Using Field-Programmable Gate Arrays to Explore Different Numerical Representations

A Use-Case on POSITs

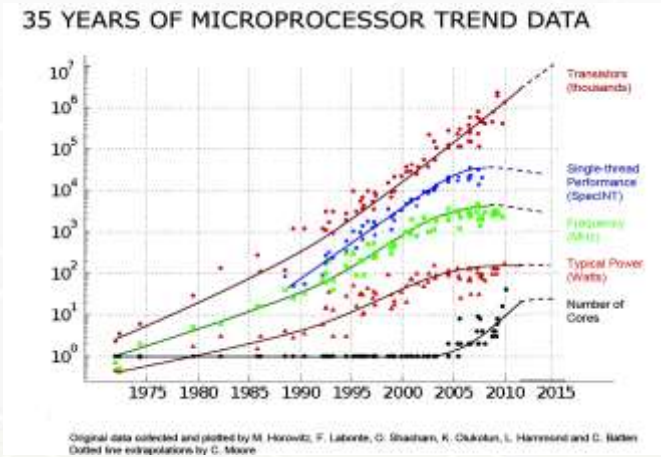
Artur Podobas (artur.podobas@riken.jp)
Processor Research Team

Overlook

- Introduction
- Reconfigurable Architectures
- POSITs
- POSITs in Hardware
- Using Custom Representations
- Application Performance
- Conclusion

Introduction

- **Moore's law ending**
  - Silicon becoming more precious
  - Challenge conventional wisdom
- **Most application memory-bound[1]**
  - Unlikely to change
- **Numerical representation candidate for optimization**
  - Reduce size (# bits) in representation
  - Improve accuracy
- **Smaller representation yields:**
  - Smaller functional units
    - More OPs/mm^2
    - More OPs/Watt
  - More values per unit bandwidth
    - A remedy for memory boundness?[2]



1 "Double-precision FPUs in High-Performance Computing: an Embarrassment of Riches?", IPDPS 2019

2 "FiNN: a Framework for Fast, Scalable Binarized Neural Network Inference", FPGA 2017

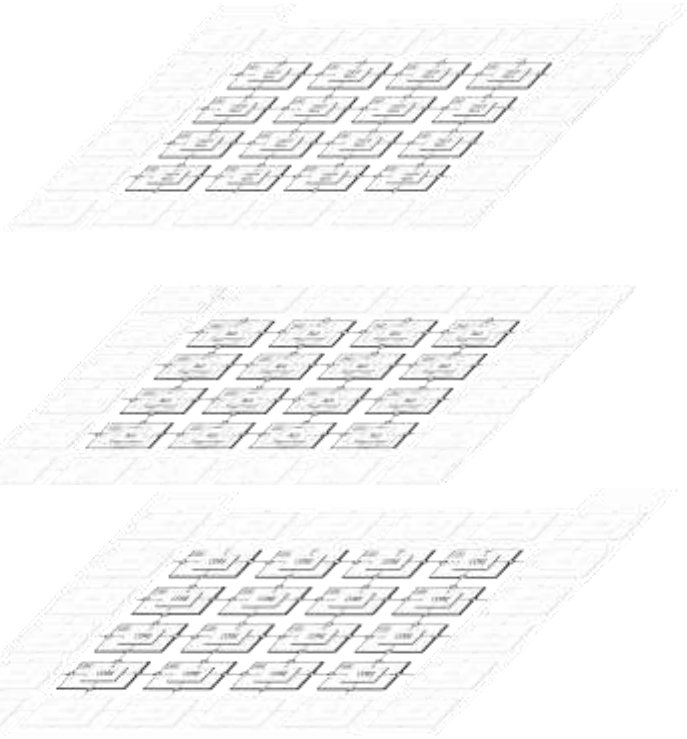
Introduction (cont.)

- **But can we use a smaller representation?**
  - … and where?
- **Insight requires experimental testing**
  - Small use-cases -> Software emulation OK!
  - Large use-cases -> Software emulation slow!
- **Testing on large use-cases requires performance**
  - Option 1: Build an ASIC
    - Expensive (at least right now)
  - Option 2: *Use Reconfigurable Architectures*

Reconfigurable Architectures

Reconfigurable Architectures

- **Reconfigurable Architectures provide means to (partially) counter the end of Moore**
- **Three types of Reconfigurable Archs:**
  - Fine-Grained architectures
    - e.g. FPGAs
  - Coarse-Grained Reconfigurable Architectures
    - e.g. ADRES, GARP
  - Coarse-Grained Reconfigurable Processors
    - e.g. RAW, Xilinx Versile
- **Fine-Grained preferable for architecture exploration**
  - Full control over flip-flops, wires, etc.



Reconfigurable Architectures (cont.)

- **Field-Programmable Gate Arrays**
- **History**
  - Created to help ASIC development
  - Found use in: Military, consumer electr., automotive
  - Today powerful compute devices
    - Capable of rivaling CPUs / GPUs

- **FPGA fabric consists of:**
  - Programmable Logic in the form of Look-Up Tables (LUTs)
  - On-chip Block RAM (BRAM) for storage
  - DSP units for fast multiplication
  - A large reprogrammable interconnect
- **Can be re-programmed to implement exciting new formats**
  - Such as POSITs

POSITs

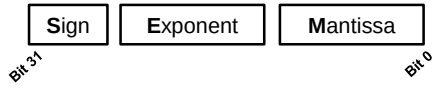
POSITs: History

- **Numbers in computers represented differently**
  - Fixed-point, 2s complement, binary-coded decimal (remember these?)
- **1985: standardization on how real numbers are represented on computers (IEEE-754)**
- **2015: "The End of Error"[1] published by prof John Gustafson**
  - Argues for a new floating-point representation called *UNUM*
  - Interval arithmetics → Complex to make hardware!
  - Variable-length encoding → Complex to make hardware!
- **~2016: UNUM (version 2) incepted**
  - No-longer variable length
  - Based on look-up table transformations → Limited size allowed (LUTs occupy a lot of space)
- **2017: POSITs (Unum version 3) introduced[2]**
  - Simplified→ Feasible to create hardware!
  - In-place replacement for IEEE-754

