Implications of FPGAs for Floating-Point HPC Systems

LACSI Workshop on Algorithm Acceleration with Reconfigurable Hardware

11 October 2005

Keith D. Underwood

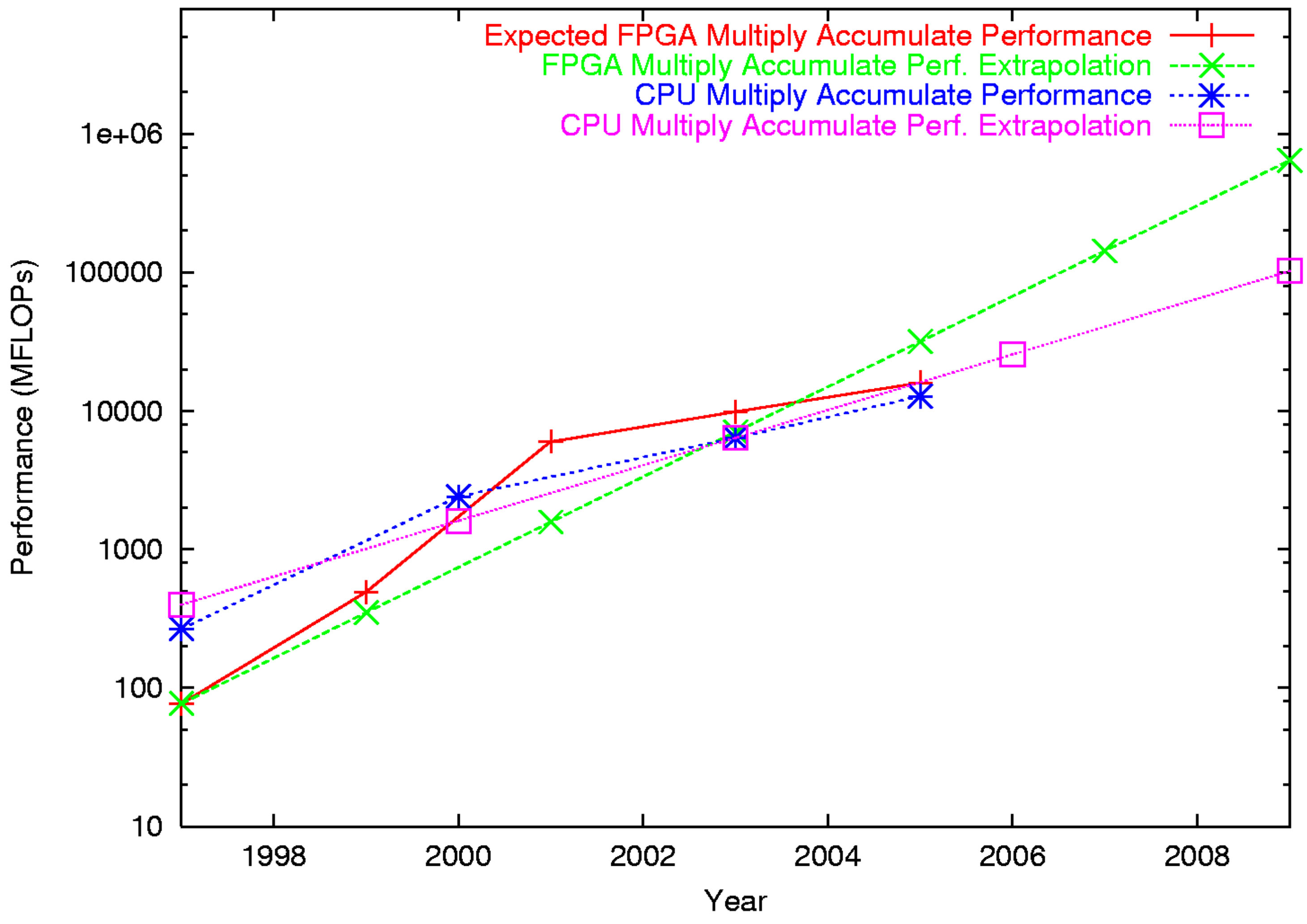
K. Scott Hemmert

Overall Message

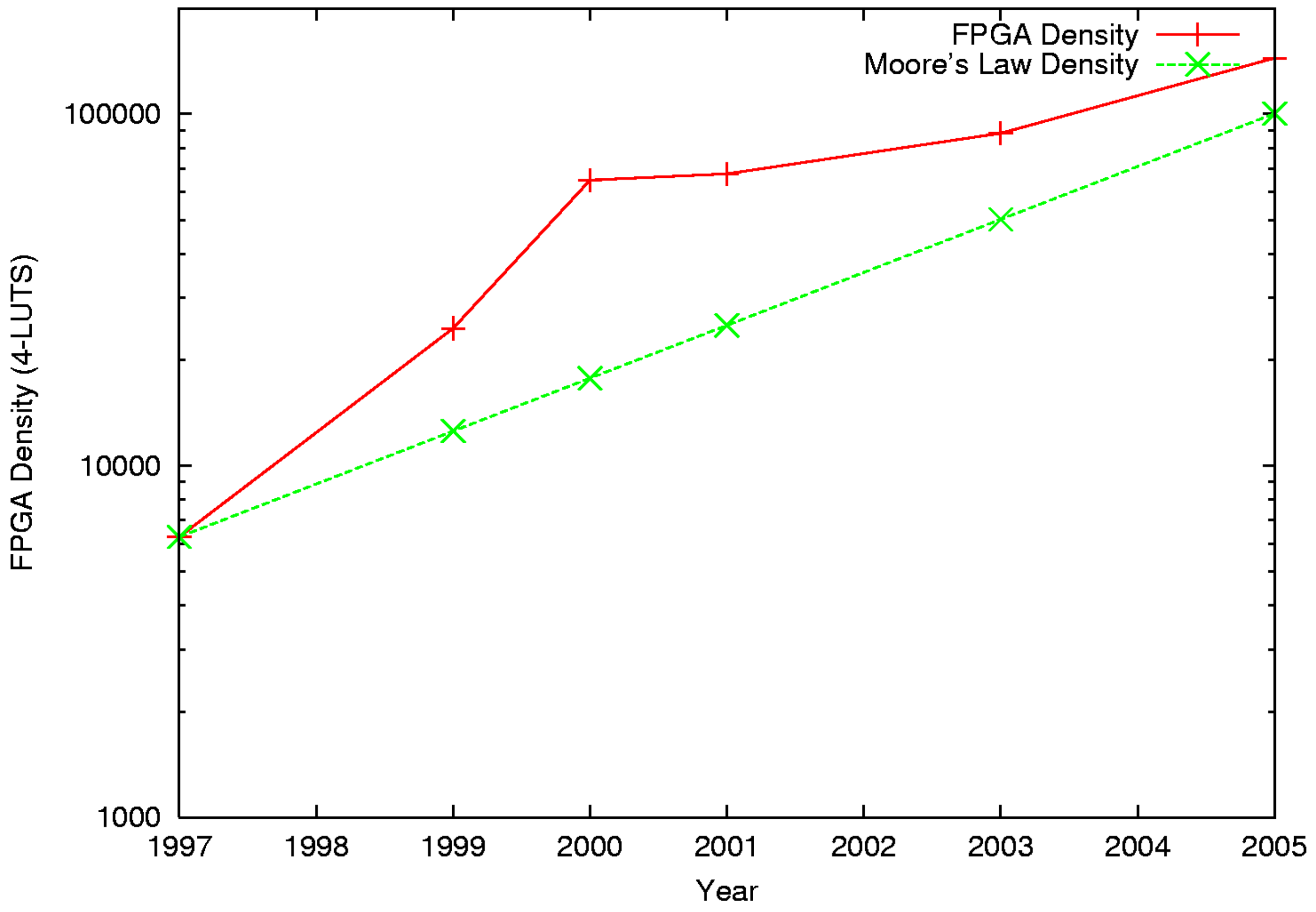
- FPGAs should be able to accelerate several types of double precision floating-point kernels
- They might even be able to accelerate applications
