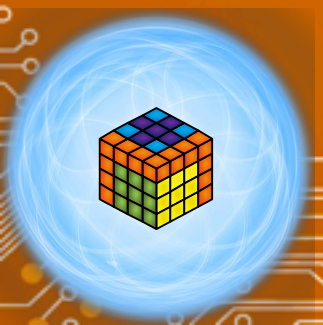
THE UNIVERSITY OF TEXAS AT AUSTIN
Cockrell School of Engineering

Floating Point Architecture Extensions for Optimized Matrix Factorization

Ardavan Pedram
Robert van de Geijn
Andreas Gerstlauer



THE UNIVERSITY OF TEXAS AT AUSTIN
UT ECE
ELECTRICAL & COMPUTER ENGINEERING

Outline

- Motivation and Vision
- Related Work and Background
- Base Architecture
- Factorization Algorithms
- Floating Point Unit Extensions
- Experimental Results
- Conclusion and Future Work

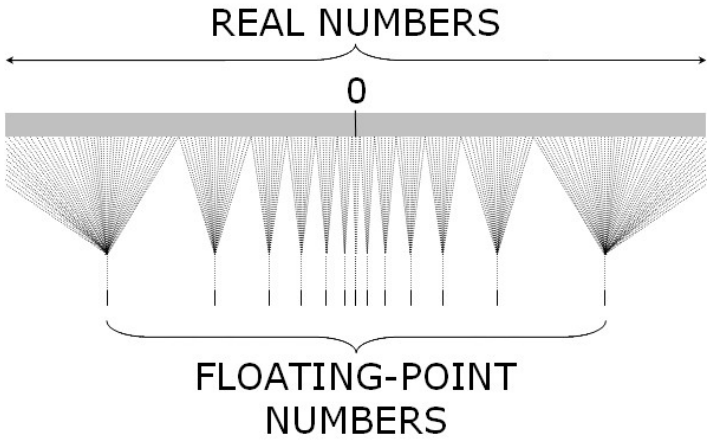
Outline

- Motivation and Vision
- Related Work and Background
- Base Architecture
- Factorization Algorithms
- Floating Point Unit Extensions
- Experimental Results
- Conclusion and Future Work

Floating Point and Numerical Stability

- Floating-point Representation is Limited

- Numerical Algorithms on CPUs
  - Ensure numerical stability
  - Avoid overflow and underflow
  - Introduce extra complexity on the critical path of algorithm
  - Sacrifice performance for numerical stability
    - Extra operations



Codesign is a Solution

- Our Goal (Three matrix factorizations)
  - Provide hardware support
  - Reduce complexity in the algorithm

- Solution
  - Provide extensions to the floating-point unit
  - Simplify the algorithm exploiting extended architecture

Outline

- Motivation and Vision
- Related Work and Background
- Base Architecture
- Factorization Algorithms
- Floating Point Unit Extensions
- Experimental Results
- Conclusion and Future Work

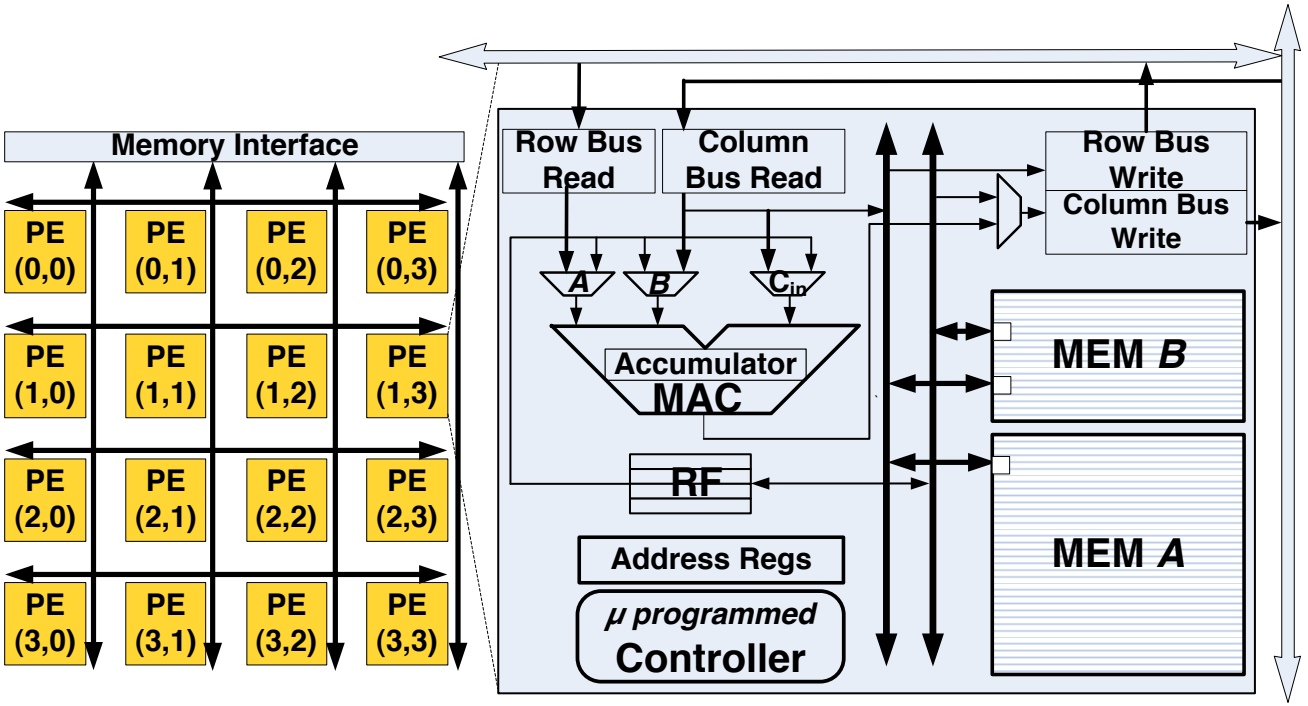
Related work

- Hybrid Solutions
  - [Volkov'08]
    - Cholesky, LU, and QR
  - [Agullo'11]
    - Multi-core Multi-GPU
    - QR
- Entirely on GPU
  - [Kerr'09]
    - Householder QR
  - [Galoppo'05]
    - LU
- FPGAs
  - [Wu'09]
    - LU without pivoting
  - LAPACKrc [Gonzalez'09]
    - Customized solvers
  - [Tai'11]
    - Scalable QR
  - [Aslan'09]
    - Householder QR
    - Unified unit