1 "The End of Error: Unum Computing", J. Gustafson, 2015

2 "Beating Floating Point at its own Game: Posit Arithmetic", J. Gustafson, I. Yonemoto SuperFri.org, 2017

POSITs: Difference to IEEE-754



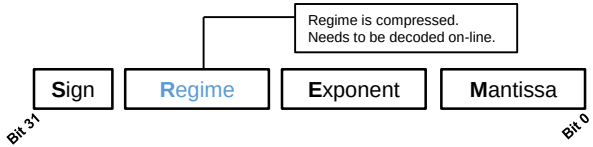
$$(-1)^{S} * 2^{(E-bias)} * 1.M$$

Pros

- Simple, Mature
- Standardized

Cons

- Many NaN and Zero representations (16,777,216 NaN in single-precision)



$$(-1)^{S} * (2^{2^{e}})^{R} * 2^{E} * 1.M$$

Pros

- Different spread of accuracy
- One zero and one NaR representation

Cons

- Parts of the format is encoded (needs decoding)

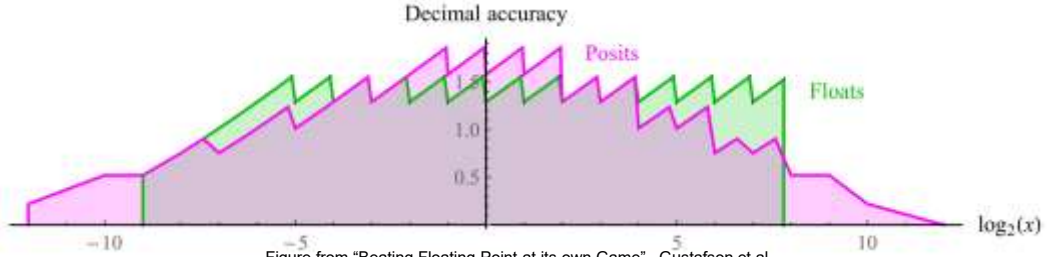
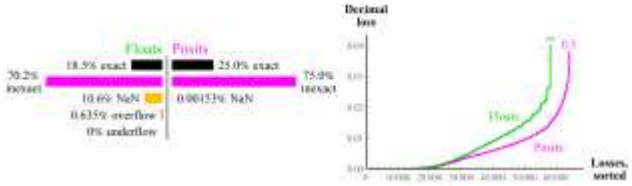


Figure from “Beating Floating Point at its own Game”, Gustafson et al.

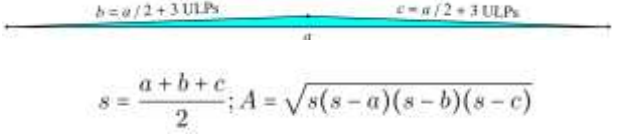
POSITs: Benefits

Computer simulations create the future

Arithmetic functions (e.g. Addition)

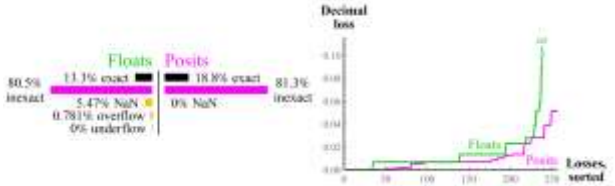


Mathematical problems (e.g. Goldberg's Thin Triangle)



$$s = \frac{a+b+c}{2}; A = \sqrt{s(s-a)(s-b)(s-c)}$$

Complex functions (e.g. Reciprocal)



Larger Real-life Problems

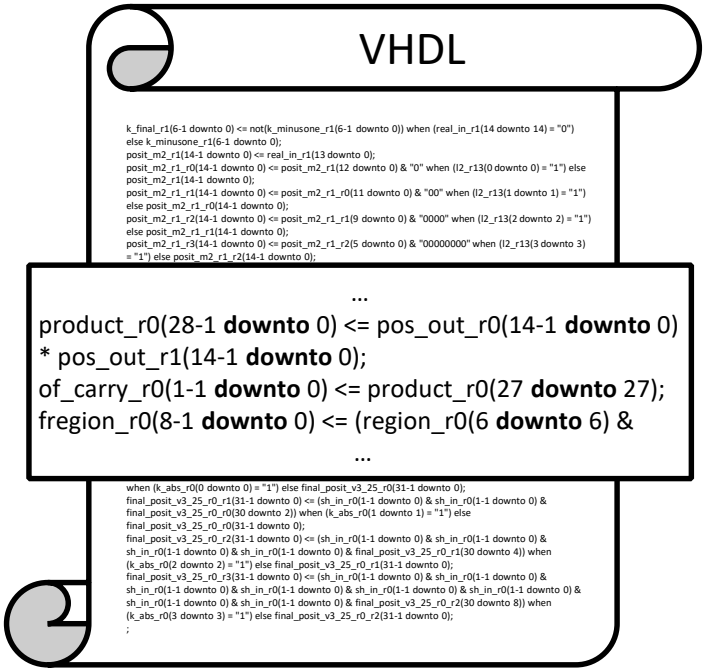
?

Figures from "Beating Floating Point at its own Game", Gustafson et al.

POSITs in Hardware

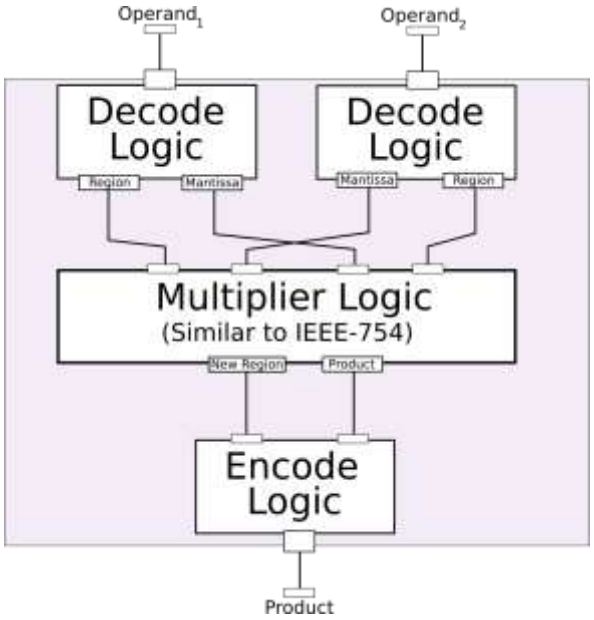
POSITs in Hardware

- **Hardware Description Languages describe FPGA execution**
  - e.g. VHDL or Verilog
    - Domain specific languages for digital design
- **We created a prototype POSIT generator**
  - Generate arbitrary sized POSIT units
    - E.g. 16-bit, 32-bit, or even 51-bit POSIT units
  - Focus on:
    - Addition/Subtraction
    - Multiplication
    - Comparisons
- **How does POSIT map to FPGAs?**
  - …and what did we learn?



