An FPGA Implementation of the Hestenes-Jacobi Algorithm for Singular Value Decomposition

Xinying Wang, Joseph Zambreno
Reconfigurable Computing Lab
Department of Electrical and Computer Engineering
Iowa State University

21st Reconfigurable Architectures Workshop
May 19-20, 2014, Phoenix, USA

Overview

- Motivation
- Theoretical Background
- Modified Hestenes-Jacobi Approach
- Our Hestenes-Jacobi SVD Architecture
  - Hestenes Preprocessor
  - Jacobi Rotation Component
  - Update Operator
  - The Cyclic Ordering
- Implementation and Evaluation
- Conclusion and Future Work

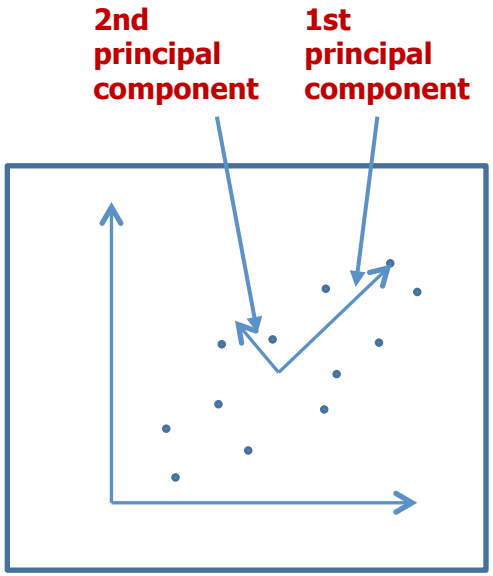
Singular Value Decomposition (SVD)

Let $A \epsilon Rm^{\times} n$, then there exist $U \epsilon Rm \times m$, $V \epsilon Rn \times n$ and $\Sigma \epsilon Rm \times n$ such that

$$A = U \Sigma V T$$

where $\Sigma = diag\ (\sigma 1, ..., \sigma r) \epsilon Rm \times n$ with singular values of A, U and V are orthogonal matrices.

- SVD is the most popular technique to perform Principal Component Analysis (PCA) for dimensionality reduction
- SVD is widely employed in many scientific and engineering applications



Singular Value Decomposition (SVD)

- Singular Value Decomposition (SVD) is a computationally-expensive procedure with computational complexity of $O(n^3)$.

- SVD Application: Background modeling from video surveillance [1]

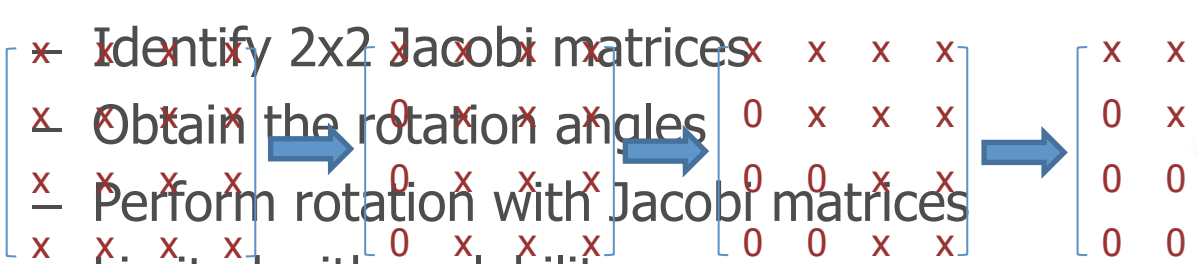
  (Using Matlab on a desktop PC with a 2.33 GHz Core 2 Duo processor and 2 GB RAM)

frames	resolution	SVD time
200	176x144	43mins
250	168x120	36mins

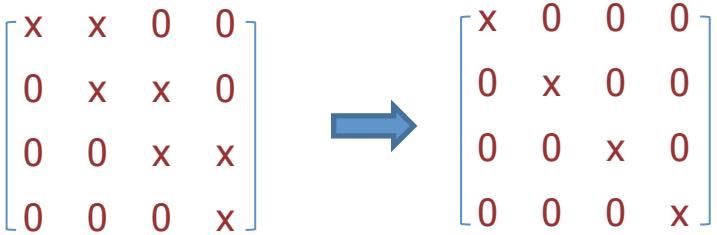
SVD Algorithms

- **Householder factorization**
  - Bi-diagonalize the matrix with Householder factorization
  - Zero out the remaining non-zero off-diagonals with recursive implicit QR factorization
  - Inherent data dependency challenges the parallelism of SVD

- **Two sided Jacobi Rotation**
  - Identify 2x2 Jacobi matrices
  - Obtain the rotation angles
  - Perform rotation with Jacobi matrices
  - Limited with scalability



$$J(p,q,\phi) = \begin{bmatrix} 1 & & & & & & 0 \\ & \ddots & & \vdots & & \iddots & \\ & & \cos\phi & \cdots & \sin\phi & & \\ & \cdots & 0 & \cdots & 0 & \cdots & \\ & & -\sin\phi & \cdots & \cos\phi & & \\ & \iddots & & \vdots & & \ddots & \\ 0 & & & & & & 1 \end{bmatrix}$$

