- One instruction opcode suffices for all supported vector lengths
- Greater scalability while preserving forward/backward compatability

Reduced Precision

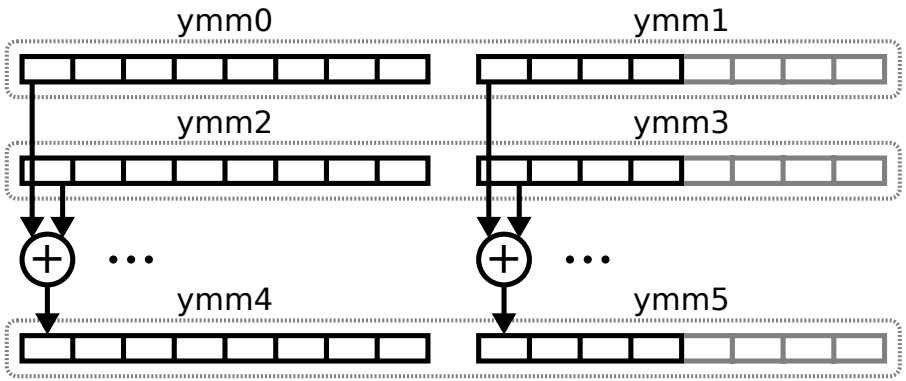
SIMD Extensions



- With fixed bit-width, registers contain a variable number of elements depending on precision
- Difficult to work with partial SIMD width (non-2^n)

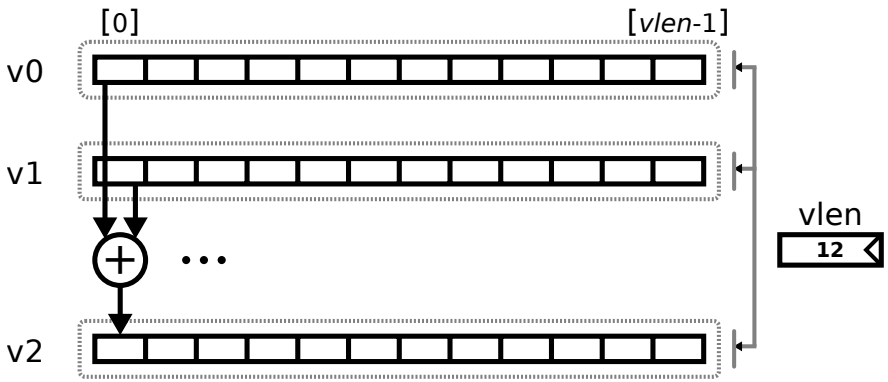
Reduced Precision

SIMD Extensions



- With fixed bit-width, registers contain a variable number of elements depending on precision
- Difficult to work with partial SIMD width (non-2^n)

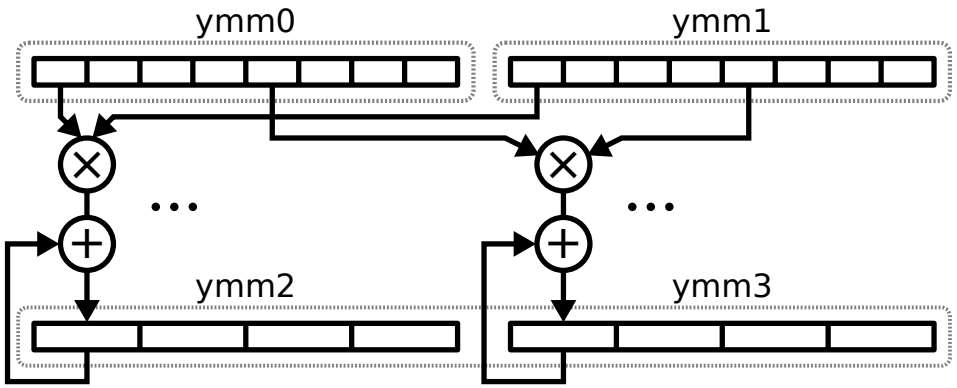
Vectors



- Hardware vectors always contain the same number of elements regardless of precision
- Avoids trailing edge case
- Programming model therefore scales more gracefully

Mixed Precision

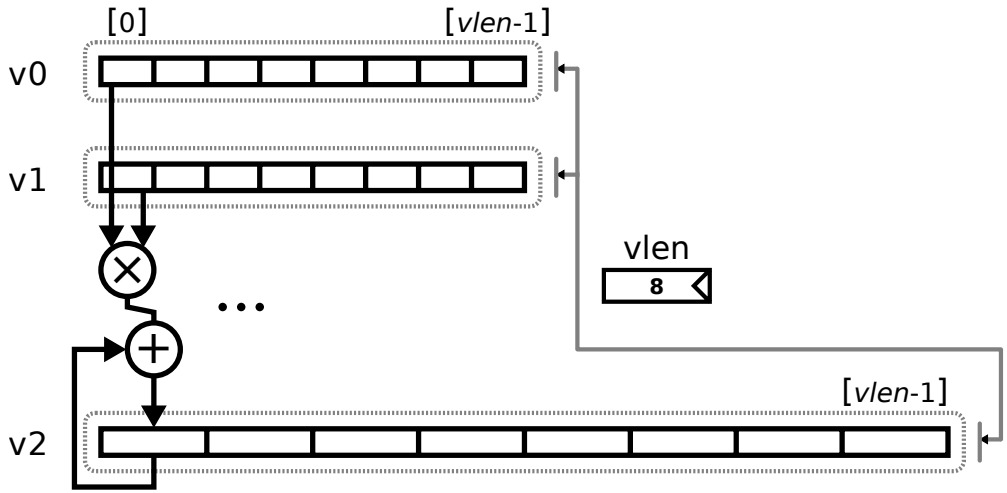
SIMD Extensions



- Separate operations for upper/lower segments of source registers (different destinations)
  - Dedicated opcode per segment
  - Alternative: extra shuffle steps
- Consequence of unequal element counts
- Exposed subword packing

Mixed Precision

Vectors



- One instruction per hardware vector regardless of specific input/output precisions

Proposal: Mixed-Precision Vector Processors

Support seamless mixed-precision computation on vector processors

Why use vectors as a foundation?

