Two-Dimensional Phase Unwrapping On FPGAs And GPUs

Sherman Braganza
Prof. Miriam Leeser

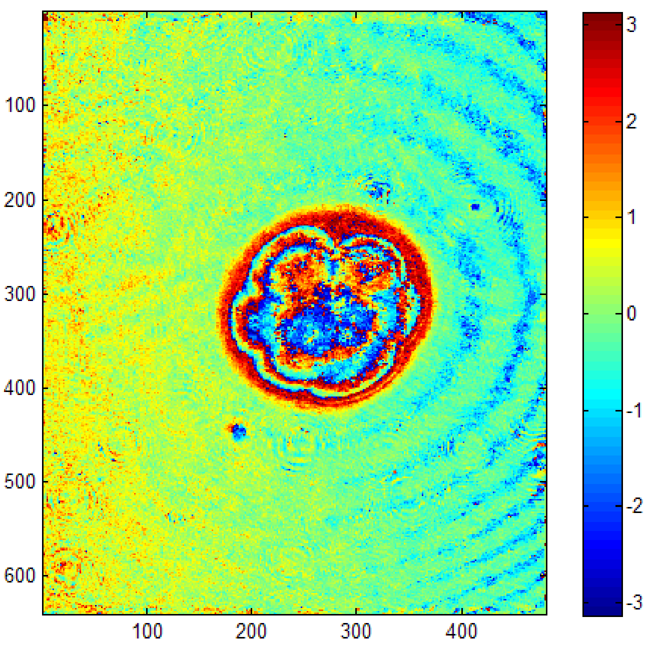
ReConfigurable Laboratory
Northeastern University
Boston, MA

Outline

- Introduction
  - Motivation
    - Optical Quadrature Microscopy
    - Phase unwrapping
- Algorithms
  - Minimum L^P norm phase unwrapping
- Platforms
    - Reconfigurable Hardware and Graphics Processors
- Implementation
  - FPGA and GPU specifics
  - Verification details
- Results
  - Performance
  - Power
  - Cost
- Conclusions and Future Work

Motivation – Why Bother With Phase Unwrapping?

- Used in phase based imaging applications
  - IFSAR, OQM microscopy
- High quality results are computationally expensive
  - Only difficult in 2D or higher
  - Integrating gradients with noisy data
  - Residues and path dependency



Wrapped embryo image

$$\begin{tikzcd}0.1 \arrow[r] & 0.3 \arrow[d] \\ -0.1 \arrow[u] & -0.3 \arrow[l]\end{tikzcd}$$

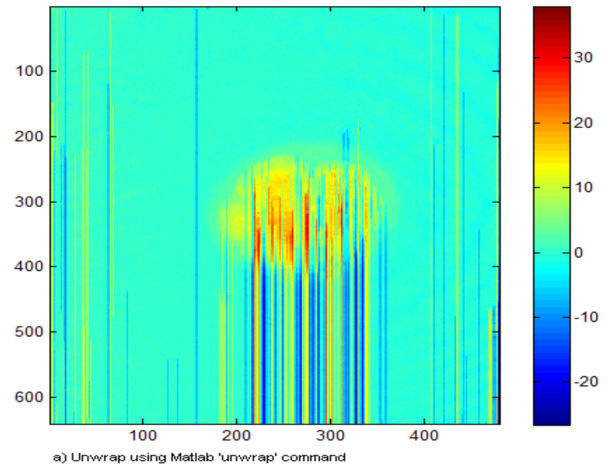
No residues

$$\begin{tikzcd}0.1 \arrow[r] & 0.3 \arrow[d] \\ -0.1 \arrow[u] & -0.2 \arrow[l]\end{tikzcd}$$

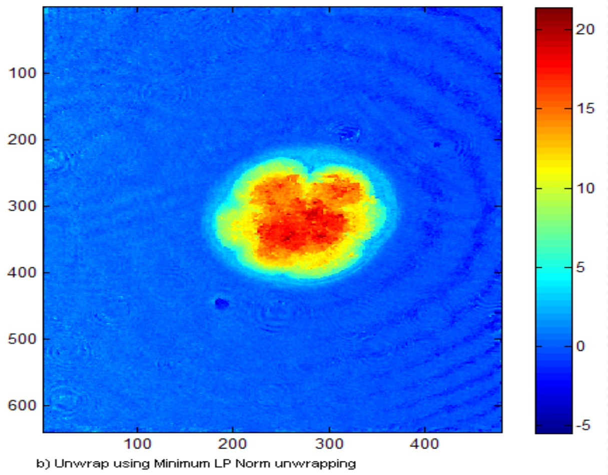
Residues

Algorithms – Which One Do We Choose?

- Many phase unwrapping algorithms
  - Goldstein’s, Flynns, Quality maps, Mask Cuts, multi-grid, PCG, Minimum LP norm (Ghiglia and Pritt, “*Two Dimensional Phase Unwrapping*”, Wiley, NY, 1998.
- We need: High quality (performance is secondary)
  - Abilitity to handle noisy data
- Choose Minimum L^P Norm algorithm: Has the highest computational cost



a) Software embryo unwrap
Using matlab ‘unwrap’



