An Asynchronous Floating-Point Multiplier

Basit Riaz Sheikh and Rajit Manohar
Computer Systems Lab
Cornell University
http://vlsi.cornell.edu/

The person that did the work!

Cornell University

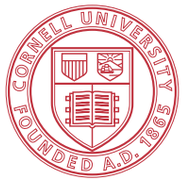
AVLSI

Motivation

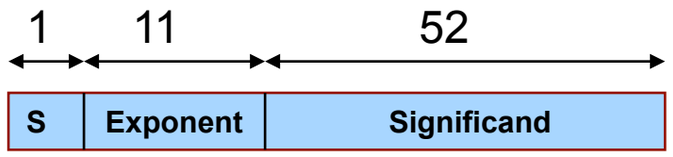
- Fast floating-point computation is important for scientific computing
  .... *and increasingly, for commercial applications*

- Floating-point operations use significantly more energy than integer operations

- Can we use asynchronous circuits to reduce the energy requirements for floating-point arithmetic?



IEEE 754 standard for binary floating-point arithmetic



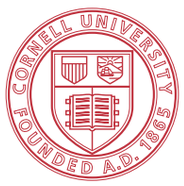
- Number is

$$(-1)^s(1.SIG) \times 2^{E-1023}$$

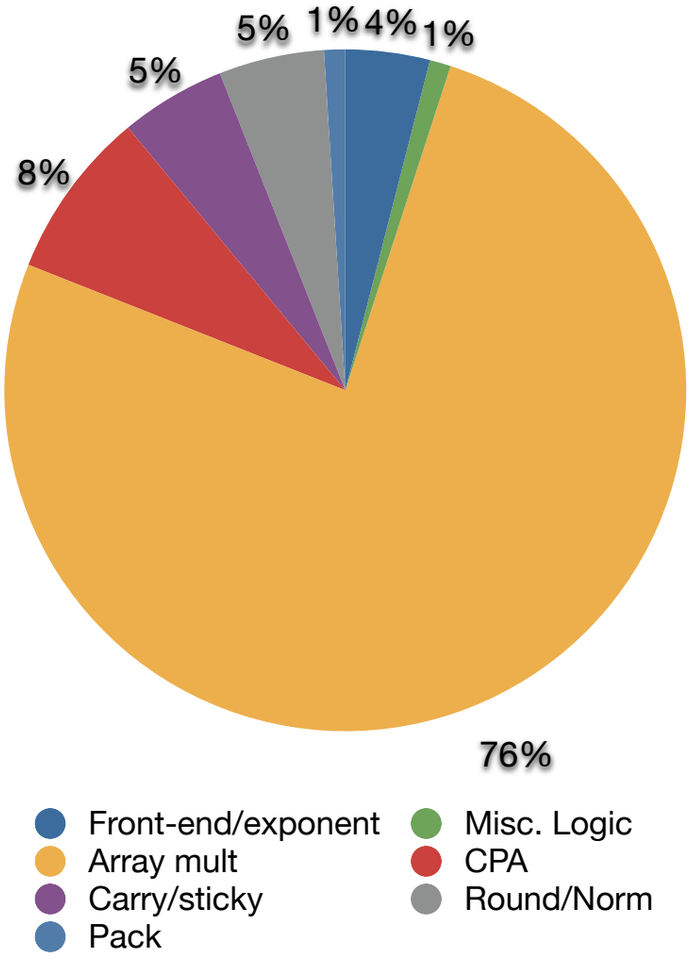
- Interesting cases
  - ❖ Denormal numbers (exponent is all zeros)

