Vector Architectures Vs. Superscalar and VLIW for Embedded Media Benchmarks

Christos Kozyrakis
Stanford University

David Patterson
U.C. Berkeley

http://csl.stanford.edu/~christos

Motivation

- Ideal processor for embedded media processing
  - High performance for media tasks
  - Low cost
    - Small code size, low power consumption, highly integrated
  - Low power consumption (for portable applications)
  - Low design complexity
  - Easy to program with HLLs
  - Scalable
- This work
  - How efficient is a simple vector processor for embedded media processing?
    - No cache, no wide issue, no out-of-order execution
  - How does it compare to superscalar and VLIW embedded designs?

Outline

- Motivation
- Overview of VIRAM architecture
  - Multimedia instruction set, processor organization, vectorizing compiler
- EEMBC benchmarks & alternative architectures
- Evaluation
  - Instruction set analysis & code size comparison
  - Performance comparison
  - VIRAM scalability study
- Conclusions

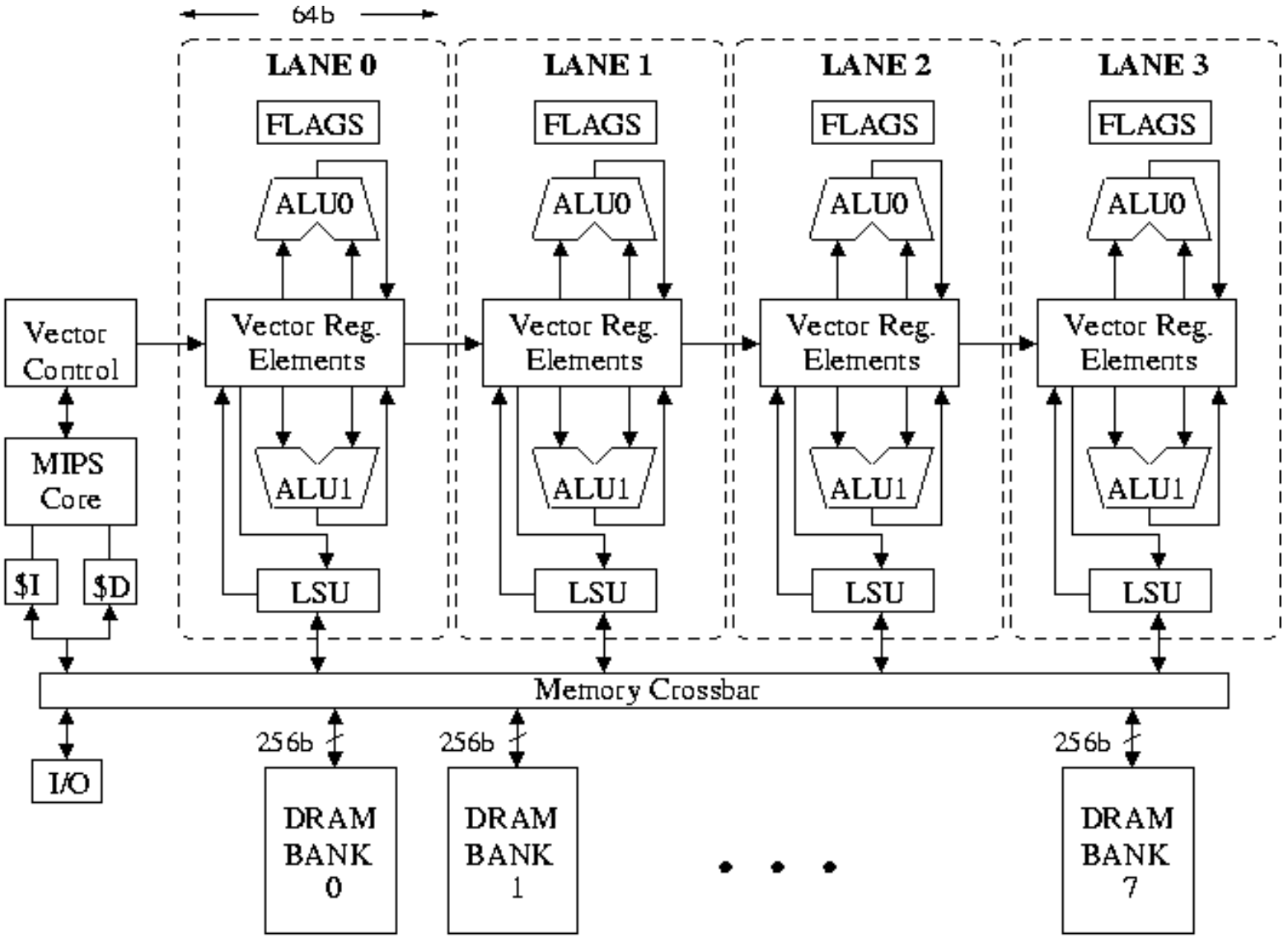
VIRAM Instruction Set

- Vector load-store instruction set for media processing
  - Coprocessor extension to MIPS architecture
- Architecture state
  - 32 general-purpose vector registers
  - 16 flag registers
  - Scalar registers for control, addresses, strides, etc
- Vector instructions
  - Arithmetic: integer, floating-point, logical
  - Load-store: unit-stride, strided, indexed
  - Misc: vector & flag processing (pop count, insert/extract)
  - 90 unique instructions