POSITs in Hardware: Multiplication

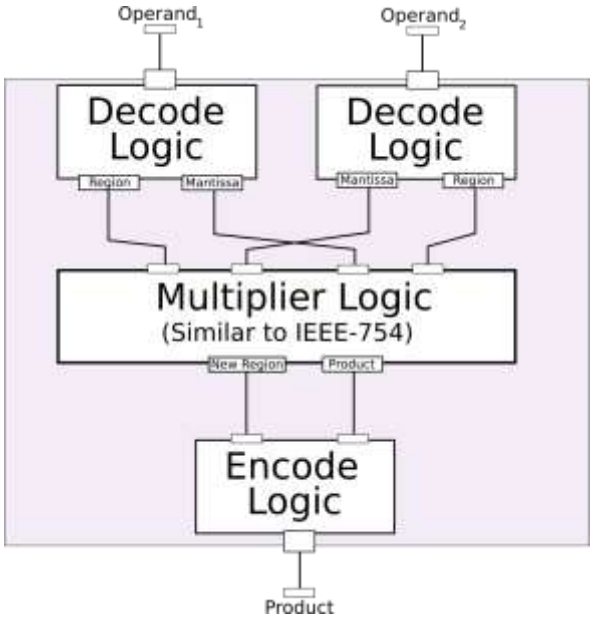
Description		POSIT 16-bit		POSIT 32-bit	
Arithmetic	*Function*	*Logic*	*DSPs*	*Logic*	*DSPs*
	Multiplication	**206**	**1**	**486**	**2**



POSITs in Hardware: Multiplication

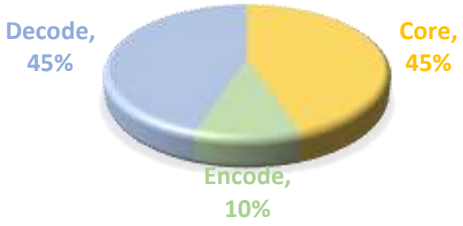
Computer simulations create the future

Description		POSIT 16-bit		POSIT 32-bit	
Arithmetic	*Function*	*Logic*	*DSPs*	*Logic*	*DSPs*
	Multiplication	**206**	**1**	**486**	**2**



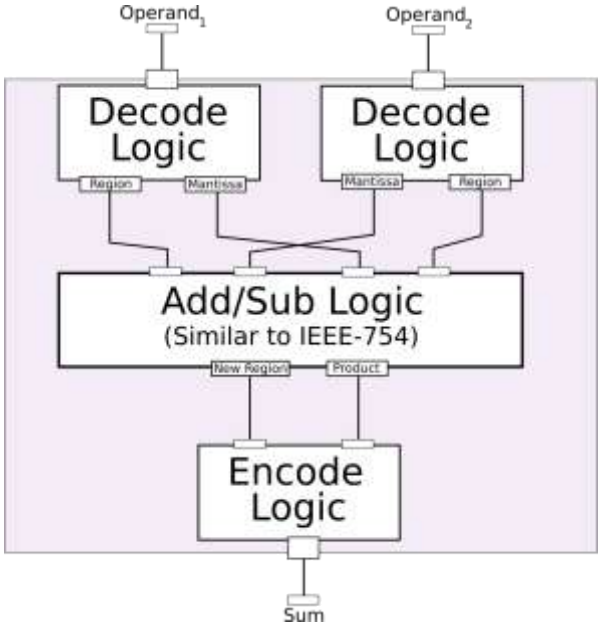
POSIT 16-BIT

POSIT 32-BIT



POSITs in Hardware: Addition/Subtraction

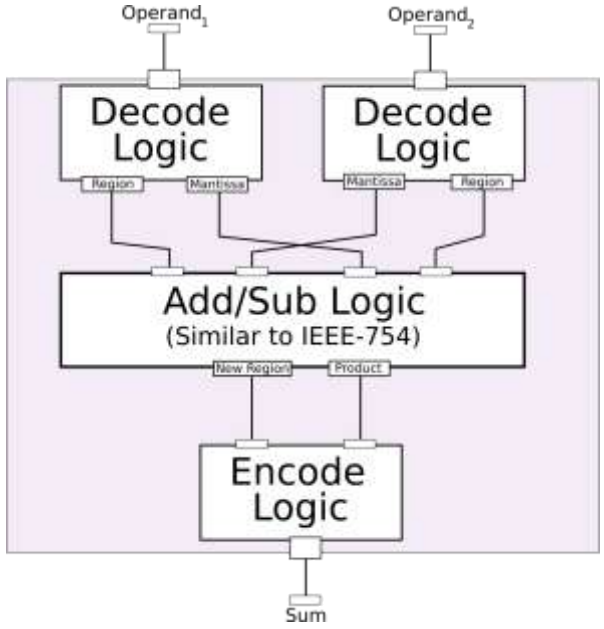
Description		POSIT 16-bit		POSIT 32-bit	
Arithmetic	*Function*	*Logic*	*DSPs*	*Logic*	*DSPs*
	Multiplication	**206**	**1**	**486**	**2**
	Add/Sub	**300**	**-**	**657**	**-**