  - Many apps user libraries (BLAS, FFT, Solvers)
  - But, most applications are more than just kernels
  - System architecture is still a major question
- Several major barriers still stand in the way
  - FPGA vendors must care about HPC
  - FPGAs must reach certain levels of reliability
  - Market is cost sensitive

Sandia National Laboratories

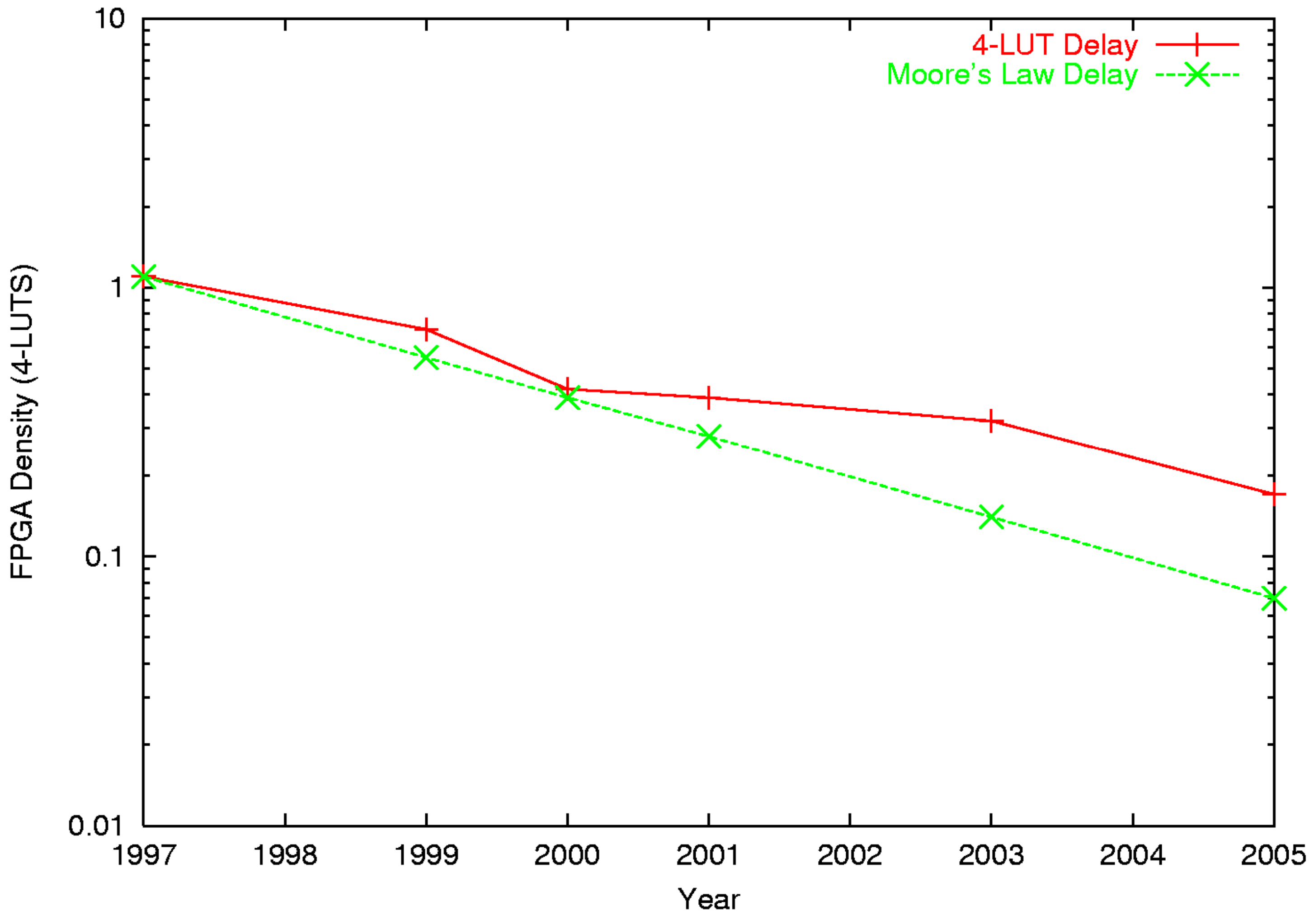
Double Precision MACC Performance Trends