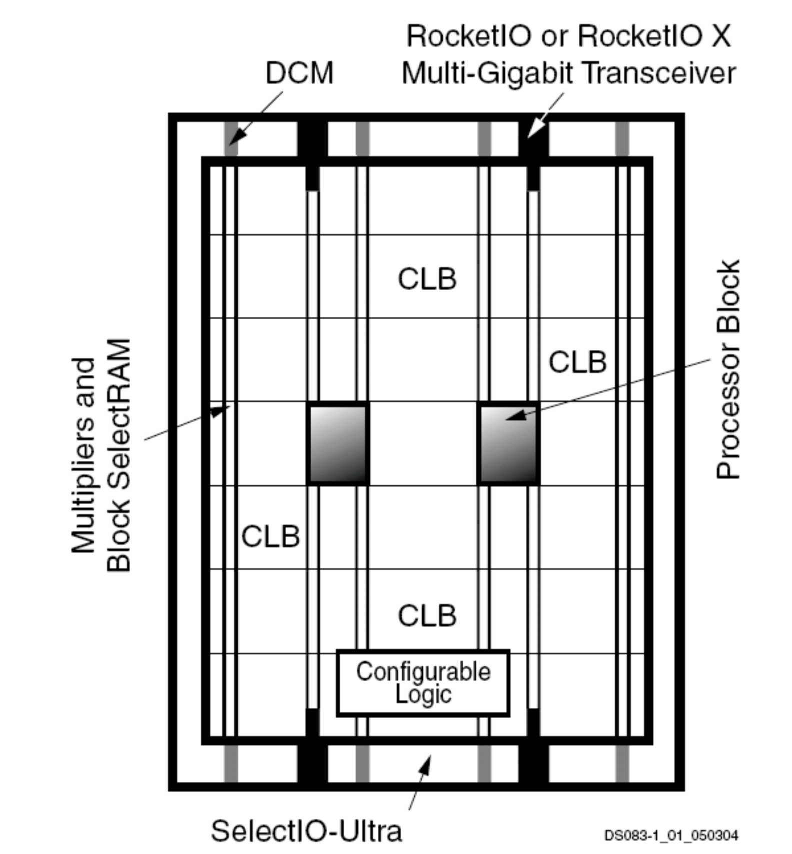
b) Software embryo unwrap
Using Minimum L^P Norm

Breaking Down Minimum LP Norm

- Minimizes existence of differences between measured data and calculated data
- Iterates Preconditioned Conjugate Gradient (PCG)
  - 94% of total computation time
  - Also iterative
  - Two steps to PCG
    - Preconditioner (2D DCT, Poisson calculation and 2D IDCT)
    - Conjugate Gradient

Platforms – Which Accelerator Is Best For Phase Unwrapping?

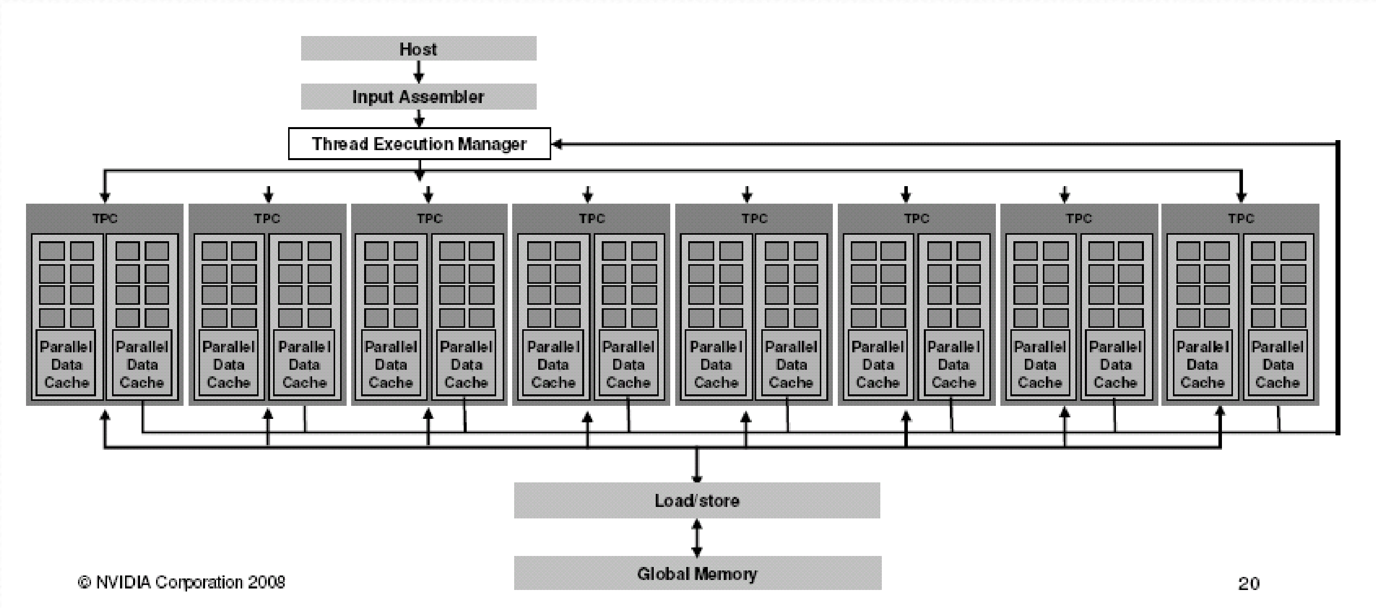
- FPGAs
  - Fine grained control
  - Highly parallel
  - Limited program memory
    - Floating point?
  - High implementation cost



Xilinx Virtex II Pro architecture
http://www.xilinx.com/

Platforms - GPUs

- GPUs
  - Data parallel architecture
    - Less flexibility
    - Floating point
  - Large program memory
    - Inter processor communication?
  - Lower implementation cost
  - Limited # of execution units



