Intra- and Inter-FPGA Programmable Multiprocessor Designs with Emphasis on Large-Scale Matrix Operations*

Sotirios G. Ziavras, Xiaofang Wang & Muhammad Z. Hasan

ECE Dept.

New Jersey Institute of Technology

Newark, NJ 07102

ziavras@njit.edu

FPGA-Based Reconfigurable Computing

- The advent of multi-million-gate FPGAs introduced a new era in reconfigurable computing
  - Complex and general-purpose (instruction-programmable) parallel computing platforms supporting floating-point arithmetic

- The peak floating-point performance of FPGAs has outnumbered (last 1-2 years) that of modern microprocessors and is growing much faster than the latter

- FPGA advantages (compared to ASIC designs)
  - Flexibility
  - Low cost
  - Ease of upgrading
  - High availability
    - Lower speed

HERA (HEterogeneous Reconfigurable Architecture) Motivation

- The performance and efficiency of an algorithm highly depend on its good match with the target hardware
- A real-world application has various subtasks with different architectural requirements
- SIMD: good for data-parallel operations
- MIMD: deals better with dynamic uncertainties

- Our target (matrix-based) applications mostly involve data-parallel operations
  - However, they can benefit in certain instances from MIMD (e.g., LU factorization & multiplication of large matrices)