- Data-level parallelism amortizes control overhead.
- Regular data access patterns assist optimizations.

Why select genuine vector processing instead of SIMD extensions?

- Better ISA abstraction allows flexibility in machine configuration while maintaining software compatability.

Relevance to Mobile Platforms

- Data-level parallelism abundant in mobile applications:
  graphics rendering, computer vision, image processing, signal processing
- Mixed-precision concepts also applicable to other architectures,
  e.g., GPUs

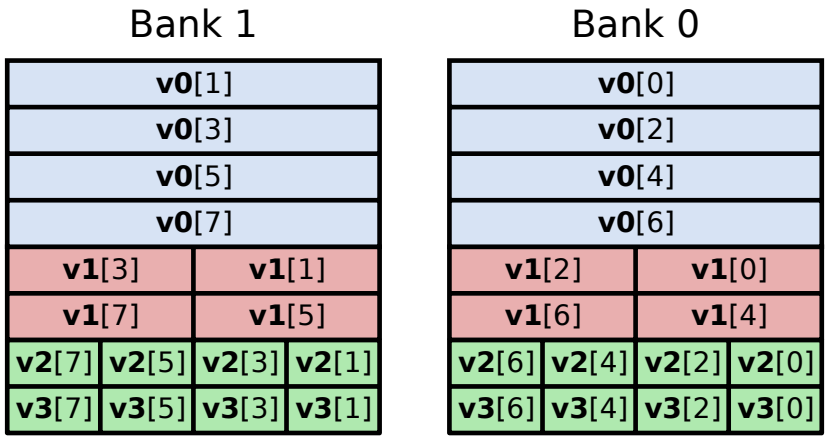
Opportunities for Microarchitectural Optimization

- Programs might not use all available vector registers.
- Storing a reduced-precision value in a wider register leaves spare bits.
- Each memory request can transfer multiple elements (memory coalescing).

MVP Configurability

We propose dynamic reconfiguration of the vector register file:

- Software specifies number of architectural registers per supported width.
  `vsetcfg 2, 1, 1, 2` (integer, double, single, half)
- Hardware subdivides a physical register into multiple narrower architectural registers as needed.
- Hardware automatically extends the maximum hardware vector length to fill the capacity of the register file.



Benefits

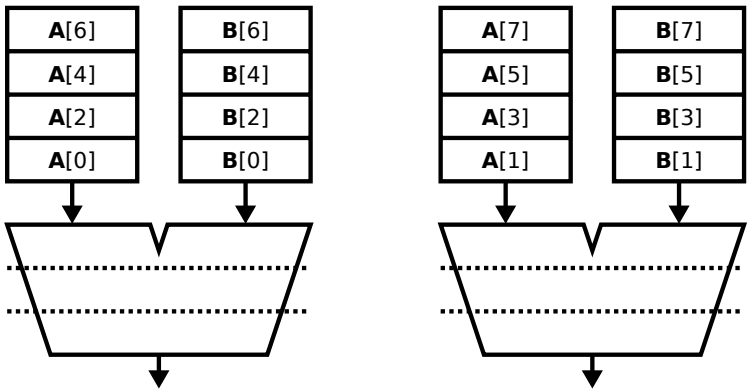
Efficiency

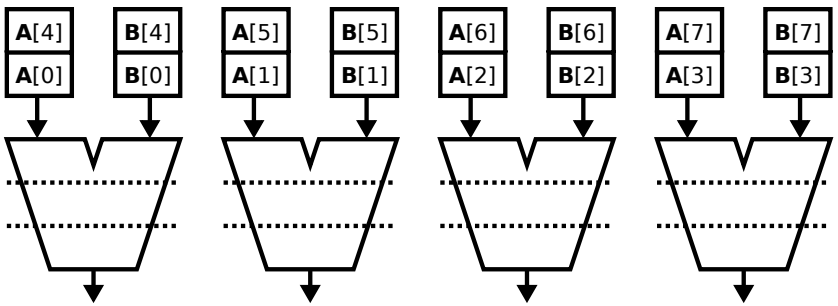
- Exchange unused architectural registers for longer hardware vectors.

Portability

- From the software perspective, architectural register widths are changing instead of packed elements being exposed.
- Subword packing occurs *implicitly* from register configuration, invisible to software.
- Implementations may ignore configuration hints and still execute the same code, albeit at reduced efficiency.