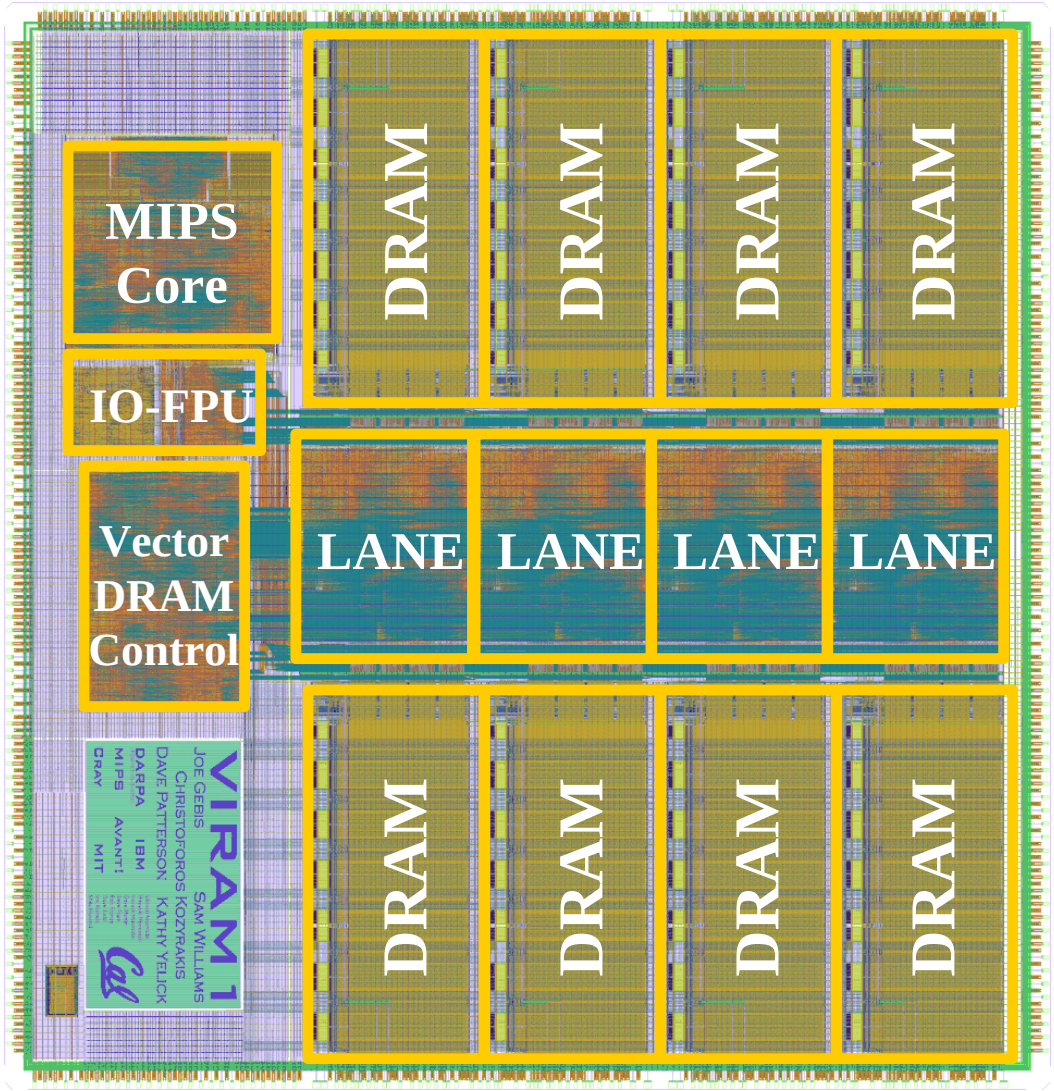
VIRAM ISA Enhancements

- Multimedia processing
  - Support for multiple data-types (64b/32b/16b)
    - Element & operation width specified with control register
  - Saturated and fixed-point arithmetic
    - Flexible multiply-add model without accumulators
  - Simple element permutations for reductions and FFTs
  - Conditional execution using the flag registers
- General-purpose systems
  - TLB-based virtual memory
    - Separate TLB for vector loads & stores
  - Hardware support for reduced context switch overhead
    - Valid/dirty bits for vector registers
    - Support for “lazy” save/restore of vector state