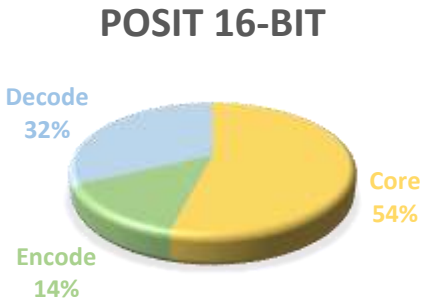


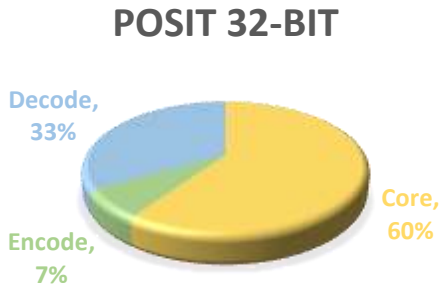
POSITs in Hardware: Addition/Subtraction

Computer simulations create the future

Description		POSIT 16-bit		POSIT 32-bit	
Arithmetic	*Function*	*Logic*	*DSPs*	*Logic*	*DSPs*
	Multiplication	**206**	**1**	**486**	**2**
	Add/Sub	**300**	**-**	**657**	**-**



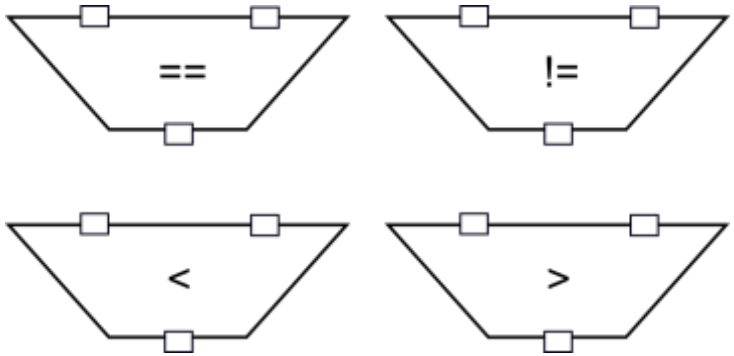




POSITs in Hardware: Comparison and Others

Computer simulations create the future

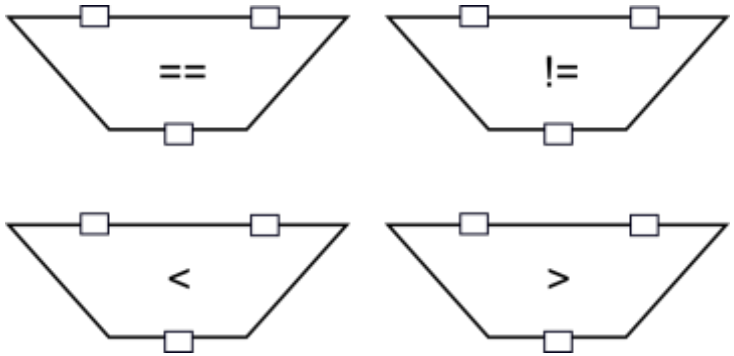
Description		POSIT 16-bit		POSIT 32-bit	
Arithmetic	*Function*	*Logic*	*DSPs*	*Logic*	*DSPs*
	Multiplication	**206**	**1**	**486**	**2**
	Add/Sub	**300**	**-**	**657**	**-**
Comparison	**OP_1 == OP_2**	**7**	**-**	**14**	**-**
	OP_1 != OP_2	**7**	**-**	**14**	**-**
	OP_1 < OP_2	**12**	**-**	**32**	**-**
	OP_1 > OP_2	**12**	**-**	**33**	**-**



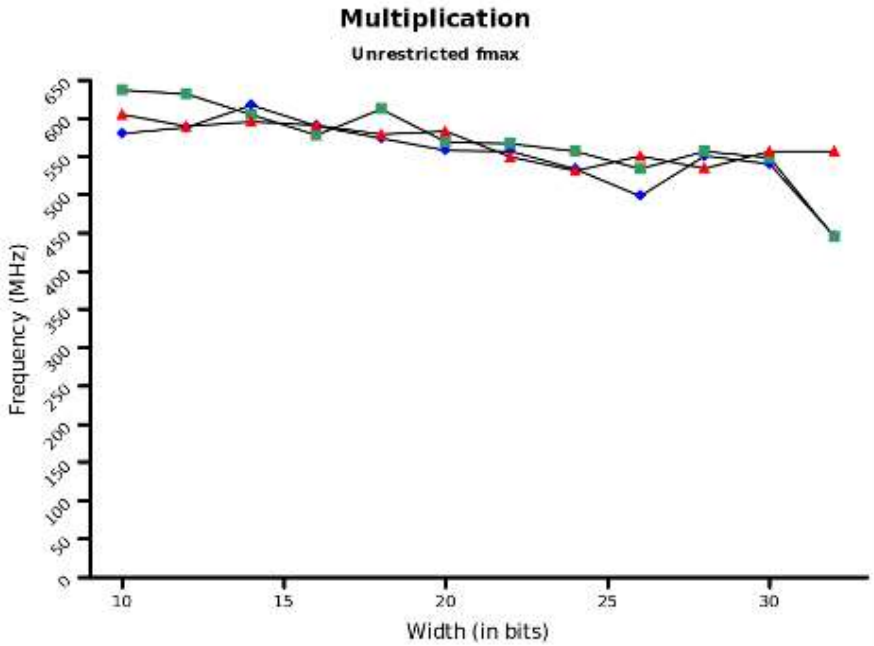
POSITs in Hardware: Comparison and Others