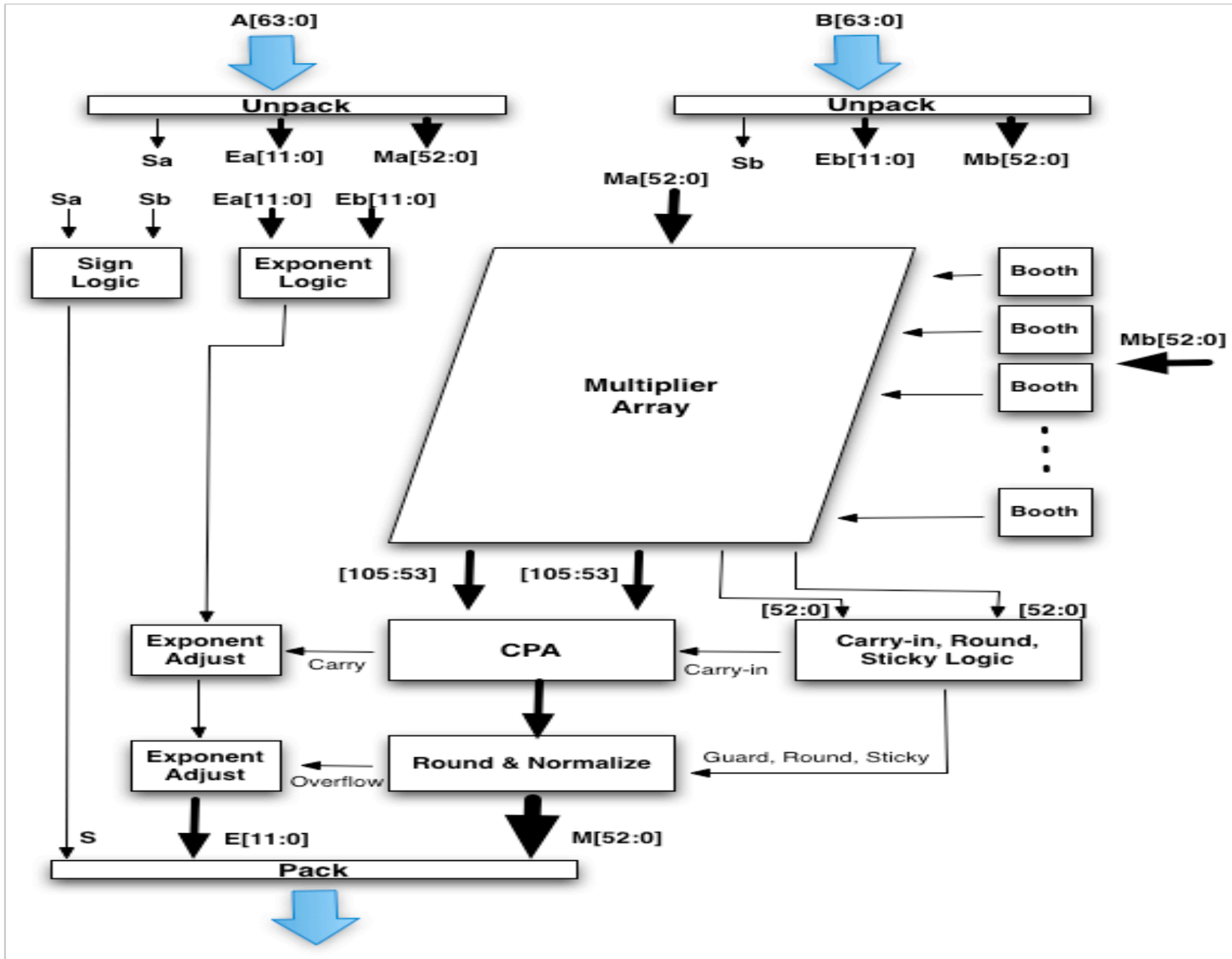
$$(-1)^s(0.SIG) \times 2^{-1022}$$

  - ❖ Special numbers: not-a-number (NaN), infinity (exponent is all ones)
  - ❖ Signed zeros



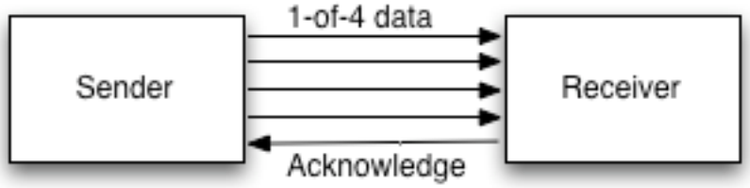
Datapath for floating-point multiplication



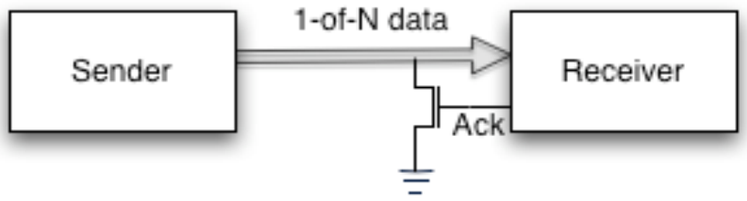


Multiplier core

- Multiplier core bits are synchronized in time
- Opportunity to use more aggressive circuit styles rather than QDI
  - ❖ Keep large timing margin in any timing assumptions
  - ❖ Study using a small 8x8 multiplier core
  - ❖ Internal protocol: single track