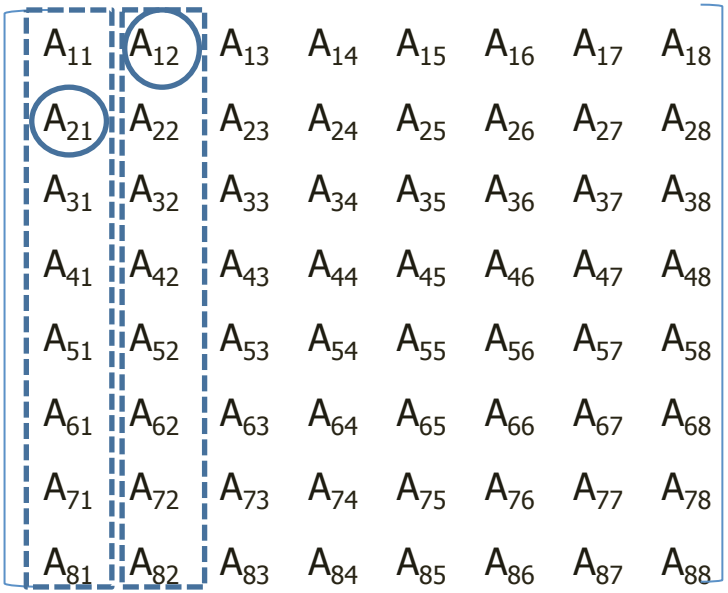


$$J^{l'} \cdot \begin{pmatrix} A_{pp} & A_{pq} \\ A_{qp} & A_{qq} \end{pmatrix} \cdot J^{r} = \begin{pmatrix} A''_{pp} & 0 \\ 0 & A''_{qq} \end{pmatrix}$$

SVD Algorithms (Cont.)

- **Hestenes-Jacobi approach**

  Hestenes discovered the equivalence between zeroing out an off-diagonal a_{ij} and orthogonalizing the i^{th} and j^{th} vectors through plane rotation



$$d_i = A_{1i}^2 + A_{2i}^2 + A_{3i}^2 + \ldots\ldots + A_{ni}^2$$
$$d_j = A_{1j}^2 + A_{2j}^2 + A_{3j}^2 + \ldots\ldots + A_{nj}^2$$
$$c_{ij} = A_{1i}*A_{1j} + A_{2i}*A_{2j} + A_{3i}*A_{3j} + \ldots\ldots + A_{ni}*A_{nj}$$