Outline

- Motivation and Vision
- Related Work and Background
- Base Architecture
- Factorization Algorithms
- Floating Point Unit Extensions
- Experimental Results
- Conclusion and Future Work

The Linear Algebra Core (LAC) [Pedram'11]



- Scalable 2-D Array of $n_r \times n_r$ Processing Elements (PEs)
  - Up to 50 GFLOPS/W efficiency @ 1GHz
  - Specialized floating-point units with 1 MAC/cycle throughput [Vangal'06]
  - Broadcast buses (no need to pipeline up to n_r=16)
  - Distributed memory architecture
  - Distributed, PE-local micro-programmed control

Outline

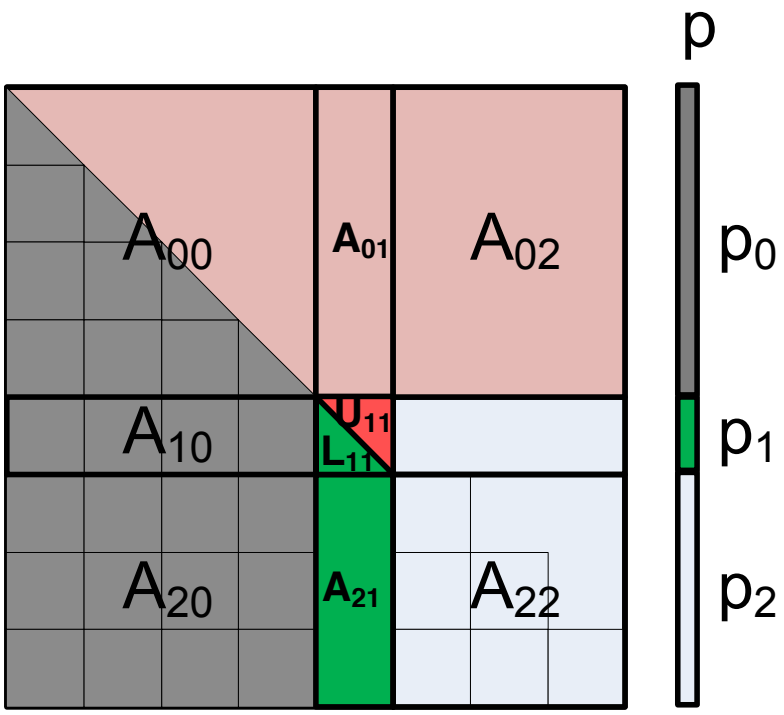
- Motivation and Vision
- Related Work and Background
- Base Architecture
- Factorization Algorithms
- Floating Point Unit Extensions
- Experimental Results
- Conclusion and Future Work

Matrix Factorizations

- LU Factorization with Partial Pivoting
  - Gaussian elimination
  - Partial pivoting due to finite precision of floating-point arithmetic
    - avoid element growth
    - avoid catastrophic cancelation
    - avoid divide by zero
- Cholesky Factorization
  - Special case of LU for Symmetric Positive Definitive (SPD) Matrices
  - Half the FLOPS of LU
- QR Factorization (Householder Reflections)
  - Most stable solution
  - Twice the FLOPS of LU

LU with Partial Pivoting

- A = LU
  - Square $A \in R^{n \times n}$
  - Unit triangular $L \in R^{n \times n}$
  - Upper triangular $U \in R^{n \times n}$
- Solution by recursively partitioning
- Rearrange (pivot) the rows of the matrix as the computation unfolds



$$\left[\begin{pmatrix} A_{11} \\ A_{12} \end{pmatrix}, p_1\right] := LU\begin{pmatrix} A_{11} \\ A_{12} \end{pmatrix}$$

LU with Partial Pivoting

Partition *A*, *L*, and *U*:

$$A = \left(\begin{array}{c|c} \alpha_{11} & a_{12}^T \\ \hline a_{21} & A_{22} \end{array}\right)$$

$$L = \left(\begin{array}{c|c} 1 & 0 \\ \hline l_{21} & L_{22} \end{array}\right) \qquad U = \left(\begin{array}{c|c} \upsilon_{11} & u_{12}^T \\ \hline 0 & U_{22} \end{array}\right)$$

α_{11} and υ_{11} are scalars and *A=LU* means:

$$\left(\begin{array}{c|c} \alpha_{11} & a_{12}^T \\ \hline a_{21} & A_{22} \end{array}\right) = \left(\begin{array}{c|c} \upsilon_{11} & u_{12}^T \\ \hline l_{21}\upsilon_{11} & L_{22}U_{22} + l_{21}u_{12}^T \end{array}\right)$$

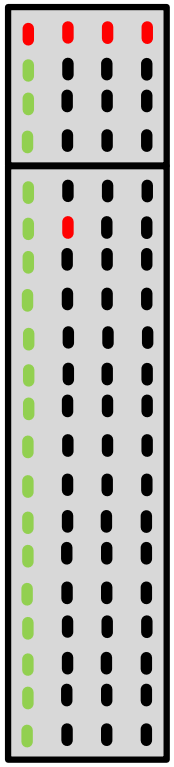
Which in turn means:

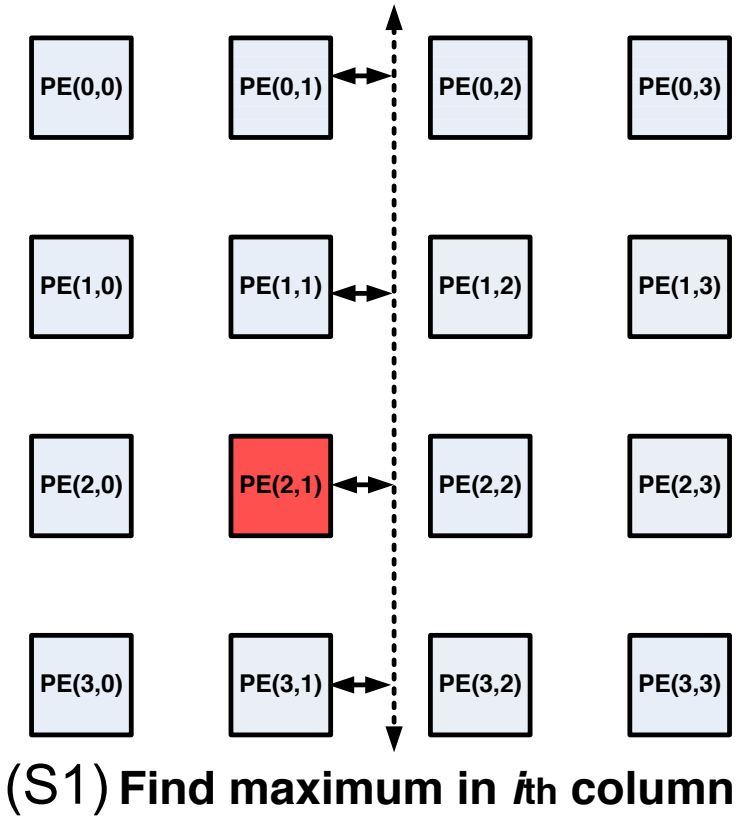
$$\left(\begin{array}{c|c} \alpha_{11} = \upsilon_{11} & a_{12}^T = u_{12}^T \\ \hline a_{21} = \upsilon_{11} l_{21} & A_{22} - l_{21}u_{12}^T = L_{22}U_{22} \end{array}\right)$$

We can thus compute *L* and *U* in place for matrix *A*:

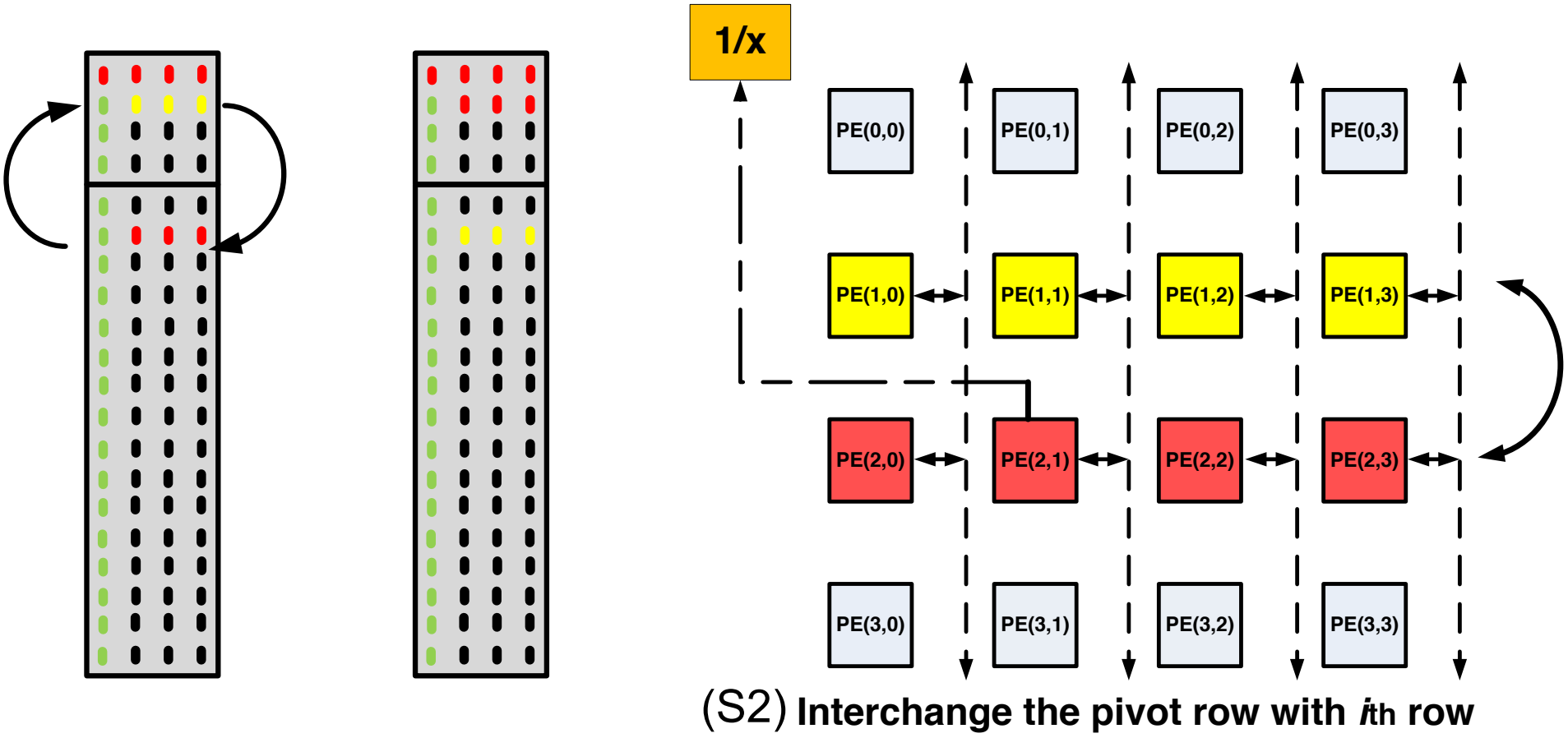
$$\left(\begin{array}{c|c} \alpha_{11} := \upsilon_{11} \text{ (no-op)} & a_{12}^T := u_{12}^T \text{ (no-op)} \\ \hline a_{21} := l_{21} = a_{21}/\upsilon_{11} & A_{22} := \mathrm{LU}(A_{22} - l_{21}u_{12}^T) \end{array}\right)$$