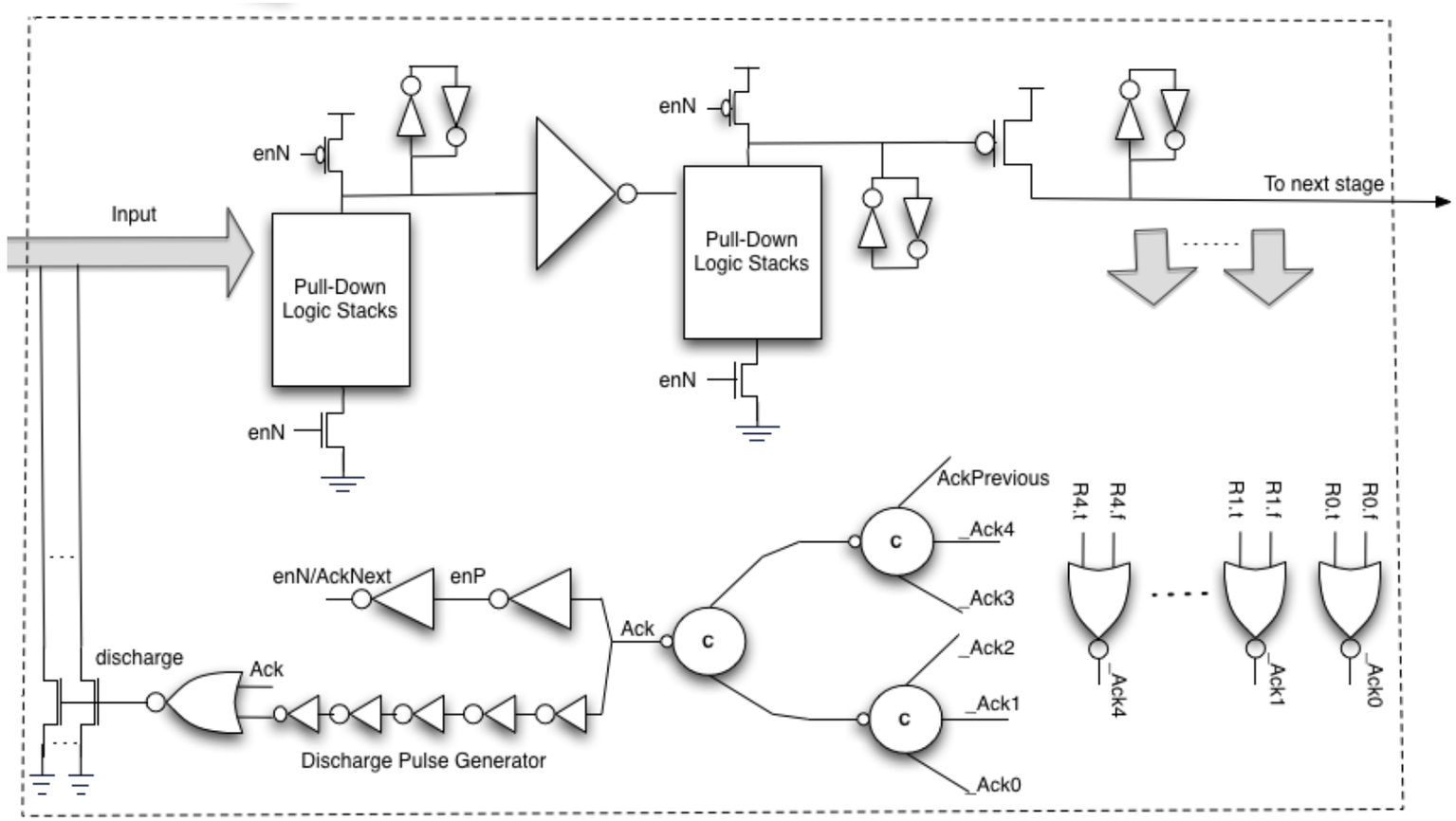


Four-Phase Handshake Protocol



Single-Track Handshake Protocol

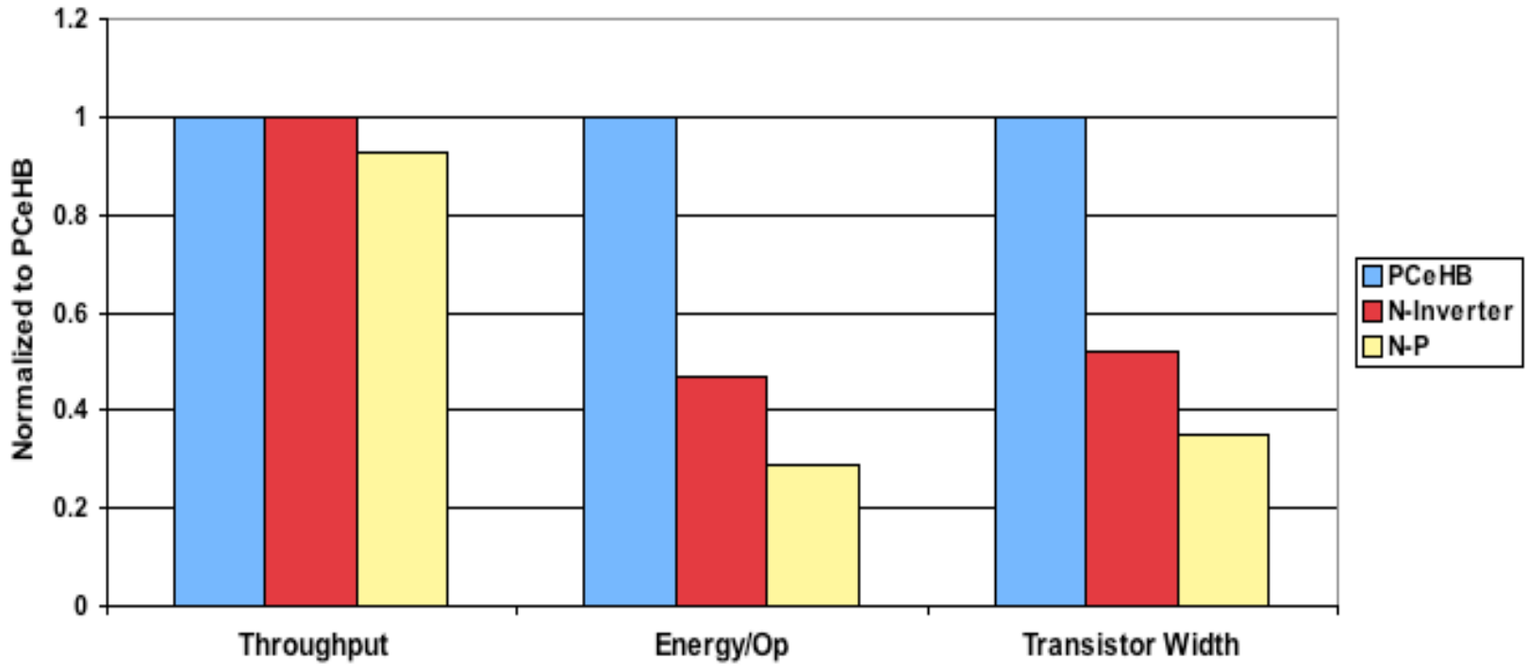
Circuit style



- Basic idea
  - ❖ Multi-stage domino, partially weak-conditioned
  - ❖ Parallel precharge (timing assumption)
  - ❖ Single-track signaling

*For more details: B. Sheikh, R. Manohar. "Energy-efficient pipeline templates for high-performance asynchronous circuits." ACM JETC, special issue on asynchrony in system design, **7**(4), December 2011.*

