FPGA Acceleration for 3D Low-Dose CT

Ming Jiang
Peking University
ming-Jiang@pku.edu.cn

MS 1 : Applied Mathematics in Tomography
Conference on Modern Challenges in Imaging
2019.08.08

Presentation based on materials/comments *from Jason Cong, Yong Cui, William Hsu, Alfred Louis, Xiuhong Li, Yun Liang, Guojie Luo, Peter Maass, Thomas Page, Linjun Qiao, Frank Natterer, Thomas Schuster, Wentai Zhang.*

Outline

- Background & Motivation
  - Energy-efficient computing with FPGA
- Asynchronous parallel iterative algorithms
  - Communication model
- FPGA implementation
  - Techniques for implementation
- Summary

Mumford-Shah functional as a regularization for image reconstruction

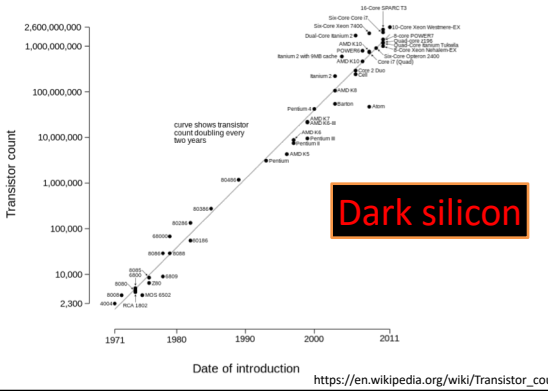
$$\int |A(f) - b|^2 + \alpha \iint_{\Omega \setminus K} |\nabla f|^2 + \beta |K|$$

- Avoid computing derivatives cross edges
- Simultaneous reconstruction of image and its segmentation
- NP-hard when A is not the identity

- Geman and Geman [1984]
- Mumford and Shah [1989]
- Ambrosio [1989]
- De Giorgi, Carriero, and Leaci [1989]
- Vese, Chan [2000]
- Chan, Esedoglu and Nikolova [2004]
- Cai, Chan, Zeng [2013]
- Chan, Yang, Zeng [2014]
- Rondi and Santosa [2001]
- Rondi [2007]
- Alexeev, and Ward [2010]
- Fornasier, March, Solombrino [2011]
- Scherzer [2011, 2014]
- Ramlau, Ring [2010]
- Klann, Ramlau [2013]
- ……
- Jiang, Maass, Page [2014]
- Storath, Weinmann; Frikel, Unser [2015]
- Hohm, Storath, Weinmann [2015]
- Carriero, Leaci, Tomarelli [2015]

An incomplete list…..

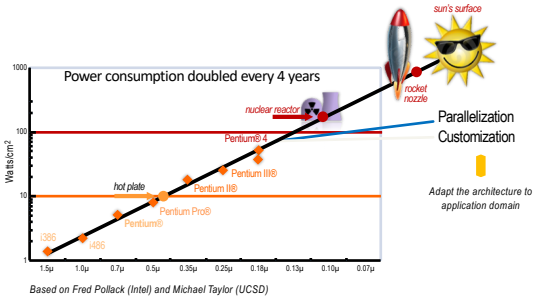
Microprocessor Transistor Counts 1971-2011 & Moore's Law



Dark silicon

https://en.wikipedia.org/wiki/Transistor_count

Customization and Specialization



Based on Fred Pollack (Intel) and Michael Taylor (UCSD)

Curtesy of Prof. Jason Cong

Energy-efficient computing with FPGA

- FPGA is another hardware accelerating approach
  - Multi-core-CPU clusters, GPU and DSP,.
- FPGA (field-programmable gate array)
  - function is defined by a user's program
  - reconfigurable for computing
  - Limited onboard memory vs others

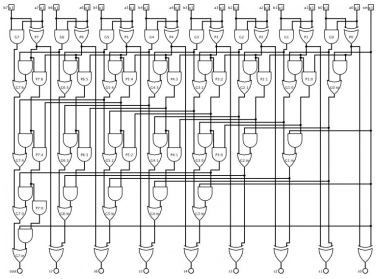
- Xilinx Virtex-7 board VC707 at 100MHz
  - 485K logic gates, 4.5MB onboard memory
  - Transistor count: 6,800,000,000
  - 4.5W (~consumer LED lamp)