VIRAM Processor Microarchitecture

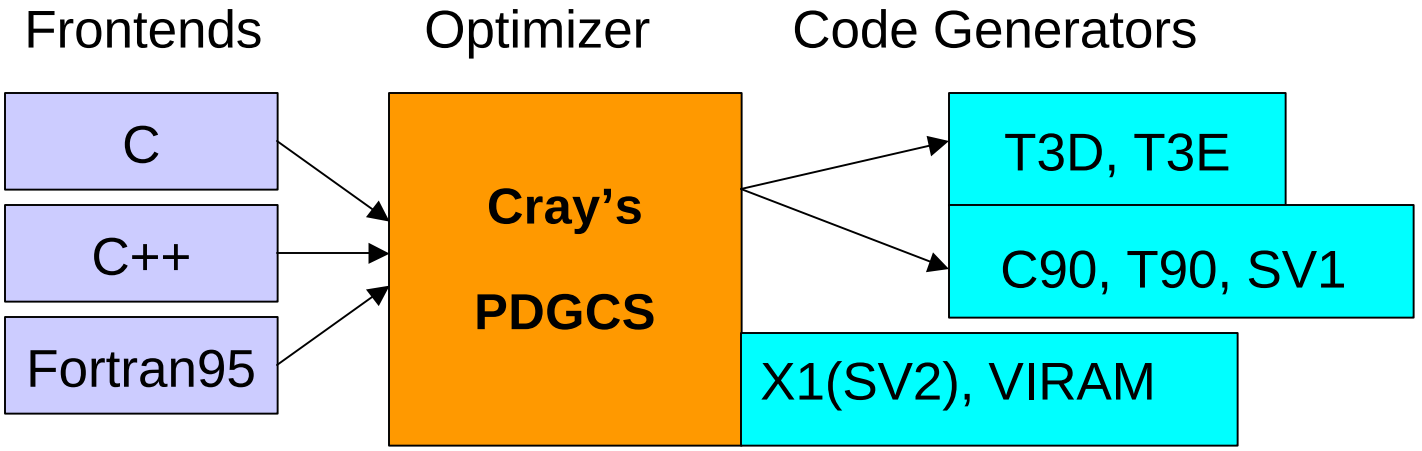


VIRAM Chip



- 0.18µm CMOS (IBM)
- Features
  - 8KB vector register file
  - 2 256-bit integer ALUs
  - 1 256-bit FPU
  - 13MB DRAM
- 325mm^2 die area
- 125M transistors
- 200 MHz, 2 Watts
- Peak vector performance
  - Integer: 1.6/3.2/6.4 Gop/s (64b/32b/16b)
  - Fixed-point: 2.4/4.8/9.6 Gop/s (64b/32b/16b)
  - FP: 1.6 Gflop/s (32b)

Vectorizing Compiler



- Based on Cray PDGCS compiler
  - Used with all vector and MPP Cray machines
- Extensive vectorization capabilities
  - Including outer-loop vectorization
- Vectorization of narrow operations and reductions