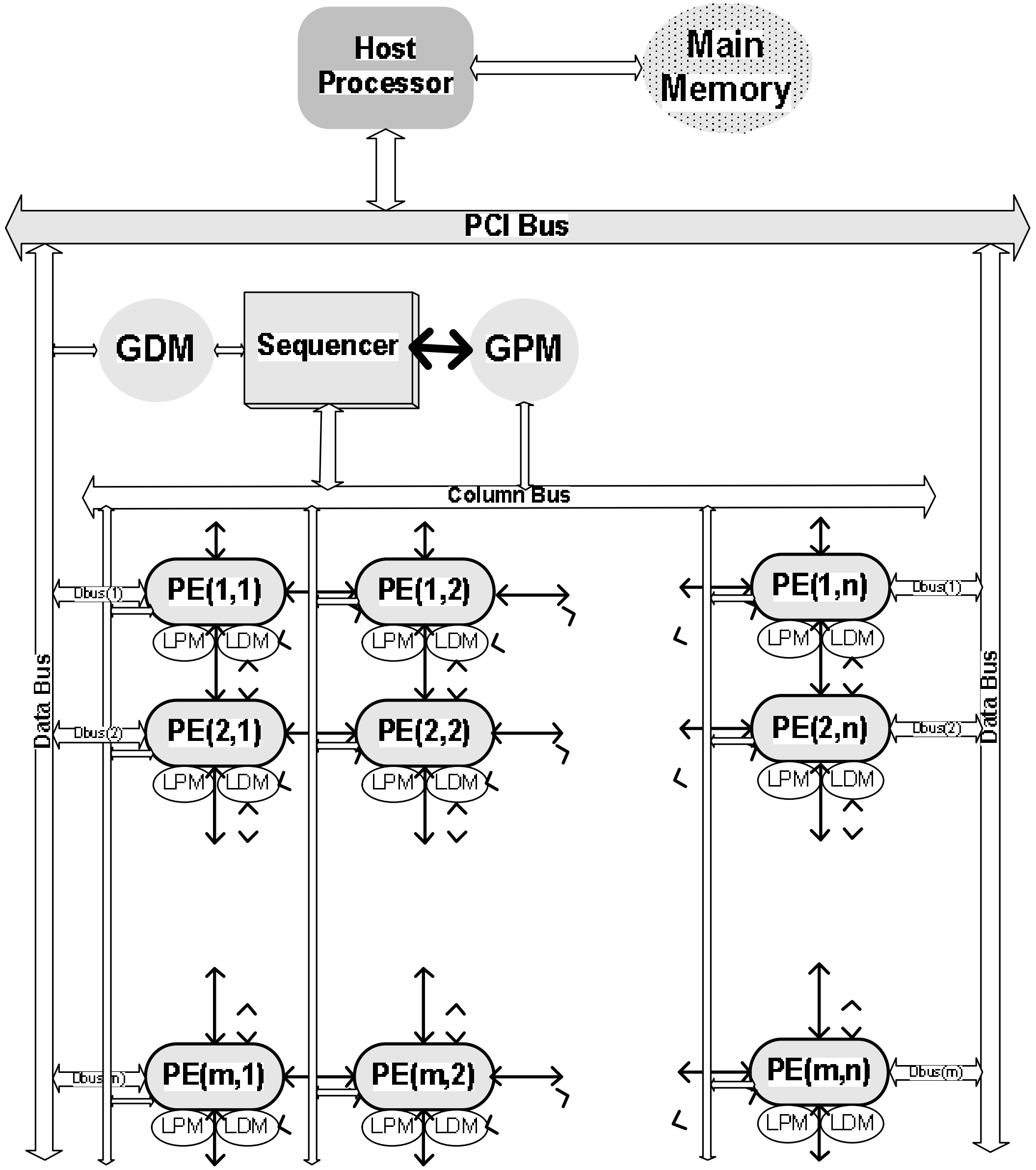
➔

- HERA: A *mixed-mode machine* reconfigured at runtime to support a variety of independent or cooperating computing modes (SIMD, MIMD, Multiple-SIMD) to better match the subtask characteristics of a single application

HERA System Architecture

- NEWS connections
- Hierarchical bus system
- PCI communication with host machine
- Shared local data memory port for local groups of 3 PEs
- Customizable in terms of PE function and total number of PEs
- Support global (PE ID) and local masking (mask register)
- PE mode switch by one instruction (*Configure/JumpI/OMR*)

*PE: Processing Element
*LDM: Local data memory
*LPM: Local program memory
*GDM: Global data memory
*GPM: Global program memory