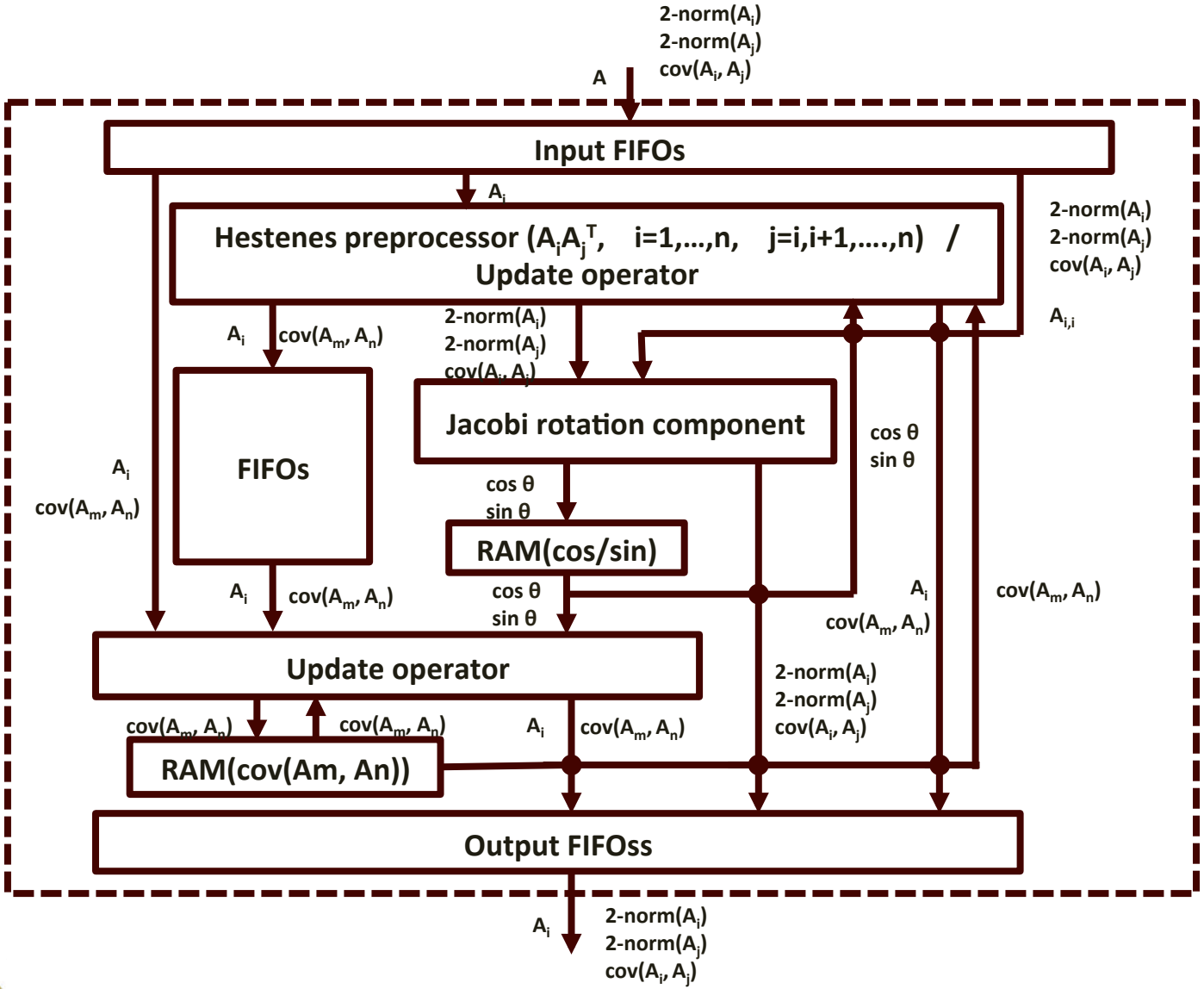
Modified Hestenes-Jacobi approach

- Our modified solution is to directly update the squared norm covariance without repetitively calculating the squared norms and covariaces
- The existing GPU implementation suffered from the iterative thread synchronizations, whose performance even worse than the CPU designs [2]
- The existing FPGA implementation with fixed point computations suffered from iterative design with duplicated computations [3]

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} & A_{15} & A_{16} & A_{17} & A_{18} \\ A_{21} & A_{22} & A_{23} & A_{24} & A_{25} & A_{26} & A_{27} & A_{28} \\ A_{31} & A_{32} & A_{33} & A_{34} & A_{35} & A_{36} & A_{37} & A_{38} \\ A_{41} & A_{42} & A_{43} & A_{44} & A_{45} & A_{46} & A_{47} & A_{48} \\ A_{51} & A_{52} & A_{53} & A_{54} & A_{55} & A_{56} & A_{57} & A_{58} \\ A_{61} & A_{62} & A_{63} & A_{64} & A_{65} & A_{66} & A_{67} & A_{68} \\ A_{71} & A_{72} & A_{73} & A_{74} & A_{75} & A_{76} & A_{77} & A_{78} \\ A_{81} & A_{82} & A_{83} & A_{84} & A_{85} & A_{86} & A_{87} & A_{88} \end{bmatrix}$$

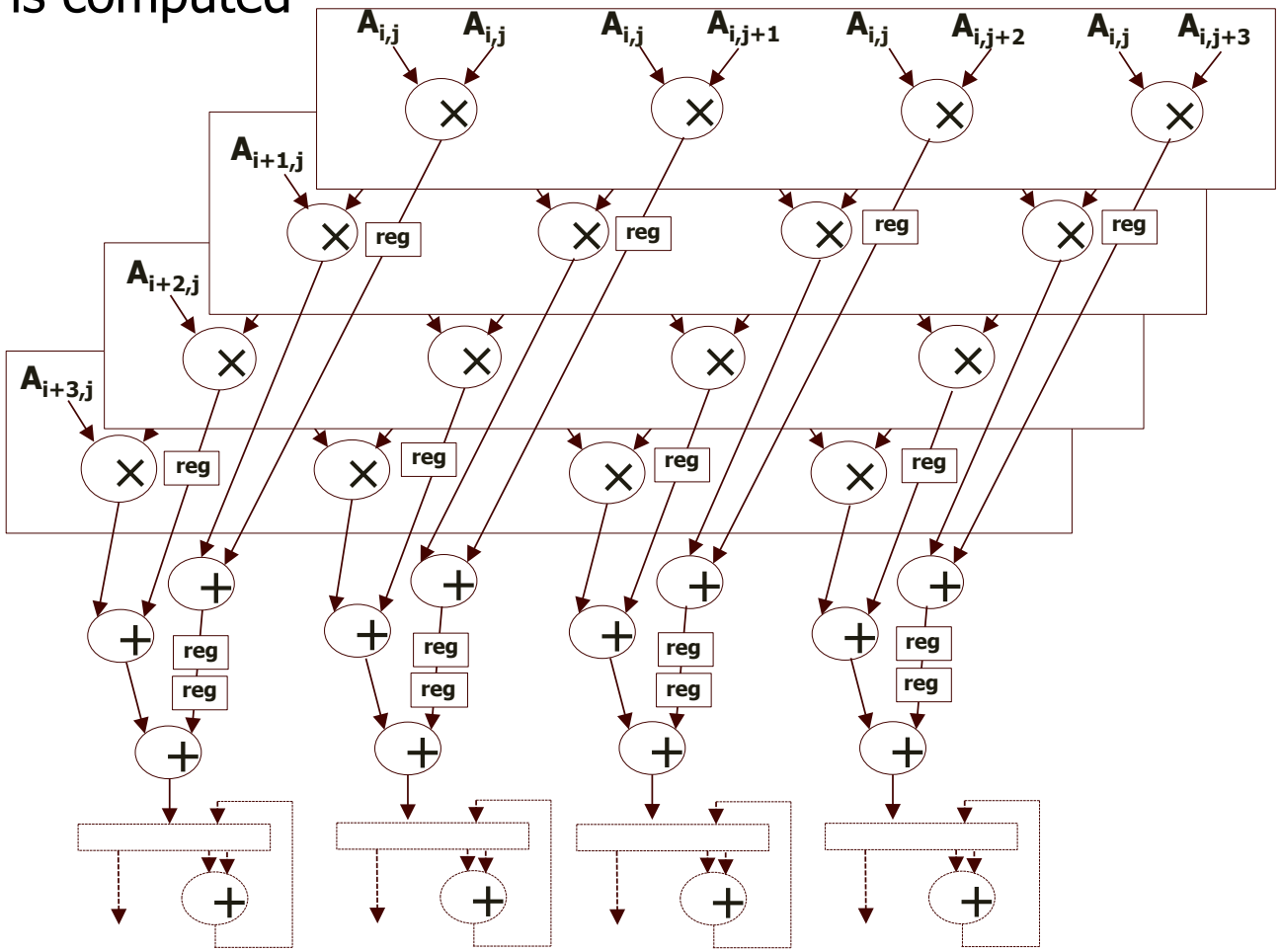
$$d_i = A_{1i}^2 + A_{2i}^2 + A_{3i}^2 + \ldots\ldots + A_{ni}^2$$
$$d_j = A_{1j}^2 + A_{2j}^2 + A_{3j}^2 + \ldots\ldots + A_{nj}^2$$
$$c_{ij} = A_{1i}*A_{1j} + A_{2i}*A_{2j} + A_{3i}*A_{3j} + \ldots\ldots + A_{ni}*A_{nj}$$