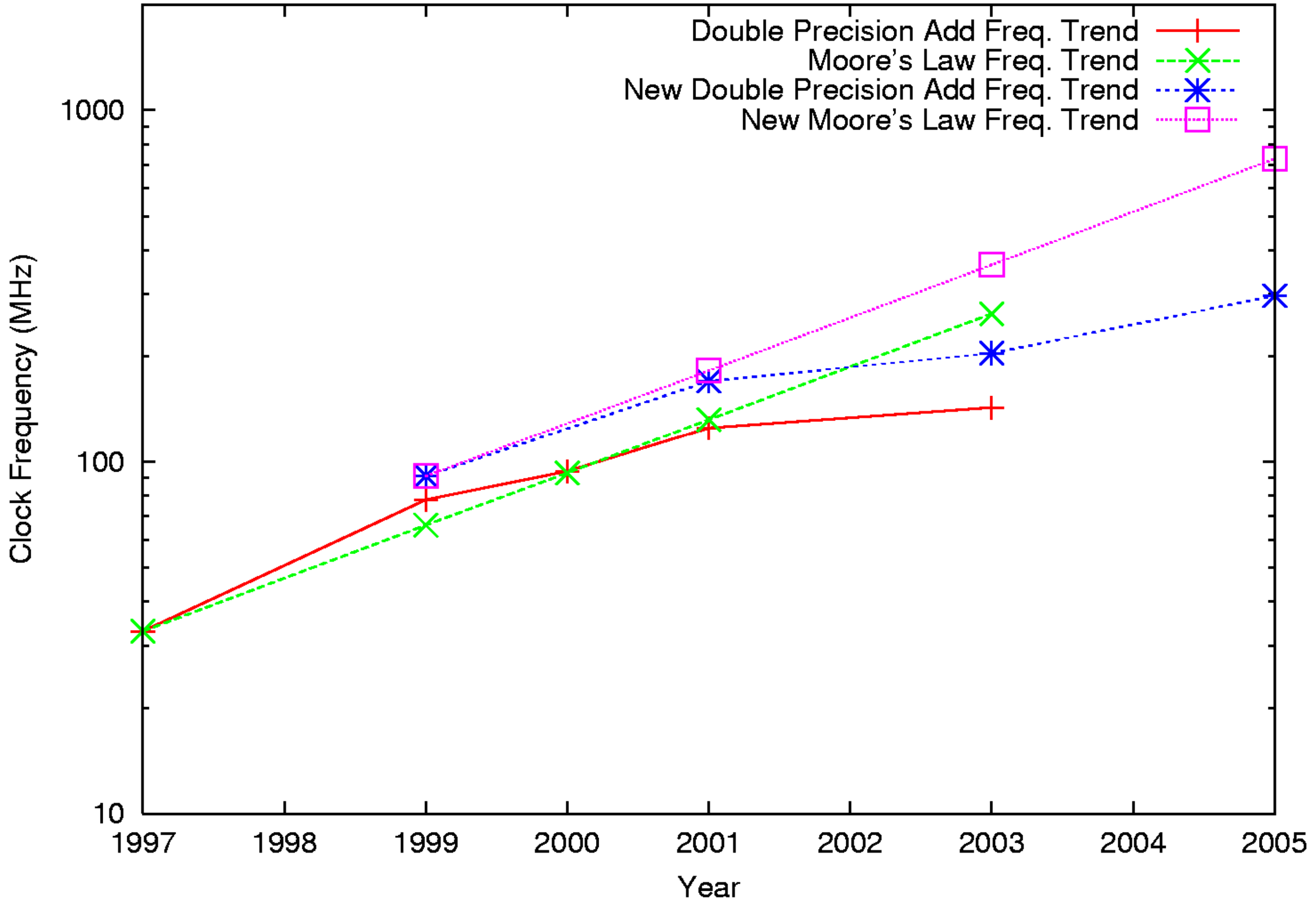
Sources of Trends – Density



Source of Trends – 4-LUT Delay



Trend Challenges – Clock Rate



Sandia National Laboratories

Keeping Up With Trends?

- Xilinx made some design decisions in Virtex-4 which limit the performance of double precision floating-point
  - Carry chain is not significantly faster than in Virtex2Pro
    - Trade off: Faster on/off times, for slower ripple time
    - This can be overcome, but requires a trade-off between clock frequency and area
  - None of the Virtex4 families have a “good” multiplier/logic mix
    - This needs to be addressed in future parts!

Sandia National Laboratories

Keeping Up With Trends?

	V2P 100-6 (original)	V2P 100-6 (current)	Predicted for 2005	V4 FX140-12 (2005)
Adder:				
Units:	40	64		105
Frequency:	140	204		296
Peak MFLOPS:	5720	13056	22880	31080
Multiplier:				
Units:	27	48		21
Frequency:	142	206		296
Peak MFLOPS:	3834	9888	17253	6216
MACC:				
Units:	15	24		21
Frequency:	140	204		296
Peak MFLOPS:	4200	9792	18900	12432

Keeping Up With Trends? What If: Better Logic/DSP48 Mix*

	V2P 100-6 (original)	V2P 100-6 (current)	Predicted for 2005	V4 FX140-12 (2005)
Multiplier:				
Units:	27	48		41
Frequency:	142	206		296
Peak MFLOPS:	3834	9888	17253	12135
MACC:				
Units:	15	24		32
Frequency:	140	204		296
Peak MFLOPS:	4200	9792	18900	18944

*The Virtex4 FX140 has 192 DSP48 slices. The proposed mix would have 288 DSP48 slices. (This is 3 columns instead of 2 columns of DSP)

Sandia National Laboratories

Reasons it Might Translate to Program Performance

- Programmable use of local storage
- Address generation can be decoupled from computation
- Large pin count for direct connection to fast, wide, external memory
- High speed signaling for integration with network and processor
- User allocates logic
- Performance can be delivered to some apps through libraries