EEMBC Benchmarks

- The de-facto industrial standard for embedded CPUs
- Used consumer & telecommunication categories
  - Representative of workload for multimedia devices with wireless/broadband capabilities
  - C code, EEMBC reference input data
- Consumer category
  - Image processing tasks for digital camera devices
  - Rgb2cmyk & rgb2yiq conversions, convolutional filter, jpeg encode & decode
- Telecommunication category
  - Encoding/decoding tasks for DSL/wireless
  - Autocorrelation compression, convolutional encoder, DSL bit allocation, FFT, Viterbi decoding

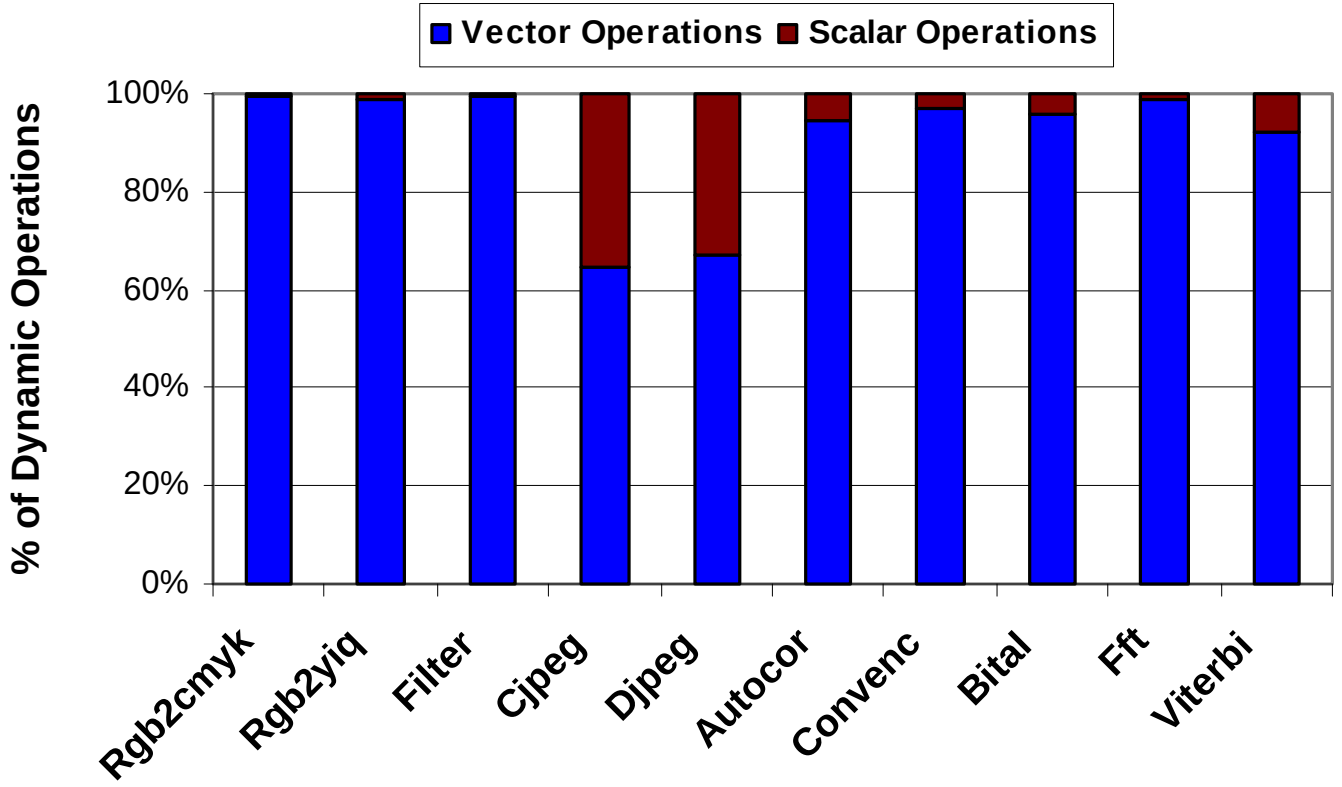
EEMBC Metrics

- Performance: repeats/second (throughput)
  - Use geometric means to summarize scores
- Code size and static data size in bytes
  - Data sizes the same for the processors we discuss
- Pitfall with caching behavior
  - Fundamentally, no temporal locality in most benchmarks
  - Repeating kernel on same (small) data creates locality
  - Unfair advantage for cache based architectures
    - VIRAM has no data cache for vector loads/stores