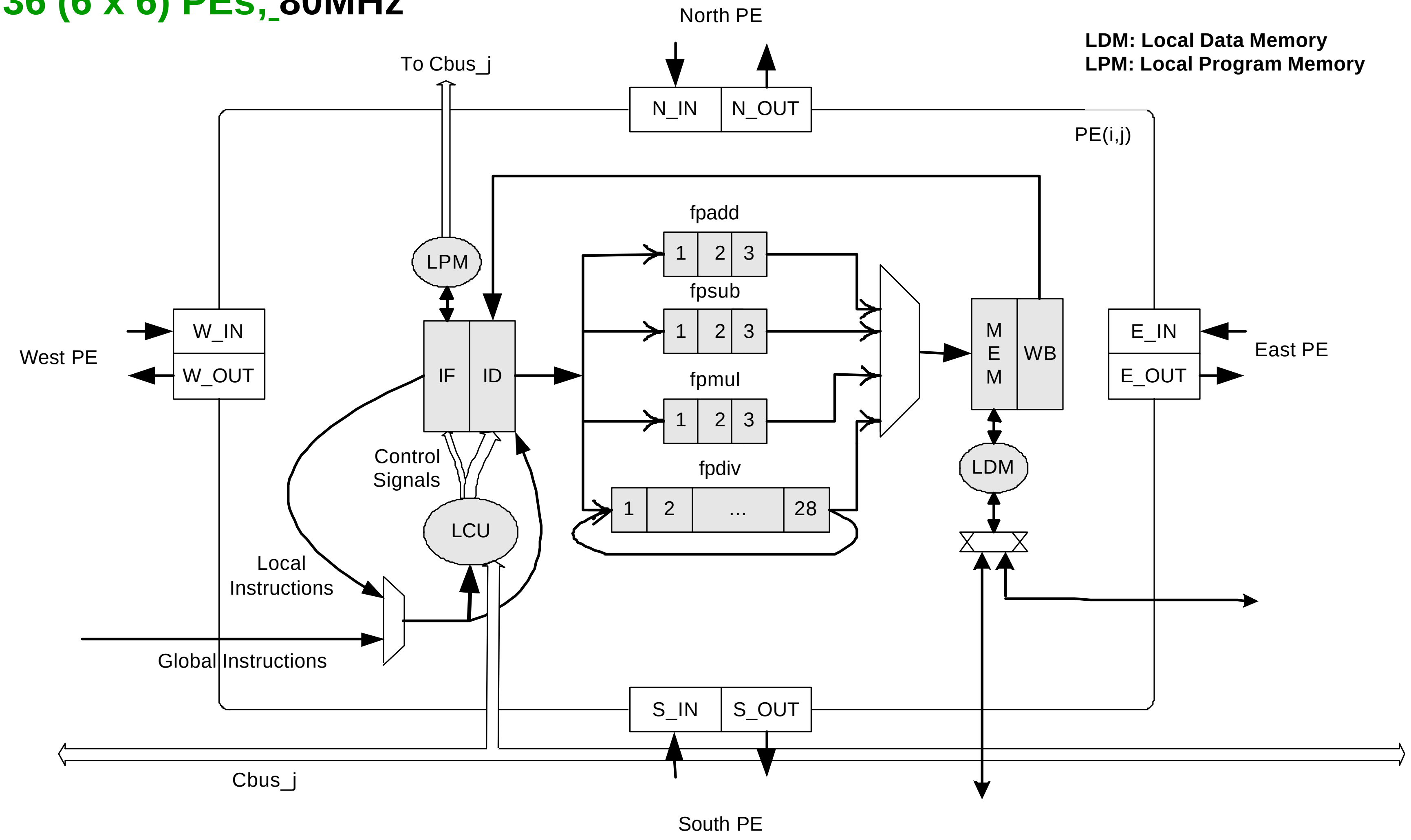


PE Microarchitecture

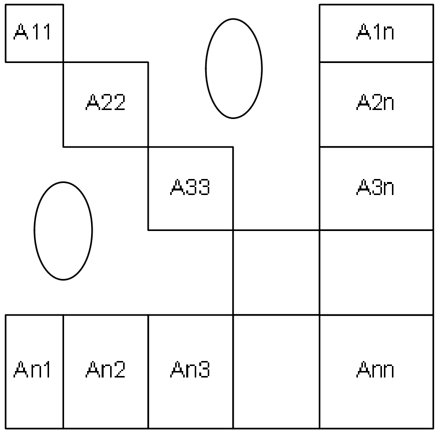
Annapolis WildStar PCI-II board (2 Xilinx Virtex II XC2V6000-5)

36 (6 x 6) PEs; 80MHz

LDM: Local Data Memory
LPM: Local Program Memory



Parallel BDB LU Factorization (sparse matrices)



?

$$\begin{Bmatrix} L_{11} & 0 & 0 & 0 & 0 \\ 0 & L_{22} & 0 & 0 & 0 \\ 0 & 0 & L_{33} & 0 & 0 \\ 0 & 0 & 0 & \ldots & \ldots \\ L_{n1} & L_{n2} & L_{n3} & \ldots & L_{nn} \end{Bmatrix} \begin{Bmatrix} U_{11} & 0 & 0 & 0 & U_{1n} \\ 0 & U_{22} & 0 & 0 & U_{2n} \\ 0 & 0 & U_{33} & 0 & U_{3n} \\ 0 & 0 & 0 & \ldots & \ldots \\ 0 & 0 & 0 & \ldots & U_{nn} \end{Bmatrix}$$

where

$$A_{kk} = L_{kk}U_{kk}$$

$$U_{kn} = L_{kk}^{-1}A_{kn}$$

$$L_{nk} = A_{nk}U_{kk}^{-1}, \; for \; k \in [1, n-1]$$

$$L_{nn}U_{nn} = A_{nn} - \sum_{k=1}^{n-1} L_{nk}U_{kn}$$

Tasks in Parallel BDB LU Factorization

- **FAC: Independent factorization of all the 3-block groups. _No data communication between PEs_**

- **MAC: Independent multiplication of the factored border block pairs and (local) accumulation of the partial products inside each PE to later produce the inner product. Every resulting product has the same size as *Ann*. This work can be overlapped with the FAC work as long as the corresponding *L* and *U* blocks are already factored**

- **PAC: Parallel (global) accumulation of the partial results in all PEs in parallel using the results of MAC tasks. This work also can be overlapped with FAC and MAC work**

- **LAST: Parallel LU factorization of the last block upon finishing all the factorization and multiplication work. The beginning of this work starts with the synchronization of the involved PEs. The last block is normally dense**