Reasons it Might NOT

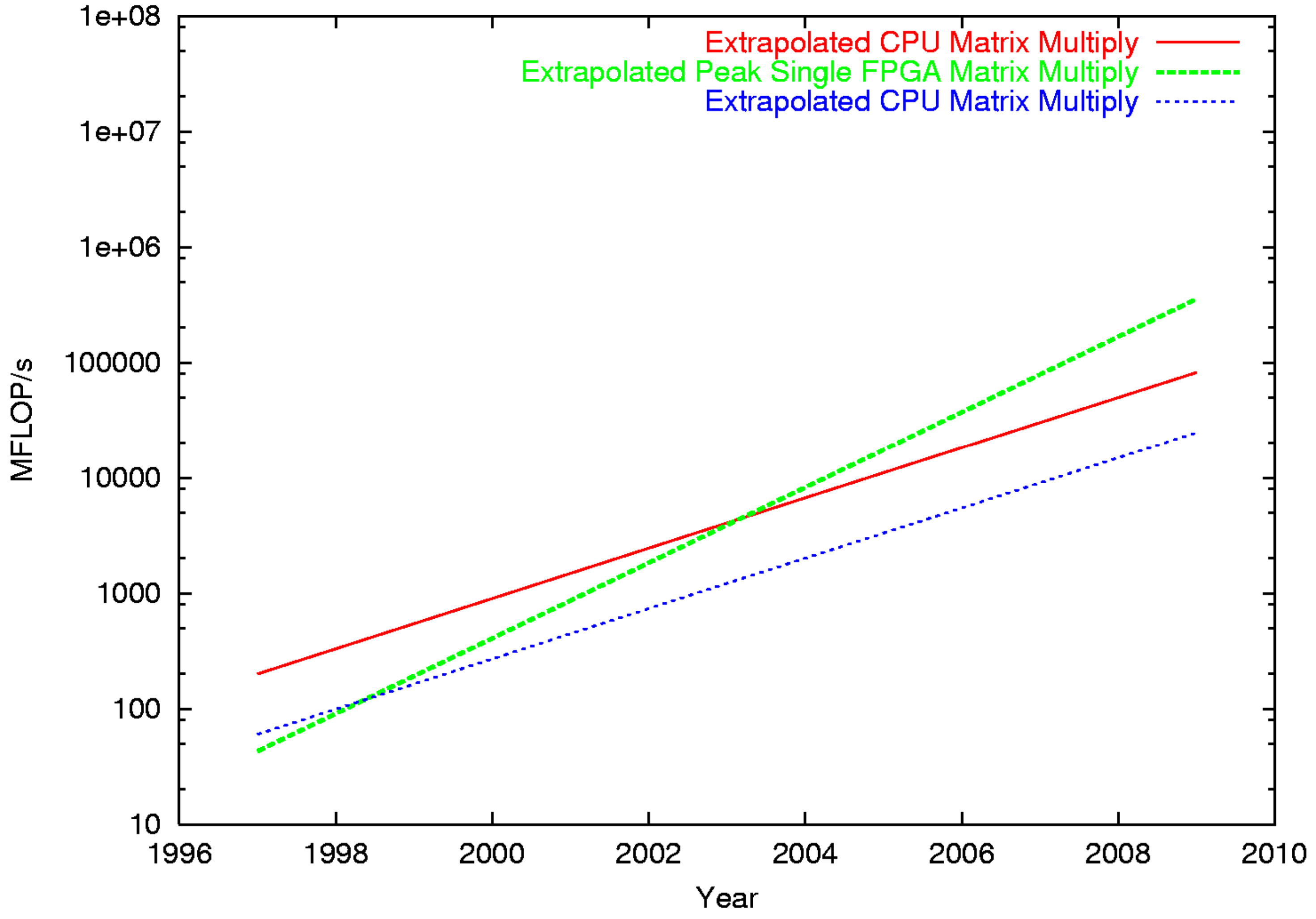
- **Must exploit significant parallelism to achieve performance**
  - **Many floating-point units**
  - **High latency FPUs**
  - **Low clock rate FPUs**
- **System integration issues**
- **Amdahl is not your friend**
- **Hard to program and “Dusty Decks” abound**
  - **Decks are big**
  - **Unfortunately, they aren’t so dusty**

Application 1: A Quantum Chemistry Code

- Uses lots of small, dense matrix multiplies
  - Consumes as much as 90% of the execution time
  - Processors are relatively bad at these
  - FPGAs are relatively good at these
- Significant challenges to overcome
  - The BLAS API is TERRIBLE for this app
  - Cost is a major constraint because this is not a huge fraction of the laboratory workload
  - Implications for network bandwidth
  - A 10X kernel performance gain yields a 5X application performance gain