Partitioning



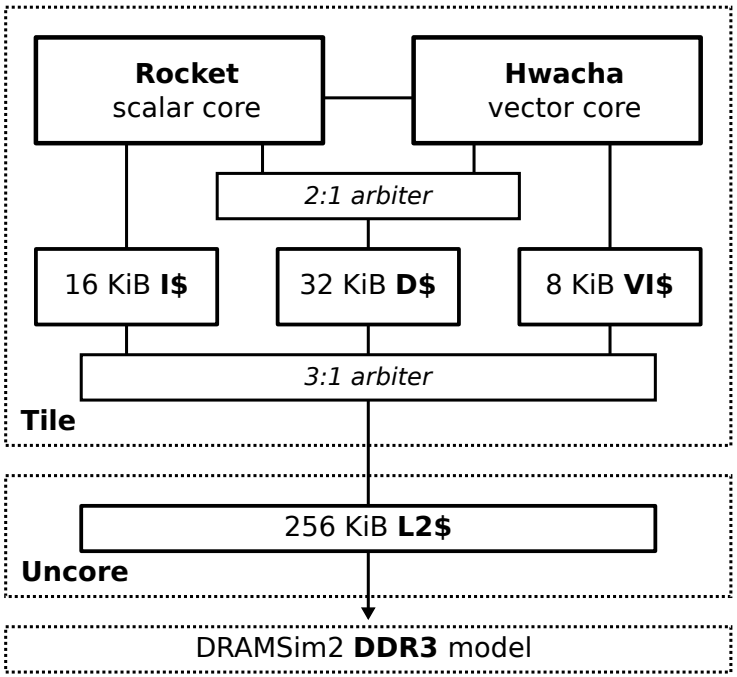


Spatial parallelism through register packing and datapath partitioning

- Improve utilization of existing resources for operand communication
  - Register file read/write ports
  - Interconnection fabric
  - Memory bandwidth
- Introduce functional units without widening overall datapath

Straightforward in a vector architecture due to intrinsic data independence

Implementation



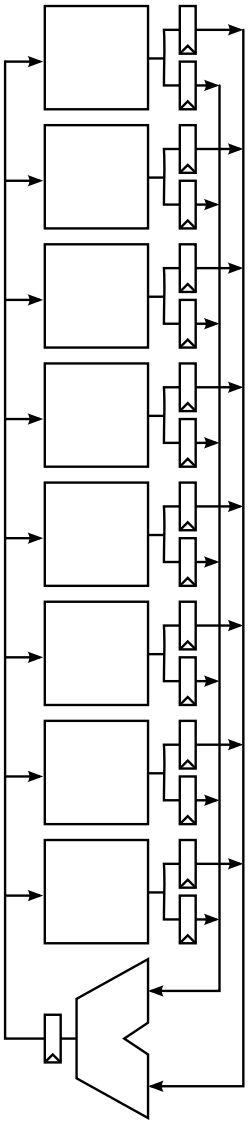
Hwacha: a decoupled vector-fetch data-parallel accelerator

Rocket: a six-stage in-order scalar control processor

Unfortunately, too time-consuming to explain the full microarchitecture – please see paper for details.

Instead, we will simply focus on the vector lane.

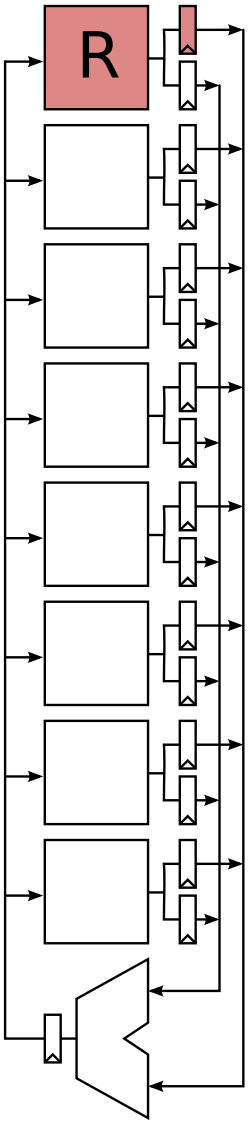
Vector Lane Organization



Bank execution

- Common misconception that vector bandwidth requirements necessitate bloated multi-ported register files
- Compact vector register file of 1R1W SRAM banks: Sustains n operands/cycle after n-cycle initial latency
- Systolic cascade of µops
- Displacement of write µop coincides exactly with functional unit latency

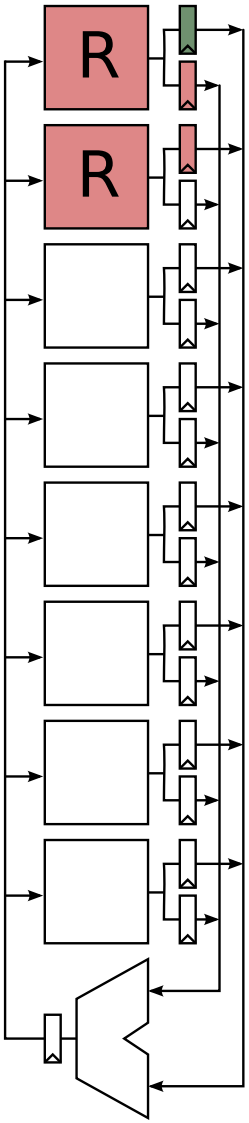
Vector Lane Organization



Bank execution

- Common misconception that vector bandwidth requirements necessitate bloated multi-ported register files
- Compact vector register file of 1R1W SRAM banks: Sustains n operands/cycle after n-cycle initial latency
- Systolic cascade of µops
- Displacement of write µop coincides exactly with functional unit latency

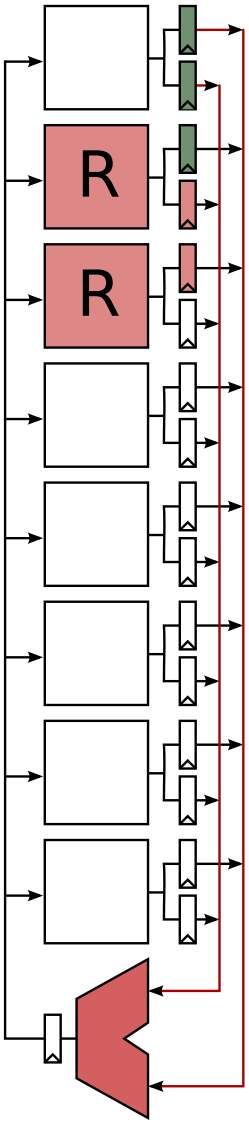
Vector Lane Organization



Bank execution

- Common misconception that vector bandwidth requirements necessitate bloated multi-ported register files
- Compact vector register file of 1R1W SRAM banks: Sustains n operands/cycle after n-cycle initial latency
- Systolic cascade of µops
- Displacement of write µop coincides exactly with functional unit latency