LU with Partial Pivoting on LAC (1)

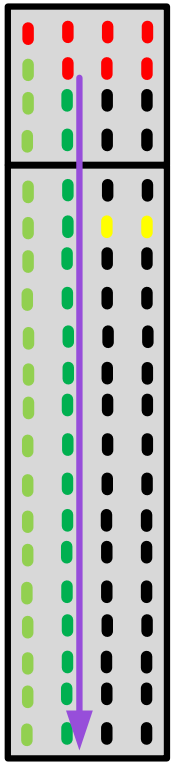


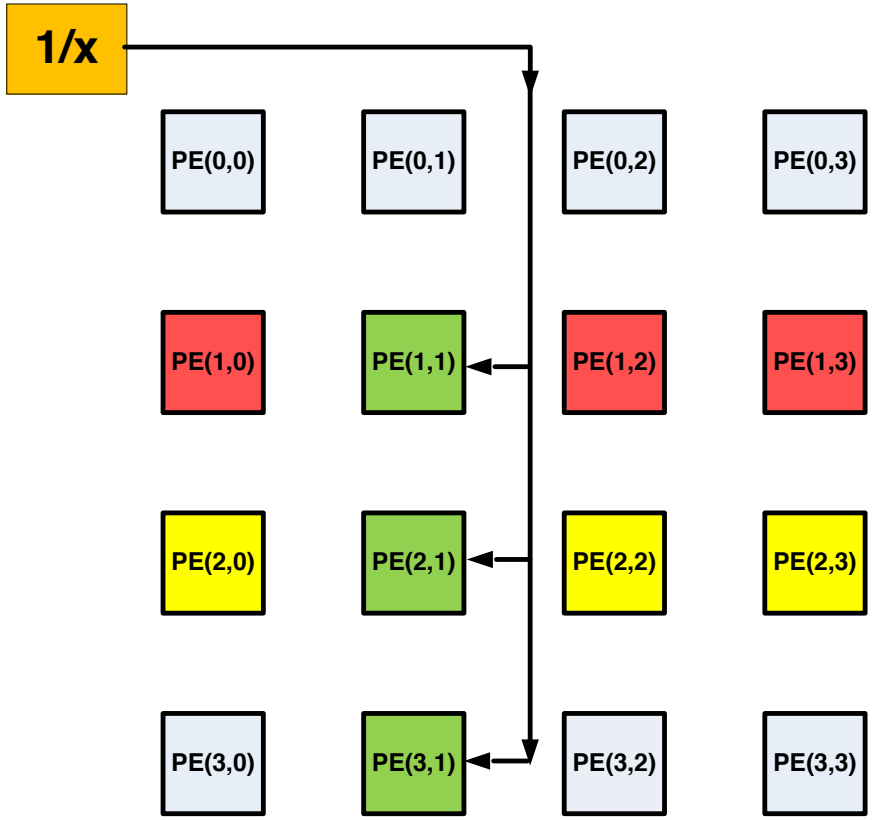


LU with Partial Pivoting on LAC (2)



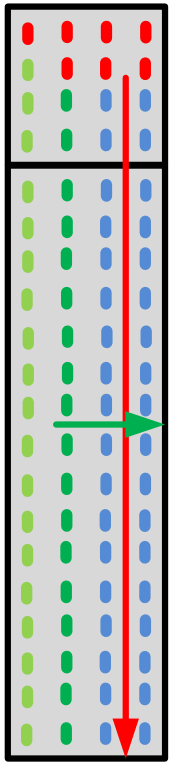
LU with Partial Pivoting on LAC (3)

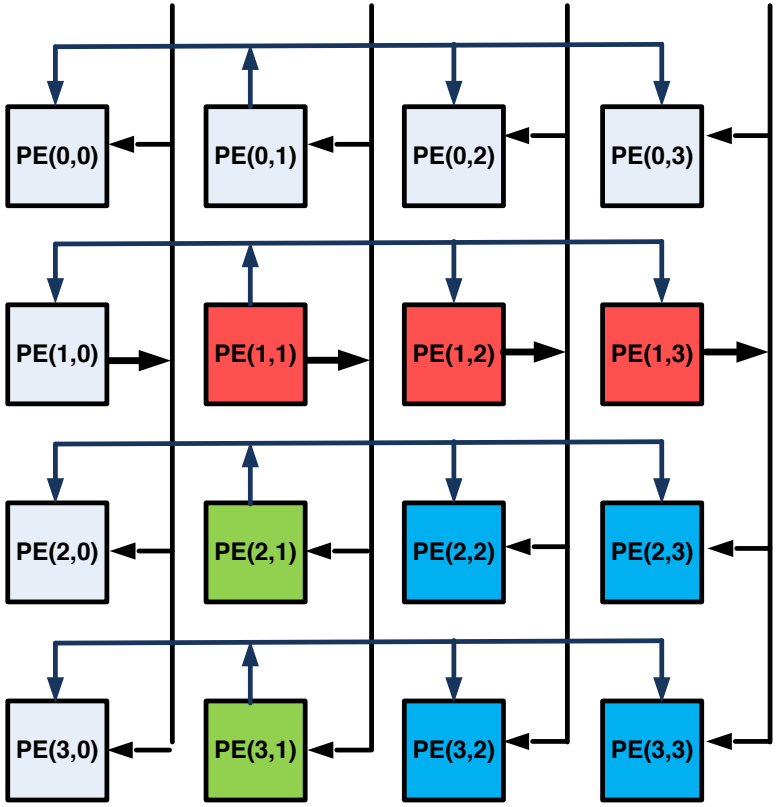




(S3) **Scale the *i*th column with pivot**

LU with Partial Pivoting on LAC (4)





