A Hardware Accelerator for Computing an Exact Dot Product

Jack Koenig, David Biancolin, Jonathan Bachrach, Krste Asanović

Berkeley Architecture Research

Berkeley
UNIVERSITY OF CALIFORNIA

Challenges with Floating Point

Addition and multiplication are not associative

$10^{20} + 1 - 10^{20} = 0$

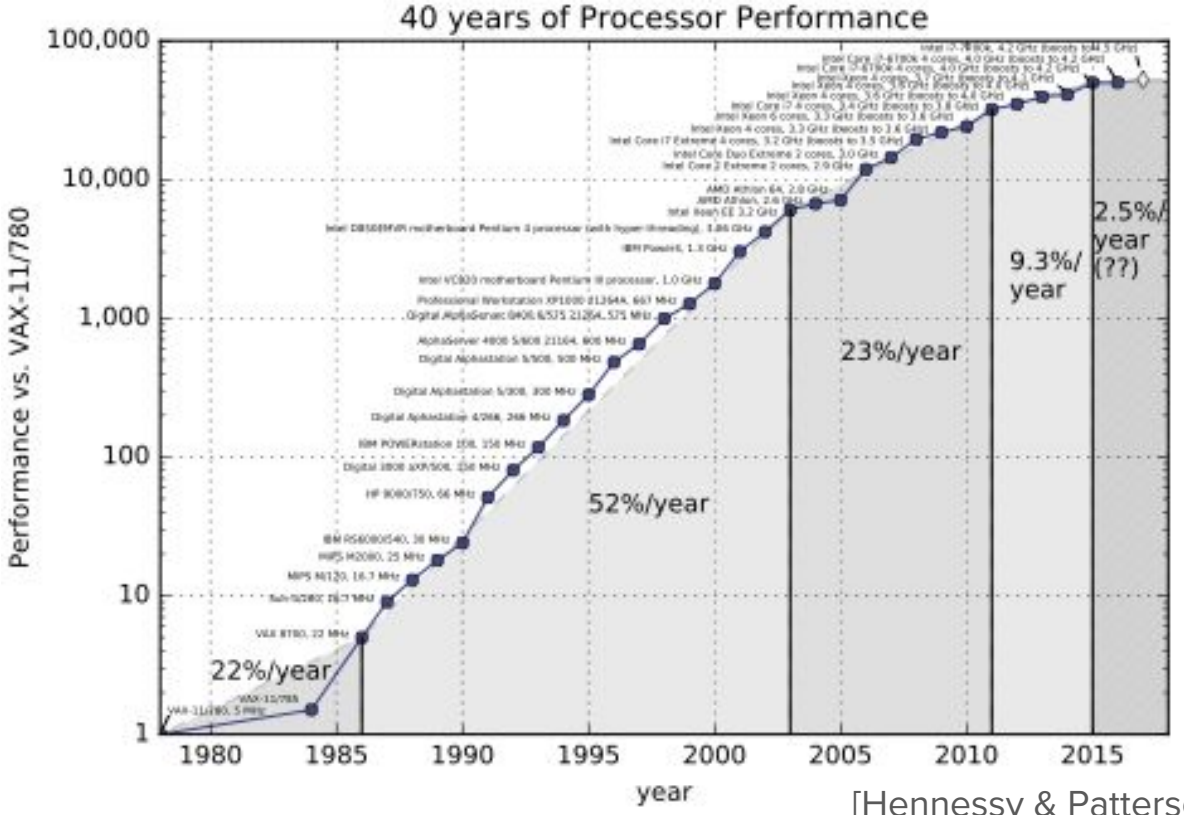
Multithreaded, not even reproducible!

$10^{20} + 1 - 10^{20} = 0$ **or** 1

Solutions

- MPFR - Exact but much slower than hardware
- ExactBLAS - Faster than MPFR, still slower than hardware
- ReproBLAS - Fast and reproducible, but not exact