Our Architecture



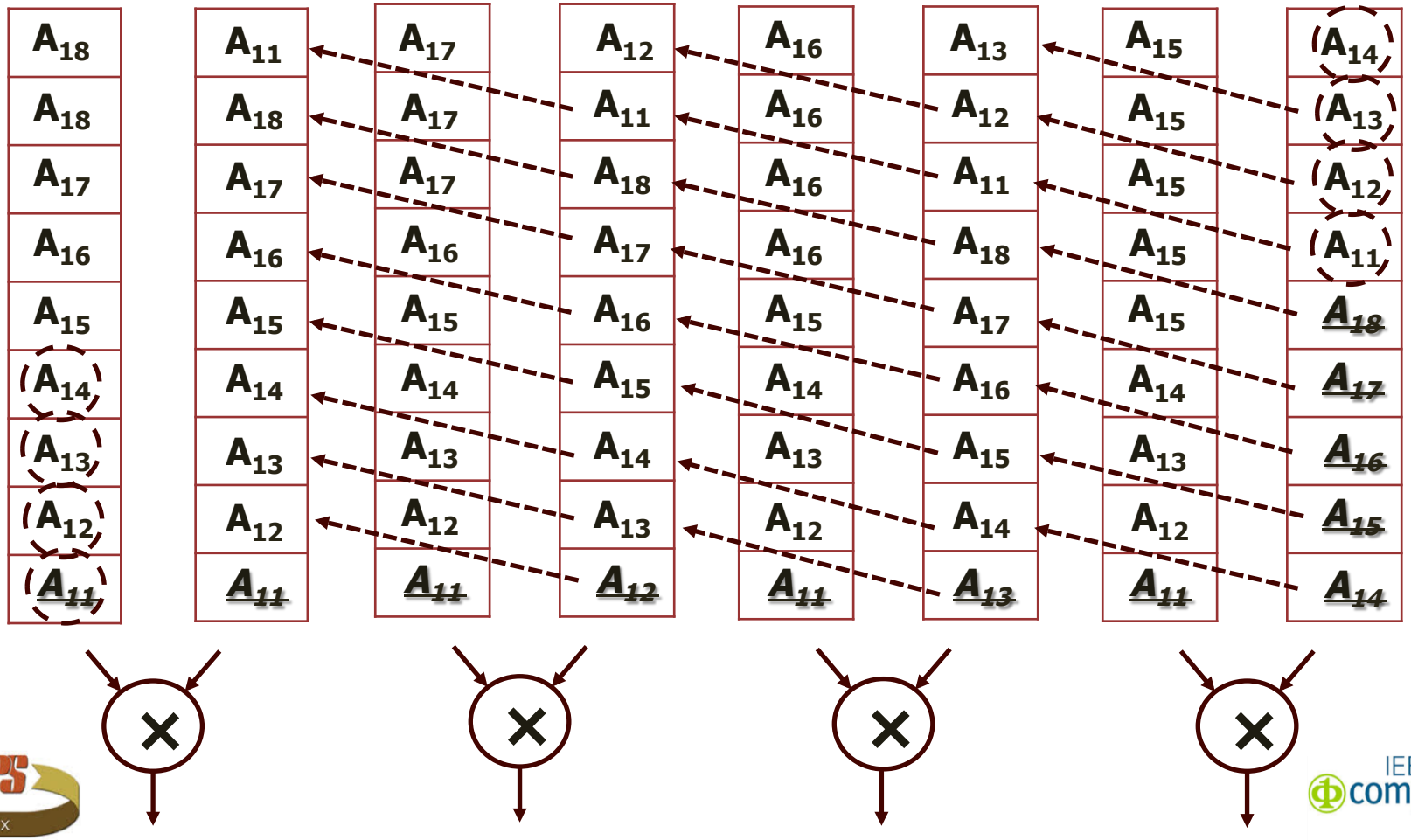
The Hestenes preprocessor

- The Hestenes preprocessor is responsible for calculating the squared column 2-norms and the covariances between column vectors, in which $A^T_i * A_j$ is computed