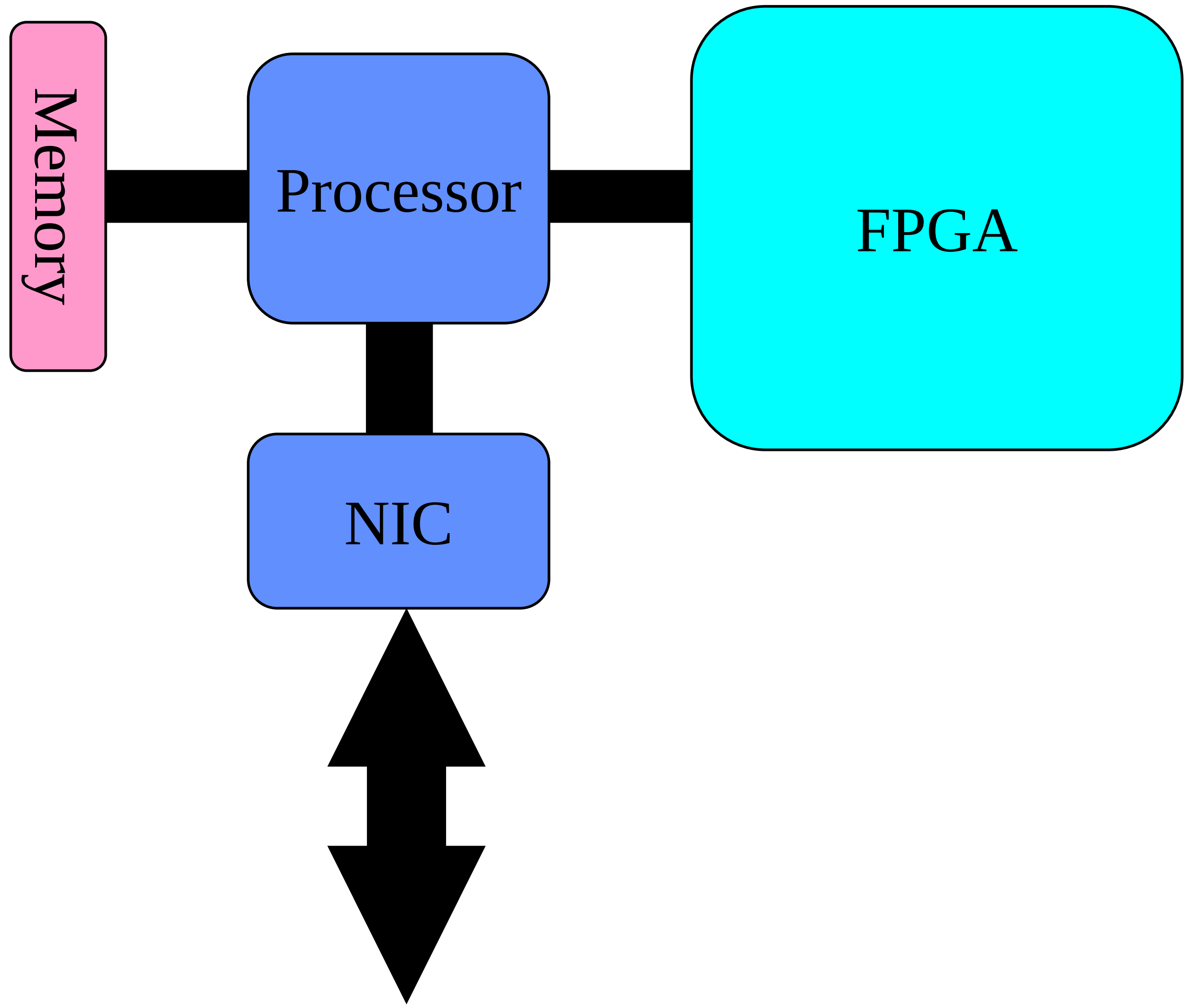
Sandia National Laboratories

Matrix Multiply Performance



Sandia National Laboratories

The Perfect Architecture (for this app….)