Moore’s Law Winding Down



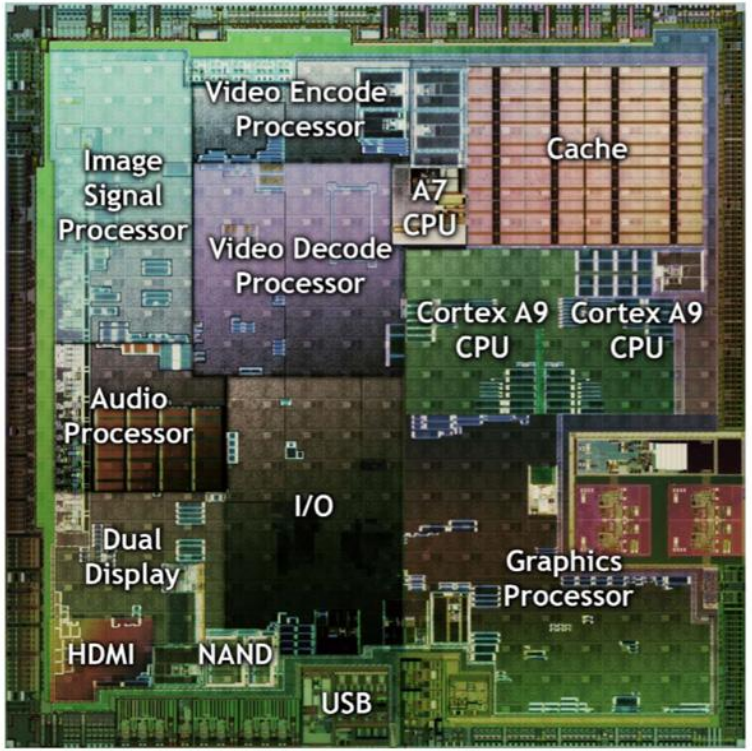
[Hennessy & Patterson, 2017]

From Moore's Law to Dark Silicon

Specialization is already here!

- System-on-Chips have billions of transistors
- Power density constraints prevent all transistors from being used at once
- Accelerators orders-of-magnitude more efficient than CPUs
- Can turn off unused specialized units to save power

⇒ Use those extra transistors for specialized hardware



NVIDIA Tegra 2

Motivation

Why Dot Product?

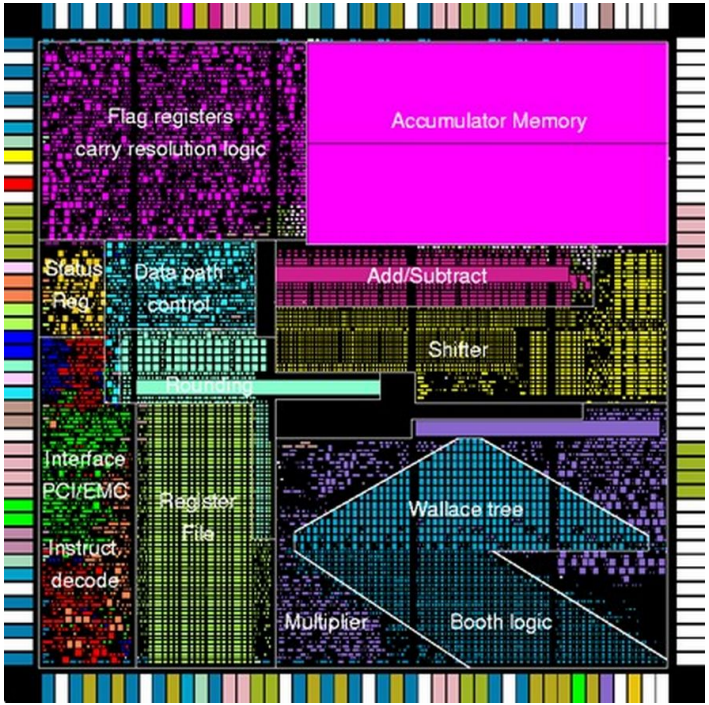
1) A kernel of many applications
2) Reduction is good candidate for exact representation

Why Exact?

Simplifies error analysis

Related Work

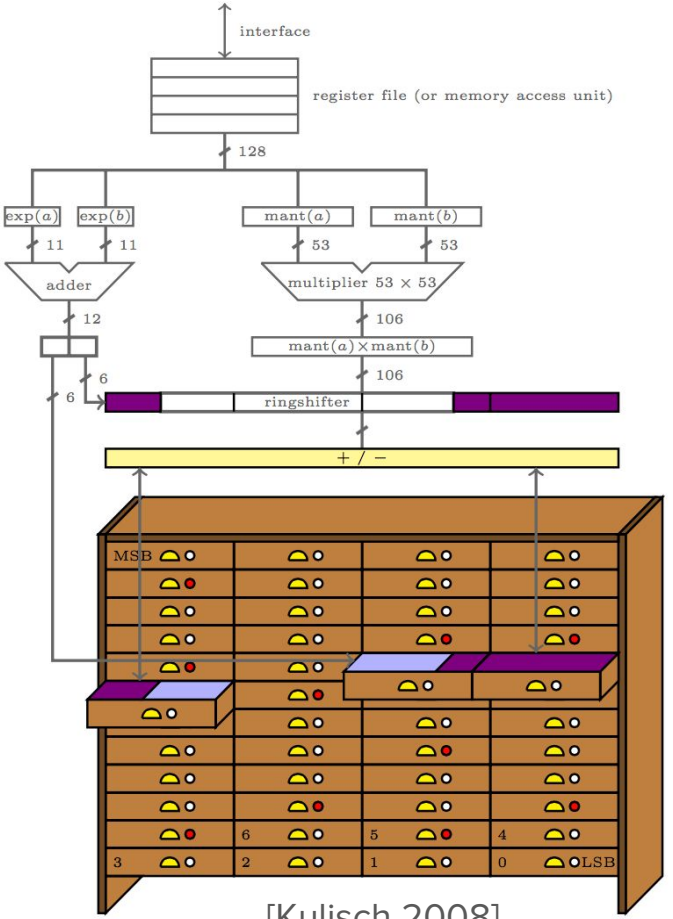
- We were heavily influenced by the work of Ulrich Kulisch et. al
  - XPA 3233 in 1994
  - PCI-based co-processor
  - 0.8um process
- Recently, Uguen and Dinechin published a design-space exploration for FPGA-hosted implementations of Kulisch's design



