ADAPTIVE SCALABLE SVD UNIT FOR FAST PROCESSING OF LARGE LSE PROBLEMS

IÑAKI BILDOSOLA & UNAI MARTINEZ-CORRAL KOLDO BASTERRETXEA
MOTIVATION

- Previous Project
  - Computational Intelligence applications
  - Real Time Computation
  - LSE problems
  - Resulting Matrices:
    - Large-scale
    - Rank-deficient
    - Ill-conditioned matrices
  - Implementation in MicroBlaze → Too Much Delay
  - Need for acceleration → parallel processing & optimized faster implementation → FPGA
- We needed an algorithm numerically robust
- Struggling with deficient matrices
- Struggling with non-square matrices
- Avoid the Inverse calculation
- Obtain the Pseudoinverse
- Good Base for Problem Reduction (future Work)
SELECTING SVD METHOD
WHY ONE-SIDED JACOBI?

- Motivation
- Selecting the Algorithm
- Selecting SVD Method
- Speaking About the Accuracy
- Improving Previous Work
- Results
- HW FPGA Implementation
- Why and How Scalable?
- Conclusion
- Future Work

• Easily Parallelizable → Jacobi
• What more?
  • Purely non-conflicting → one-sided
  • Optimizing the managed unit size → one-sided
• Main features
  • Based on Column Pairs Orthogonalization
  • Given’s rotations by Rutishauser formulas

\[
\tan(2\theta_{ij}^k) = \frac{2 * (A_{ij}^k * A_{ij}^k)}{||A_{ij}^k||^2 - ||A_{ij}^k||^2} \quad AV = W \quad A = USV^T
\]
SPEAKING ABOUT THE ACCURACY IMPOSED CONDITIONS

- Motivation
- Selecting the Algorithm
- Selecting SVD Method
- Speaking About the Accuracy
- Improving Previous Work
- Results
- HW FPGA Implementation
- Why and How Scalable?
- Conclusion
- Future Work

- Computing Precisions → Sets The Maximum
- Matrix Conditioning → Impacts on the Accuracy
  - $K(A) = 10^k ; CP = 10^m ; Solution = 10^{m-k}$.
  - Matrix Size → Possible Accumulated error.
  - $K(A) \&$ Matrix Size Impact close to $CP$ → No Solution or very degraded
SPEAKING ABOUT THE ACCURACY: IMPOSED CONDITIONS

- Motivation
- Selecting the Algorithm
- Selecting SVD Method
- Speaking About the Accuracy
- Improving Previous Work
- Results
- HW FPGA Implementation
- Why and How Scalable?
- Conclusion
- Future Work

Randomly generated matrices $f(\text{Size}, k(A))$

Errors: Our Algorithm ($'$) in Single Vs Matlab($''$) in double

- Singulars = Maximum Normalized Error $= \frac{\sigma' - \sigma''}{\sigma''}$
- Inverse $= \|A_{inv}' - A_{inv}''\|^2$
- Remainder $= \|A - \text{SVD}\|^2$

![Graph showing error comparison](image1)
![Graph showing error comparison](image2)
- Motivation
- Selecting the Algorithm
- Selecting SVD Method
- Speaking About the Accuracy
- Improving Previous Work
- Results
- HW FPGA Implementation
- Why and How Scalable?
- Conclusion
- Future Work

**Threshold Value**

- Iterative Algorithm Finisher → Orthogonalization
- User Defined Parameter → Time & Accuracy Trade-off
- Error Saturation Phenomenon → (Imposed Conditions)
IMPROVING PREVIOUS WORK
LEARNING FROM OTHERS

- Motivation
- Selecting the Algorithm
- Selecting SVD Method
- Speaking About the Accuracy
- Improving Previous Work
- Results
- Why and How Scalable?
- Conclusion
- Future Work

• **Brent and Luck**
  - Highlighted the column norm importance
  - Normalized the Threshold → Adapting to the columns’ norm → Actually calculating the cosine: \( \frac{A_i \cdot A_j^T}{\|A_i\| \|A_j\|} < \text{Threshold} \)

• **Hestenes**
  - Swap the columns → Active Sorting → \( f(\text{column norm}) \)
IMPROVING PREVIOUS WORK
ADDING OUR TOUCH

- Increased Adaptability
  - Realizing that the “Inverse Error” lies on small columns
  - Being Fussier with them → Harder Threshold
  - With Easier Threshold → Same Solution Accuracy
  - Not rotating in vain the big columns
  - AMN:
    \[
    \frac{A_i \cdot A_j^T}{\|A_i\| \|A_j\|} = \cos(A_i, A_j) < \text{Threshold} \cdot \min(\|A_i\|, \|A_j\|)
    \]
  - AAMN:
    \[
    \frac{A_i \cdot A_j^T}{\|A_i\| \|A_j\|} = \cos(A_i, A_j) < \text{Threshold} \cdot \|A_j\|
    \]
IMPROVING PREVIOUS WORK BEING HW FRIENDLY

• Initially Two Angles Calculation:
  • The Decision $\rightarrow f(\text{cosine})$
  • The Rotation $\rightarrow f(\text{Rutishhauser})$
  • Cos & RutisHauser $\rightarrow$ both $f(\text{columns and its norms})$
• Killing two bird with one stone
  Decision and rotation $\rightarrow f(\text{RutisHauser})$
• Readaptation $\rightarrow$ More sensitive
• AARH: $\theta_{ij} < Threshold \cdot \|A_j\|^2$
• Avoiding root squares
- Motivation
- Selecting the Algorithm
- Selecting SVD Method
- Speaking About the Accuracy
- Improving Previous Work