- *How to make the best use of it?*

https://en.wikipedia.org/wiki/LED_lamp

7 of 40

Logical circuits of an 8-bit Adder

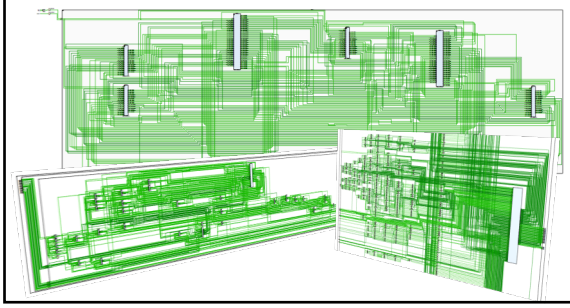


Gate-Level Diagram of an 8-bit Kogge-Stone Parallel Prefix Adder
http://venividiwiki.ee.virginia.edu/.....

8 of 40

FPGA Implementation of Mumford-Shah Imaging Algorithm

XCT Reconstruction with Mumford-Shah: 200K LUT (logic look-up tables)



9 of 40

High-level Synthesis C->FPGA

- Energy-efficient accelerator-rich architecture
- Optimization under the performance, power, and cost constraints
- Extensive use of accelerators (algorithmic blocks)
- Limited onboard memory with full control including precision
  - Extendable
- VivadoHLS of Xilinx
  - https://www.xilinx.com/products/design-tools/vivado.html
- *How to design algorithms for FPGA*
  - *to enable the advantages of FPGA*

Cong et al, *High-Level Synthesis for FPGAs: From Prototyping to Deployment*, 2011.

10 of 40

Outline

- Background & Motivation
  - Energy-efficient computing with FPGA
- Asynchronous parallel iterative algorithms
  - Communication model
- FPGA implementation
  - Techniques for implementation
- Summary

11 of 40

Iterative methods

- Many algorithms have the structure
  $$x(t+1) = f(x(t)), \qquad t = 0, 1, \cdots$$
  $$x(t) \in \mathbb{R}^n, f: \mathbb{R}^n \to \mathbb{R}^n.$$
- Component form
  $$x_i(t+1) = f_i\big(x_1(t), \cdots, x_n(t)\big), i = 1, \cdots, n.$$
- It can be parallelized by letting each one of n processors update a component x_i by f_i,
  - at each stage, the i-th processor
    - has the value of all components of $x(t)$ on which f_i depends,
    - computes the new value $x_i(t+1)$,
    - communicates it to other processors in order to start the next iteration.

Bertsekas and Tsitsiklis, Parallel and distributed computation : numerical methods. 1989.

12 of 40

Block iterative methods

- A coarse-grained parallelization of iterative methods with p processors.
- Each iteration
- $f_i: \mathbb{R}^n \to \mathbb{R}^{n_i}, \sum_{i=1}^{p} n_i = n,$
  $$x_i(t+1) = f_i\big(x(t)\big), \qquad i = 1, \cdots, p.$$
- Each f_i is updated by one of the p processors.
- Reasons
  - there may be too few processors available.
  - block-parallelization reduces the communication expense.

Bertsekas and Tsitsiklis, Parallel and distributed computation : numerical methods. 1989.

Jacobi and Gauss-Seidel Iterations

- Jacobi-type iteration
- all components, *simultaneously*, updated and made available for next iteration

$$x_i(t+1) = f_i\big(x_1(t), \cdots, x_n(t)\big)$$

- Gauss-Seidel Iteration
- components are updated, one at a time, using the most recently computed values of other components

$$x_i(t+1) = f_i\big(x_1(t+1), \cdots, x_{i-1}(t+1), x_i(t), \cdots, x_n(t)\big)$$

- updated in a cyclic order from 1 to n (or p for block iterative methods).
- One update of all components is a sweep.

Bertsekas and Tsitsiklis, Parallel and distributed computation : numerical methods. 1989.

Gauss-Seidel Iteration

- They incorporate the newest available information.
- Hence, they sometimes converge faster than the corresponding Jacobi-type algorithms.

- Gauss-Seidel algorithms can have different updating orders for f_i.
- In addition to the cyclic order, there are other orders.

Bertsekas and Tsitsiklis, Parallel and distributed computation : numerical methods. 1989.
Censor and Zenios, Parallel Optimization: Theory, Algorithms and Applications, 1997.