[XPA 3233]

Principle of Operation

- Fixed point representation of entire space
  - $1 + 2 \times (2^{bits(exp)} + bits(mant))$
  - 2100 bits to represent 1 double-precision number
  - 4200 bits for product of 2 doubles
  - 88 bits to preclude overflow
  - 4288 bits in our complete representation (CR)
- Accumulation
  - Fetch elements of each vector from memory
  - Calculate product of the mantissas and sum of the exponents
  - Use sum of exponents to align product of mantissas with complete representation
  - Accumulate
  - Propagate carry or borrow if necessary



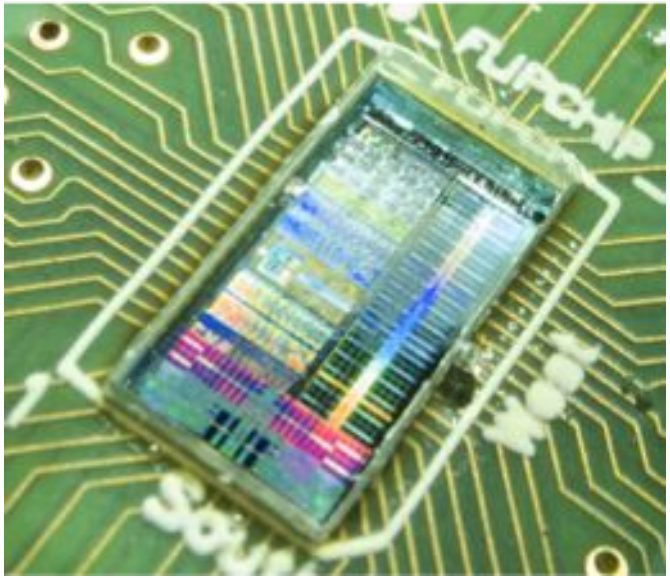
[Kulisch 2008]

System Architecture

Rocket Chip Generator

- A RISC-V processor generator
- RISC-V is an open-source, extensible ISA
- Provides Rocket Custom Coprocessor Interface (RoCC)

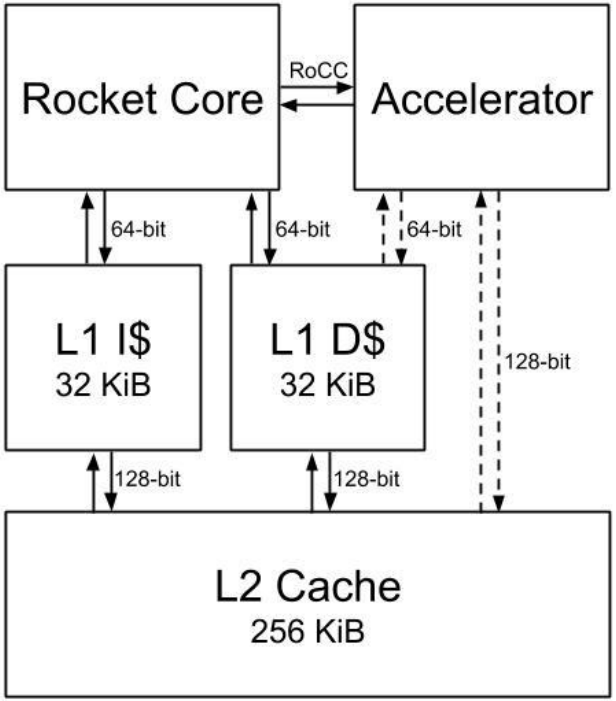
Used in over 12 academic tapeouts and at least 1 commercial tapeout



EOS 22 (2014)