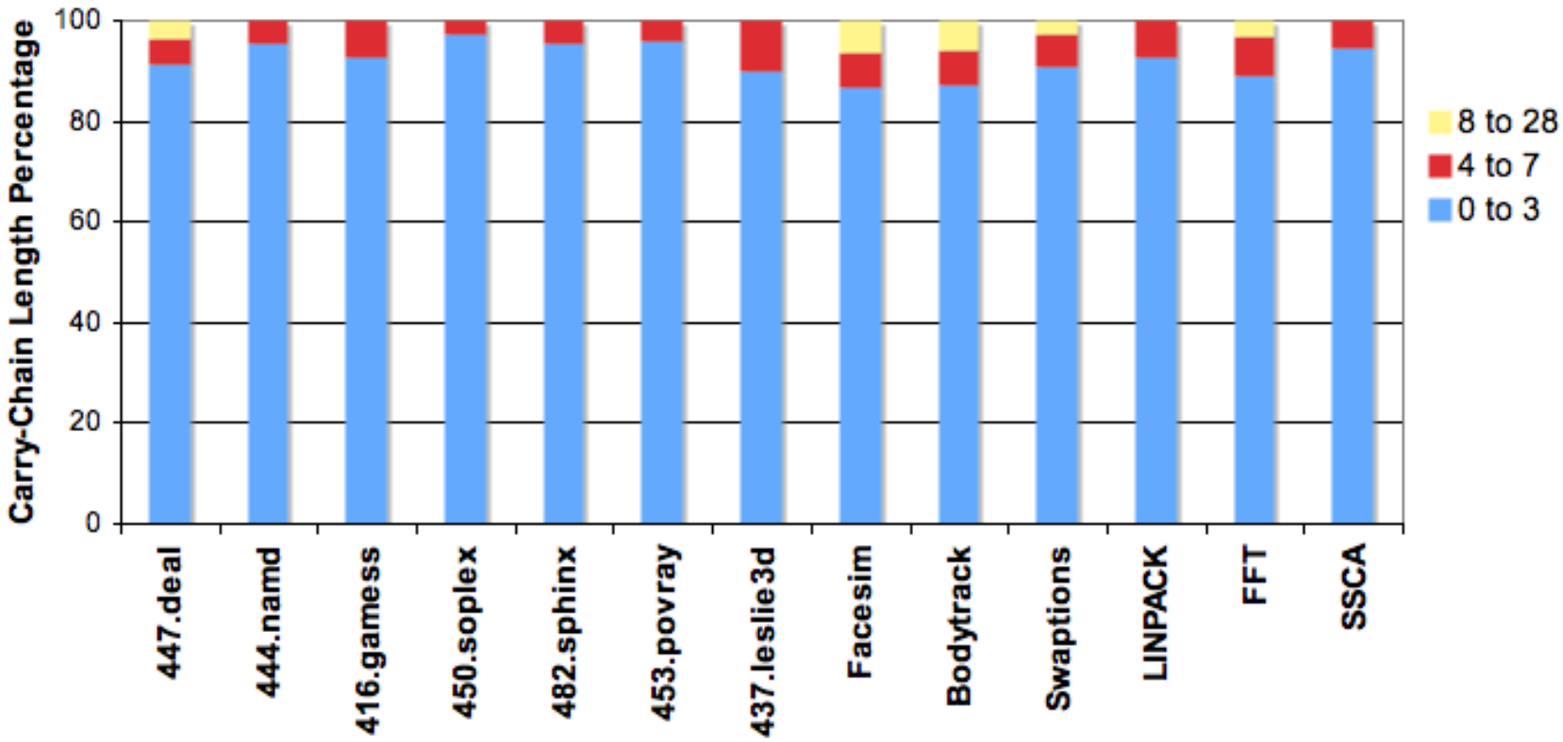
Comparison to QDI stage



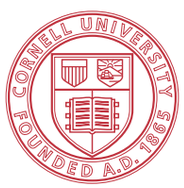
- Two styles
  - ❖ Logic followed by inverter (N-inverter)
  - ❖ Logic followed by logic (N-P)
- Reduction in energy and area while preserving most of the throughput
  - ❖ Tightest timing requirement: 7 FO4 v/s 2.5 FO4

Multiplier core

- Radix 4 Booth v/s Radix 8
  - ❖ Radix 4: 0, ±Y, ±2Y
  - ❖ Radix 8: 0, ±Y, ±2Y, ±3Y, ±4Y
- Often, the cost of the 3Y calculation out-weighs benefits



Carry-chain length statistics (radix 4 ripple)