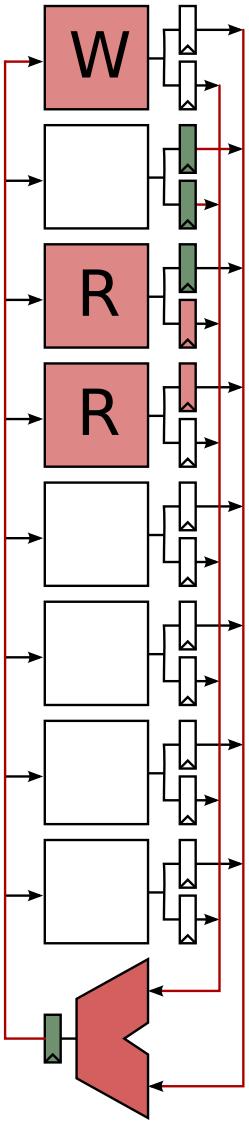
Vector Lane Organization



Bank execution

- Common misconception that vector bandwidth requirements necessitate bloated multi-ported register files
- Compact vector register file of 1R1W SRAM banks: Sustains n operands/cycle after n-cycle initial latency
- Systolic cascade of µops
- Displacement of write µop coincides exactly with functional unit latency

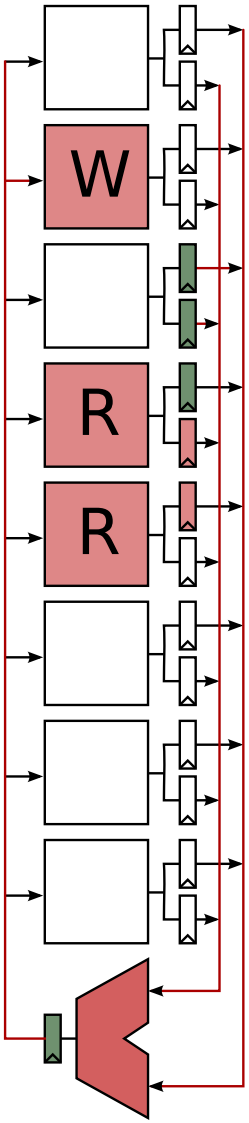
Vector Lane Organization



Bank execution

- Common misconception that vector bandwidth requirements necessitate bloated multi-ported register files
- Compact vector register file of 1R1W SRAM banks: Sustains n operands/cycle after n-cycle initial latency
- Systolic cascade of µops
- Displacement of write µop coincides exactly with functional unit latency

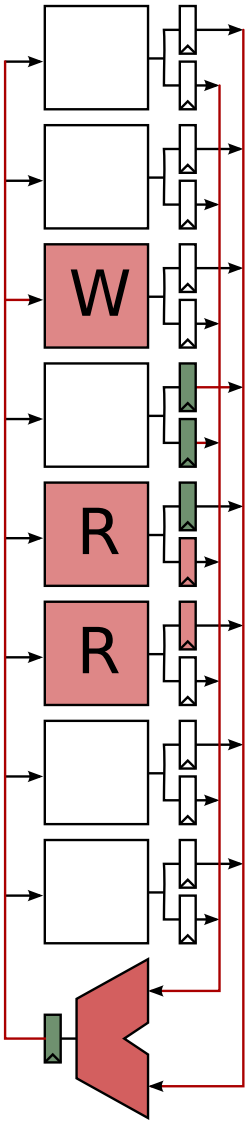
Vector Lane Organization



Bank execution

- Common misconception that vector bandwidth requirements necessitate bloated multi-ported register files
- Compact vector register file of 1R1W SRAM banks: Sustains n operands/cycle after n-cycle initial latency
- Systolic cascade of µops
- Displacement of write µop coincides exactly with functional unit latency

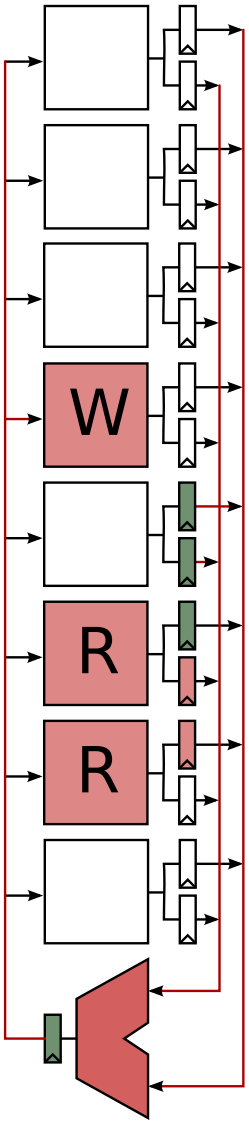
Vector Lane Organization



Bank execution

- Common misconception that vector bandwidth requirements necessitate bloated multi-ported register files
- Compact vector register file of 1R1W SRAM banks: Sustains n operands/cycle after n-cycle initial latency
- Systolic cascade of µops
- Displacement of write µop coincides exactly with functional unit latency

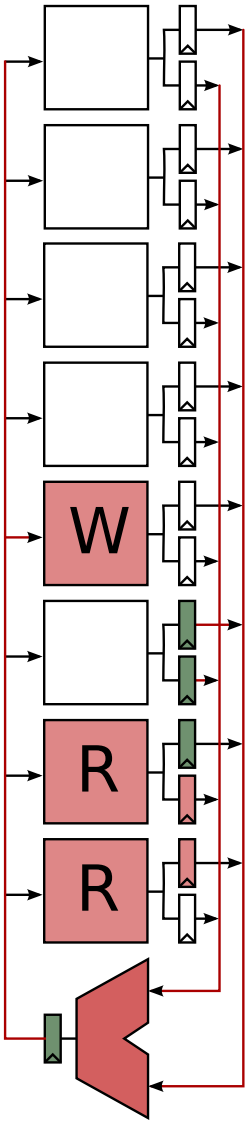
Vector Lane Organization



Bank execution

- Common misconception that vector bandwidth requirements necessitate bloated multi-ported register files
- Compact vector register file of 1R1W SRAM banks: Sustains n operands/cycle after n-cycle initial latency
- Systolic cascade of µops
- Displacement of write µop coincides exactly with functional unit latency