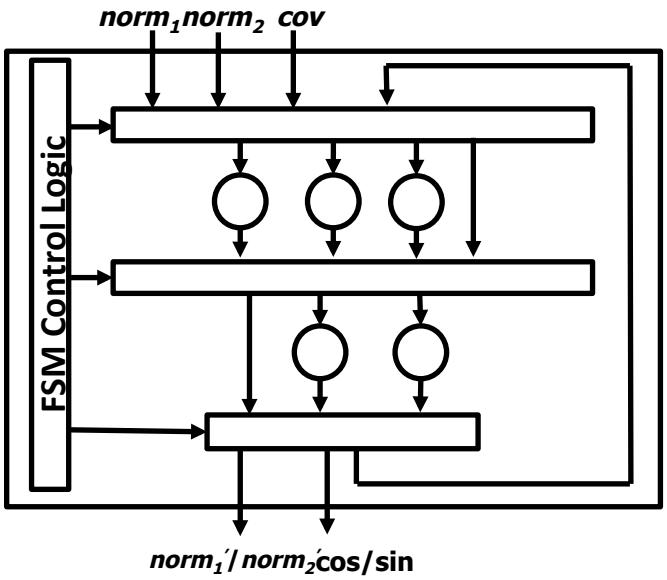
The Hestenes preprocessor (Cont.)

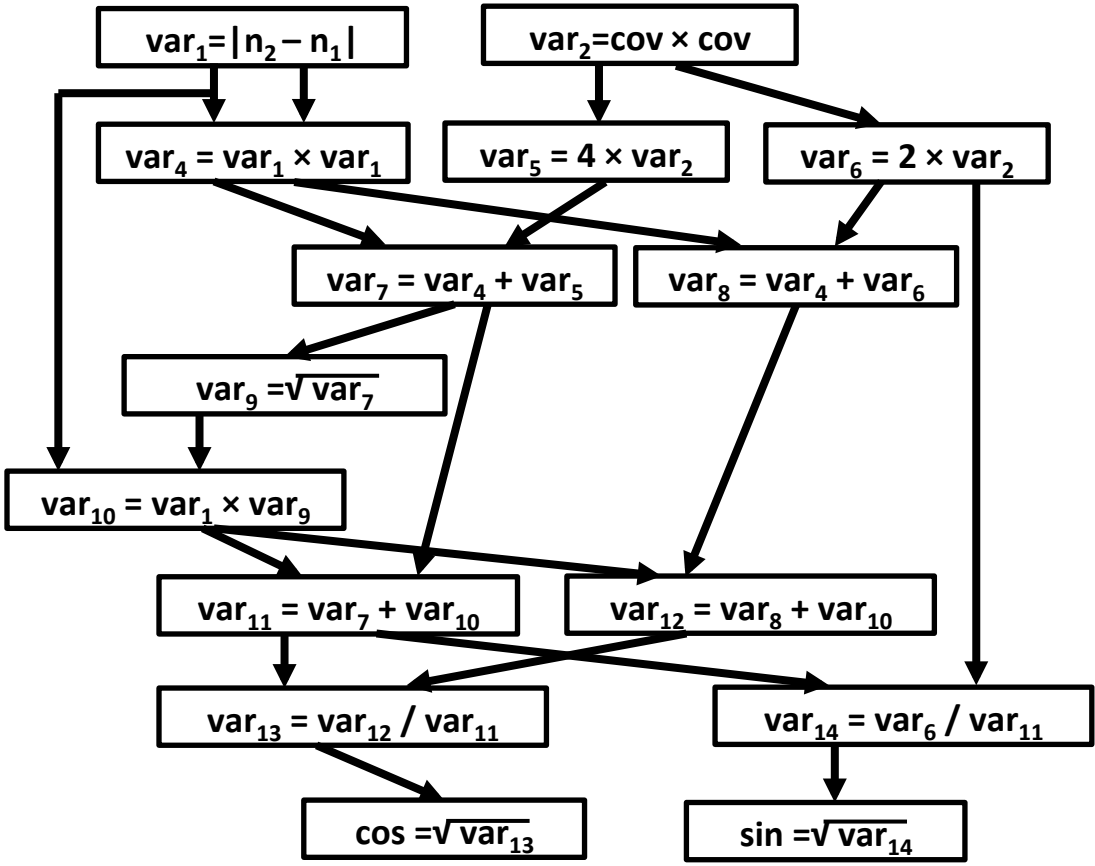
- The Hestenes preprocessor is responsible for calculating the squared column 2-norms and the covariances between column vectors, in which $A^T_i * A_j$ is computed



The Jacobi Rotation Component

- Jacobi rotation component performs the orthogonal transformation between two column vectors through a series of operations on their squared column 2-norms and the covariance between them