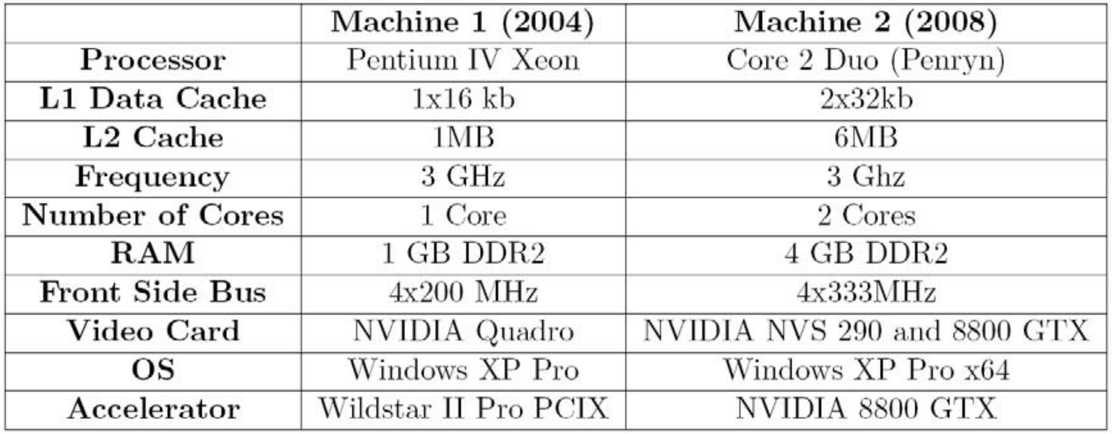
G80 Architecture [nvidia.com/cuda]

Platform Comparison

FPGAs	GPUs
•Absolute control: Can specific custom bit-widths/architectures to optimally suit application	•Need to fit application to architecture
•Can have fast processor-processor communication	•Multiprocessor-multiprocessor communication is slow
•Low clock frequency	•Higher frequency
•High degree of implementation freedom => higher implementation effort. VHDL.	•Relatively straightforward to develop for. Uses standard C syntax
•Small program space. High reprogramming time	•Relatively large program space. Low reprogramming time.

Platform Description

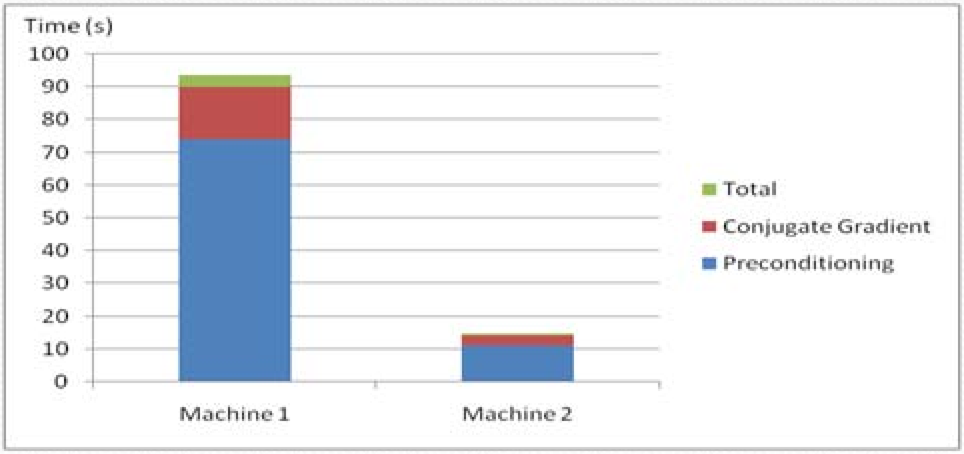
- FPGA and GPU on different platforms 4 years apart
- Effects of Moore’s Law



	Machine 1 (2004)	Machine 2 (2008)
Processor	Pentium IV Xeon	Core 2 Duo (Penryn)
L1 Data Cache	1x16 kb	2x32kb
L2 Cache	1MB	6MB
Frequency	3 GHz	3 Ghz
Number of Cores	1 Core	2 Cores
RAM	1 GB DDR2	4 GB DDR2
Front Side Bus	4x200 MHz	4x333MHz
Video Card	NVIDIA Quadro	NVIDIA NVS 290 and 8800 GTX
OS	Windows XP Pro	Windows XP Pro x64
Accelerator	Wildstar II Pro PCIX	NVIDIA 8800 GTX

Platform specifications

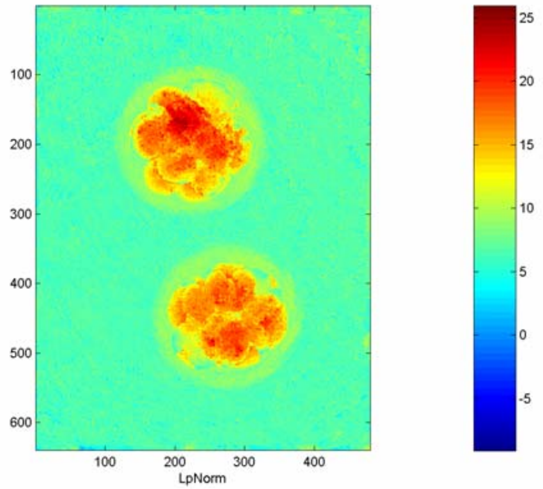
- Machine 3 in the *Results: Cost* section has a Virtex 5 and two Core2Quads



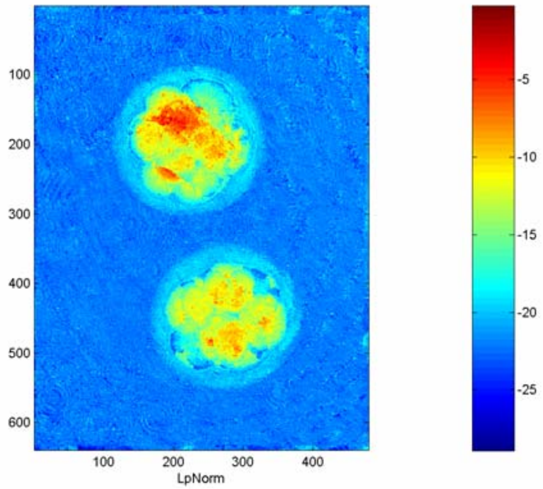
Software unwrap execution time

Implementation: Preconditioning On An FPGA

- Need to account for bitwidth
  - Minimum of 28 bit needed – Use 24 bit + block exponent
- Implement a 2D 1024x512 DCT/IDCT using 1D row/column decomposition
- Implement a streaming floating point kernel to solve discretized Poisson equation