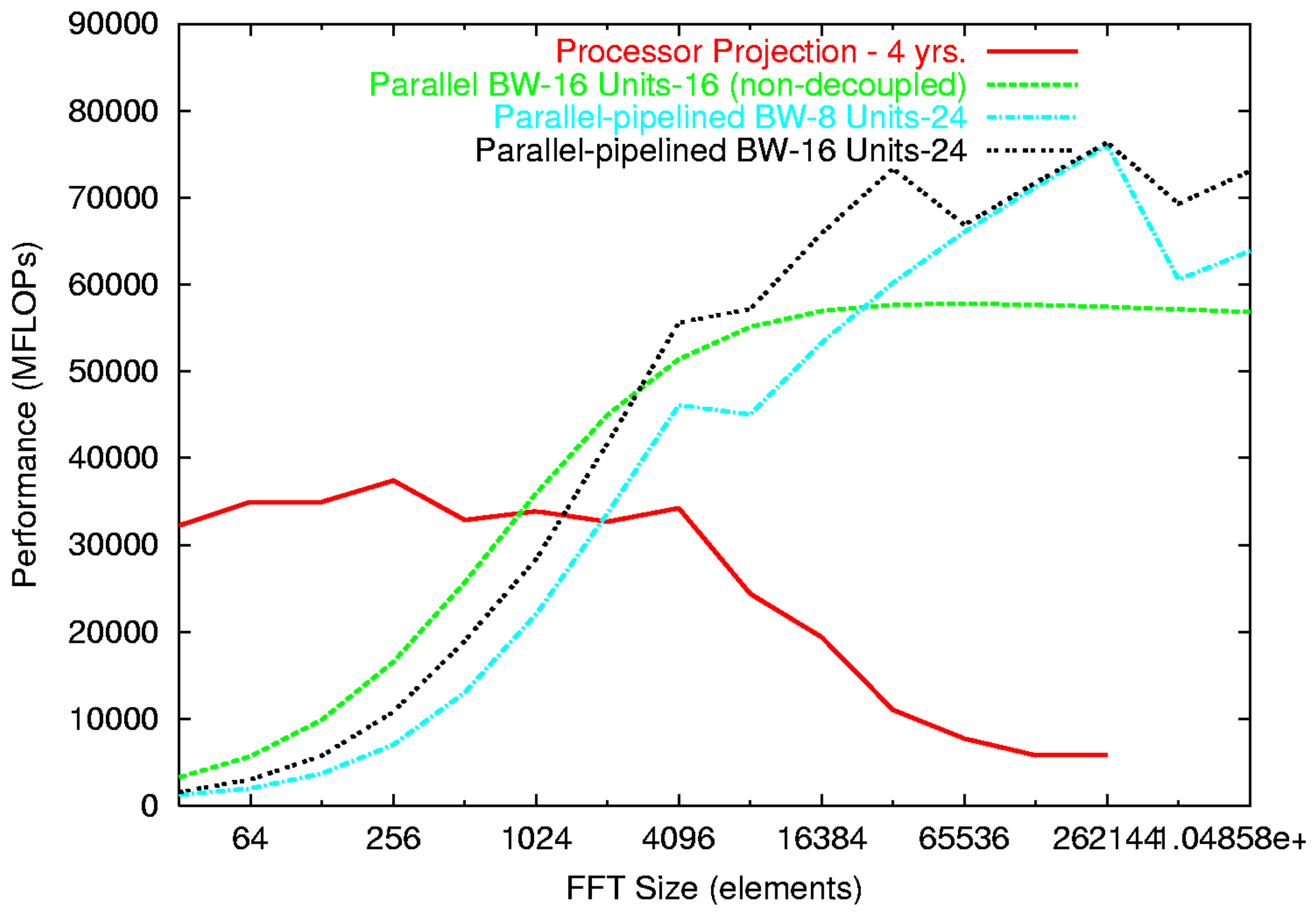
Application 2: Molecular Dynamics

- One potential gain: uses 3D-FFTs

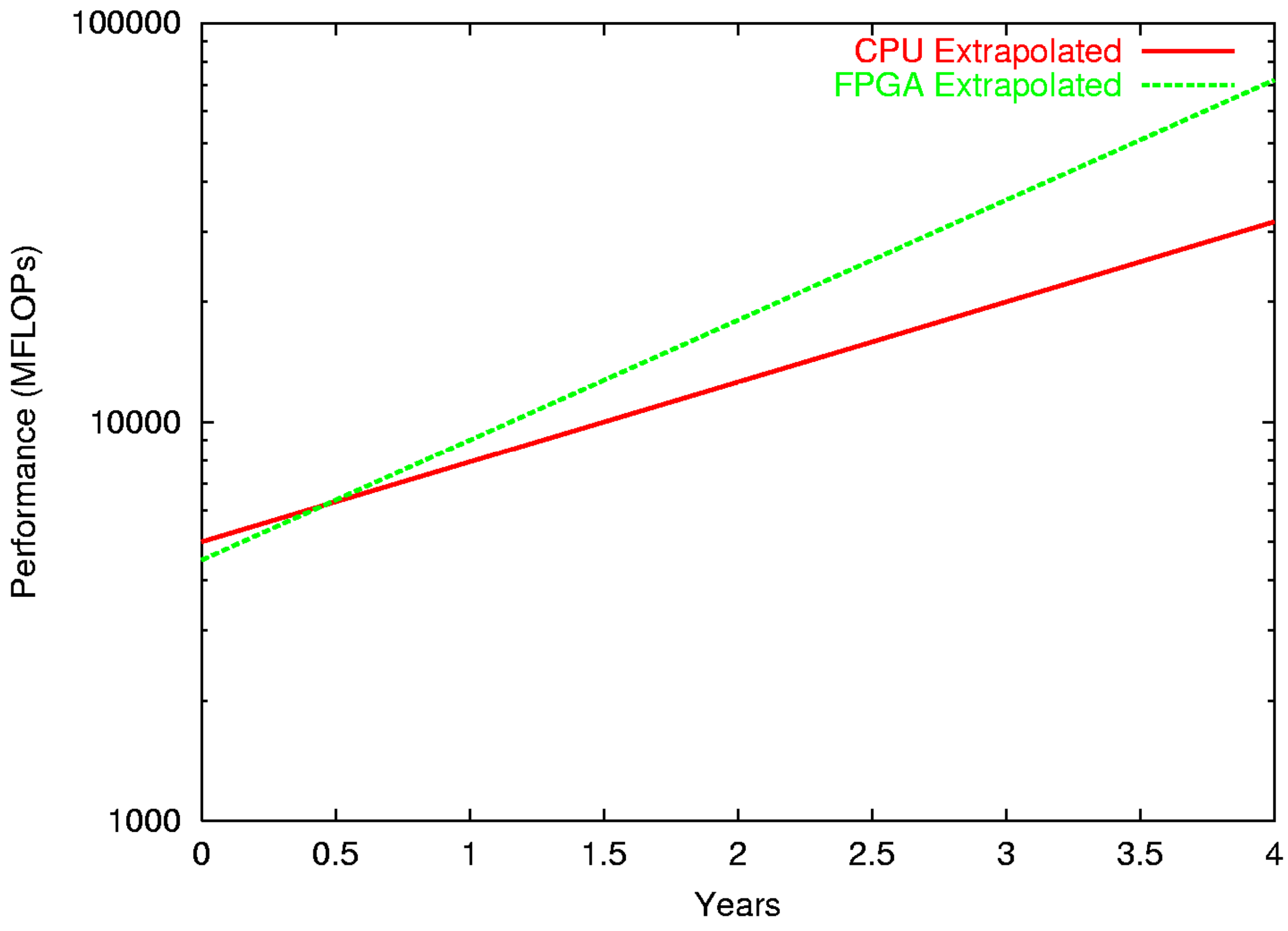
- Challenges
  - The 3D-FFT is a parallel 3D-FFT
  - Each dimension is small and is currently written to call lots of 1D-FFTs (another broken API)
  - Oh, and lots of communications (matrix transposes)
  - And it’s only 15-20% of the app…