Mixed-Mode Scheduling (1)

Step 1:

- **Identify 3-block groups of comparable size and put them into the same task queue**
- **Configure HERA into *M-SIMD* based on the task information**
- **Assign 3-block groups from a queue to PEs in the same *SIMD* group, and perform the FAC and MAC work on these groups until the number of remaining 3-block groups is less than the number of PEs (i.e., 36)**

Step 2: Assign the remaining 3-block groups in such a way that groups of comparable size go to the same column of PEs and every PE has the largest possible number of idle nearest neighbors. This is an effort to facilitate the following *PAC* work. If necessary, reconfigure the system into a different *M-SIMD* layout

Step 3: A PE is reconfigured into *MIMD* as soon as it finishes its work and no more 3-block group is in the task queue

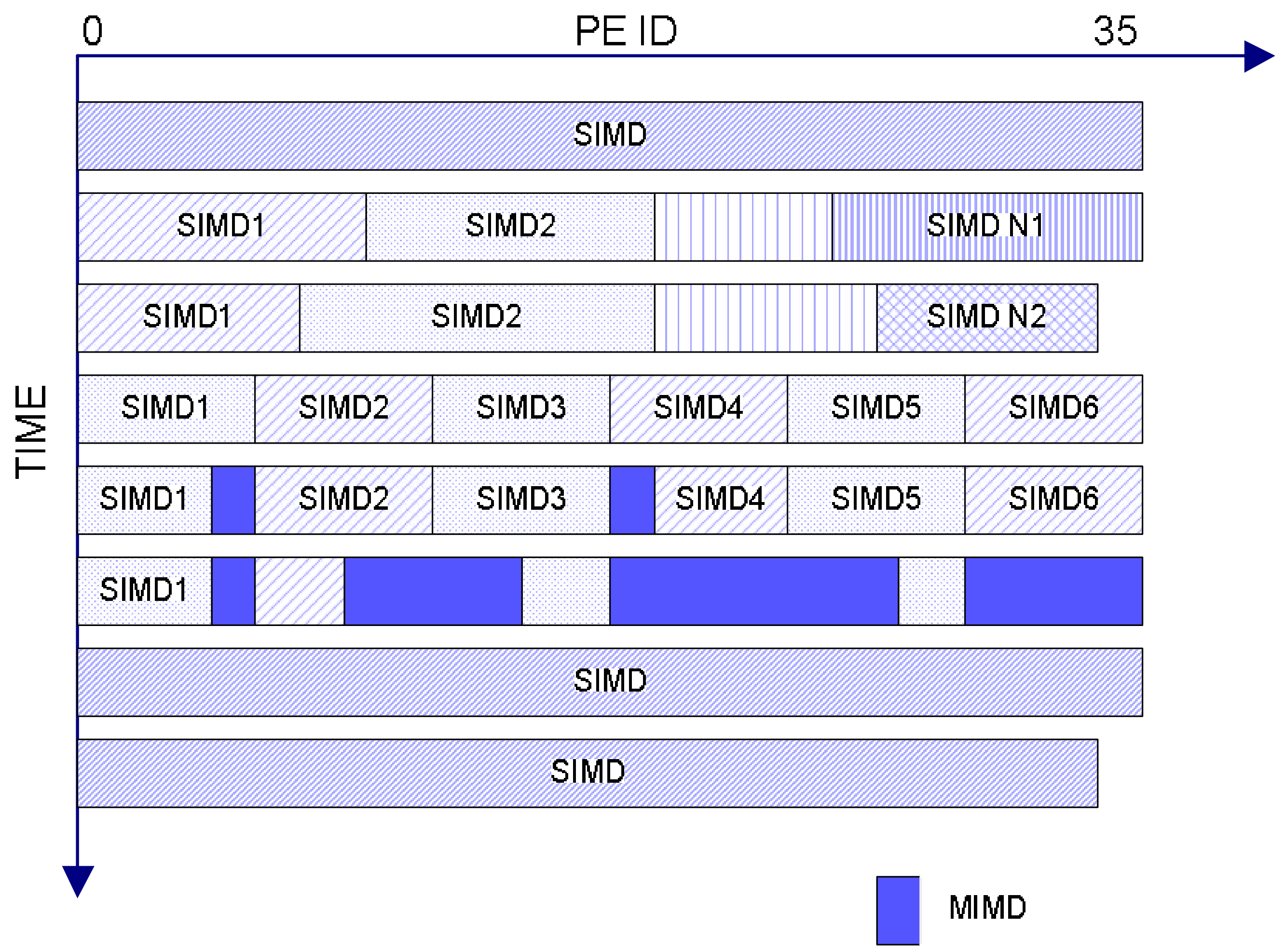
Mixed-Mode Scheduling (2)