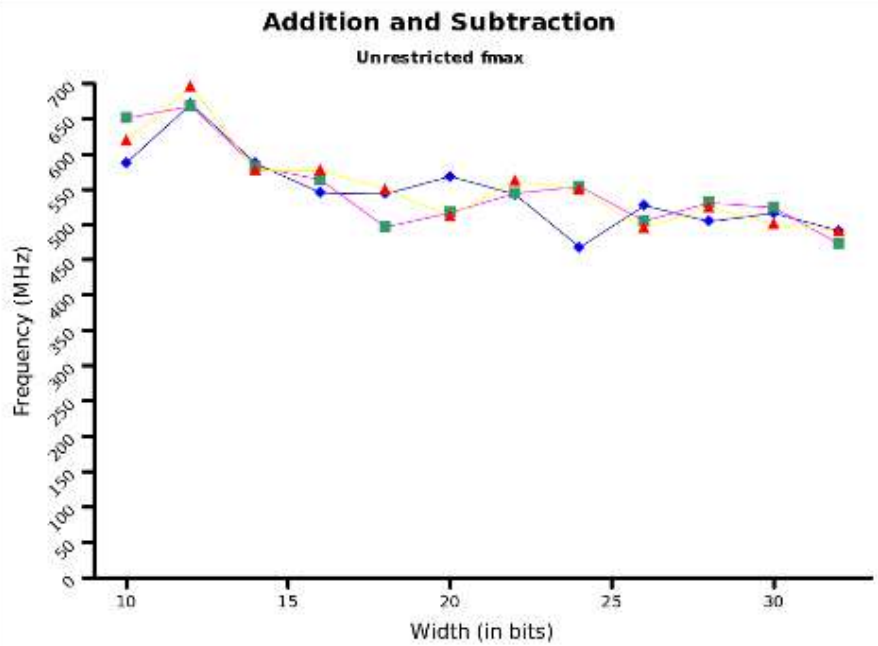
Computer simulations create the future

Description		POSIT 16-bit		POSIT 32-bit	
Arithmetic	*Function*	*Logic*	*DSPs*	*Logic*	*DSPs*
	Multiplication	**206**	**1**	**486**	**2**
	Add/Sub	**300**	**-**	**657**	**-**
Comparison	**OP_1 == OP_2**	**7**	**-**	**14**	**-**
	OP_1 != OP_2	**7**	**-**	**14**	**-**
	OP_1 < OP_2	**12**	**-**	**32**	**-**
	OP_1 > OP_2	**12**	**-**	**33**	**-**
Other	**abs(OP_1)**	**17**	**-**	**33**	**-**
	max(0,OP_1)	**8**	**-**	**16**	**-**



POSITs in Hardware: Isolated Performance



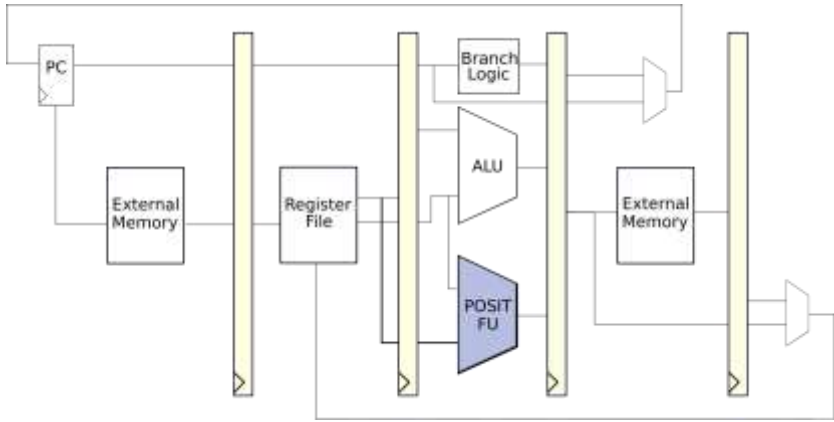


References:

[1] *"Hardware Implementation of POSITs and Their Application in FPGAs", Podobas et al., RAW-18*

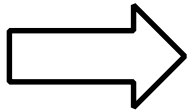
Using Custom Representations

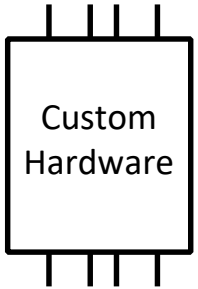
- **1) Replace Floating-Point Unit (FPU) in Soft-Core**
  - **Pros:** Access through ISA , Access to software
  - **Cons:** Change the compilation infrastructure
- **2) Build a CGRA Overlay**
  - **Pros:** Can run very fast, Low reconfiguration overhead
  - **Cons:** Poor existing compilation infrastructure
- **3) Use High-Level Synthesis**
  - **Pros:** Runs fast, Write C/C++ code
  - **Cons:** Manually insert POSIT calls, Long Compilation times