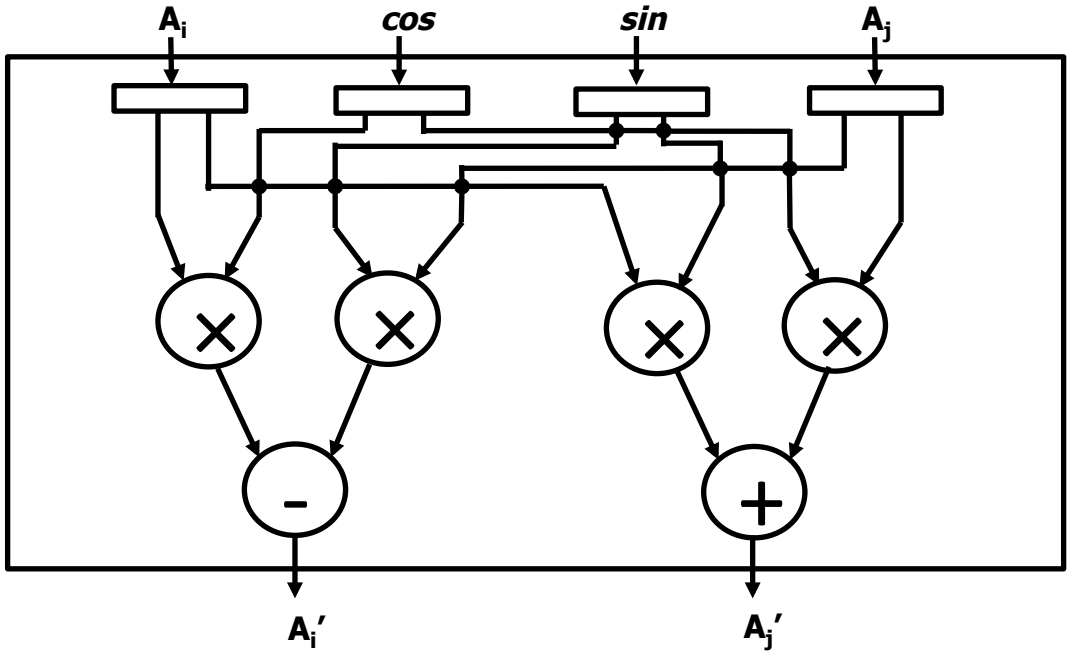


Update Operator

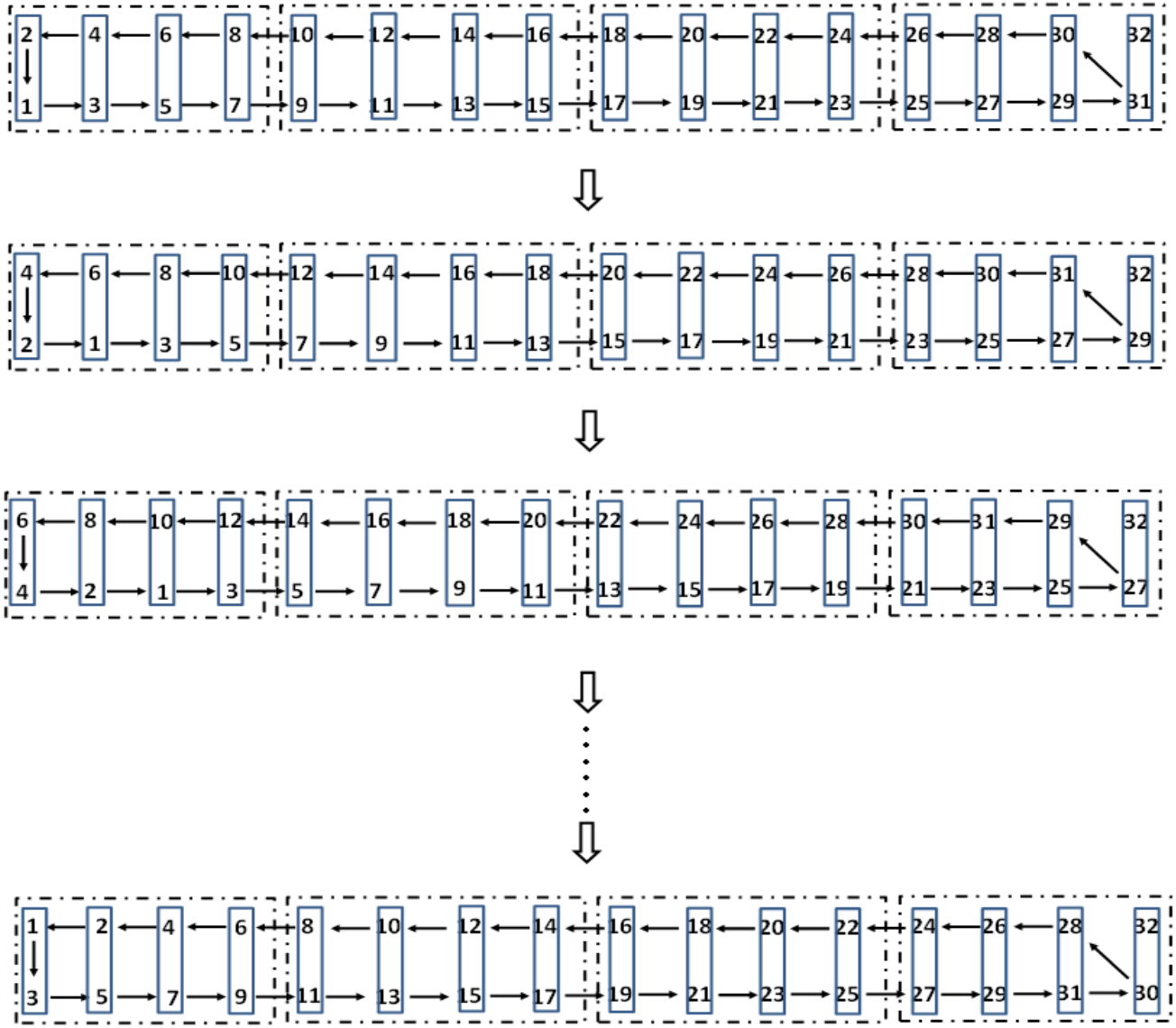
- The Update operator is responsible for updating column elements and covariance which are affected by the processed rotations
- To optimize the use of hardware resources, the Hestenes preprocessor is able to be reconfigured to function as multiple update kernels