Step 4: Assign each PE in *MIMD* to the multiplication of a pair of (row and column) factored border blocks. Since the LDM has a shared port with its east and south neighbors, every idle PE will help its neighbors after it finishes its own work; no data transfer incurs in this process

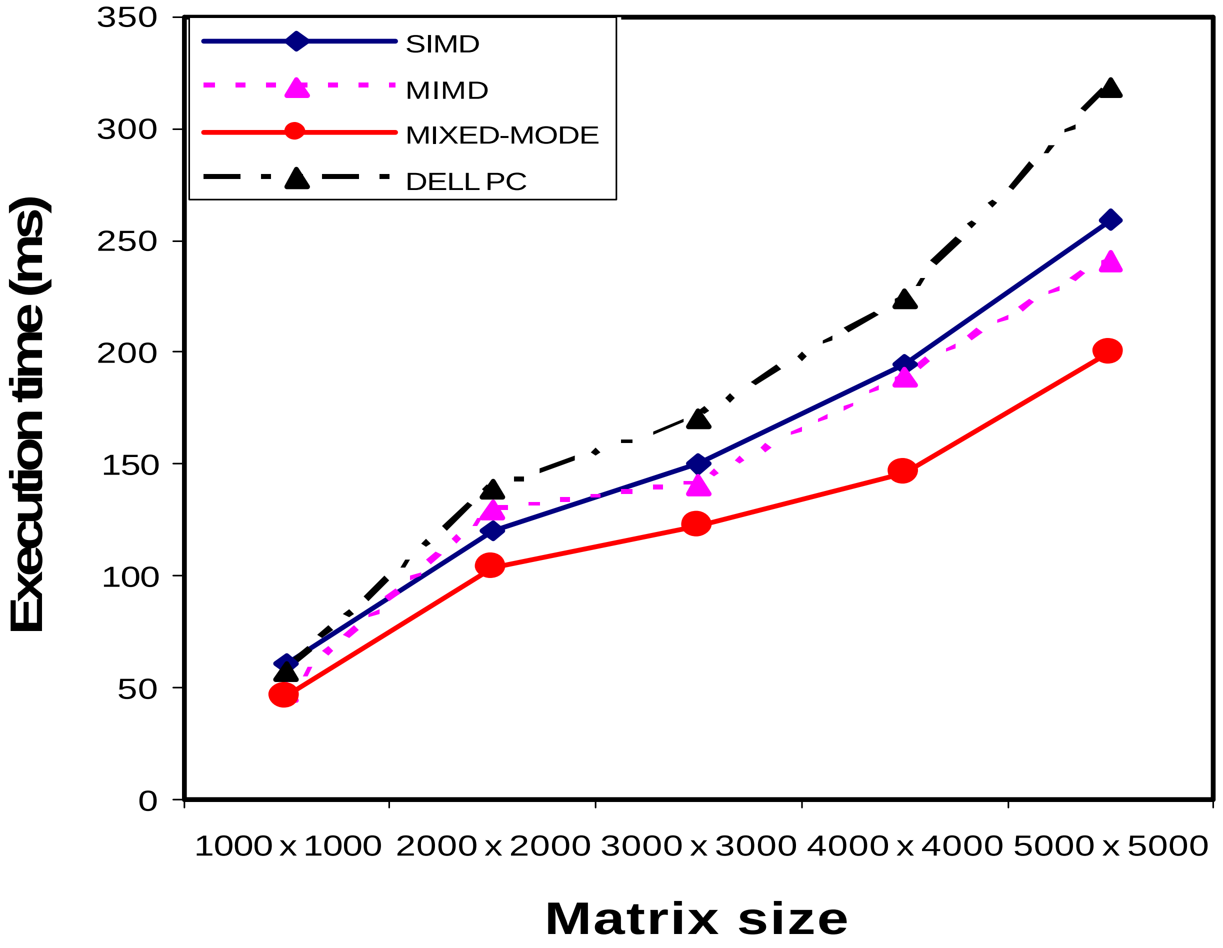
Step 5: After the factorization of all the 3-block groups and the multiplication of factored border blocks, reconfigure all the PEs again into the *SIMD* mode to carry out the *PAC* work