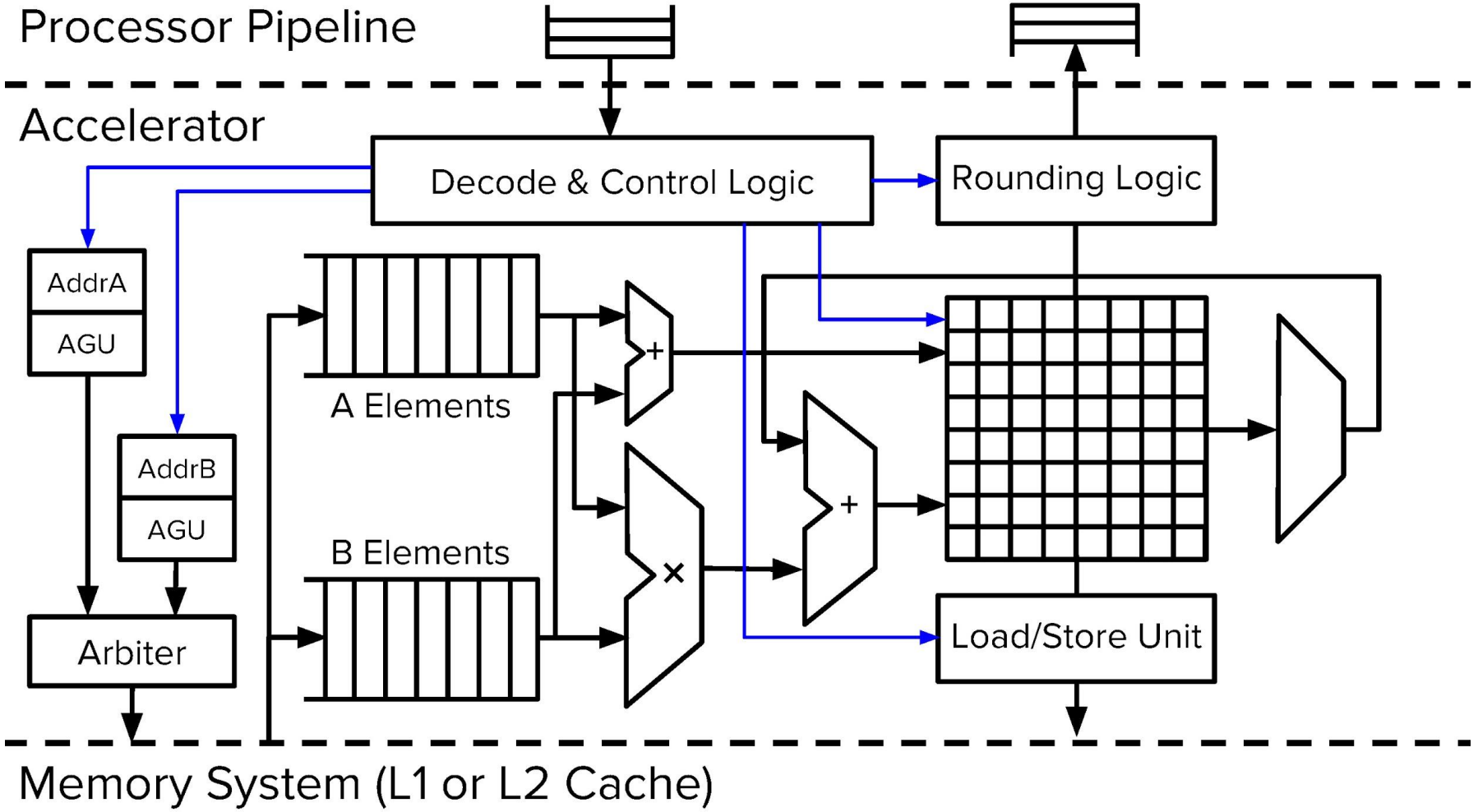
RoCC Accelerator

- Integrated with Rocket Chip via RoCC
  - 5 stage in-order pipeline (Rocket)
  - 32 KiB L1 instruction and data caches
  - 256 KiB L2 cache
- Instructions are fetched by Rocket core and forwarded to the accelerator
- Memory interface is parameterized for
  - 64-bit L1 cache interface
  - 128-bit L2 cache interface



Instructions

Name	Description
CLR_CR	Clear the complete register
RD_DBL/RD_FLT	Round the complete register and return the result to a general-purpose register
LD_CR	Loads a complete register from memory
ST_CR	Stores the complete register to memory
ADD_CR	Adds a complete register in memory to the current value
PRE_DP	Initializes vector base address registers
RUN_DP	Specifies vector length; instructs accelerator to begin computation


Processor Pipeline
Accelerator
Decode & Control Logic
Rounding Logic
AddrA
AGU
AddrB
AGU
A Elements
B Elements
+
×
+
Arbiter
Load/Store Unit
Memory System (L1 or L2 Cache)