27 bit software unwrap



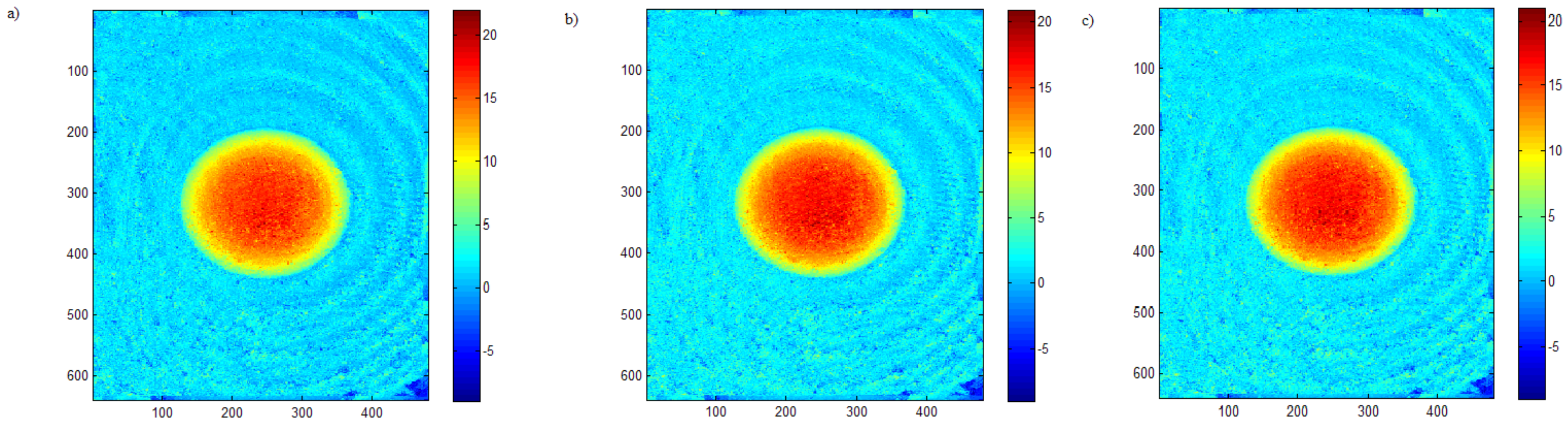
28 bit software unwrap

Minimum L^P Norm On A GPU

- NVIDIA provides 2D FFT kernel
  - Use to compute 2D DCT
- Can use CUDA to implement floating point solver
  - Few accuracy issues
- No area constraints on GPU
  - Why not implement whole algorithm?
- Multiple kernels, each computing one CG or L^P norm step
- One host to accelerator transfer per unwrap

Verifying Our Implementations

- Look at residue counts as algorithm progresses
  - Less than 0.1% difference
- Visual inspection: Glass bead gives worst case results