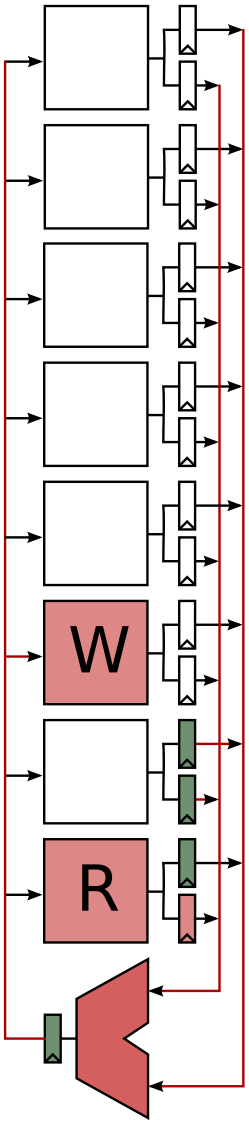
Vector Lane Organization



Bank execution

- Common misconception that vector bandwidth requirements necessitate bloated multi-ported register files
- Compact vector register file of 1R1W SRAM banks: Sustains n operands/cycle after n-cycle initial latency
- Systolic cascade of µops
- Displacement of write µop coincides exactly with functional unit latency

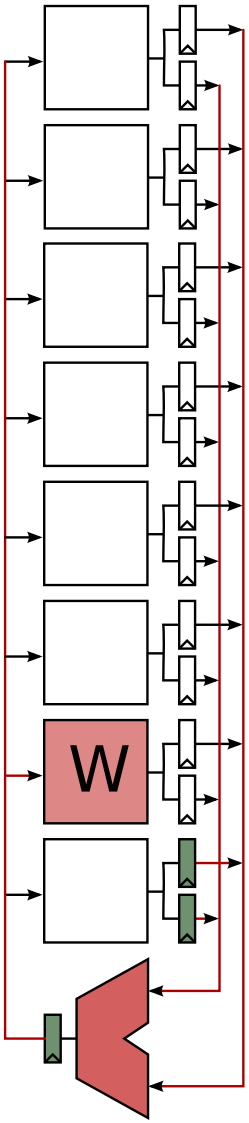
Vector Lane Organization



Bank execution

- Common misconception that vector bandwidth requirements necessitate bloated multi-ported register files
- Compact vector register file of 1R1W SRAM banks: Sustains n operands/cycle after n-cycle initial latency
- Systolic cascade of µops
- Displacement of write µop coincides exactly with functional unit latency

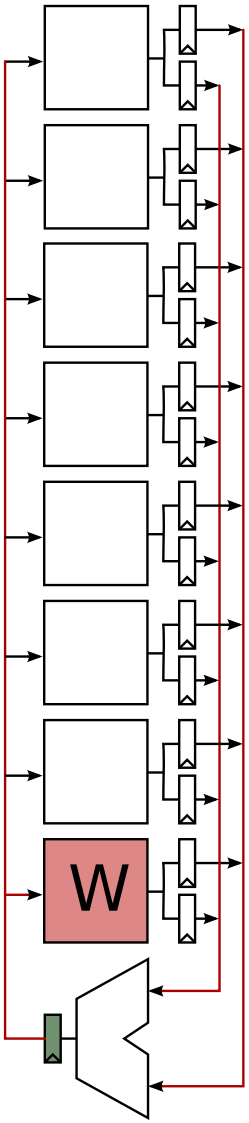
Vector Lane Organization



Bank execution

- Common misconception that vector bandwidth requirements necessitate bloated multi-ported register files
- Compact vector register file of 1R1W SRAM banks: Sustains n operands/cycle after n-cycle initial latency
- Systolic cascade of µops
- Displacement of write µop coincides exactly with functional unit latency

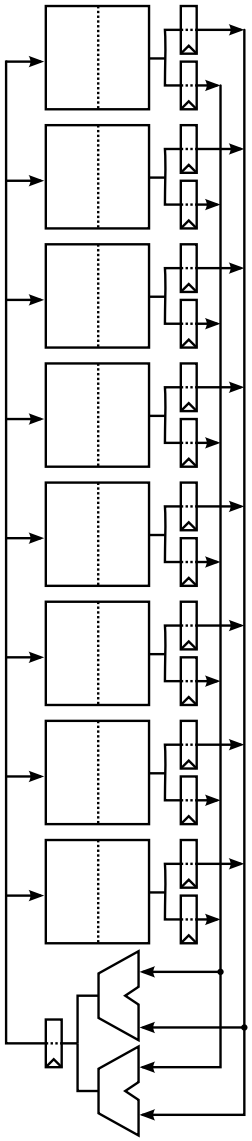
Vector Lane Organization



Bank execution

- Common misconception that vector bandwidth requirements necessitate bloated multi-ported register files
- Compact vector register file of 1R1W SRAM banks: Sustains n operands/cycle after n-cycle initial latency
- Systolic cascade of µops
- Displacement of write µop coincides exactly with functional unit latency