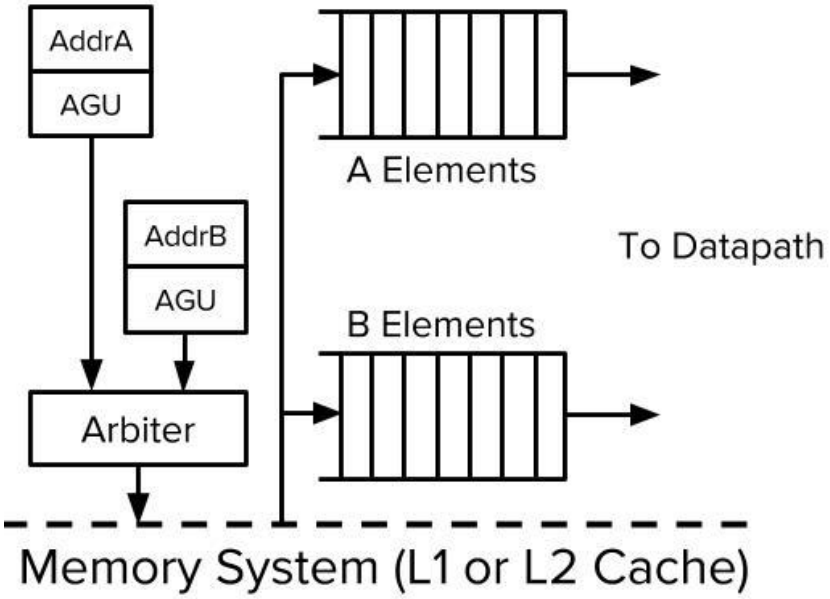
Control & Memory Unit

Control Unit

- Decodes instructions
- Rounds complete register

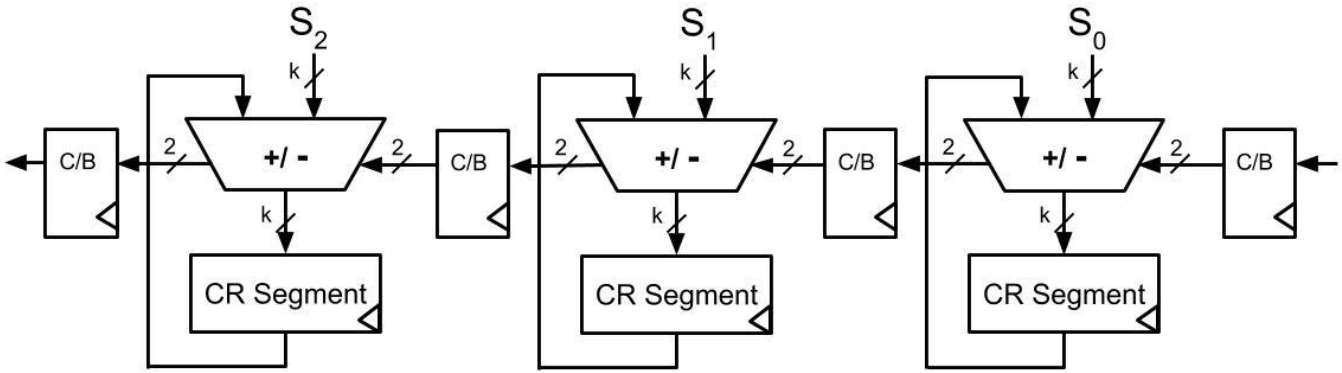
Memory Unit

- Fetches operands from memory and re-orders responses to feed to datapath
- Parameterized for 64-bit L1 or 128-bit L2 interface