Parallelization of Jacobi and Gauss-Seidel Iterations

- Jacobi-type iteration
- all components, *simultaneously*, updated with the current values and made available for next iteration
- $$x_i(t+1) = f_i\big(x_1(t), \cdots, x_n(t)\big), \quad t = 0, 1, \cdots$$
- Synchronous update: next iteration waits until all updates are conducted

- Gauss-Seidel Iteration
- each update is computed with the latest available values of other components

$$x_i(t+1) = f_i(x(\tau_i(t))), \quad 0 \le \tau_i(t) \le t, t = 0, 1, \cdots$$

- The latest available may not be available, but earlier values, due to communication delay.
- Asynchronous update: iterations starts as soon as any recent values are available.

Bertsekas and Tsitsiklis, Parallel and distributed computation : numerical methods. 1989.
Avron, et al, Revisiting asynchronous linear solvers: provable convergence rate through randomization, Journal of ACM, 2015.

Stochastic gradient descent method

$$\min_x F(x) = \sum_i F_i(x)$$

- Instead of using full gradient, partial gradients of components are used.
- Its sequential version is

$$x(t+1) = f_i\big(x(t)\big) = x(t) - \lambda \nabla F_i(x(t))$$
$$f_i: \mathbb{R}^n \to \mathbb{R}^n$$

- Its asynchronous version is

$$x(t+1) = x(\tau_i(t)) - \lambda(t) \nabla F_i(x(\tau_i(t))), \tau_i(t) \le t.$$

- Convex constraint leads extra projections onto convex sets.
- Incremental gradient descent
- Ordered-subset algorithm in engineering
- Block-iterative method in image reconstruction

Kiwiel, Convergence of approximate and incremental subgradient methods for convex optimization, SIAM J. Optim., 2004.
Liu, et al, An Asynchronous Parallel Stochastic Coordinate Descent Algorithm, J. of Machine Learning Research, 2015.
Hannah and Yin, More Iterations per Second, Same Quality – why Asynchronous Algorithms may Drastically Outperform Traditional Ones, 2017. Preprint.

Kaczmarz Algorithm

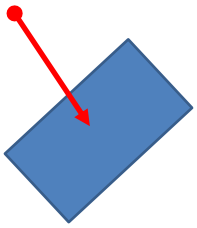
- To solve systems of linear equations,

$$Ax = b$$

- find (relaxed) projections onto each hyperplane iteratively,

$$x_j^{n+1} = x_j^n + \lambda_n A_{j,i} \frac{b_i - A_i x^n}{|A_i|^2}$$

- *It is an Gauss-Seidel iteration.*
  - Frank Natterer and Alfred Louis
- Also called the algebraic reconstruction technique (ART)



$$x^{n+1} = x^n + \lambda_n \frac{b_i - (A_i x^n)}{|A_i|^2} A_i^{\mathrm{tr}}$$

$$A_i x^{n+1} = A_i x^n + \lambda_n \frac{b_i - A_i x^n}{|A_i|^2} A_i A_i^{\mathrm{tr}}$$
$$= A_i x^n + \lambda_n (b_i - A_i x^n)$$
$$= b_i, \quad \text{if } \lambda_n = 1.$$

Kaczmarz Algorithm

- *It is a stochastic gradient descent.*

$$L(x) = \frac{1}{2} \sum_{i=1}^{m} \frac{\|A_i x - b_i\|^2}{|A_i|^2}$$

$$f_i(x) = \frac{1}{2} \frac{\|A_i x - b_i\|^2}{|A_i|^2}$$

$$\nabla_j f_i(x) = A_{j,i} \frac{A_i x - b_i}{|A_i|^2}$$

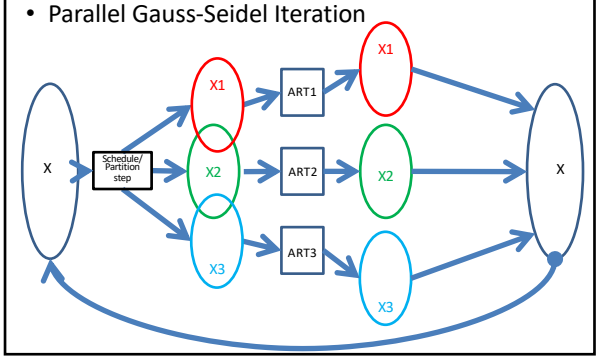
$$x_j^{n+1} = x_j^n + \lambda_n A_{j,i} \frac{b_i - A_i x^n}{|A_i|^2}$$

Its sequential version converges under the diminishing condition.
Its asynchronous iteration is not well covered in literature.