Embedded Processors

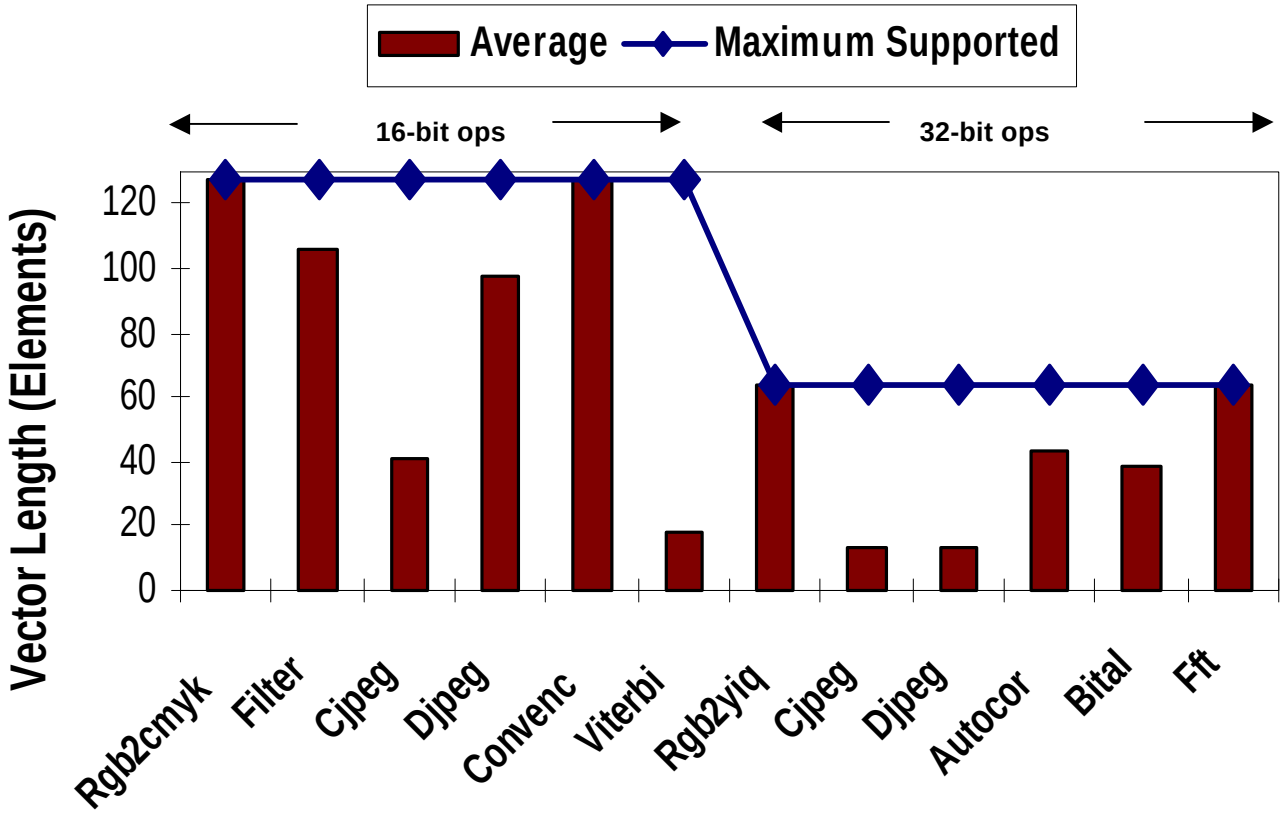
Architecture		Chip	Issue Width	OOO	Cache L1I/L1D/L2 (KB)	Clock (MHz)	Power (W)
VIRAM	Vector	VIRAM	1	–	8/—/—	200	2.0
x86	CISC	K6-IIIE+	3	√	32/32/256	550	21.6
PowerPC	RISC	MPC7455	4	√	32/32/256	1000	21.3
MIPS	RISC	VR5000	2	–	32/32/—	250	5.0
Trimedia	VLIW+ SIMD	TM1300	5	–	32/16/—	166	2.7
VelociTI	VLIW+ DSP	TMS320C6 203	8	–	96/512/—	300	1.7