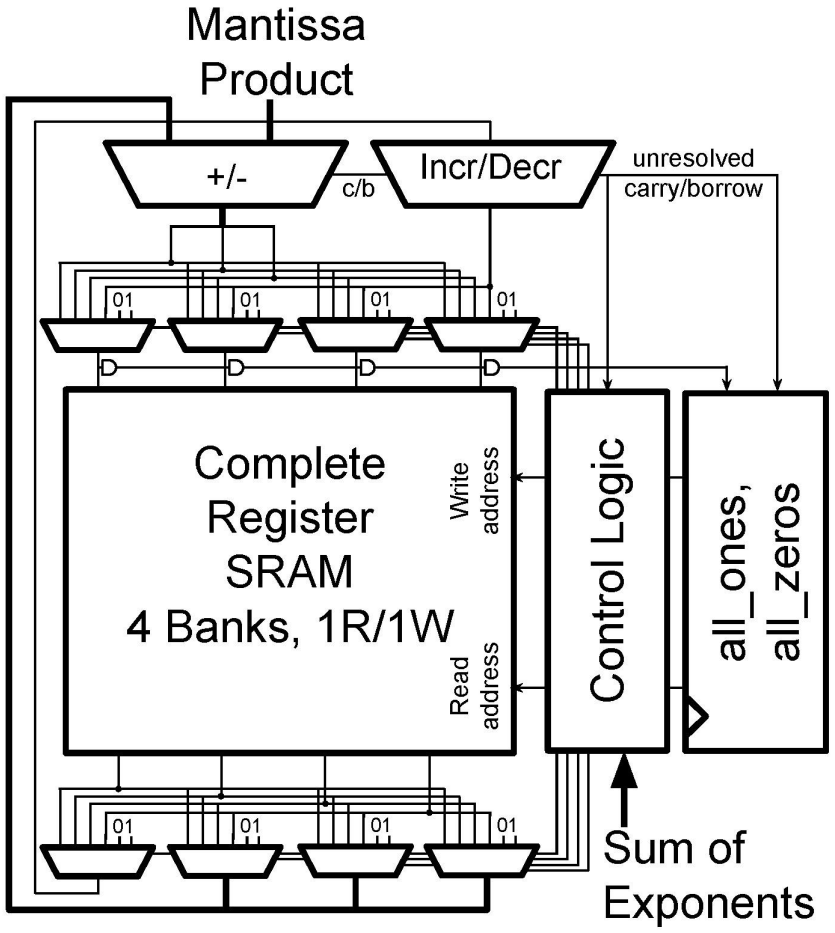
Segmented Accumulator

- Divide complete register into segments
- Each segment gets its own adder
- Accumulates a portion of the product of the mantissas and incoming carry/borrow



Centralized Accumulator

- Uses a single adder
- For double, product of mantissas gives 104-bit summand
- Read appropriate 4 words from accumulator based on sum of exponents
- Add summand to lower-order 3 words, propagated carry/borrow into 4th
- Stall if carry or borrow propagates beyond
- all_ones, all_zeros helps with propagation



Methodology & Evaluation Overview

Performance evaluation requires both cycles-per-dot-product and cycle time.

1) Cycles-per-dot-product:
   - Simulate the SoC in RTL simulation, measure execution time in cycles
2) Cycle time:
   - Push SoC through synthesis and P&R, determine critical path, area

Design space exploration over three parameters:

Complete Register (Centralized, Segmented)

Cache Interface (L1 = 64 bit, L2 = 128 bit)

Operand Precision (Double, Float)

Measuring Cycles-Per-Operation

- simulate entire SoC in RTL simulation (Synopsys VCS)
- microbenchmark: random vectors uniformly in the mantissa and exponent spac
- measure cycles-per-element (CPE) of software libraries on a host with similar caches:

Software libraries:

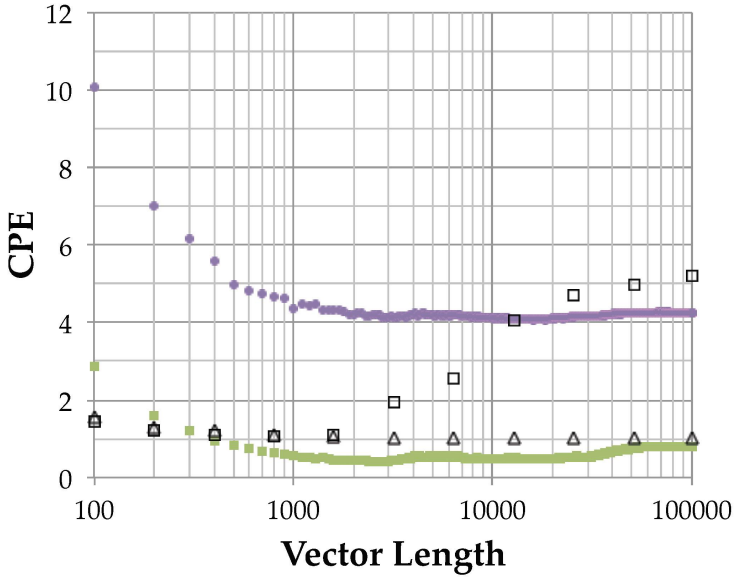
- ReproBLAS
- Intel MKL

Host machine: Intel Xeon E5-2667

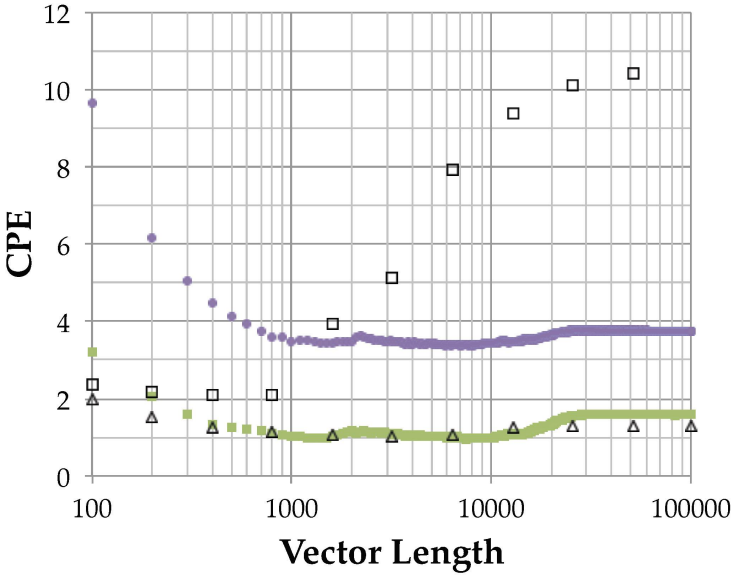
- caches: 32 KiB L1 D$, 256 KiB unified L2
- ISA extensions: SSE 4.1, 4.2, AVX