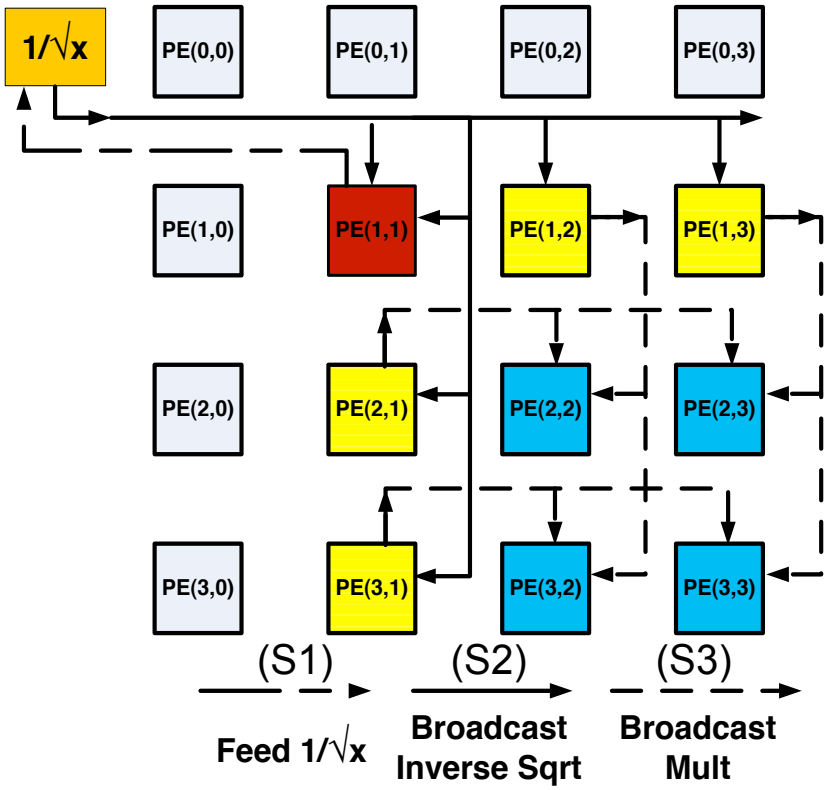
(S4) **Rank-1 update**

LU with Partial Pivoting on LAC

- Pivoting on the Panel to Avoid
  - Division by zeros on diagonal
  - Element growth
- Sequential Algorithm
  - Except for GEMM updates
- Special Demands
  - Reciprocal (1/X) unit
  - Find maximum element of a vector

Cholesky on LAC

- $A = LL^T$
  - Symmetric Positive Definite (SPD) matrix $A \in R^{n \times n}$
  - Lower triangular matrix $L \in R^{n \times n}$
- Sequential Algorithm
  - Except for GEMM updates
- Simple Control
  - 7-stage state machine in PEs
- Special Demands
  - Inverse Square-root unit
  - Square root unit



Householder QR

- *A = QR (Householder)*
  - Square matrix $A \in R^{n \times n}$
  - Orthogonal matrix $Q \in R^{n \times n}$
  - Upper triangular matrix $R \in R^{n \times n}$
- Householder Transformation
  - Vector-Norm is the key operation
- Special Demands
  - Normalization

$$x = \chi_0, \cdots, \chi_{n-1}$$

$$\|x\|_2 = \sqrt{\chi_0^2 + \chi_2^2 + \ldots + \chi_{n-1}^2}$$

$$t = \max |x_i| ;$$

$$y = x/t = 1/t * (\chi_0, \cdots, \chi_{n-1});$$

$$\|x\|_2 := t \times \|y\|_2$$

Outline

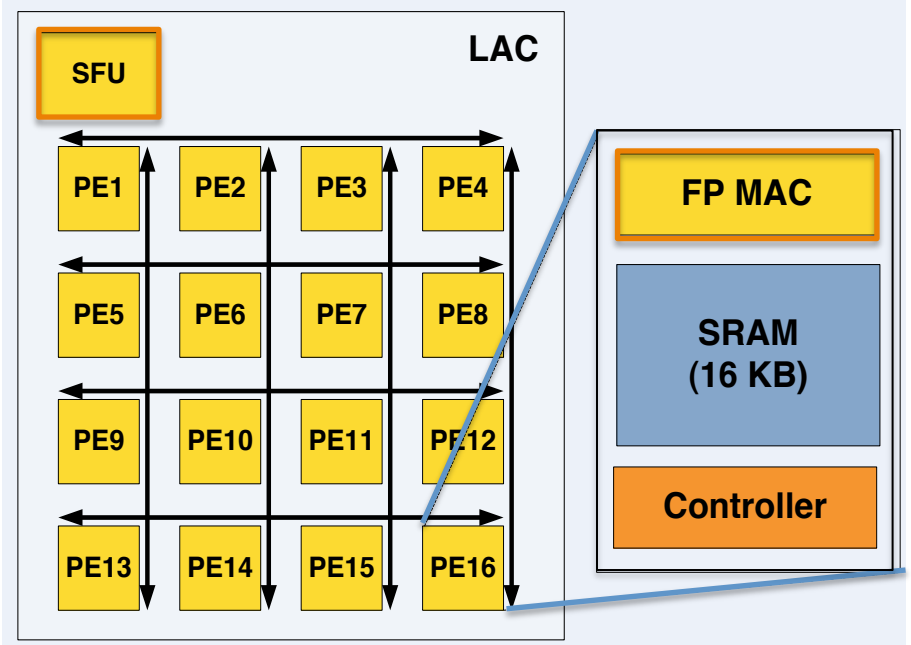
- Motivation and Vision
- Related Work and Background
- Base Architecture
- Factorization Algorithms
- Floating Point Unit Extensions
- Experimental Results
- Conclusion and Future Work