Degree of Vectorization



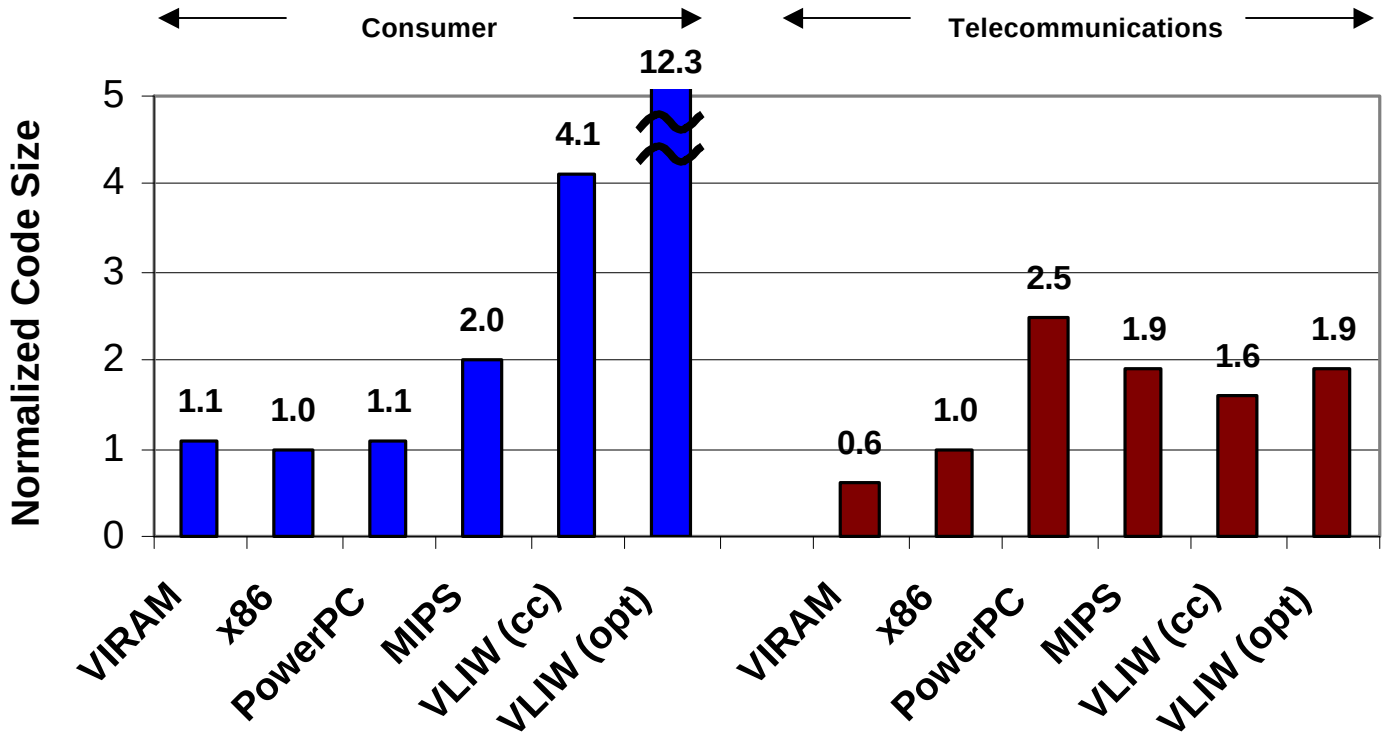
- Typical degree of vectorization is 90%
  - Even for Viterbi decoding which is partially vectorizable
  - ISA & compiler can capture the data parallelism in EEMBC
  - Great potential for vector hardware