Comparison: CPE vs Vector Length

Single Precision



Double Precision



MKL

RBLAS

C_L2_D

C_L1_D

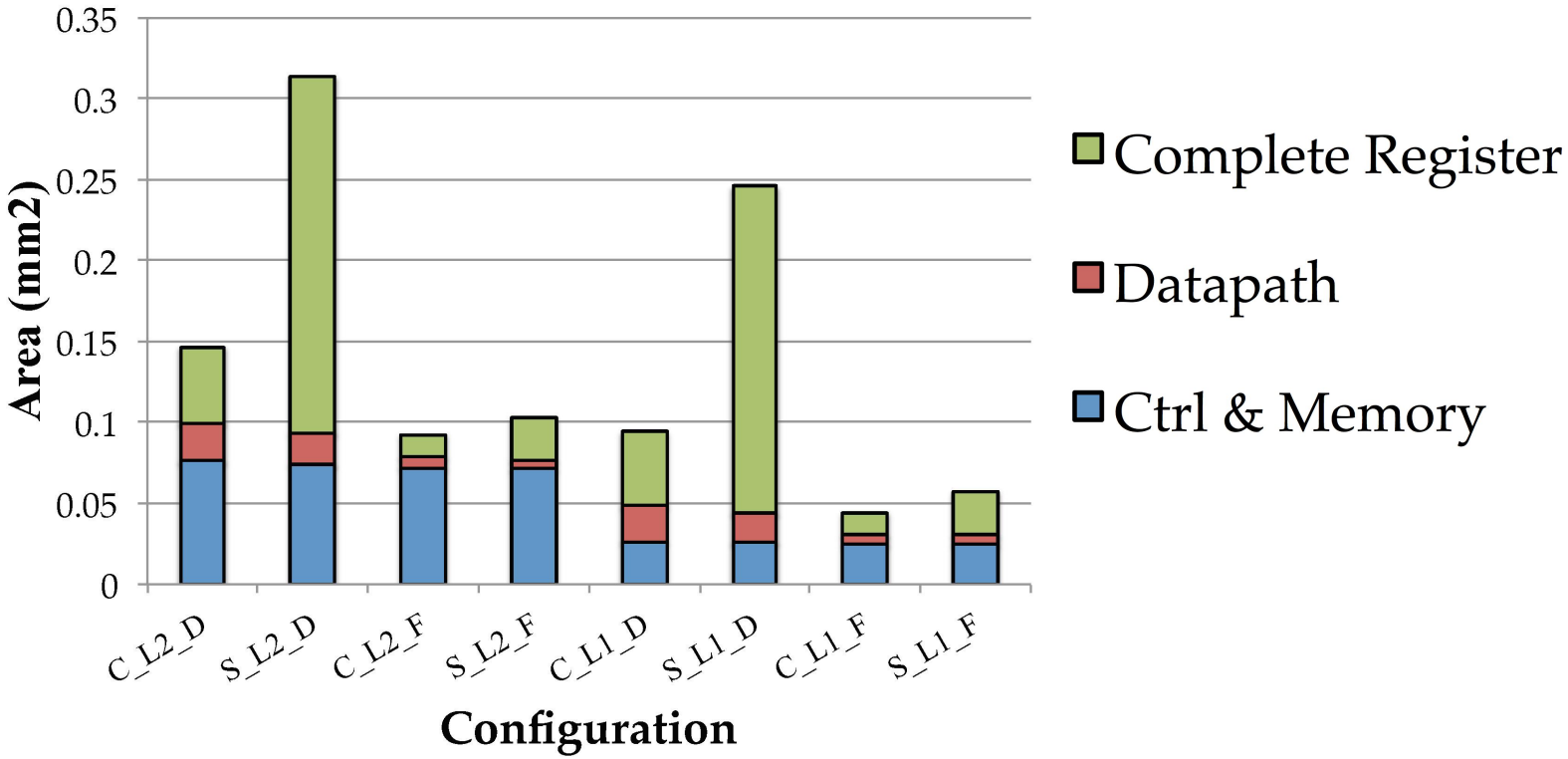
VLSI Evaluation

Push the complete SoC through CAD flow, measure cycle time and area.