Vector Lane Organization



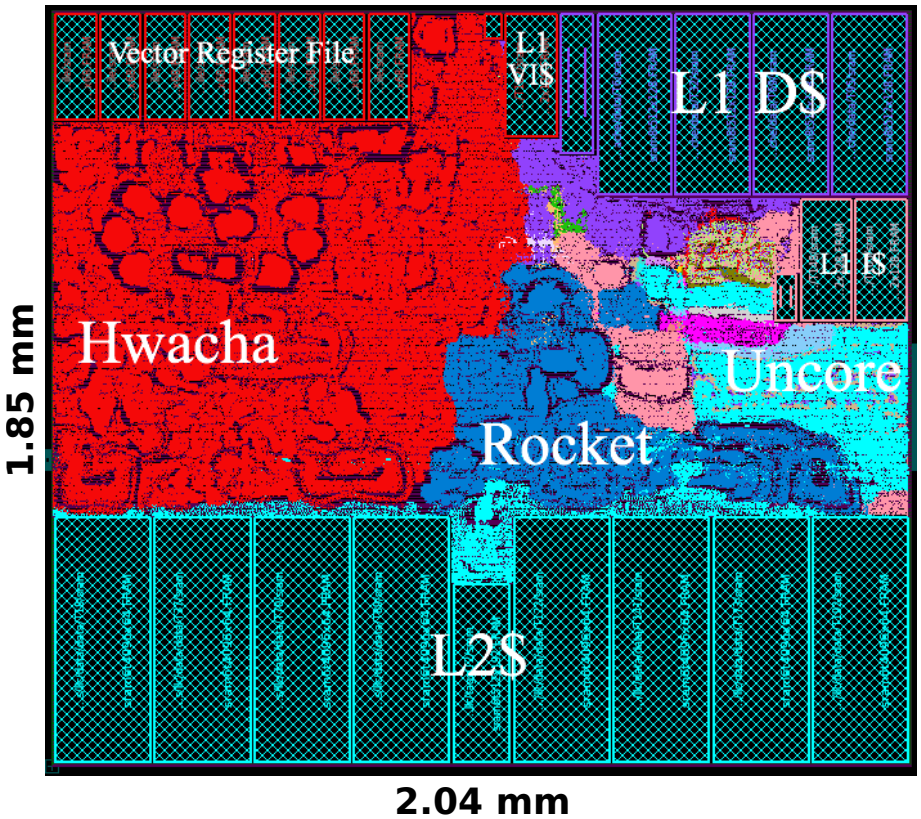
Most complexity lies in the control, not datapath.

Subword packing implications

- Chaining operations of mismatched throughput
- Different read/write rates for mixed-precision operations

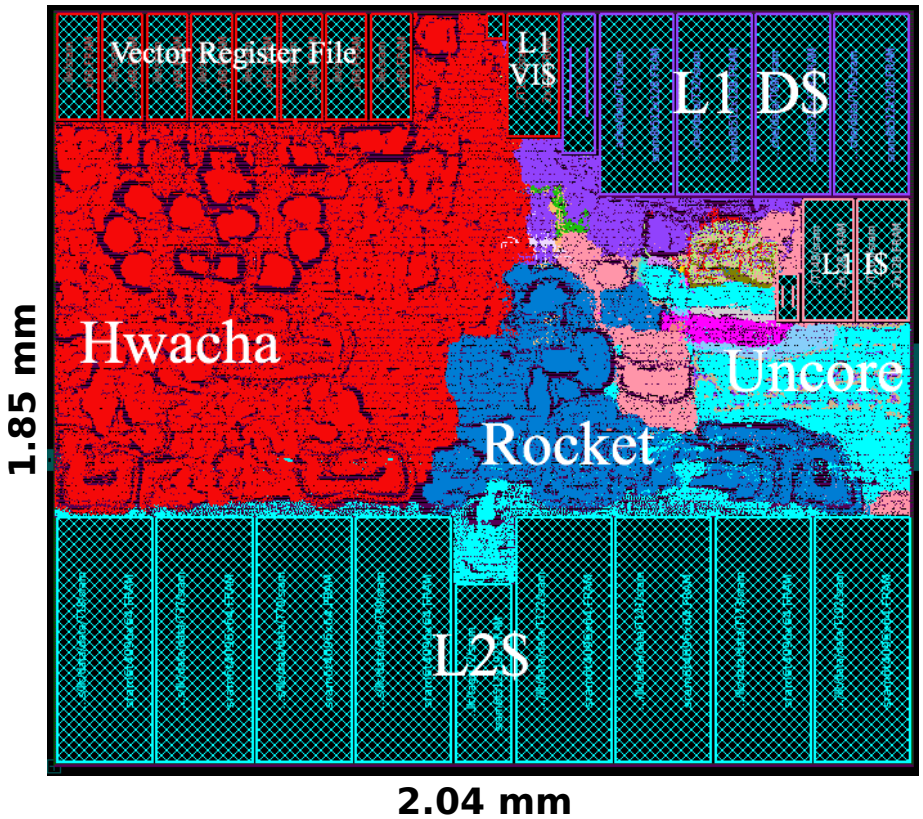
Physical Design

- Chisel RTL description
- ASIC tool flow: Synopsys Design Compiler, IC Compiler
- TSMC40GPLUS technology
- SRAMs modeled with CACTI
- Area: 3.79 mm^2
- Critical path through uncore: −0.2 ns worst negative slack when targeting 1 ns clock period



Area Impact

- 0.2 mm^2 overhead
- 17.3% in Hwacha
- 5.8% in overall chip
- FMA units: increase of 0.086 mm^2