Vector Length



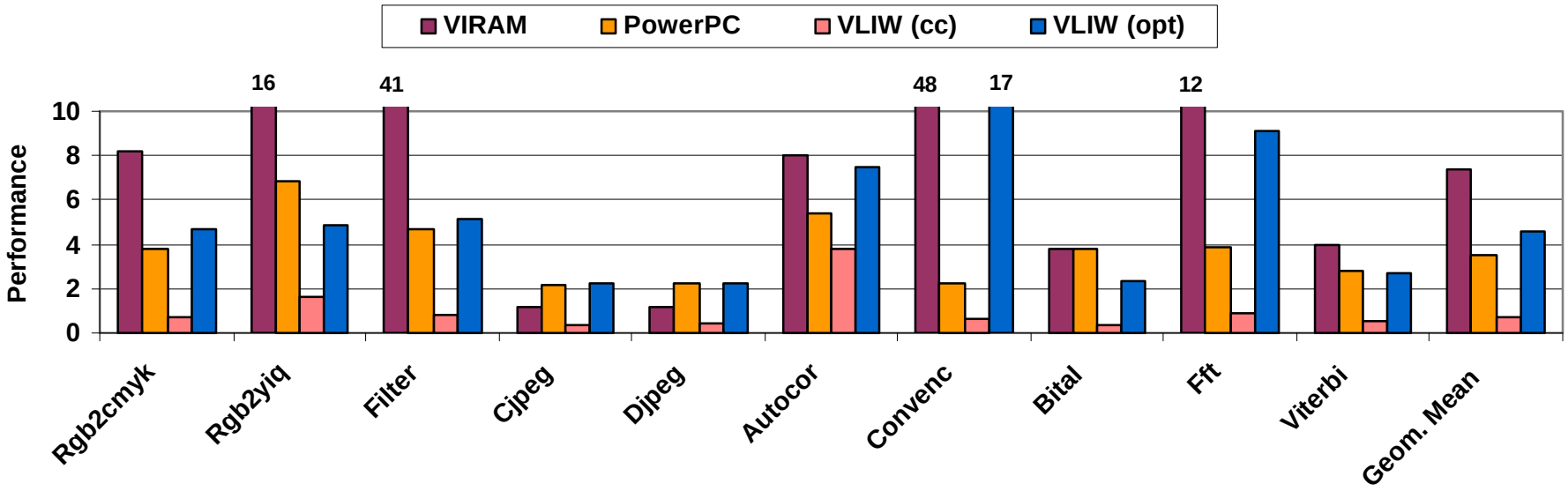
- Short vectors
  - Unavoidable with Viterbi, artifact of code with Cjpeg/Djpeg
- Even shortest length operations have ~20 16-bit elements
  - More parallelism than 128-bit SSE/Altivec can capture

Code Size



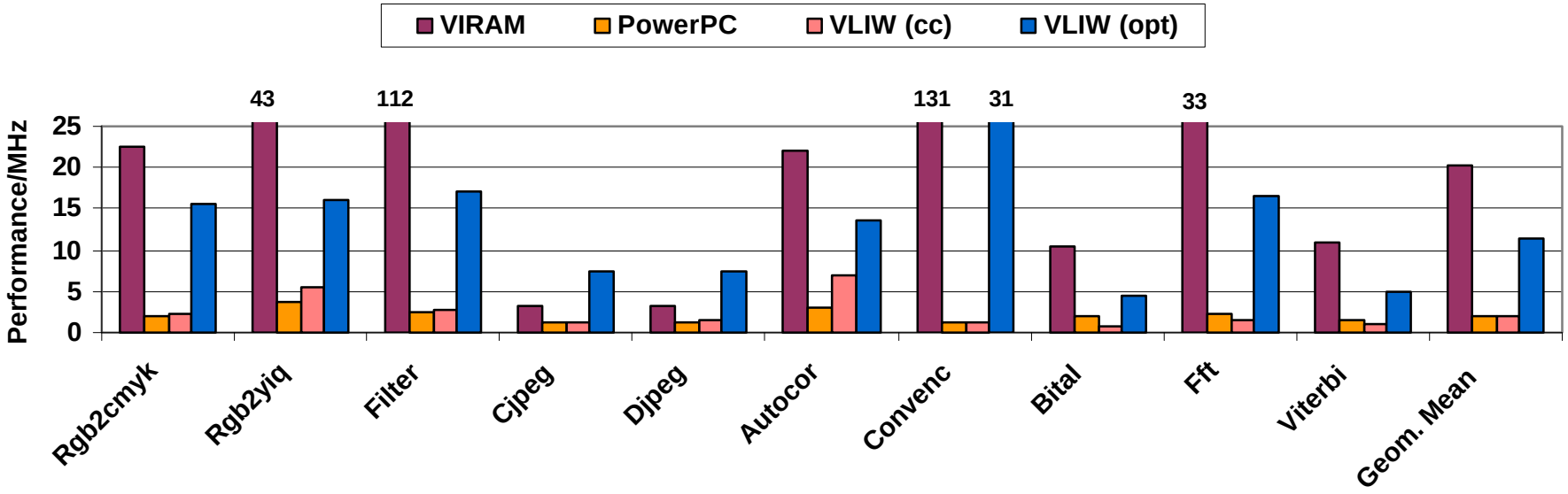
- VIRAM high code density due to
  - No loop unrolling/software pipelining, compact expression of strided/indexed/permutation patterns, no small loops
- Vectors: a “code compression” technique for RISC