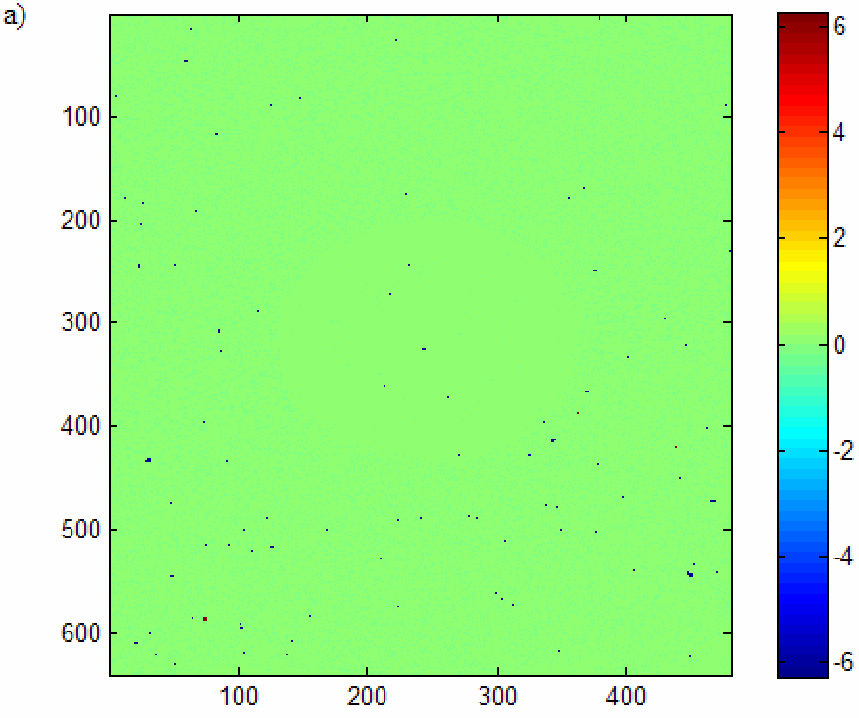
Software unwrap

GPU unwrap

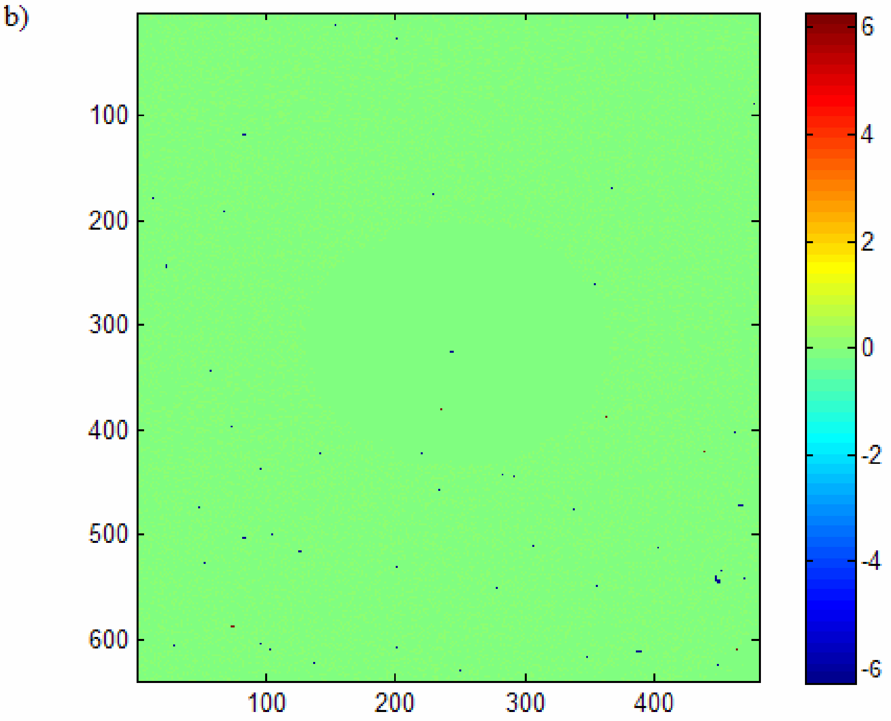
FPGA unwrap

Verifying Our Implementations

- Differences between software and accelerated version



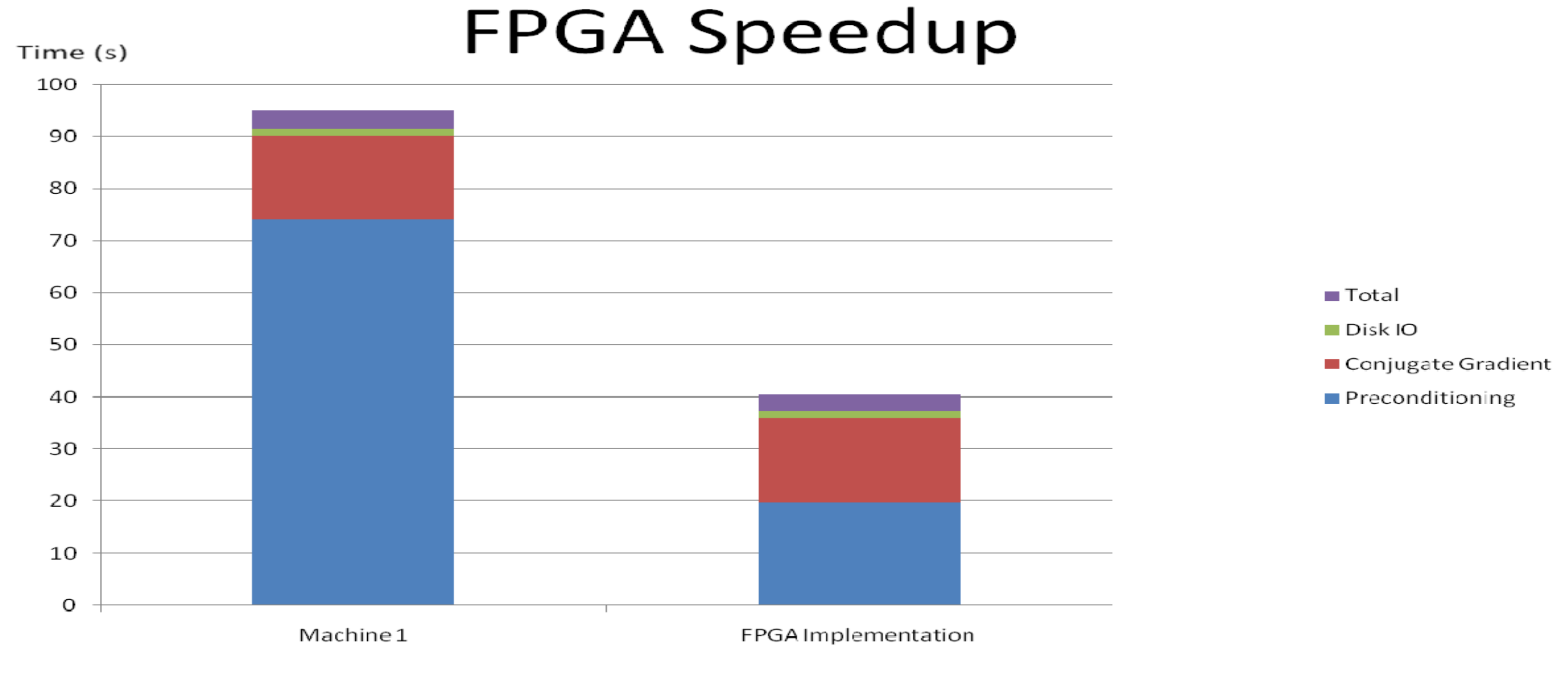
GPU vs. Software



FPGA vs. Software

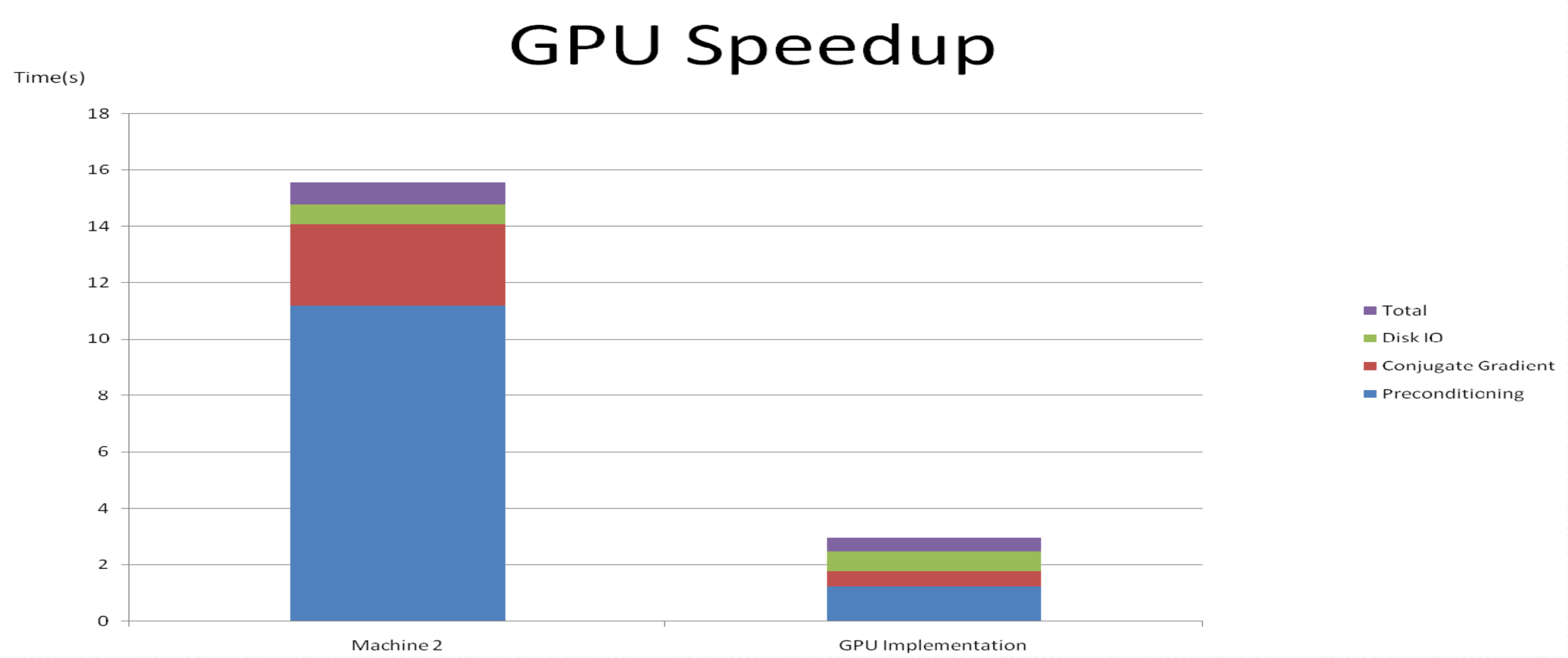
Results: FPGA

- Implemented preconditioner in hardware and measured algorithm speedup