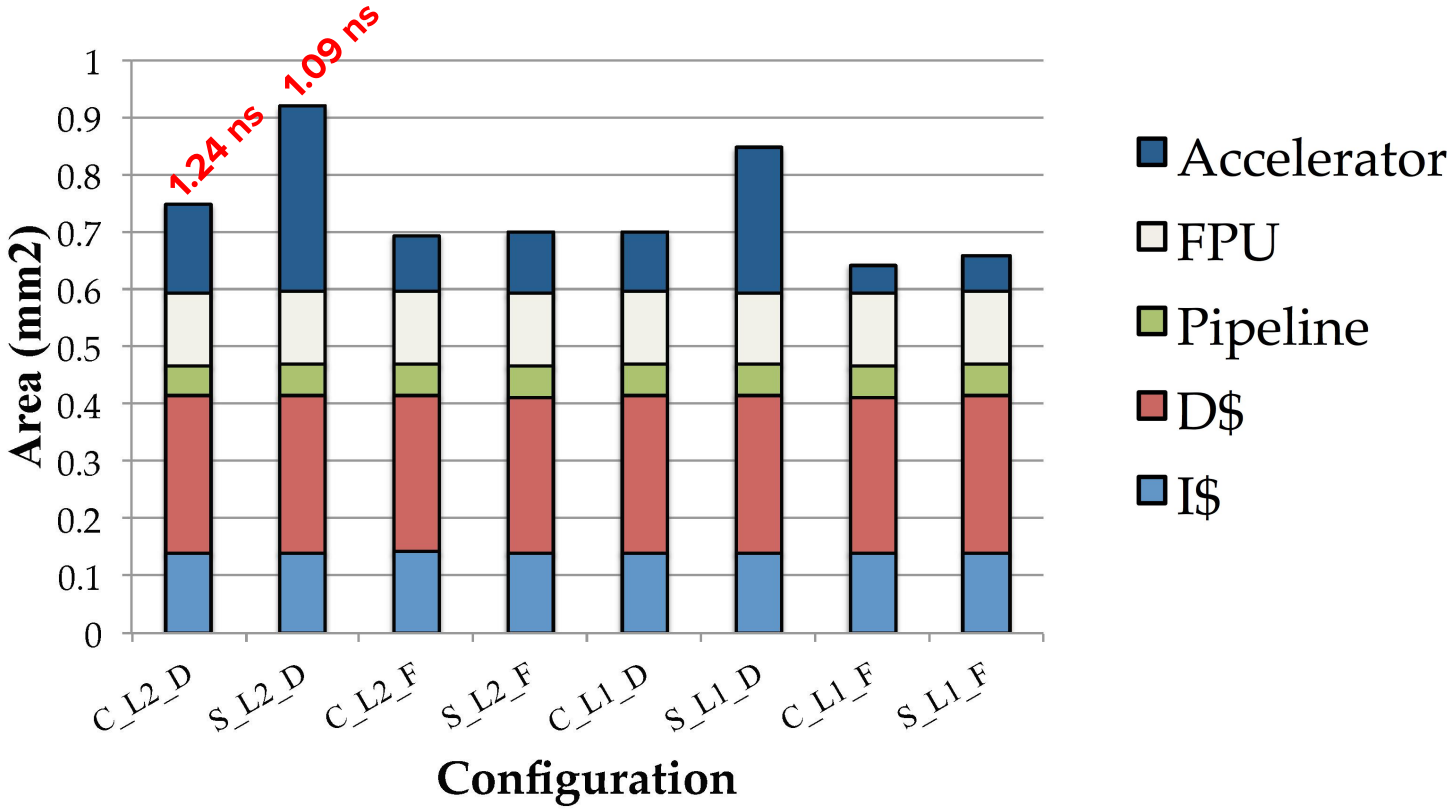
Flow details:

- Synthesis: Synopsys Design Compiler
- Place & Route: Synopsys IC Compiler
- Technology: TSMC 45nm
- No SRAM compiler
  - generate timing and area models using CACTI

Area Breakdown of Accelerator



Area Breakdown of Core excluding L2



Outstanding Questions & Future Work

- How to use effectively in BLAS-2 and BLAS-3 kernels?
  - Must amortize overhead of accelerator setup
  - Cost of saving intermediate exact results is high
- Measure energy and compare to software libraries.
  - Compare to software libraries.

Conclusion

- Realizable with modest area costs
- Easily saturates available memory bandwidth

Strong case for integration in application specific SoCs; more careful evaluation required to motivate integration in general-purpose machines

Acknowledgements

- Special thanks to Jim Demmel, William Kahan, Hong Diep Nguyen, and Colin Schmidt
- This research was partially funded by DARPA Award Number HR0011-12-2-0016 and ASPIRE Lab industrial sponsors and affiliates Intel, Google, HPE, Huawei, LGE, Nokia, NVIDIA, Oracle, and Samsung.