Summary of Extensions

- Cholesky
  - Inverse square-root
  - Square-root
- LU with Partial Pivoting
  - Maximum finder
  - Reciprocal
- QR (Vector-Norm)
  - Option 1:
    - Maximum finder
    - Reciprocal
  - Option 2:
    - Extend the floating-point representation
    - Add one extra bit to the exponent

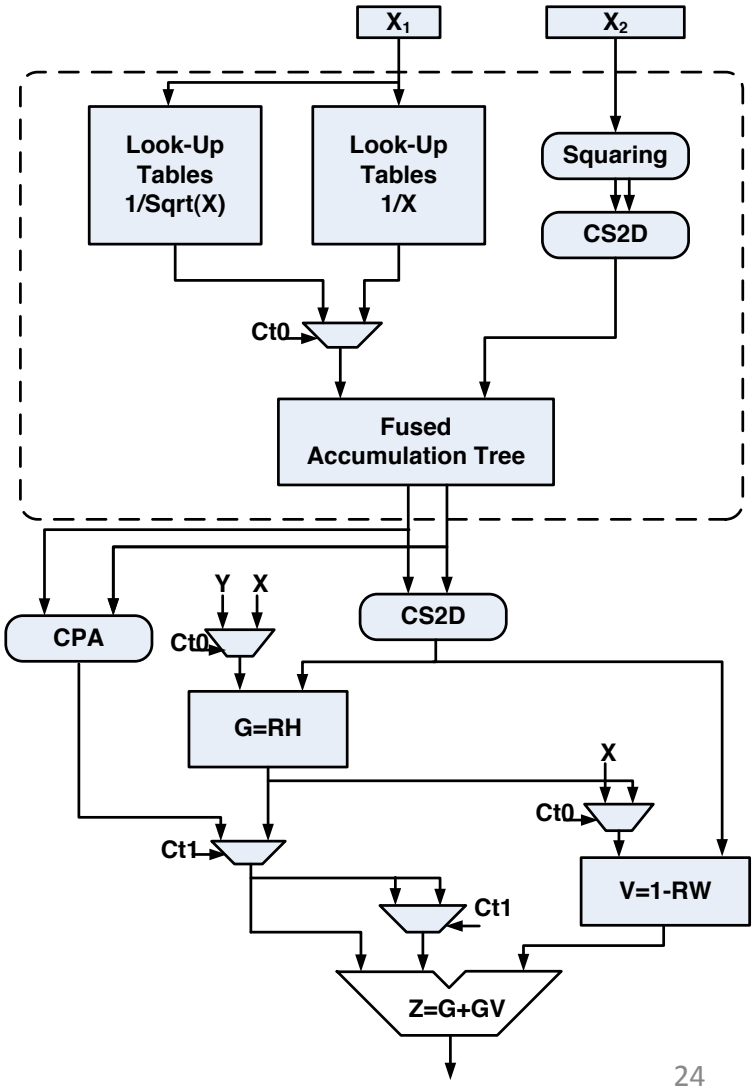
Modifications to the LAC

- Special Function Unit (SFU)
  - Inverse square-root
  - Square-root
  - Reciprocal (1/X)
- FP MAC
  - Maximum Finder
  - Exponent Extension



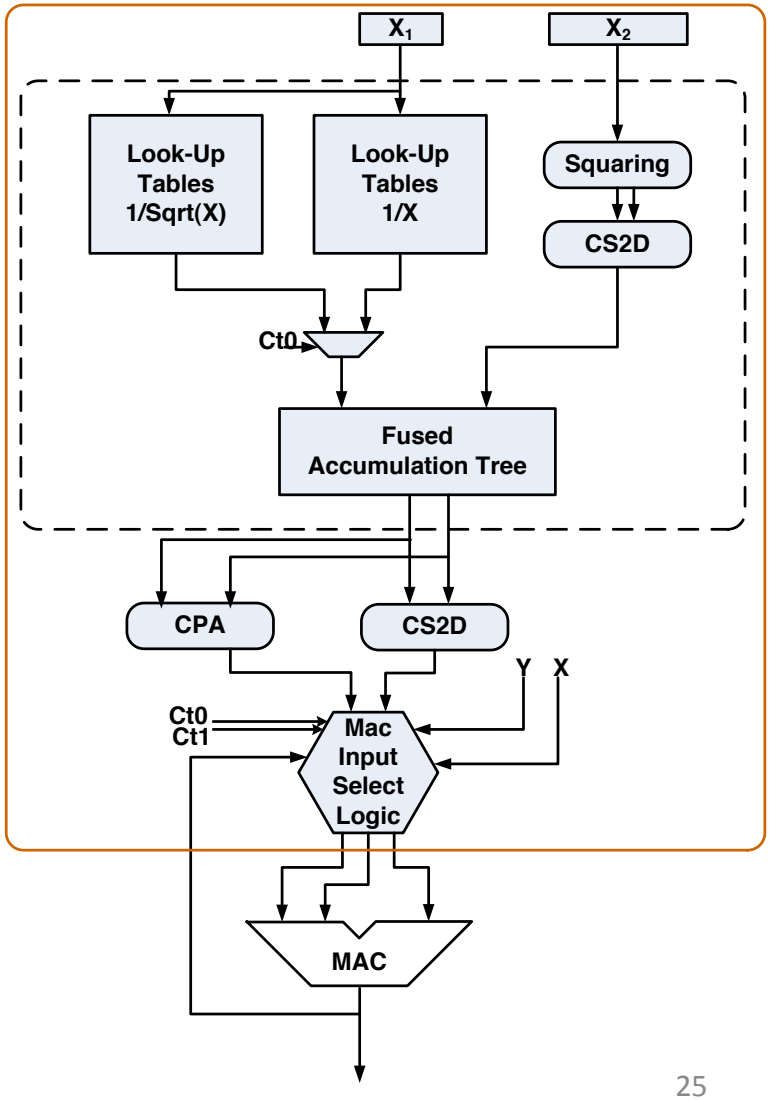
Special Function Unit (SFU) [Piñeiro'02]