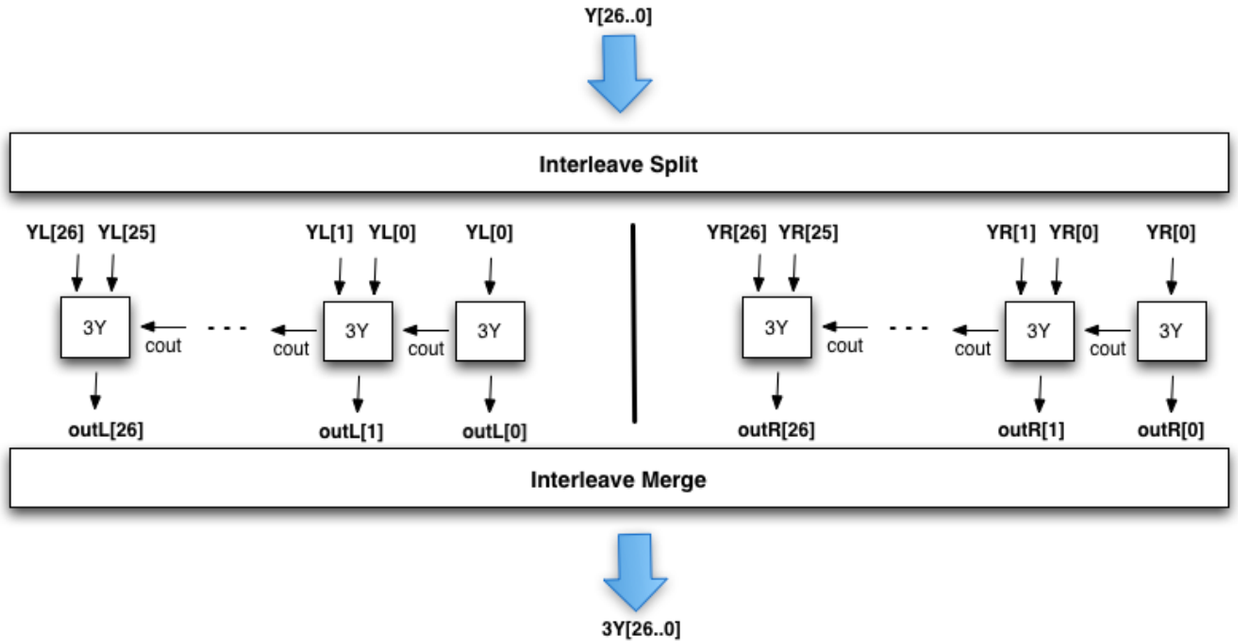


Cornell University

AVLSI

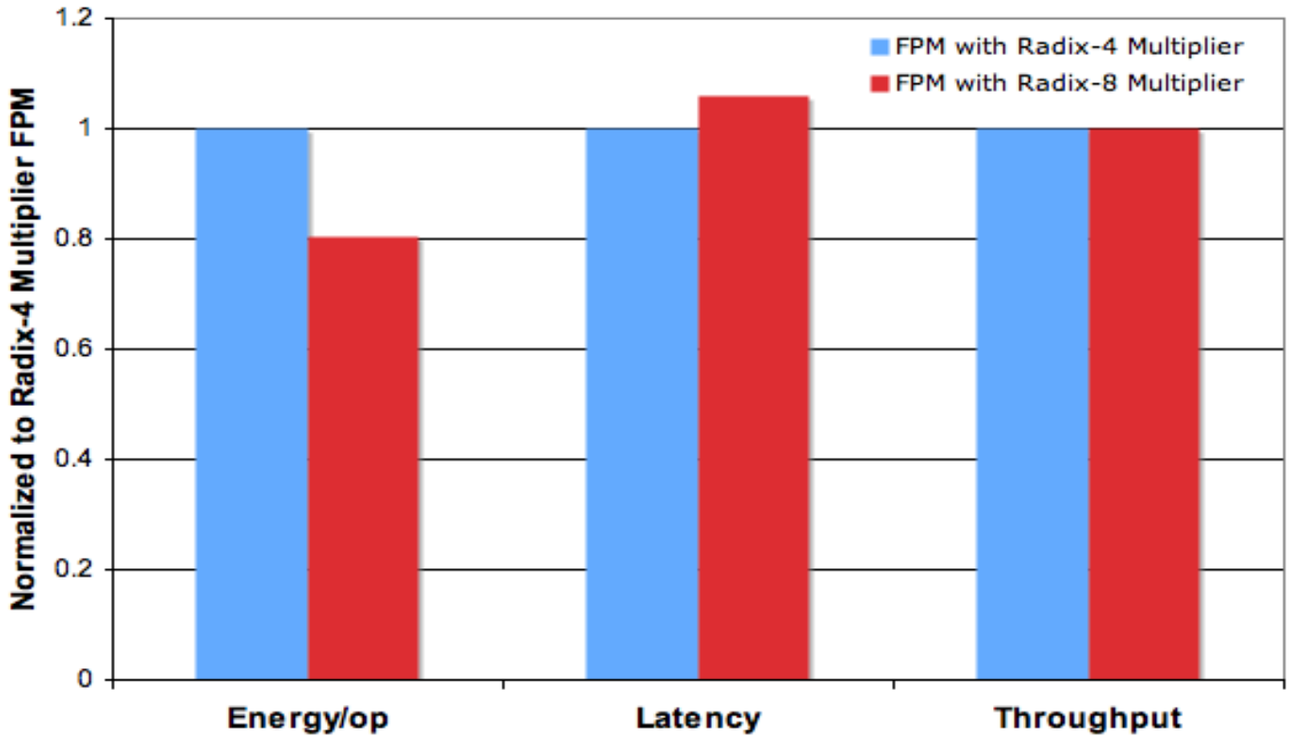
3Y adder

- Ripple carry adder with interleaving hides most stalls



- Much cheaper in energy compared to a full high-performance adder *(Kogge-Stone + carry select)*
  - ❖ 68.1% lower energy
  - ❖ 8.3% less latency