- Maximum speedup assuming zero preconditioning calculation time : 3.9x
  - We get 2.35x on a V2P70, 3.69x on a V5 (projected)



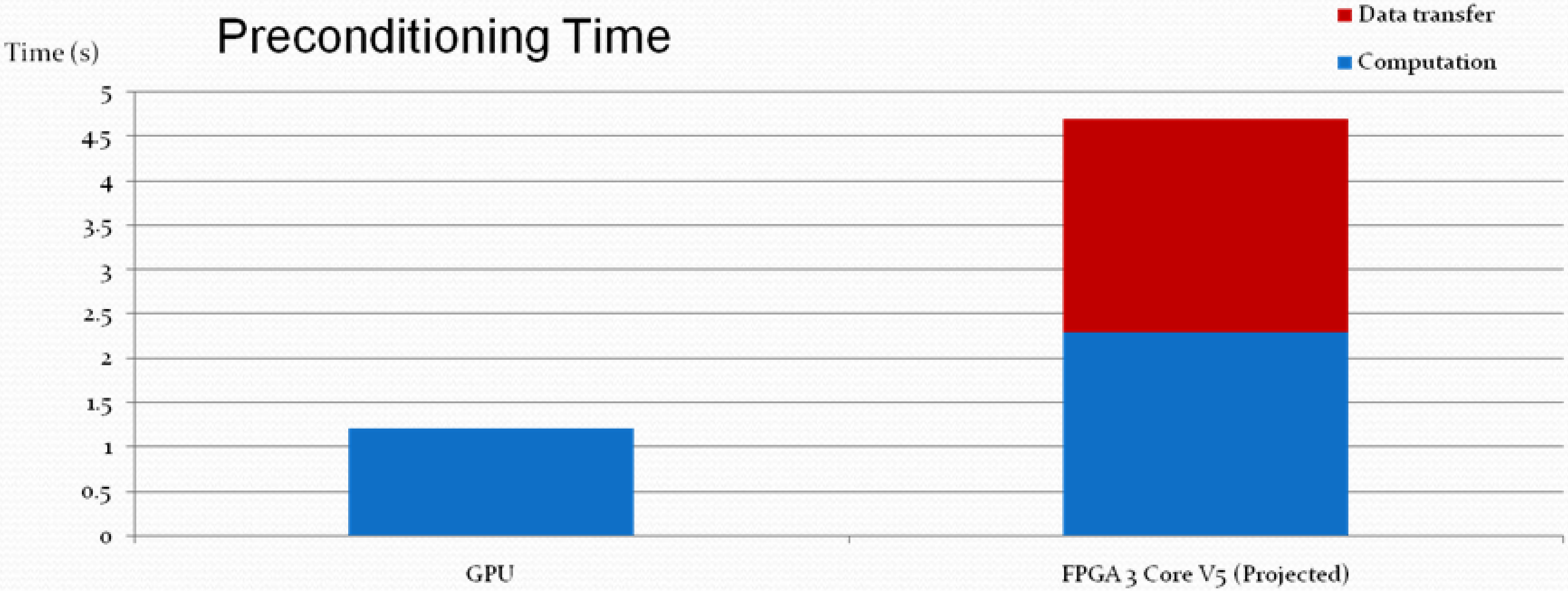
Results: GPU

- Implemented entire LP norm kernel on GPU and measured algorithm speedup
- Speedups for all sections except disk IO
- 5.24x algorithm speedup. 6.86x without disk IO