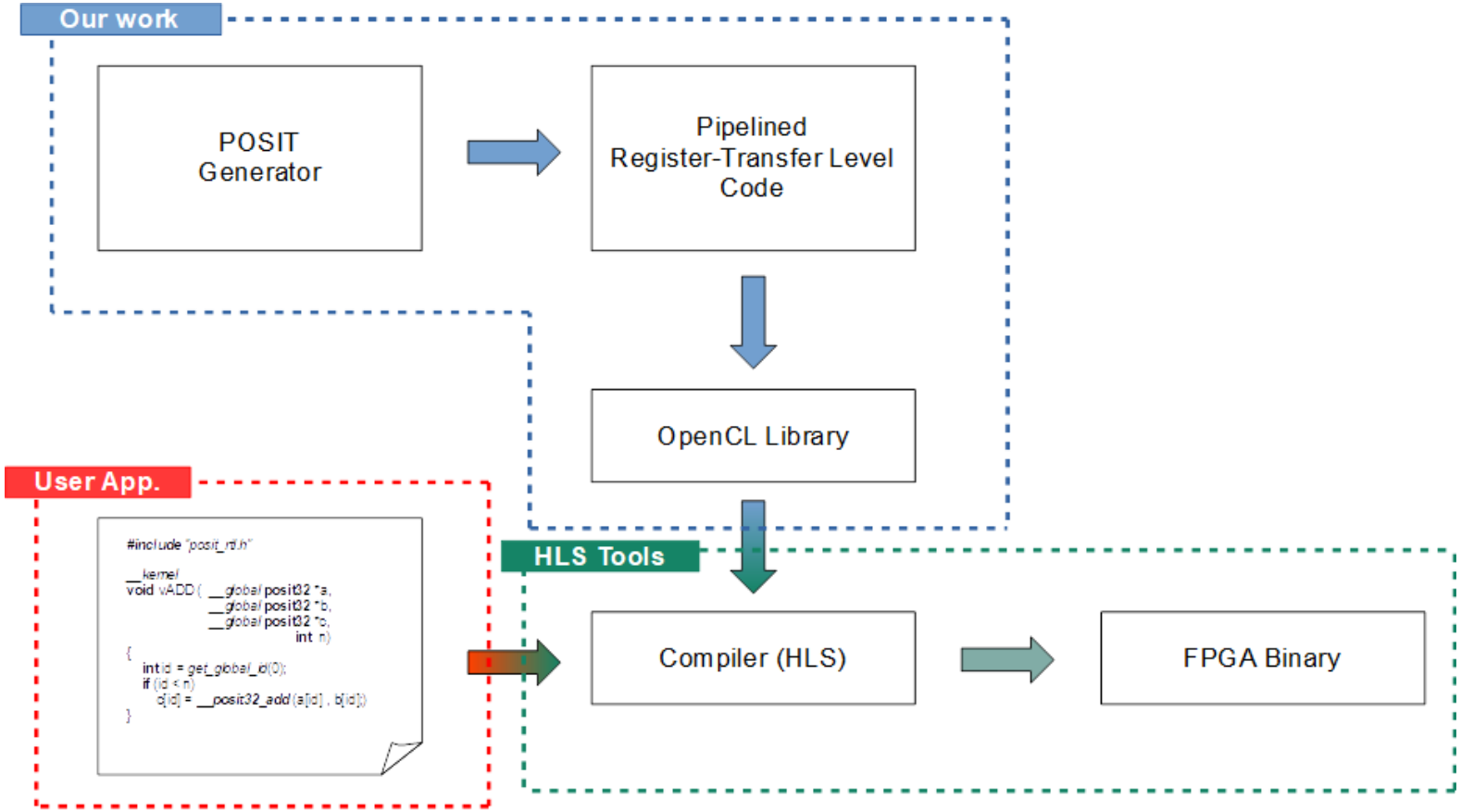
C/C++/OpenCL

```
__kernel
void vADD ( __global float *a,
            __global float *b,
            __global float *c,
                     int  n)
{
   int id = get_global_id(0);
   if (id < n)
      c[id] = a[id] + b[id];
}
```





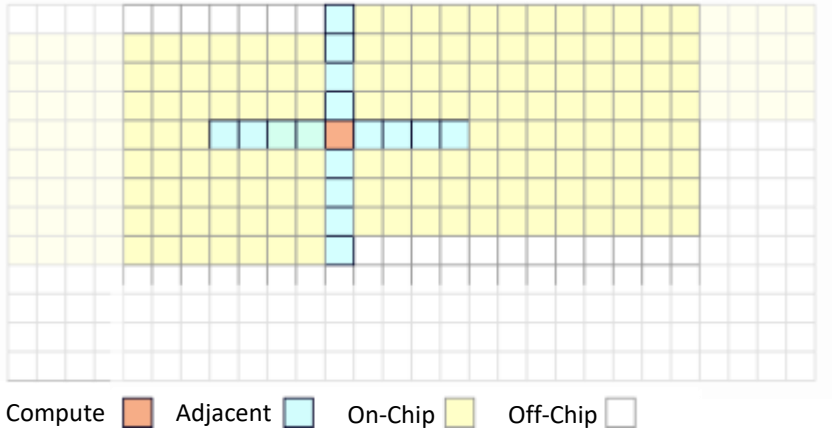
Using Custom Representations



Application Performance

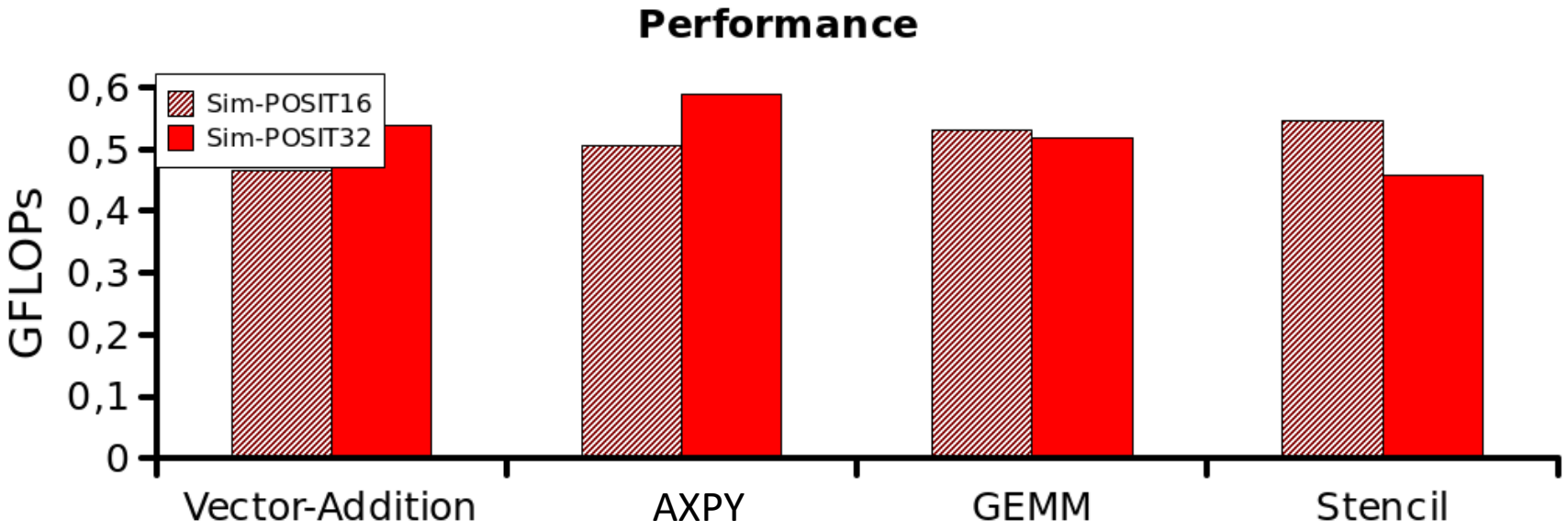
Experimental Setup

- **FPGAs used:**
  - Stratix V FPGA (2010)
  - Arria 10 FPGA (2014)
  - Intel OpenCL 18.0
- **Software Emulation Library**
  - https://gitlab.com/cerlane/SoftPosit
  - Host CPU: i7-8700 @ 3.2 GHz w/ 12 threads
  - All examples parallelized using OpenMP
- **Applications:**
  - Vector-Addition (128 M elements)
  - Vector-Scalar product (?axpy, 128 M elements)
  - Matrix Multiplication (?gemm, 2048x2048)
    - Based on Intel's provided example
  - Stencil computations (rad-4, 8192x256)
    - Shift-registers and Vectorization optimizations