- Supported operations
  - Division
  - Reciprocal
  - Square-root
  - Invert square-root
- Implementation
  - Multiplicative method
  - 29-30 bit approximation look-up
  - Single iteration of modified Goldschmidt



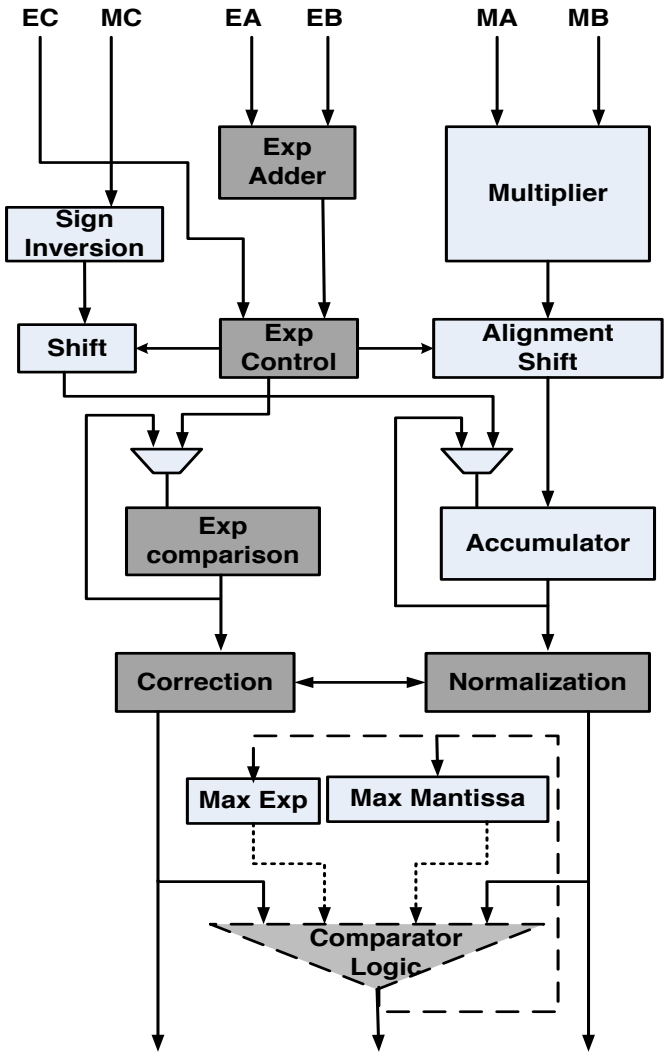
Merged SFU with MAC

- For Linear Algebra
  - No need for high throughput
  - No resource conflict
- Utilize the Existing MAC Unit
- Save Area and Power
- Augment Diagonal PEs
  - Avoid extra communications
  - Simpler control



Extensions to the Base MAC in [Jain'10]

- LU with Pivoting
  - Keep the maximum value produced so far
    - Reduce the search space for pivot
    - Only n_r values to compare among PEs
  - Simple logic for comparator
    - Activated with normalization
    - Not on the critical path
    - Causes no extra delay
- Vector-norm
  - An extra exponent bit (gray)
  - Result feeds SFU
  - SFU produces typical FP value



Outline

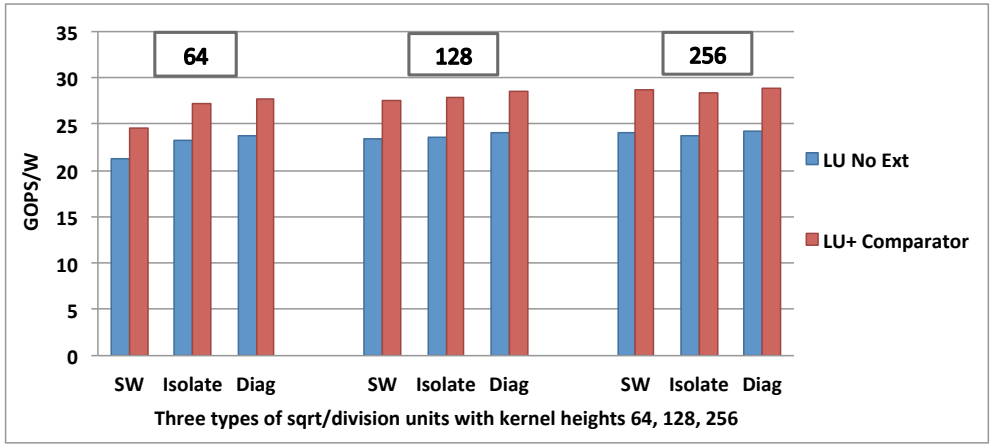
- Motivation and Vision
- Related Work and Background
- Base Architecture
- Factorization Algorithms
- Floating Point Unit Extensions
- Experimental Results
- Conclusion and Future Work