**Results**
- HW FPGA Implementation
- Why and How Scalable?
- Conclusion
- Future Work

### Initialization: Problem Size, Flag & Counters

- \( i = 0 \); \( CONV = 1 \)
- \( i++ \)
- \( j = i+1 \)
- \( \text{swap} \)
- \( \| A_i \| < \| A_j \| \)
- \( j++ \)
- \( j = ns \& \| A_j \| = \text{null} \)
- \( \| A_i \| = \text{null} \)
- \( T \)
- \( F \)
- \( ns = i-1 \)
- \( ns = j-1 \)

### Iteration : Swap & Null Columns Management

- \( \text{compute angle (6)} \)
- \( \text{threshold } 2^{-tk} \)
- \( \text{evaluate} \)
- \( \text{CONV} = 0 \text{ Givens' rotation} \)
- \( j < nl \)
- \( F \)
- \( T \)
- \( \text{CONV} = 1 \)
- \( F \)
- \( T \)
- \( \text{Factorization (4) & (5)} \)
- \( [U,S,V] = \text{svd}(A) \)
- \( \text{STOP} \)

### Iteration : Decision, Rotation & Actualization

### Finish: Matrix Factorization

\[
\sigma_i = \|W(:,i)\| \quad U(:,i) = \frac{W(:,i)}{\sigma_i}
\]
COMPARATION ANALYSIS TOOLBOX

Column Arrangement
Selection of the Comparation

Error Evolution
Results
RESULTS
COMPARING WITH THE REST

- Motivation
- Selecting the Algorithm
- Selecting SVD Method
- Speaking About the Accuracy
- Improving Previous Work
- Results
  - HW FPGA Implementation
  - Why and How Scalable?
- Conclusion
- Future Work

Obtaining Same Accuracy ➔ Easier Threshold
Obtaining Same Accuracy ➔ Less Rotations
Obtaining a Better Result ➔ The Higher the K(A)

Testing With Real Matrices & Obtaining Expected Results
RESULTS
COMPARING WITH THE REST

• Savings in Number of Rotations

<table>
<thead>
<tr>
<th>CN</th>
<th>Fixed to AARH</th>
<th>B&amp;L to AARH</th>
<th>ABL to AARH</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,00E+01</td>
<td>24,29%</td>
<td>21,02%</td>
<td>2,24%</td>
</tr>
<tr>
<td>1,00E+02</td>
<td>30,38%</td>
<td>24,48%</td>
<td>4,15%</td>
</tr>
<tr>
<td>1,00E+03</td>
<td>44,08%</td>
<td>36,50%</td>
<td>18,00%</td>
</tr>
<tr>
<td>1,00E+04</td>
<td>52,72%</td>
<td>39,29%</td>
<td>19,32%</td>
</tr>
</tbody>
</table>

-Motivation
-Selecting the Algorithm
-Selecting SVD Method
-Speaking About the Accuracy
-Improving Previous Work
-Results
- HW FPGA Implementation
-Why and How Scalable?
-Conclusion
-Future Work
Double Data-Flow:

- Primary: Linear array to manage Ai/Aj
- Secondary: Asynchronous full-duplex shared bus to manage Vi/Vj
- FIFO between PUs
HW FPGA IMPLEMENTATION
PROCESSING UNIT

PU Design:
• Evaluation: Computing square Euclidean norms and vector multiplication, swapping and deciding
• Cordic: Theta calculation and rotations’ performing
• Cache: L(m+n) and R(max(m,2n))

-Motivation
-Selecting the Algorithm
-Selecting SVD Method
-Speaking About the Accuracy
-Improving Previous Work
-Results
-HW FPGA Implementation
-Why and How Scalable?
-Conclusion
-Future Work
WHY AND HOW SCALABLE?

- Motivation
- Selecting the Algorithm
- Selecting SVD Method
- Speaking About the Accuracy
- Improving Previous Work
- Results
- HW FPGA Implementation
- Why and How Scalable?
- Conclusion
- Future Work

- No limited to specific matrices and HW: Generic Solution
  - Different sizes
  - Different Shapes
  - Different budgets

- Architecture
  - Based on basics processing units PUs
  - PUs variable on quantity
  - From 2 to n/2
<table>
<thead>
<tr>
<th></th>
<th>xc6slx45-3fgg484</th>
<th>xc7k160t-3fbg484</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area</td>
<td>56 %</td>
<td>86 %</td>
</tr>
<tr>
<td>DSPs</td>
<td>93 %</td>
<td>60 %</td>
</tr>
<tr>
<td>RAM</td>
<td>62 %</td>
<td>44 %</td>
</tr>
<tr>
<td>Matrix Size : K(A)</td>
<td>300x100 ; $10^2$</td>
<td>750 x 250</td>
</tr>
<tr>
<td>PUs</td>
<td>9</td>
<td>60</td>
</tr>
<tr>
<td>Frequency</td>
<td>55 MHz</td>
<td>90 MHz</td>
</tr>
<tr>
<td>Processing Time</td>
<td>60 ms</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(5-6 sweeps, 8-16 ms/sweep)</td>
<td></td>
</tr>
</tbody>
</table>

Word-Length : 18 bits
CONCLUSION

• AAMN and AARH proposed outperforming previous proposals.
  • Small Columns Important Columns
  • Same Accuracy Less Rotations
  • User-defined Accuracy -> Threshold
  • HW Friendly
• An implemented parallel processing scheme proposed:
  • Linear Array of PUs
  • Scalable
  • Double Data-Flow
FUTURE WORK

• Online reduction of problem size
• Improve sorting
• Optimize PU design
  • Improved CORDIC realization (Redundant arithmetic (Ercegovac) or Square root and division free (Gotze))
  • Ad-hoc online estimators
    • Atan
    • Square norm
THANK YOU VERY MUCH

• Questions or Details ?