19 of 40

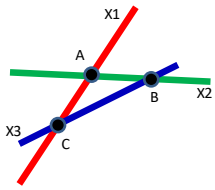
Asynchronous Parallel Kaczmarz (I)

- Parallel Gauss-Seidel Iteration



20 of 40

Asynchronous Parallel Kaczmarz (II)



X1	A, C	t1	A1, C1
X2	A, B	t2	A2, B2
X3	C, B	t3	C3, B3

In the next update of X1, will C1 or C3 be used?

- Not the conventional Kaczmarz method.
- More accelerators, more asynchronous updates with conflicts.

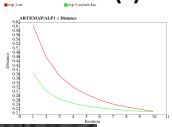
- The hope is that later updates will resolve the conflicts, even slowly.
- The solution is to use small or diminishing relaxations,

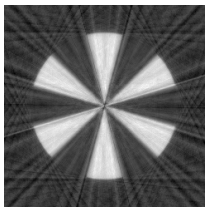
$$\frac{1}{t} \quad \sum \frac{1}{t} = +\infty$$

21 of 40

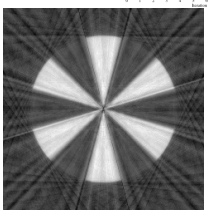
Experiments – Kaczmarz method(I)

- $\lambda_1 = 0.05, \lambda_{n+1} = \frac{A}{D+10n}, A = 1$
- No noise.
- 10 iteration
- Display window not adjusted





Kaczmarz

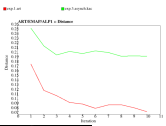


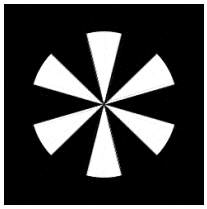
Asynchronous parallel Kaczmarz with 3 threads

22 of 40

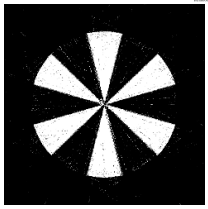
Experiments – Kaczmarz method(IV)

- $\lambda_n = 1.5$
- Gaussian noise: 0.05
- 10 iteration
- Display window = [0 1]





Kaczmarz



Asynchronous parallel Kaczmarz with 3 threads

23 of 40

Heuristics for Asynchronous Parallelism

- Heuristics
  - 欲速则不达:
  - Haste does not bring success.
  - Haste makes waste.
  - Eile mit Weile. (from Google translate)
  - Immer mit der Ruhe. (from Thomas Schuster)

Egyptian laborers lift the Stargate out of the ground in 1928
http://stargate.wikia.com/wiki/Tau'ri

- Examples
  - Lifting heavy object by a group people
  - Painting a big picture by a group of painters

- My lessons
  - ✓ Do not use full strength, but do it *slowly and slowly*.
  - ✓ No waiting does not mean to be hash.

- Expected performance
  - reconstruction images in early iterations even with small or diminishing relaxation, as ordered-subset methods do.

24 of 40

Convergence results (I)

- Linear system of equations

$$Ax = b$$

- [C. M.] A symmetric and positive definite
  - $\rho(A) < 1$: convergence.
  - $\rho(A) \geq 1$: no convergence.
- [L. W. S.] Consistent system under the same spectral condition.

- Kaczmarz algorithm, or ART, does not fit.

Rosenfeld, A case study in programming for parallel-processors, Communications of the ACM, 1969. (research report in 1967).
Chazan and Miranker, Chaotic relaxation, Linear Algebra and its Applications, 1969.
Liu, Wright, and Sridhar, An Asynchronous Parallel Randomized Kaczmarz Algorithm, 2014. Preprint.

Convergence results (II)

- $f: \mathbb{R}^n \to \mathbb{R}^n$ contraction.
- Components update
  - $f_i: \mathbb{R}^n \to \mathbb{R}^{n_i}$, $\sum_{i=1}^{p} n_i = n$,
    $x_i(t+1) = f_i(x(t))$
- Extension of [C. M.]
- Convergence for consistent linear systems
  - weakly diagonal dominated matrices
  - *irreducible* nonnegative matrix and positive solutions.
- Kaczmarz does not fit.

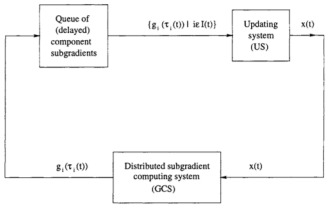
Bertsekas and Tsitsiklis, Parallel and distributed computation : numerical methods. 1989.
Frommer, On asynchronous iterations, Journal of Computational and Applied Mathematics, 2000.
Frommer and Spiteri, On linear asynchronous iterations when the spectral radius of the modulus matrix is one. Topics in numerical analysis, 2001.