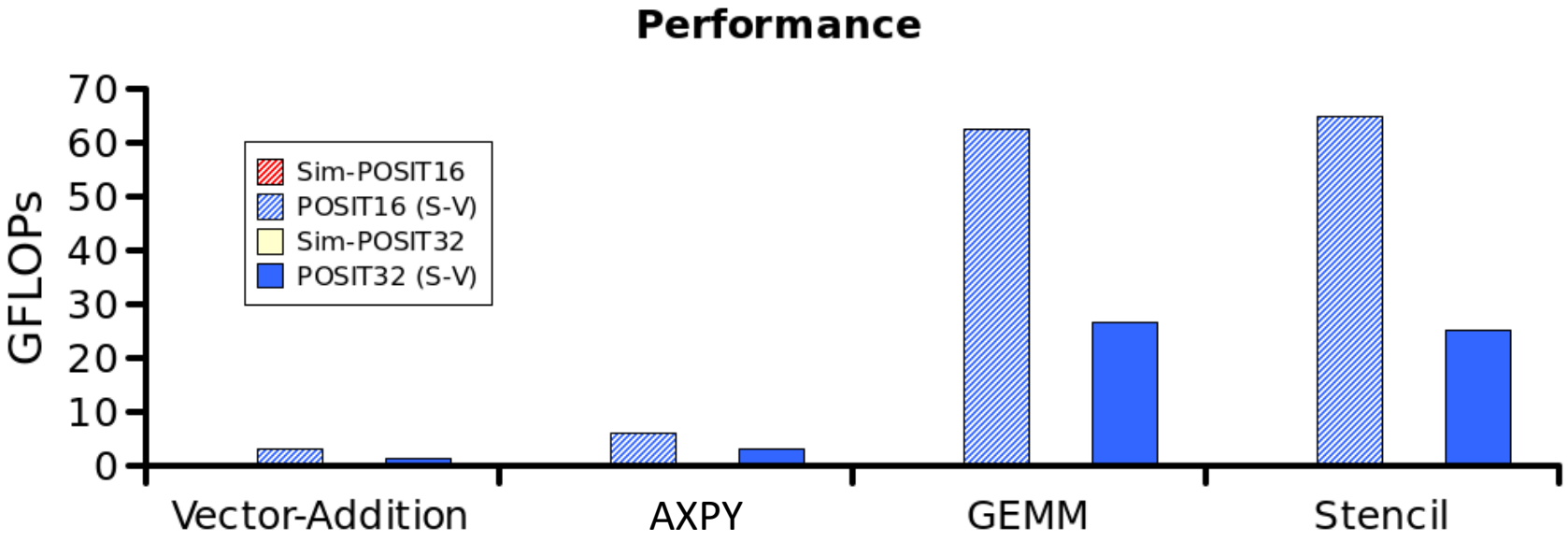
References:
[1] *"Combined Spatial and Temporal Blocking for High-Performance Stencil Computation on FPGAs using OpenCL",* H.R. Zohouri, A. Podobas, S. Matsuoka, FPGA'18
[2] *"High-Performance High-Order Stencil Computation on FPGAs Using OpenCL",* H.R. Zohouri, A. Podobas, S. Matsuoka, RAW'18

Application Performance (cont.)



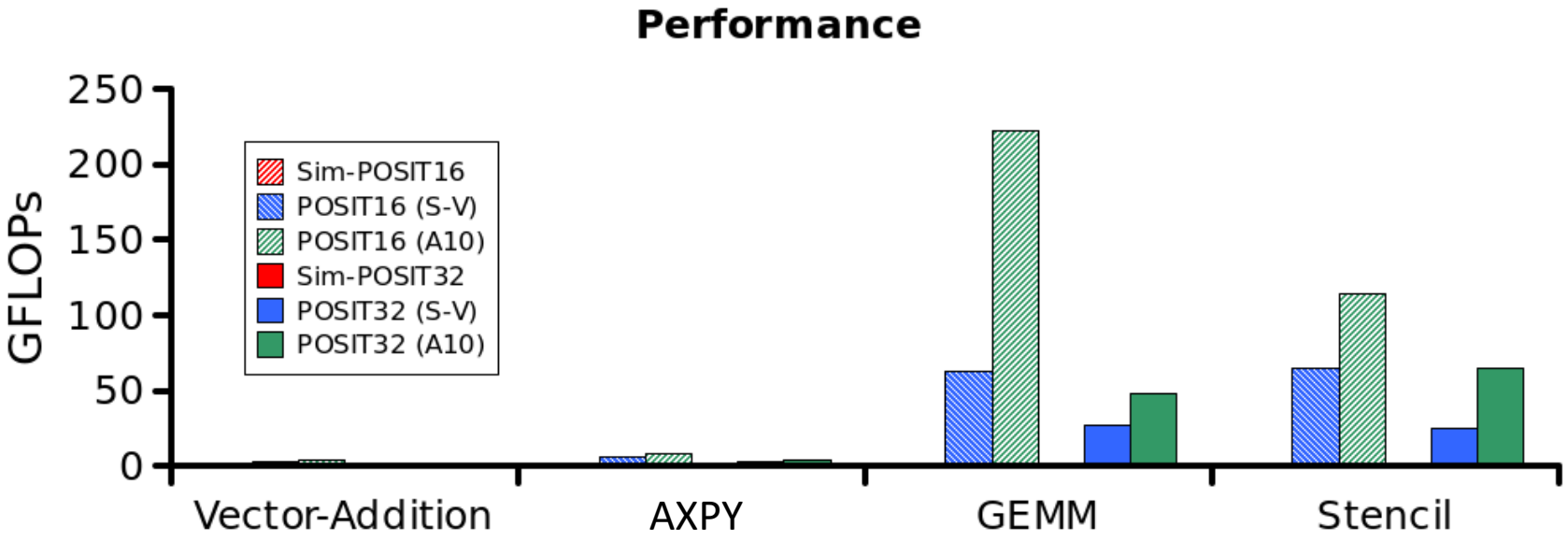
Software Emulation: Compute-bound performance at ~500 MFLOP/s.