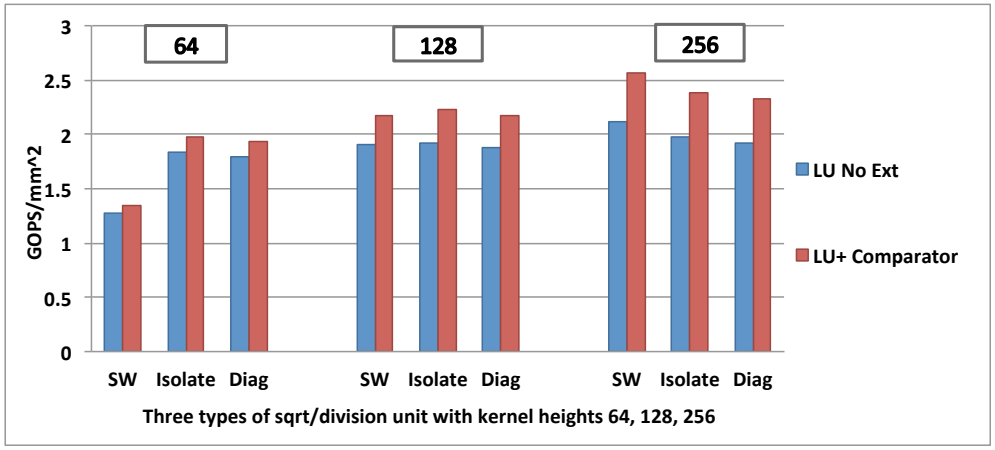
Power and Performance Analysis

- Component Selections
  - @ 1GHz target frequency
  - MAC units (45nm) [Galal'10]
  - Storage model with CACTI 6.0
    - Pure SRAM Model
  - Interconnect
    - CACTI 6.0
    - AMBA AHB [Lahiri'04]
    - [Wolkotte.2009]
- Two Case for MAC Extension
  - Only maximum finder
  - Extended exponent bit
- Three Cases of SFU Support
  - No SFU (μ-programmed in SW)
  - Isolated separate SFU
  - Diagonal PEs
    - Merged unit
    - Simpler control
- Unblocked Inner Kernels
  - $n_r \times n_r$ Cholesky
  - $k \times n_r$ LU and QR
    - k = 64, 128, 256

LU with Partial Pivoting

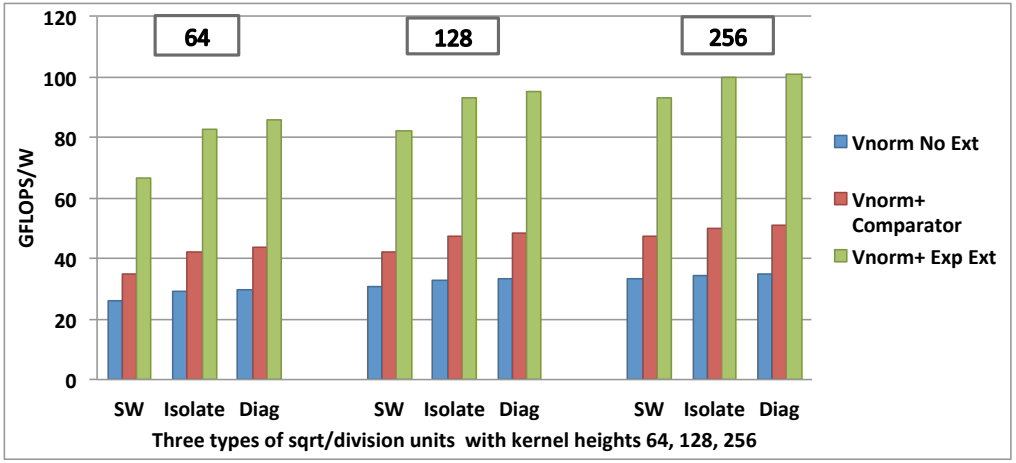
- Inner Kernel Size $k \times n_r$:
  - k = 64, 128, 256
- No Benefit From SFU
  - Reciprocal and pivoting performed concurrently
- Pivoting is Dominating
- Benefits of Comparator
  - 20% speed up
  - 15% energy savings

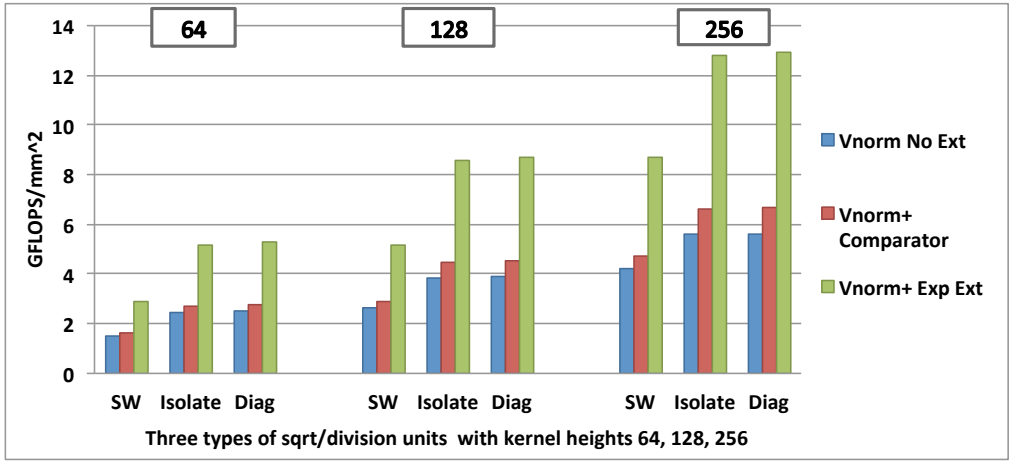




Vector Norm

- Inner Kernel Size $k \times n_r$:
  - k = 64, 128, 256
- SFU
  - 30% speedup
  - 10% energy savings
- Exponent Bit
  - 100% speedup
  - 60% energy savings





Outline

- Motivation and Vision
- Related Work and Background
- Base Architecture
- Factorization Algorithms
- Floating Point Unit Extensions
- Experimental Results
- Conclusion and Future Work

Conclusions and Future Work

- Matrix Factorizations
  - Cholesky
  - LU with Partial Pivoting
  - Householder QR
- FPMAC Modifications
  - Decrease/Facilitate algorithm complexity
- Extension
  - Inverse Square-root
  - Reciprocal
  - Extended exponent
  - Maximum finder

➢ Integration of the Core Into a Heterogeneous System

➢ Block Level Algorithmic Analyses

➢ Compare Trade-offs
  - Efficiency vs. Flexibility

Thank You and Welcome to Texas

FPMAC[Vangal et al.2006]