One convergence result

- Asynchronous version

  $x(t+1) = x(t) - \lambda(t)\nabla f_i(x(\tau_i(t)))$.

- Convexity
- Diminishing relaxations
- Bounded delay
  - $\lambda \sim \frac{1}{n^q}, q \le 1$.
  - $t \le \tau_i(t) + D$.



Nedić, Bertsekas, Borkar, Distributed asynchronous incremental subgradient methods, 2001.

Diversified convergence results

- Different asynchronous mode
  - Master or without master?
  - Update all or partial solution
- Different combinations of obj
  - Fidelity + regularization
- Different communication assu
  - Mostly ambiguous
  - Not well modelled w.r.t compu
- Different relaxations
  - Fixed or diminishing?
- Deterministic or probabilistic
- Low precision implementatio
  - Big data applications typically

(a) Retrieve the current vector from the shared memory, say y.
(b) Compute the convex combination of y with the current vector in the local memory, say z:

$$z = \alpha_i y + (1-\alpha_i)z.$$

Store z in the local memory and in the shared memory.
(c) Perform the ith step of the ART algorithm on z, namely, compute

$$z := P_i z + \frac{\omega f_i}{R_i^T R_i} R_i.$$

(d) Go to (a).

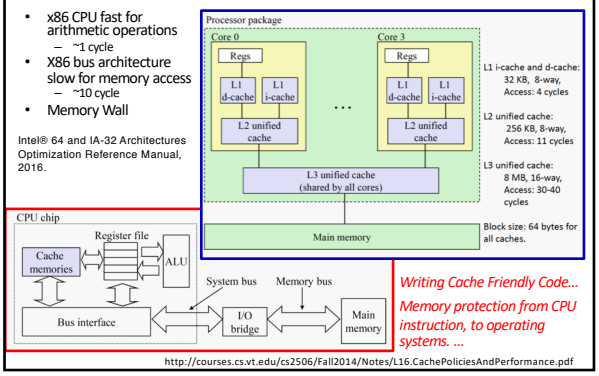
It is further assumed that the communication time between local and shared memories is negligibly small relative to the updating time in steps (b) and (c) and that no two processors access the shared memory at the same time.

Liu, Wright, et al, An Asynchronous Parallel Stochastic Coordinate Descent Algorithm, J. of Machine Learning Research, 2015.
Hannah and Yin, More Iterations per Second, Same Quality – why Asynchronous Algorithms may Drastically Outperform Traditional Ones, 2017. Preprint.
De Sa, et al, Understanding and Optimizing Asynchronous Low-Precision Stochastic Gradient Descent, ISCA'17, 2017.

CPU Operation vs Memory Access

- x86 CPU fast for arithmetic operations
  - ~1 cycle
- X86 bus architecture slow for memory access
  - ~10 cycle
- Memory Wall

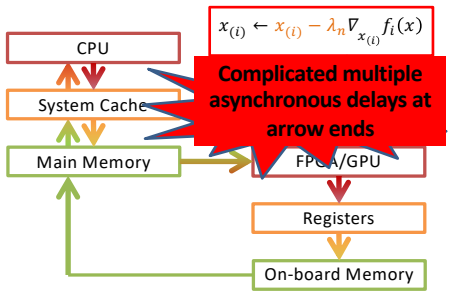
Intel® 64 and IA-32 Architectures Optimization Reference Manual, 2016.



Writing Cache Friendly Code…
Memory protection from CPU instruction, to operating systems. …

http://courses.cs.vt.edu/cs2506/Fall2014/Notes/L16.CachePoliciesAndPerformance.pdf

Asynchronous updates in data flow



$$x(t+1) = x(\tau_i(t)) - \lambda(t)\nabla f_i(x(\tau_i(t))), \tau_i(t) \le t.$$

Outline

- Background & Motivation
  - Energy-efficient computing with FPGA
- Asynchronous parallel iterative algorithms
  - Communication model
- FPGA implementation
  - Techniques for implementation
- Summary

31 of 40

High-level Synthesis C->FPGA

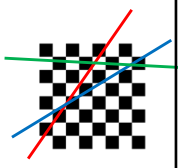
- Energy-efficient accelerator-rich architecture
- Can be optimized under the performance, power, and cost constraints
- Extensive use of accelerators (algorithmic blocks)
- Limited onboard memory vs others
  - Extendable
- VivadoHLS of Xilinx
  - https://www.xilinx.com/products/design-tools/vivado.html
- *How to design algorithms for FPGA?*
  - ➢ Asynchronous iterative parallel algorithms