Sandia National Laboratories

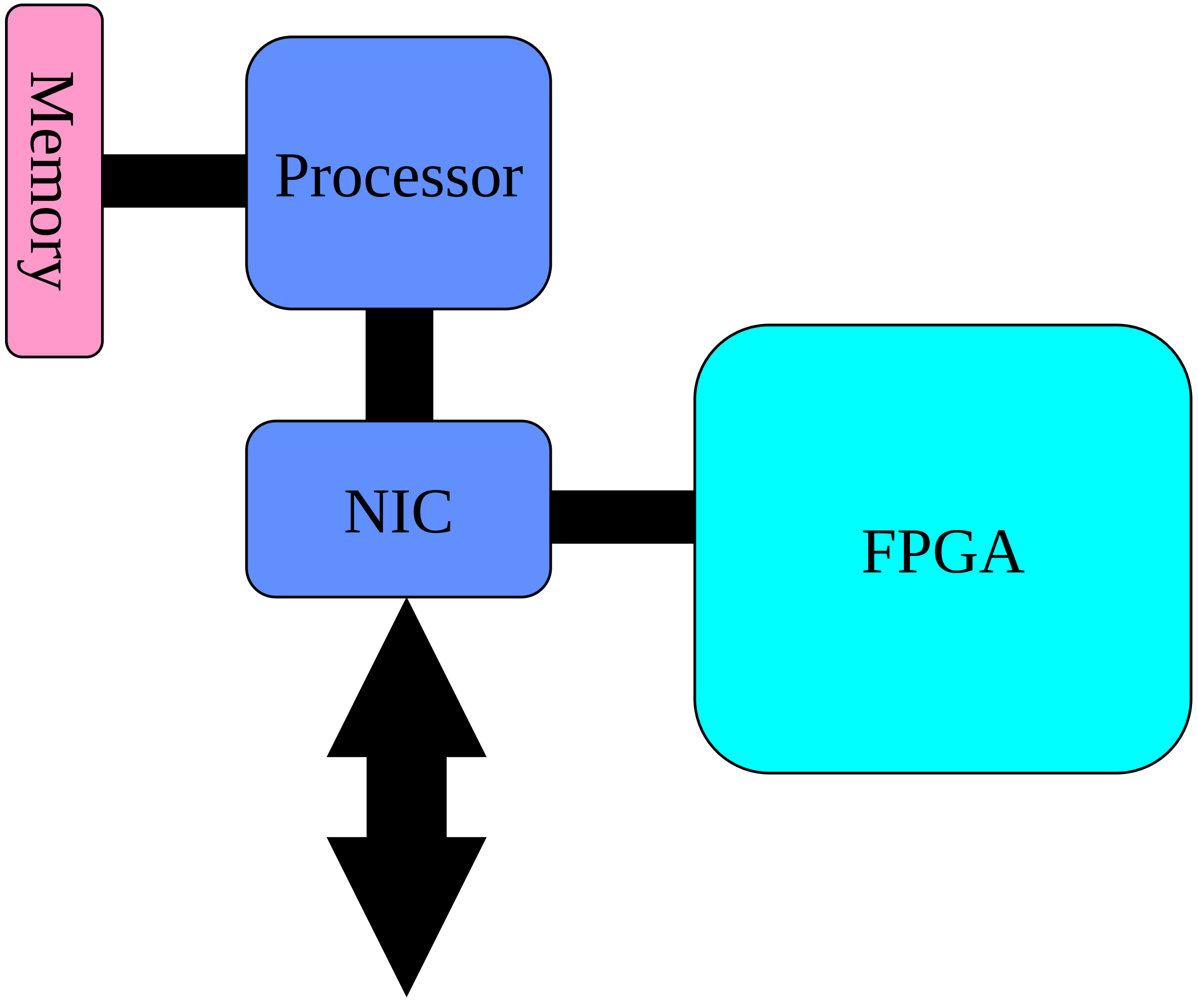
Future FFT Performance versus CPUs



Streaming, Small FFT Performance Graph



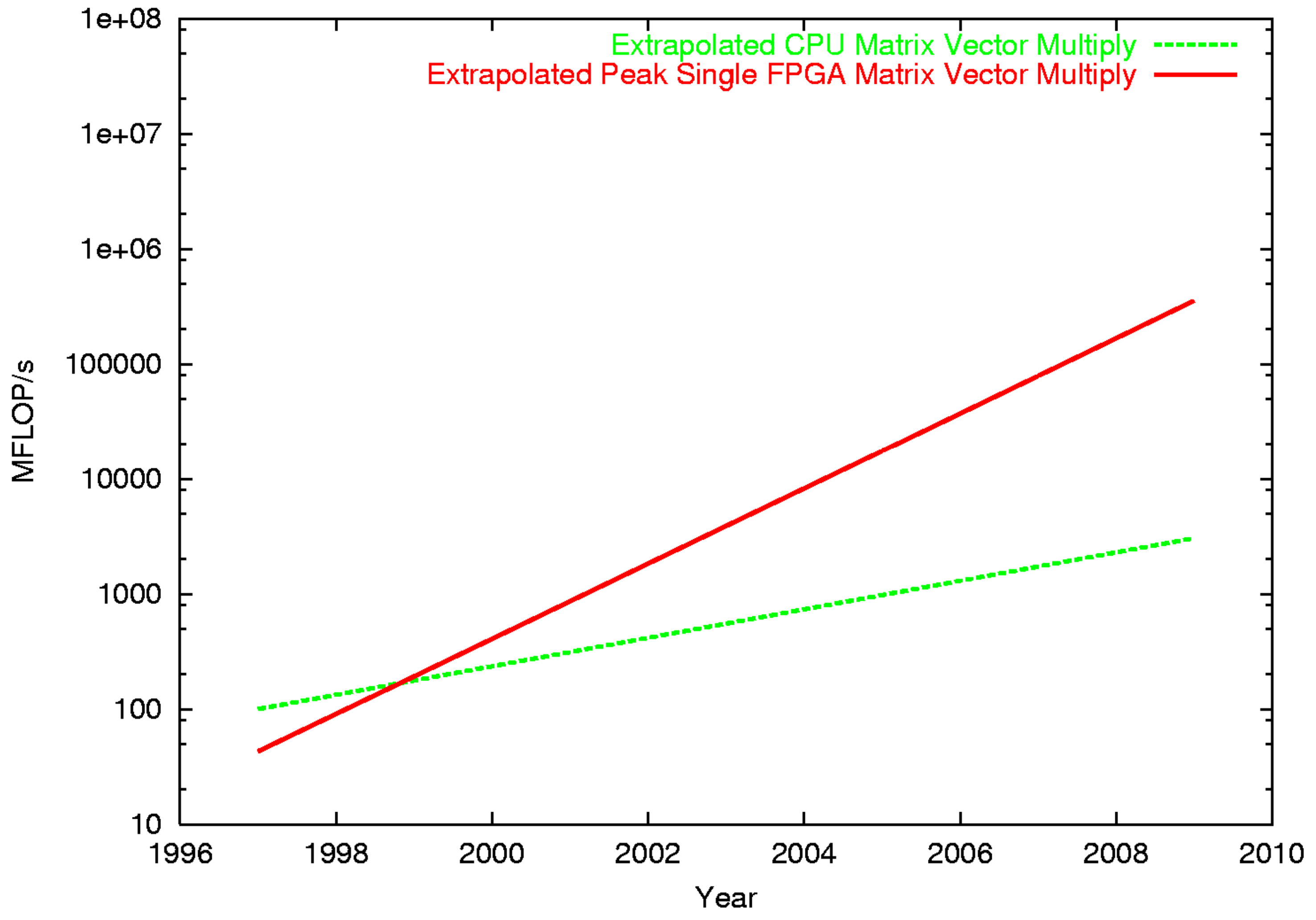
The Perfect Architecture (for this app….)