$$A'i = Ai \times cos - Aj \times sin$$

$$A'_j = Ai \times sin + Aj \times cos$$



The Cyclic Order for Vector Pairing



Performance analysis

- A single Xilinx Virtex-5 XC5VLX330 FPGA on Convey HC-2 system is used
- The system is tested by executing at 150Mhz for 6 iterations, which is believed sufficient for achieving convergence with certain thresholds

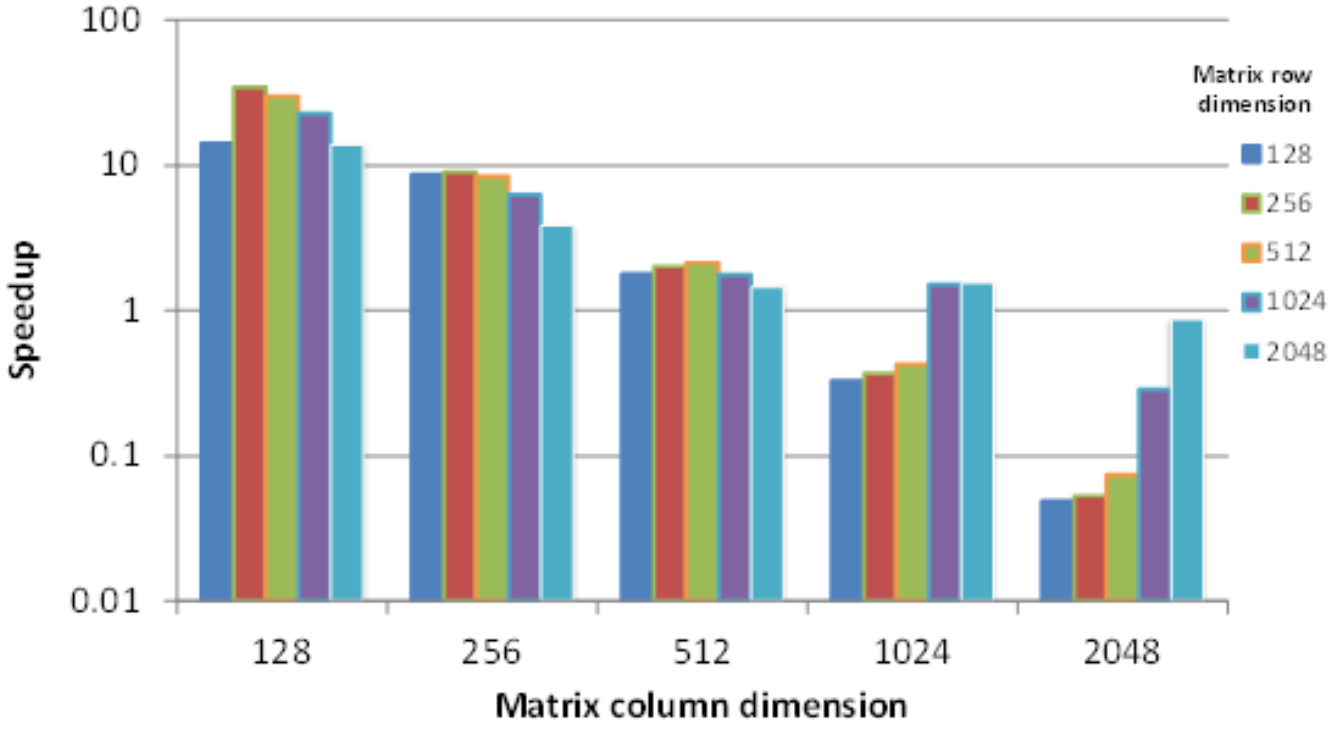
Execution time in seconds.

$m \backslash n$	128	256	512	1024
128	4.39×10^{-3}	6.30×10^{-3}	1.01×10^{-2}	1.79×10^{-2}
256	2.52×10^{-2}	3.30×10^{-2}	4.84×10^{-2}	7.94×10^{-2}
512	1.70×10^{-1}	2.01×10^{-1}	2.63×10^{-1}	3.87×10^{-1}
1024	1.23	1.35	1.61	2.01

- The experimental results demonstrate that the execution time grows significantly as the number of matrix columns increases, which determines the number of covariance, whose computation dominates the overall performance

Performance analysis (Cont.)