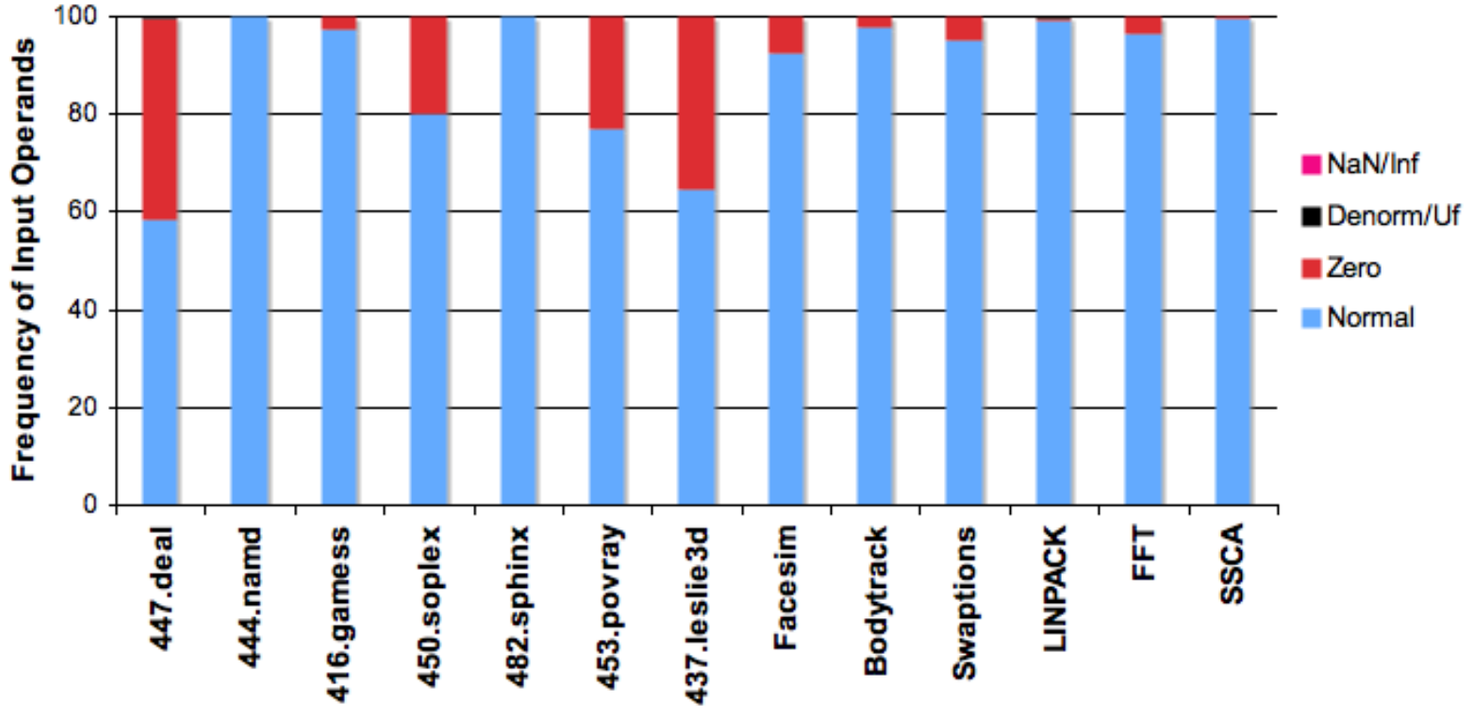
Multiplier core



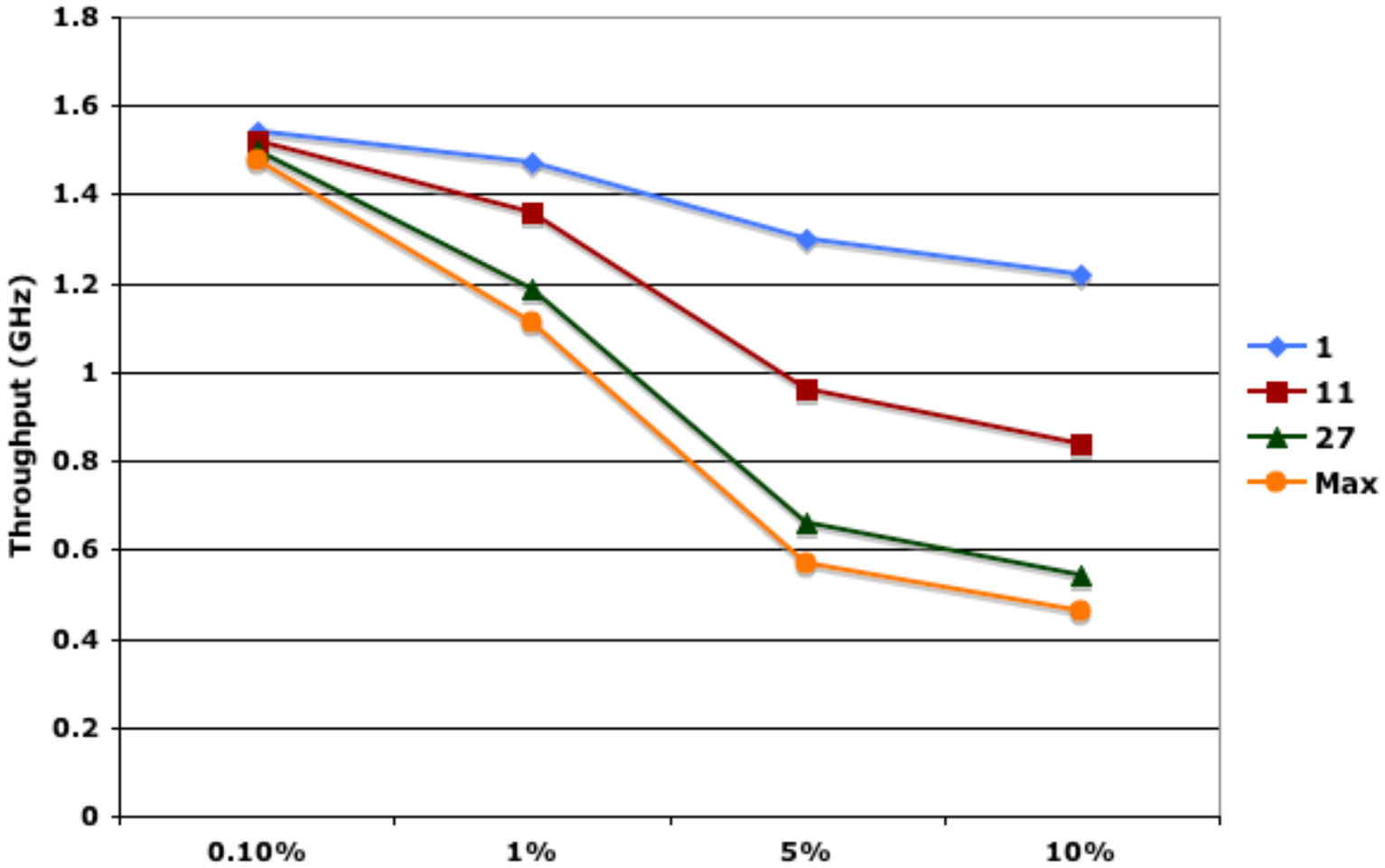
- 3 : 2 compressor trees
- Latency: higher (≈ +6%) due to 3Y adder
- Energy: lower (≈ -20%) due to fewer partial product bits