Application 3: Parallel Sparse Matrix Solve

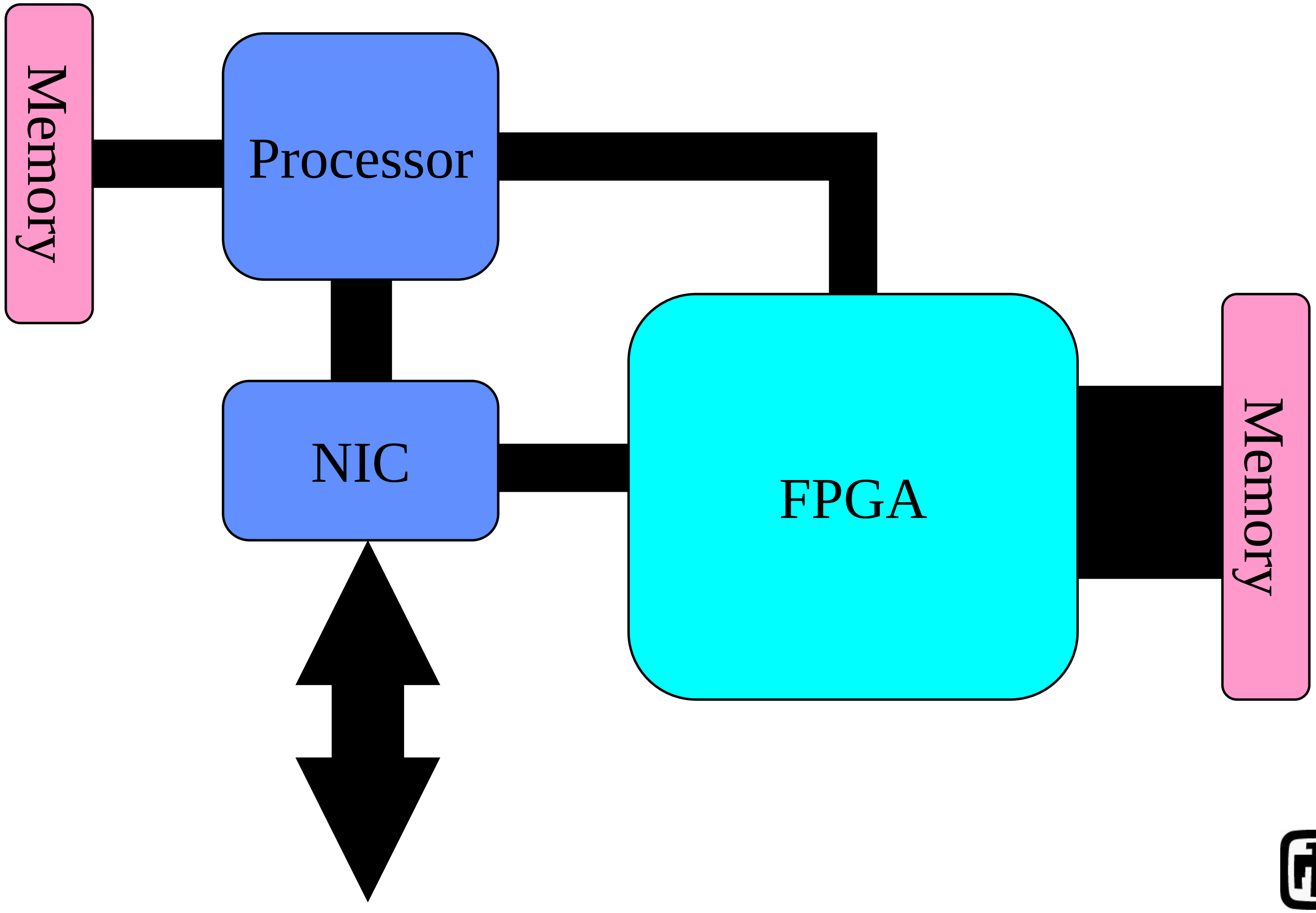
- **Ok, so that isn't an app…**
  - **It does take as much as 75% of some applications execution time**
  - **It is better suited to FPGAs than microprocessors**
    - **High memory bandwidth needs**
    - **Indirected loads (a = b[c[d]])**
- **It does have significant challenges**
  - **Data tends to live in app**
  - **Needs network integration**

(Dense) Matrix Vector Multiply on FPGAs



Sandia National Laboratories

The Perfect Architecture (for this app….)



Major Challenges Remain

- This is a subset of applications
  - Many apps don’t have “kernels”
  - Most of those don’t even follow the traditional 90/10 model (90% of the time in 10% of the code)
- System Architecture must be *stable* and *general*
  - Applications will not recode for 4 boutique architectures
  - If migration from one generation to the next is “hard”, applications will drop the platform
  - A standard, useful API
- Reliability is probably the single biggest challenge in major HPC systems

Reliability

- FPGAs have high susceptibility to Cosmic Rays
  - With large numbers of nodes, the upset rate will be too high to ignore

- FPGAs do not have:
  - A way to detect cosmic ray hits
  - A way to correct bit errors
  - A way to prevent memory commits after a cosmic ray hit

“Smoke Proof” Verification

- Thermal disruption
  - Bad User: Connect all flip-flops in 500 MHz toggle chain. Thermal load melts solder.
  - Good User: Transient load trips current limit and crashes node.
  - How do we prevent both and provide “clean” exit path?
- Part destruction
  - Drive output while enabling input

Conclusions

- FPGAs on track to dramatically outperform CPUs on double precision, floating point operations
  - Performance growth may slow if vendors do not keep floating point in mind when developing new architectures
  - Performance growth may accelerate if vendors decide to focus on floating point.
- Important questions still to answer
  - What floating-point friendly, commercially viable improvements can be made to FPGA architectures?
  - Is there a sufficiently general, sufficiently cost-effective, system architecture?
  - What do the “right” APIs look like for standard library operations?

Sandia National Laboratories

Questions?