Comparison: Performance



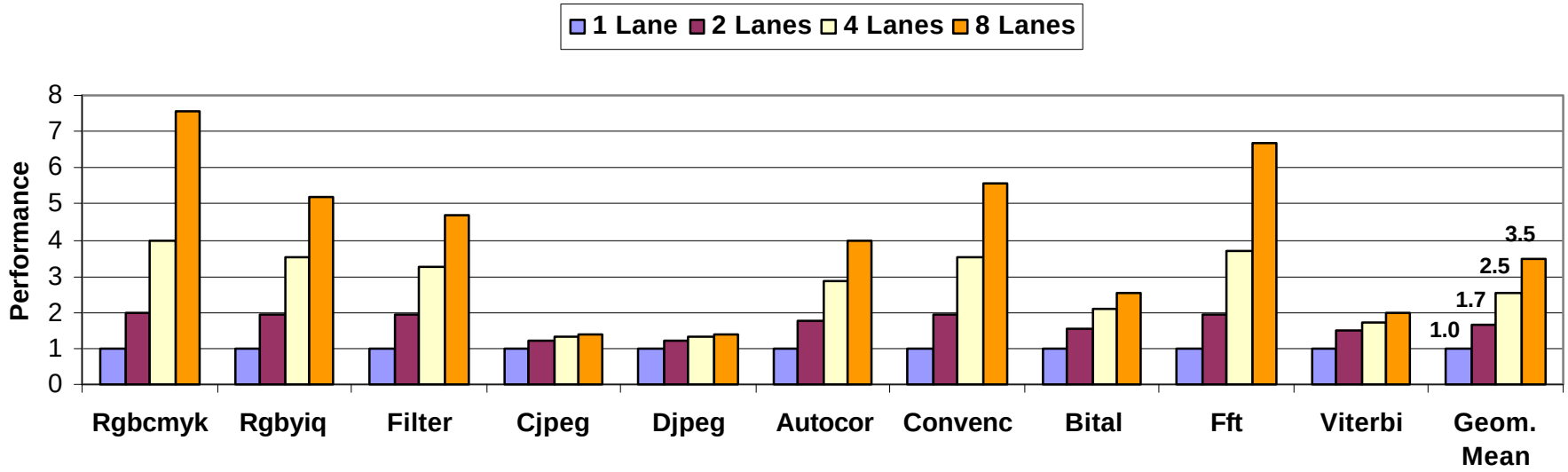
- 200MHz, cache-less, single-issue VIRAM is
  - 100% faster than 1-GHz, 4-way OOO processor
  - 40% faster than manually-optimized VLIW with SIMD/DSP
- VIRAM can sustain high computation throughput
  - Up to 16 (32-bit) to 32 (16-bit) arithmetic ops/cycle
- VIRAM can hide latency of accesses to DRAM

Comparison: Performance/MHz



- VIRAM Vs 4-way OOO & VLIW with compiler code
  - 10x-50x with long vectors, 2x-5x with short vectors
- VIRAM Vs VLIW with manually optimized code
  - VLIW within 50% of VIRAM
  - VIRAM would benefit from the same optimizations
- Similar results if normalized by power or complexity

VIRAM Scalability



- Same executable, frequency, and memory system
- Decreased efficiency for 8 lanes
  - Short vector lengths, conflicts in the memory system
  - Difficult to hide overhead with short execution times
- Overall: 2.5x with 4 lanes, 3.5x with 8 lanes
  - Can you scale similarly a superscalar processor?

Conclusions

- Vectors architectures are great match for embedded multimedia processing
  - Combined high performance, low power, low complexity
  - Add a vector unit to your media-processor!
- VIRAM code density
  - Similar to x86, 5-10 times better than optimized VLIW
- VIRAM performance
    - With compiler vectorization and no hand-tuning
  - 2x performance of 4-way OOO superscalar
    - Even if OOO runs at 5x the clock frequency
  - 50% faster than manually-optimized 5 to 8-way VLIW
    - Even if VLIW has hand-inserted SIMD and DSP support