Preliminary Benchmarking

Simulation of the post-layout gate-level netlist + PrimeTime

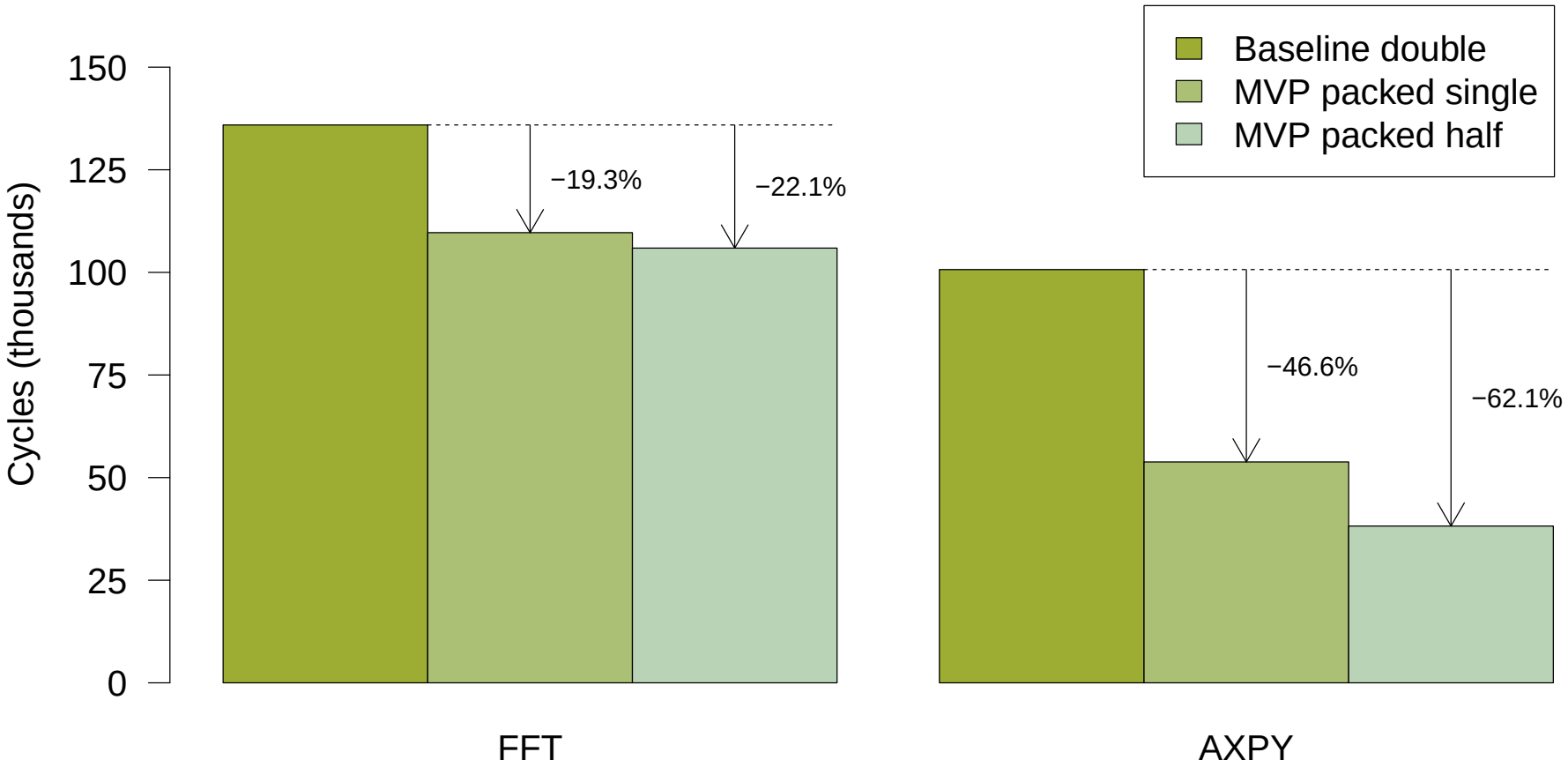
FFT: 1024-point radix-2 fast Fourier transform

- Bounded by scatter/gather (bit-reversal permutation)
- Relative improvement from FMA throughput only
- Smaller memory footprint from reduced precision