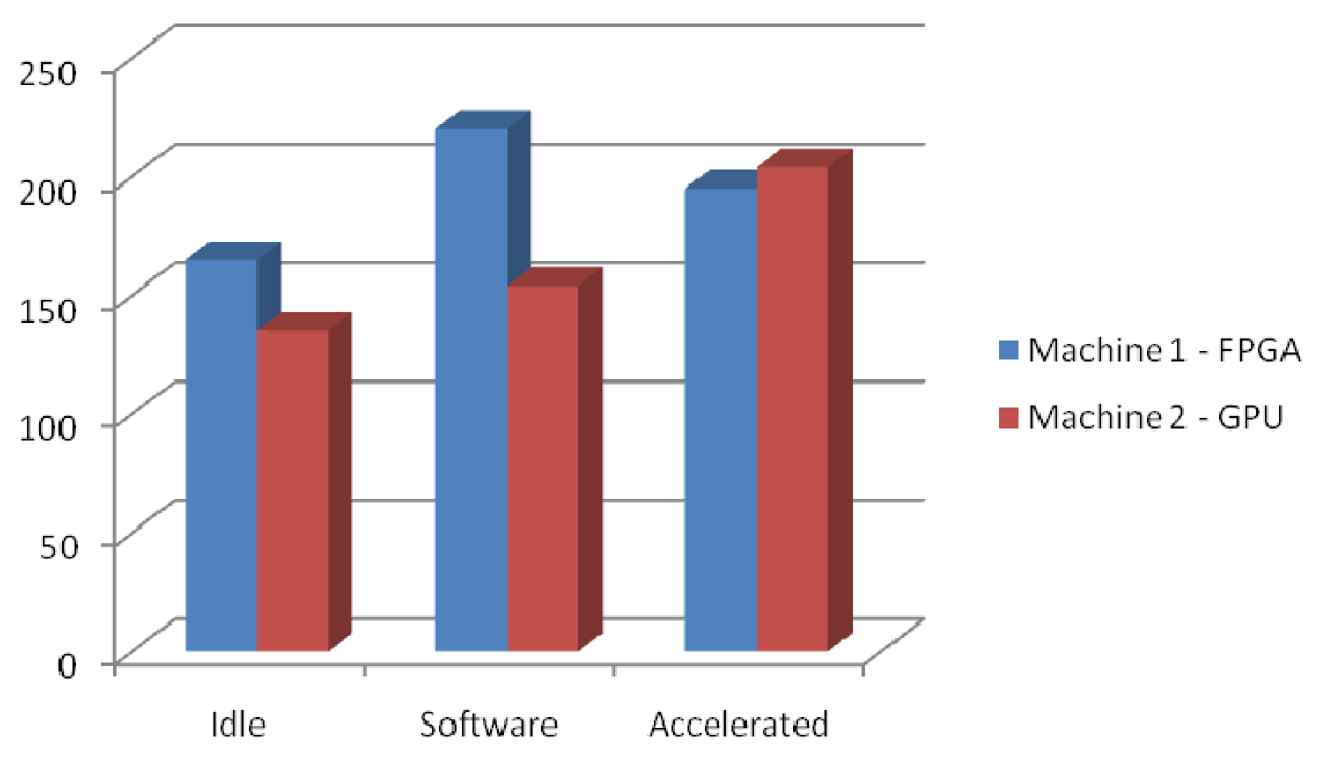
Results: FPGAs vs. GPUs

- Preconditioning only
- Similar platform generation. Projected FPGA results.
- Includes FPGA data transfer, not GPU
  - Buses? Currently use PCI-X for FPGA, PCI-E for GPU