Step 6: Factor the last block in *SIMD*

Typical BDB Execution Mode for Large BDB Matrices



Performance



Conclusions

- New generation FPGAs brought about a new era in reconfigurable computing & multiprocessor-on-chip designs
  - Viable and cost-effective approaches in building high-performance parallel computing platforms for scientific computing
- Mixed-mode parallelism can match better the requirements of applications throughout execution
  - Better performance than SIMD or MIMD
  - Appropriate for applications with irregular computation and communication patterns, and/or frequent conditional executions
- HERA yields high performance
  - Dynamic reconfiguration to match the application
  - User-programmable (software)
  - Dynamic mixed-mode execution (SIMD, MIMD, combination M-SIMD)
  - Low-cost

Vector Processor for the W-Matrix Method

(Solution of Linear Equations: Ax=b)

- Let $A = L\,D\,U$
- Let $L^{-1} = W^L$ & $U^{-1} = W^U \Rightarrow x = W^U D^{-1} W^L b$
- Solution in 3 steps: $W^L b = z$, $D^{-1} z = y$ & $W^U y = x$.
- Let $L = L_1 L_2 \ldots L_n$, where L_i is an identity matrix having the i-th column the same as that of L $\Rightarrow$

  $W^L = L^{-1} = (L_1 L_2 \ldots L_n)^{-1} = L_n^{-1} \ldots L_2^{-1} L_1^{-1} = W_n \ldots W_2 W_1$, where W_i is L_i with the signs of the off-diagonal elements reversed.
- $x = (L_1^t)^{-1}(L_2^t)^{-1} \ldots (L_n t)^{-1} D^{-1} L_n^{-1} \ldots L_2^{-1} L_1^{-1} b = (W_1^t)(W_2^t) \ldots (W_n^t) D^{-1} W_n \ldots W_2 W_1 b$ or $x = (W_a^t)(W_b^t) D^{-1} W_b W_a b$. The number of non-zero elements increases as we combine the W factors, reducing the number of serial steps.

Vector Operations for the W-matrix Method

Operations	Effect
Counting the number of non-zero elements	Fast ordering of rows for more parallelism
Sorting the rows based on the above counts	
Selecting a row based on the minimum count	
Creating pseudo-columns from many partially filled columns	Fast access of the elements
Accessing only the diagonal elements	Fast access of the elements
Creating a vector from all non-zero elements	Fast access of the elements
Creating a vector from the column indices of all non-zero elements of a matrix	
Creating a vector from the last row numbers of partitions of a matrix *	
Adding only certain elements of a vector with certain elements of ***another vector ****	Fast multiplication of a matrix with a vector
Adding only certain elements of a vector with certain other elements of the ***same vector***	
Multiplication followed by addition*	Reduce the number of memory accesses
Loading-Adding-Storing with the same indices *	Reduce the number of indirect memory accesses

IEEE Power Benchmarks

Test System	Cycles Needed		Cycle Savings (%)
	With standard instructions	With sparse handling instructions	
14-bus (27.55% NZ)	2110	1563	25.924
30-bus (12% NZ)	9834	7550	23.225
57-bus (6% NZ)	36872	29218	20.758

Two- & Three-Stage Vector-Chaining

Opera-tions	Cycles Needed		Savings (%)
	W/O Chain-ng	With Chaining	
Load-Mult	87	63	27.5
Add-Store	50	25	50

	Cycles Needed		Savi-ngs (%)
	W/O Chaining	With Chaini-ng (Load-Mult-Prefetch)	
14 bus	1644	1474	10.34
30 bus	7514	6734	10.38