AXPY: matrix-vector $Ax + y$, dimension $n = 128$

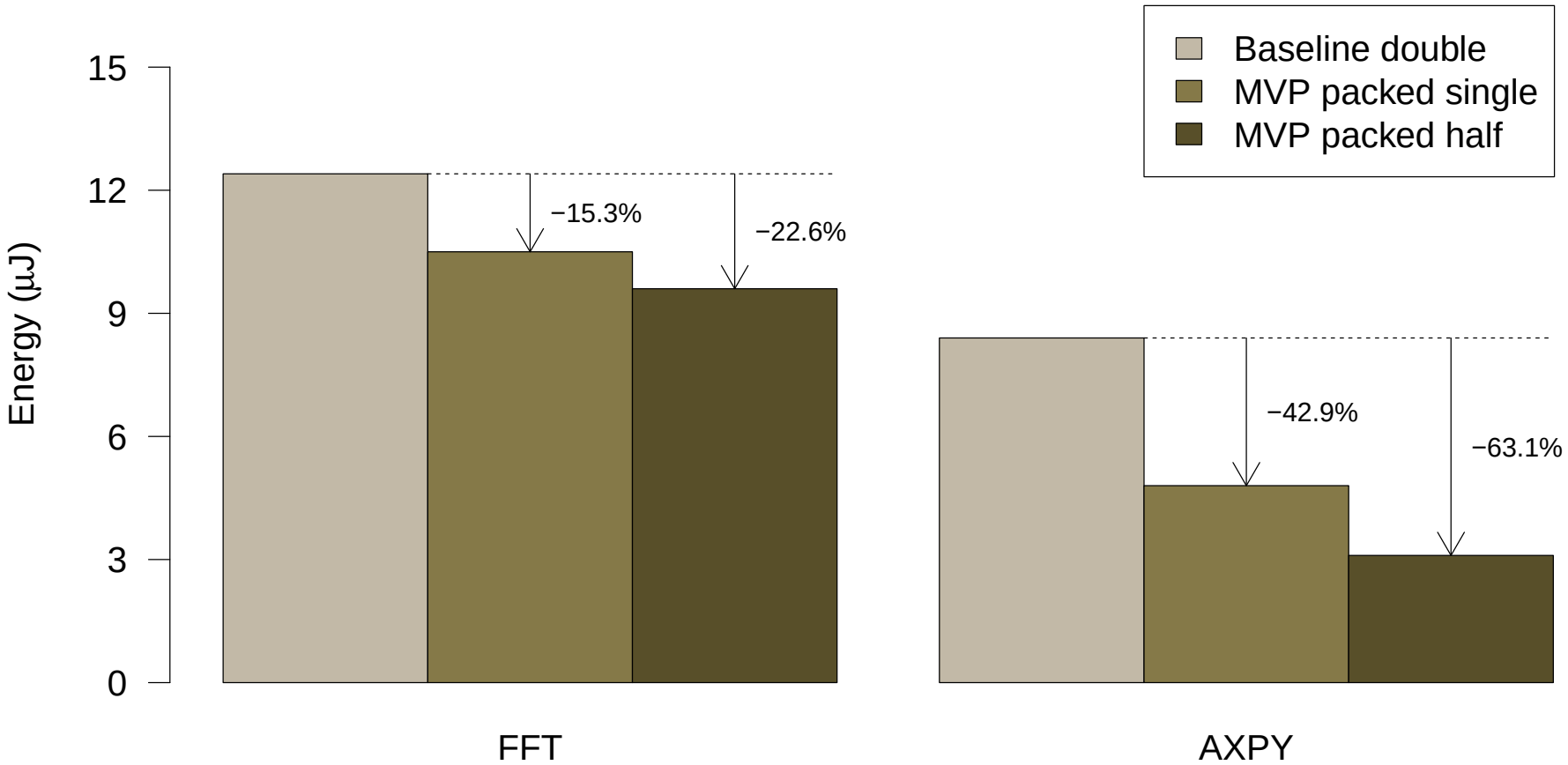
- Unit-stride vectors favorable to VMU
- Memory access coalescing
- Cache prefetching

Performance



(*lower is better*)

Energy



(*lower is better*)

Conclusion

- Mixed-precision computation offers significant benefits in performance and energy efficiency for programmable accelerators.
- Vector architectures yield a more convenient abstraction than SIMD extensions for expressing mixed-precision code.

Conclusion

- Mixed-precision computation offers significant benefits in performance and energy efficiency for programmable accelerators.
- Vector architectures yield a more convenient abstraction than SIMD extensions for expressing mixed-precision code.

Future Work

- Integrate with Precimonious, a tool to tune floating-point precisions, to automate configuration
- Investigate more variety of mixed-precision applications e.g., Smith-Waterman

Questions?

- Mixed-precision computation offers significant benefits in performance and energy efficiency for programmable accelerators.
- Vector architectures yield a more convenient abstraction than SIMD extensions for expressing mixed-precision code.

Future Work

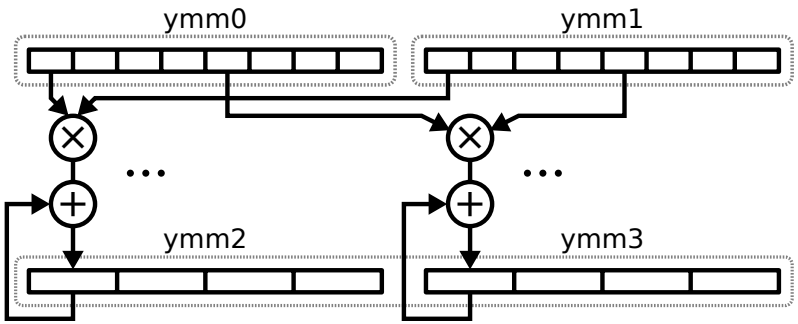
- Integrate with Precimonious, a tool to tune floating-point precisions, to automate configuration
- Investigate more variety of mixed-precision applications e.g., Smith-Waterman

Acknowledgements

Research is partially funded by DARPA Award Number HR0011-12-2-0016, the Center for Future Architecture Research, a member of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA, and ASPIRE Lab industrial sponsors and affiliates Intel, Google, Nokia, NVIDIA, Oracle, and Samsung. Any opinions, findings, conclusions, or recommendations in this paper are solely those of the authors and do not necessarily reflect the position or the policy of the sponsors.

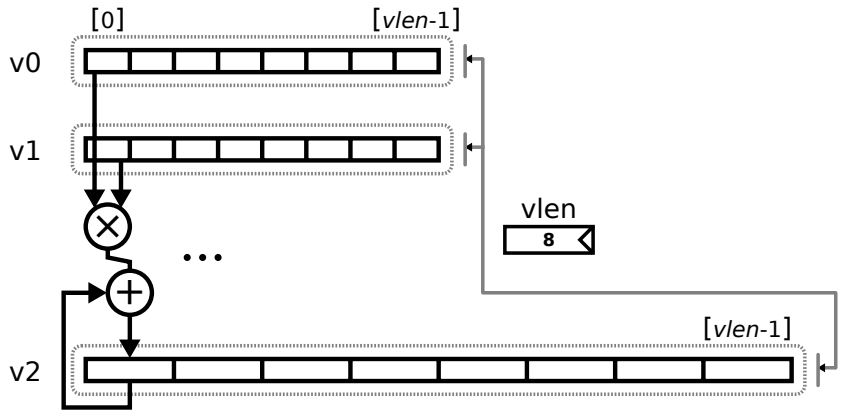
Review

SIMD Extensions



- Opcode-encoded fixed bit-width
- Number of elements per SIMD register varies at different precisions

Vectors



- Flexible vector length register
- Length expressed in terms of elements
- Same number of elements per vector register regardless of precision