Denormal arithmetic

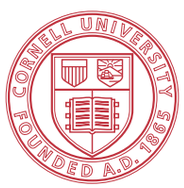
- Two scenarios
  - ❖ Inputs are denormal
  - ❖ Inputs are small, and result is denormal (“underflow”)
- Separate datapath to handle these cases
  - ❖ Slow, iterative shifter
  - ❖ Output of final adder is re-directed to either the normal datapath or denormal datapath



Denormal arithmetic penalty

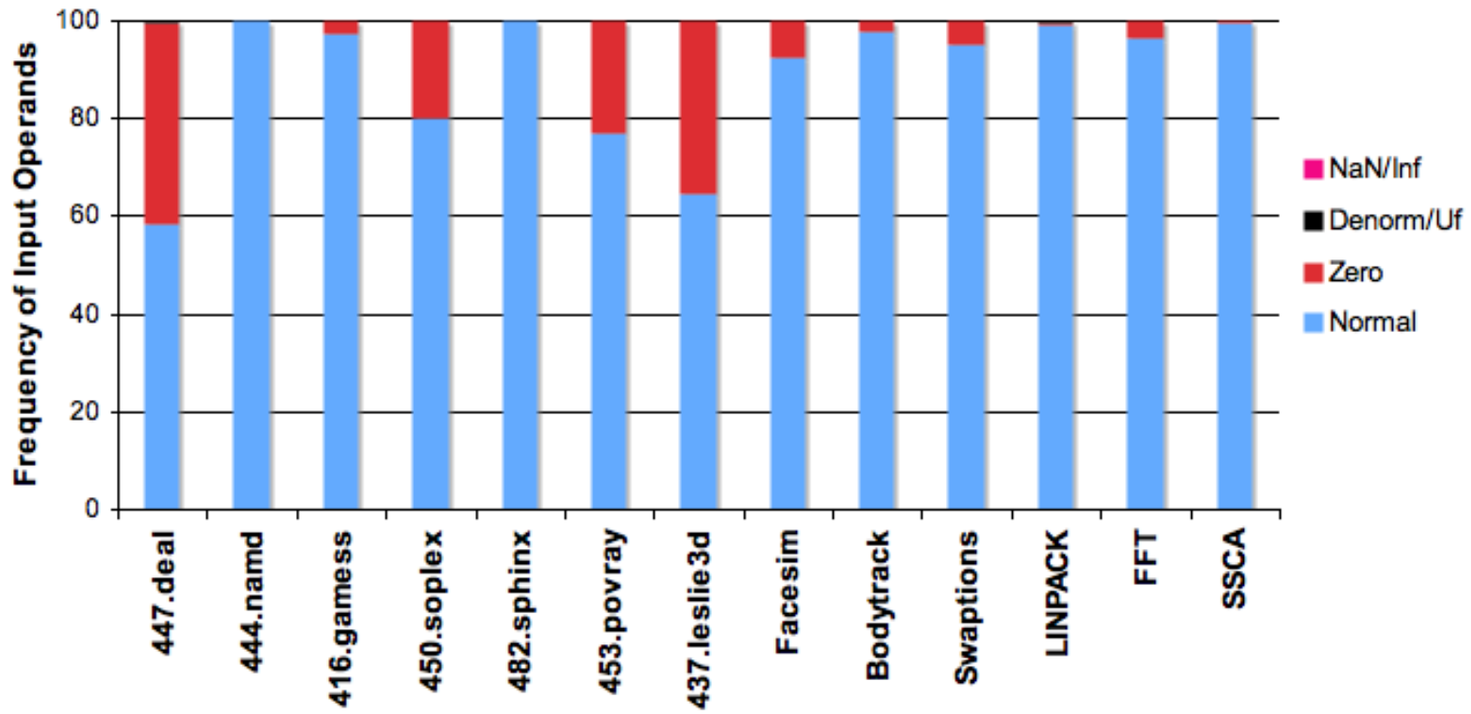


- Throughput drop is negligible for benchmarks



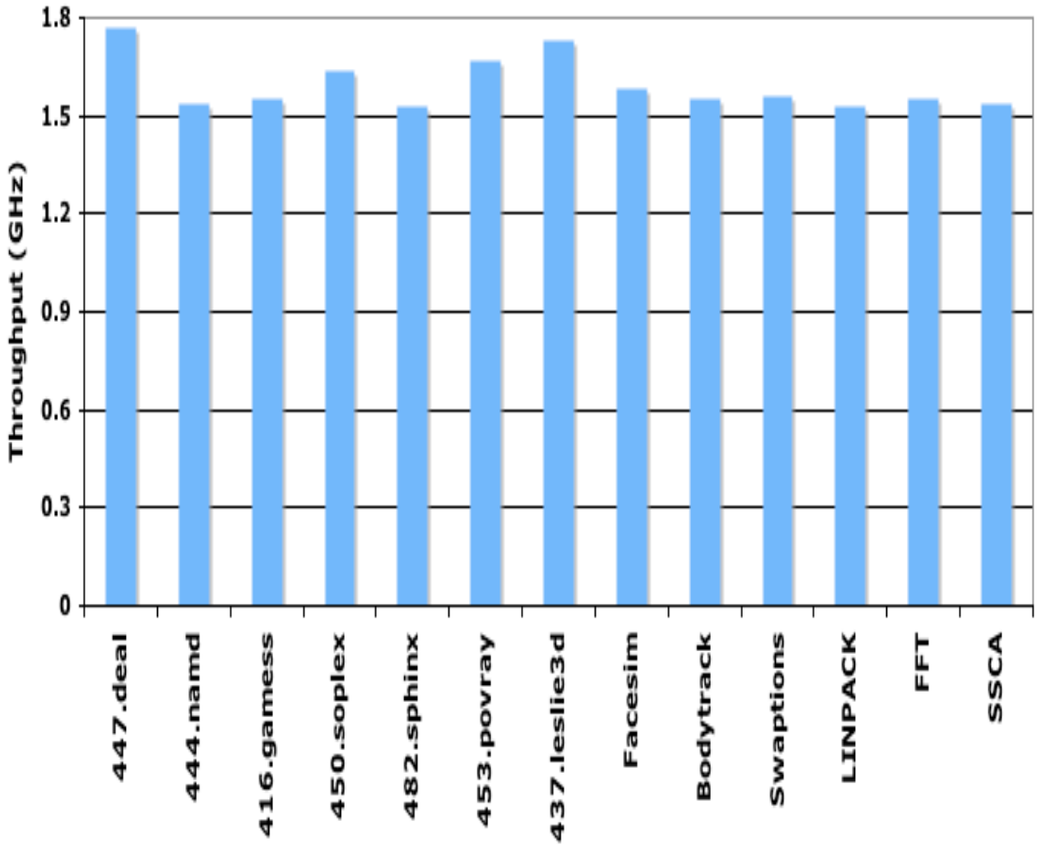
Zero-bypass datapath

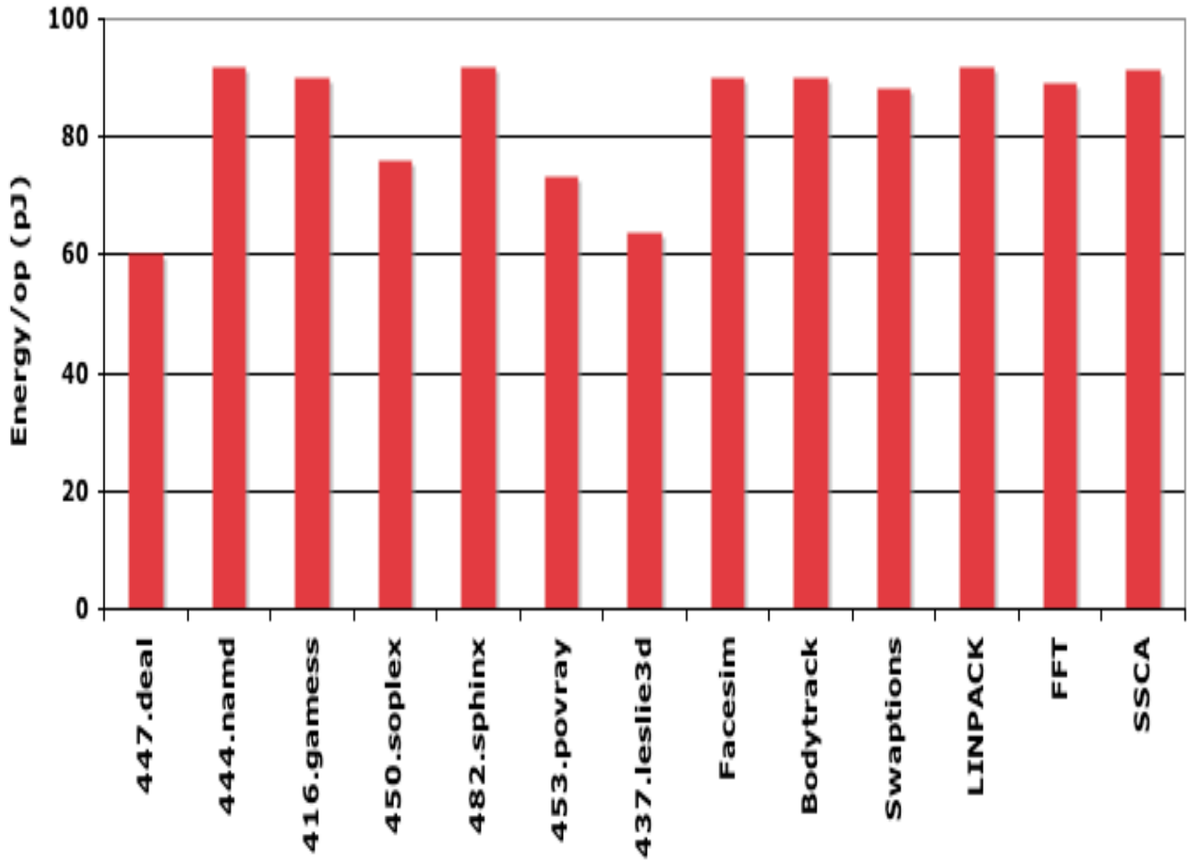
- For some applications, a non-trivial number of operands are zero
- Example: matrices with a small number of non-zero elements
  - ❖ Efficient sparse matrix codes use “mostly dense” sub-matrices
- Can avoid most of the energy required



Results

- 65nm process (TT, 1V)
  - ❖ 92.1 pJ/op, 1.5 GHz (leakage: 1.6mW @ 90C)





- Synchronous FPM by Quinnell (65nm SOI, 1.2V)
  - ❖ 280 pJ/op, 0.67 GHz, 701ps latency @ 1.3V

For reference: Synopsys Designware FPM: 9.5x higher latency

Summary

- We have designed a double-precision floating-point multiplier
- Techniques used to reduce energy
  - ❖ Radix 8 array with simpler 3Y adder
  - ❖ Circuits modified to reduce handshake energy
  - ❖ Slow denormal arithmetic
  - ❖ Zero bypass
- Future work
  - ❖ Fused multiply-add
  - ❖ New techniques to reduce multiplier energy?
- Thanks to NSF

The person that did the work!