Application Performance (cont.)



Stratix V: Up-to 160x performance over software emulation approaches.

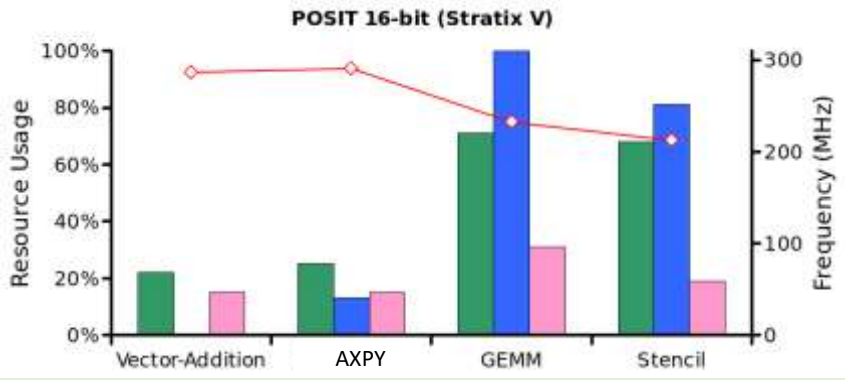
Application Performance (cont.)

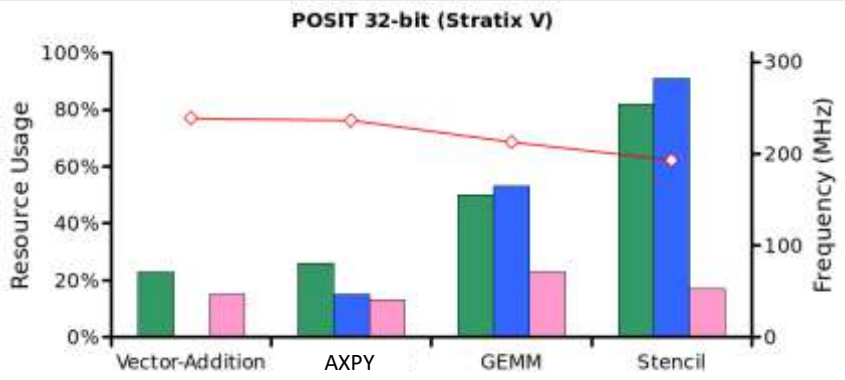


Arria 10: Nearly 500x performance over software emulation.

Resouce Utilization

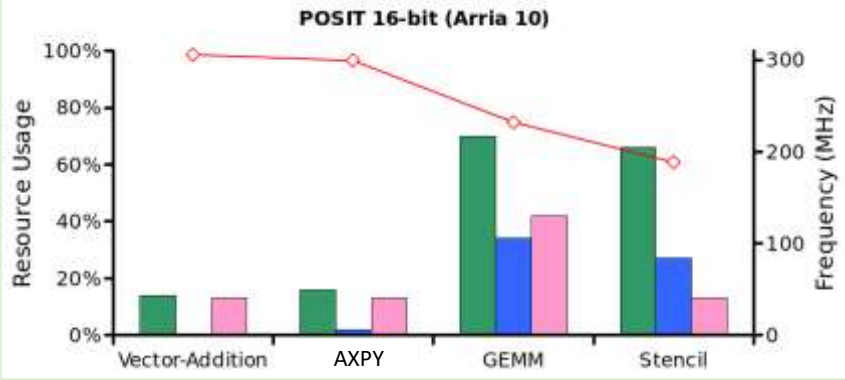
Stratix V

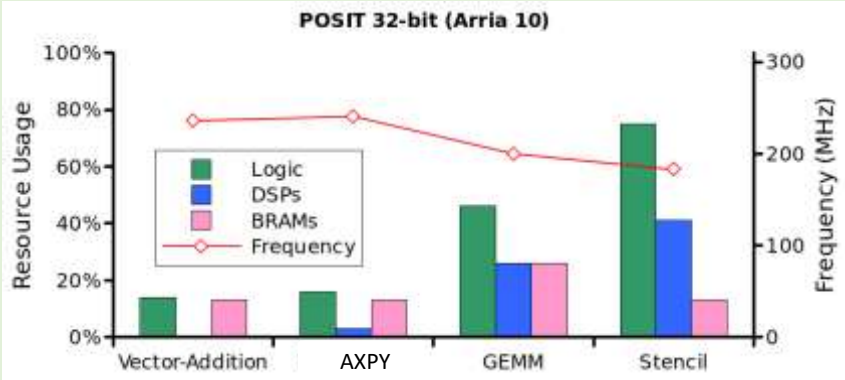




Our designs occupy most of the Stratix V resources.

Arria 10





Arria 10 under-utilized with more room for improvement!

Conclusion

Conclusion

- **Exciting times for architectural research**
  - Old wisdom challenged and (possibly) improved
- **Reconfigurable Architecture can help exploring new concepts**
  - Exploring numerical representations ideal for FPGAs
    - Help understand what is needed before proceeding to ASIC
  - Example shown with POSITs
    - Performance (MHz)
    - Area / Occupancy
  - Demonstrated arguable high-performance
    - 200+ GFLOP/s @ 16-bit POSITs
    - 60+ GFLOP/s @32-bit POSITs
    - (Likely) even higher on modern Xilinx Ultrascale+ or Stratix 10 platforms
- **Thank you for your attention!**