- The dimensional speedups that can be achieved range from 3.8x to 43.6× for matrices with column sizes from 128 to 256 and row dimensions from 128 to 2048



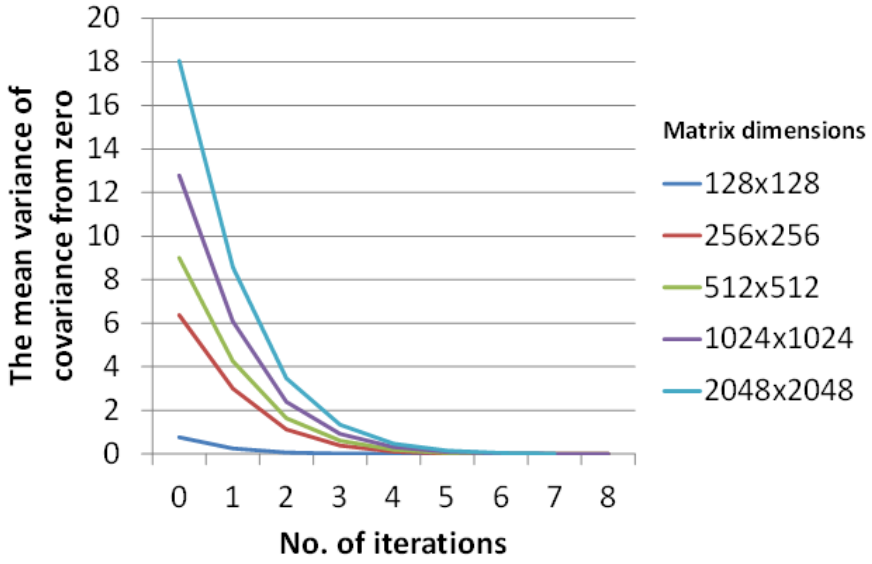
Performance analysis (Cont.)

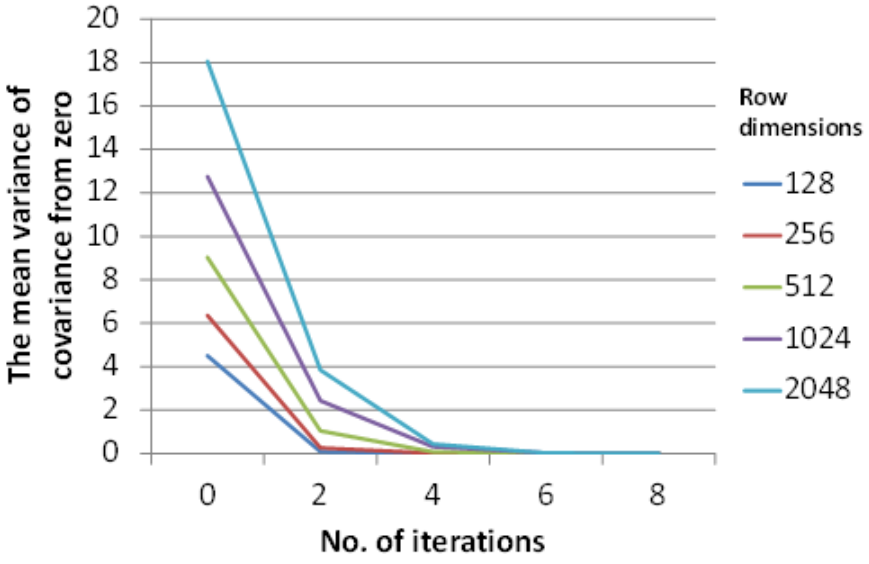
- The GPU-based implementation, which ran 106.90ms and 1022.92ms to decompose a 128×128 and a 256×256 matrix respectively, failed to achieve any speedup compared to a conventional software solution [2]

- The FPGA-based design was devised to perform fixed-point operations, which can only analyze the matrices with the size up to 32×128 due to the limitation of on-chip memory. It takes 24.3143ms to decompose the largest analyzed matrix with the dimensions of 32×127 [3]

Convergence Analysis

- The mean absolute deviations from zero of the covariance after being processed by a number of iterations are analyzed and reasonable convergence can be achieved within 6 iterations of operations for matrices of dimensions up to 2048





Conclusion and Future Work

- An FPGA-based hardware architecture is proposed to perform Singular Value Decomposition with Hestenes-Jacobi algorithm
- Dimensional speedups range from 3.8x to 43.6x for matrices with column dimensions from 128 to 256 and row sizes from 128 to 2048
- Our analysis provides direction for potential improved solution and our design will be extended to accelerate SVD applications

An FPGA Implementation of the Hestenes-Jacobi Algorithm for Singular Value Decomposition

Xinying Wang, Joseph Zambreno
Reconfigurable Computing Lab
Department of Electrical and Computer Engineering
Iowa State University

21st Reconfigurable Architectures Workshop
May 19-20, 2014, Phoenix, USA

References

- [1] E. J. Candes, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?" *The Computing Research Repository*, vol. 0912.3599, 2009.
- [2] C. Kotas and J. Barhen, "Singular Value Decomposition utilizing parallel algorithms on graphical processors," in *Proceedings of OCEANS 2011*, Sept. 2011, pp. 1–7.
- [3] L. Ledesma-Carrillo, E. Cabal-Yepez, R. de J Romero-Troncoso, A. Garcia-Perez, R. Osornio-Rios, and T. Carozzi, "Reconfigurable FPGA-Based unit for Singular Value Decomposition of large m x n matrices," in *Proceedings of International Conference on Reconfigurable Computing and FPGAs (ReConFig)*, Nov.-Dec. 2011, pp. 345–350.