32 of 40

Beam-based SGD for Mumford-shah Regularization

$$\int |R(f)-b|^2 + \alpha \int v^2 |\nabla f|^2 + \beta \int \varepsilon |\nabla v|^2 + \frac{(1-v)^2}{4\varepsilon}$$

$$= \sum_m f(x, b_m) = \sum_m f_m(x)$$



- Gamma-approximation by edge indicator
- Beam-based gradient descent,

$$x_{(i)} \leftarrow x_{(i)} - \lambda_n \nabla_{x_{(i)}} f_i(x_{(i)})$$

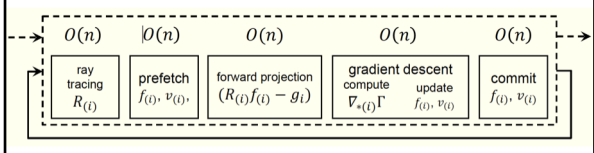
- Alternative minimization algorithm for f and v
- Enable fine-grained parallelism, natural for CT
- Accelerators with minimum data communication
- Reduce communication latency and back-projection computation

FPGA Acceleration for 3D Low-Dose Tomographic Reconstruction, submitted to IEEE Transactions on Parallel and Distributed Systems, 2019.

33 of 40

Implementation Techniques (I)

- Pipeline for streaming dataflow



- Achieve Load balance in the hardware pipeline.

34 of 40

Tile-based memory optimization

- Frequent large sized I/O of image and projection data in 3D.
  - Memory intensive.
- Tile the image and projection data
  - Prefetching and buffering strategy.
  - *Mathematical problem has not formulated.*
- Achieve a high data reuse rate
- Save the memory bandwidth
- Increase performance

Linjun Qiao, Poster at this workshop.

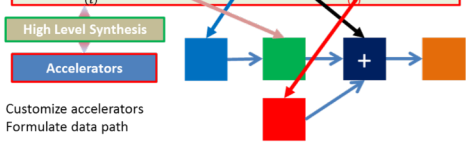
35 of 40

FPGA for XCT with Mumford-Shah

$$\|A(x)-b\|^2 + R(x) = \sum_m f_m(x)$$

$$x_{(i)} \leftarrow x_{(i)} - \lambda_n \nabla_{x_{(i)}} f_i(x)$$

$$\nabla_{x_{(i)}} f_i(x) = 2A_i{}^T(A_i(x)-b) + \nabla_{x_{(i)}} R(x)$$

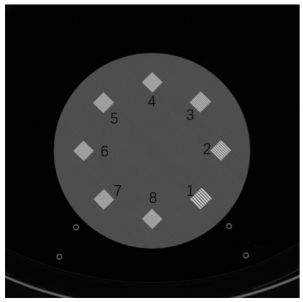


Customize accelerators
Formulate data path

36 of 40

Low-dose CT Experment

- CT ACR 464 phantom on the
- SOMATOM Definition AS at UCLA.
- Voltage: 120KV
- Current: 215, 150, 100, or 50mA.



37 of 40

Results (SSIM)

Data set	method	mean	max	min	std
215mA	ours	0.991	0.996	0.942	0.0611
	vendor	1.000	1.000	1.000	0.0000
150mA	ours	0.981	0.989	0.965	0.0077
	vendor	0.997	0.998	0.995	0.0010
100mA	ours	0.972	0.990	0.901	0.0258
	vendor	0.997	0.998	0.995	0.0011
50mA	ours	0.976	0.989	0.920	0.0188
	vendor	0.982	0.998	0.995	0.0298

- 8.5 minutes to reconstruct a typical 3D lung image with a quality comparable with the vendor's result.

38 of 40

Outline

- Background & Motivation
  - Energy-efficient computing with FPGA
- Asynchronous parallel iterative algorithms
  - Communication model
- FPGA implementation
  - Techniques for implementation
- Summary

39 of 40

Summary

- Mumford-Shah regularization for XCT
  - FPGA implementation with a quality comparable with the vendor's result.
- Asynchronous parallel computation is necessary for architectures with rich computing units
  - multi-core CPU/GPU
  - accelerators rich FPGA
  - multi-node supercomputer
- Algorithms with communication model
  - Memory access/communication expense should not be ignored

40 of 40

Thank you for your attention.

- Supported by
  - National Science Foundation of China (61520106004).
  - National Basic Research and Development Program of China (973 Program) (2015CB351803).