Results: Power

- GPU power consumption increases significantly
- FPGA power decreases

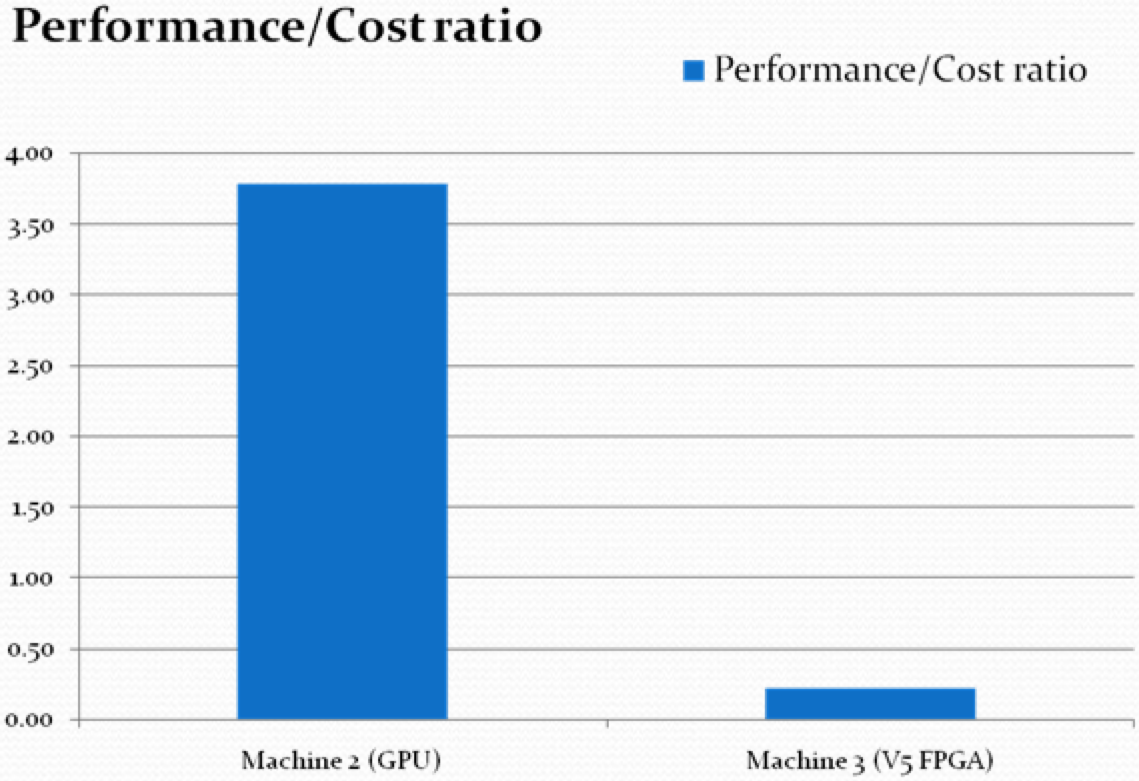


Power consumption (W)

Cost

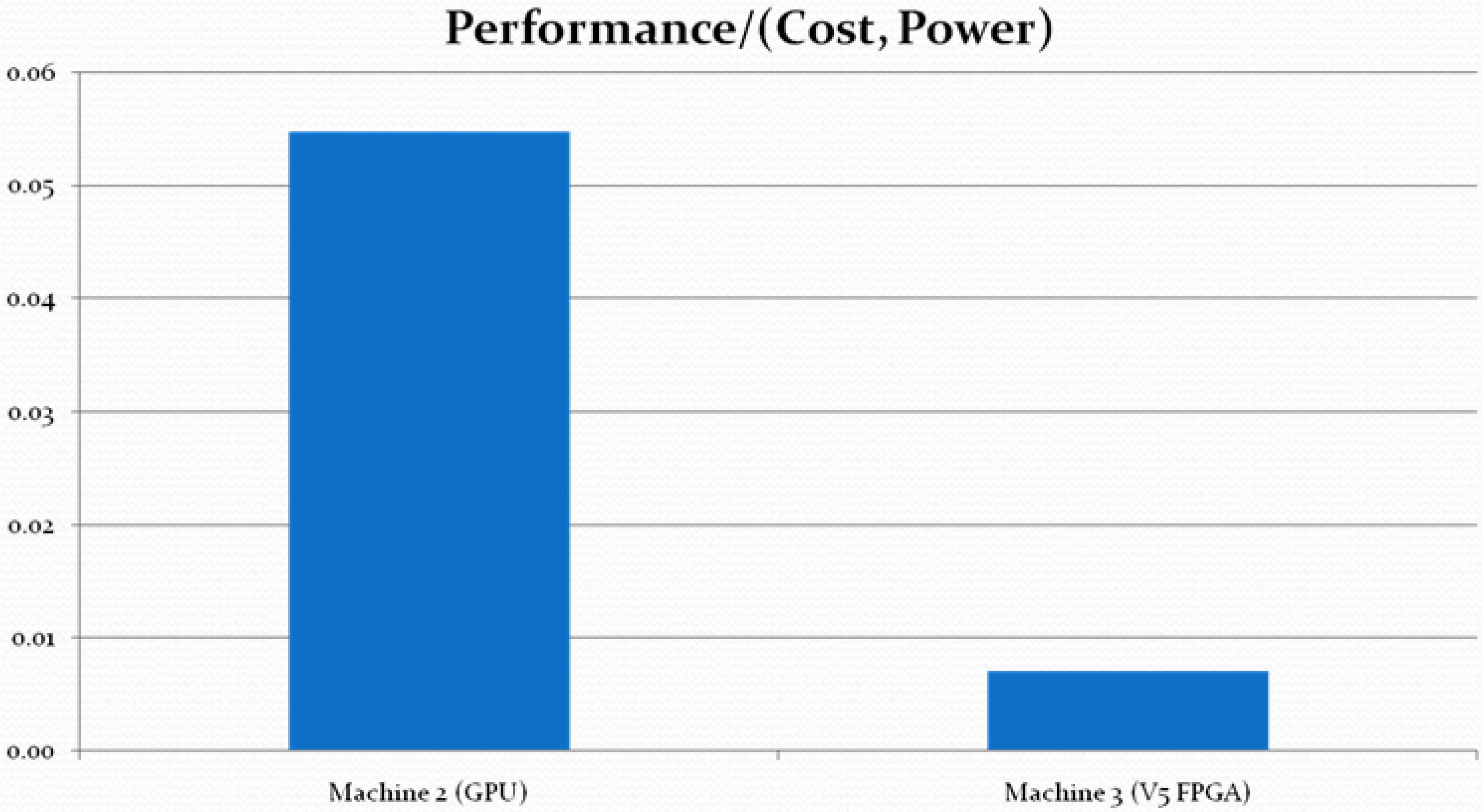
Machine 2	$2200
Machine 3	$10000

- Machine 3 includes an AlphaData board with a Xilinx Virtex 5 FPGA platform and two Core2Quads
- Performance is given by $1/T_{exec}$
  - Proportional to FLOPs



Performance To Watt-Dollars

- Metric to include all parameters



Conclusions And Future Work

- For phase unwrapping GPUs provide higher performance
  - Higher power consumption
- FPGAs have low power consumption
  - High reprogramming time
- OQM: GPUs are the best fit. Cost effective and faster:
  - Images already on processor
  - FPGAs have a much stronger appeal in the embedded domain

- Future Work
  - Experiment with new GPUs (GTX 280) and platforms (Cell, Larrabee, 4x2 multicore)
  - Multi-FPGA implementation

Thank You!

Any Questions?

Sherman Braganza (braganza.s@neu.edu)
Miriam Leeser (mel@coe.neu.edu)
Northeastern University ReConfigurable Laboratory
http://www.ece.neu.edu/groups/rcl