Hardware Acceleration for Load Flow Computation

Jeremy Johnson, Chika Nwankpa and Prawat Nagvajara
Kevin Cunningham, Tim Chagnon, Petya Vachranukunkiet

Computer Science and Electrical and Computer Engineering
Drexel University

Eigth Annual CMU Conference on the Electricity Industry
March 13, 2012

Overview

- Problem and Goals
- Background
- LU Hardware
- Merge Accelerator Design
- Accelerator Architectures
- Performance Results
- Conclusions

Problem

- **Problem:** Sparse Lower-Upper (LU) Triangular Decomposition performs inefficiently on general-purpose processors (irregular data access, indexing overhead, less than 10% FP efficiency on power systems) [3, 12]

[3] T. Chagnon. Architectural Support for Direct Sparse LU Algorithms.

[12] P. Vachranukunkiet. Power Flow Computation using Field Programmable Gate Arrays.

Problem

- **Problem:** Sparse Lower-Upper (LU) Triangular Decomposition performs inefficiently on general-purpose processors (irregular data access, indexing overhead, less than 10% FP efficiency on power systems) [3, 12]
- **Solution:** Custom designed hardware can provide better performance (3X speedup over Pentium 4 3.2GHz) [3, 12]

---

[3] T. Chagnon. Architectural Support for Direct Sparse LU Algorithms.

[12] P. Vachranukunkiet. Power Flow Computation using Field Programmable Gate Arrays.

Problem

- **Problem:** Sparse Lower-Upper (LU) Triangular Decomposition performs inefficiently on general-purpose processors (irregular data access, indexing overhead, less than 10% FP efficiency on power systems) [3, 12]
- **Solution:** Custom designed hardware can provide better performance (3X speedup over Pentium 4 3.2GHz) [3, 12]
- **Problem:** Complex custom hardware is difficult, expensive, and time-consuming to design

---

[3] T. Chagnon. Architectural Support for Direct Sparse LU Algorithms.

[12] P. Vachranukunkiet. Power Flow Computation using Field Programmable Gate Arrays.

Problem

- **Problem:** Sparse Lower-Upper (LU) Triangular Decomposition performs inefficiently on general-purpose processors (irregular data access, indexing overhead, less than 10% FP efficiency on power systems) [3, 12]
- **Solution:** Custom designed hardware can provide better performance (3X speedup over Pentium 4 3.2GHz) [3, 12]
- **Problem:** Complex custom hardware is difficult, expensive, and time-consuming to design
- **Solution:** Use a balance of software and hardware accelerators (ex: FPU, GPU, Encryption, Video/Image Encoding, etc.)

---

[3] T. Chagnon. Architectural Support for Direct Sparse LU Algorithms.

[12] P. Vachranukunkiet. Power Flow Computation using Field Programmable Gate Arrays.

Problem

- **Problem:** Sparse Lower-Upper (LU) Triangular Decomposition performs inefficiently on general-purpose processors (irregular data access, indexing overhead, less than 10% FP efficiency on power systems) [3, 12]
- **Solution:** Custom designed hardware can provide better performance (3X speedup over Pentium 4 3.2GHz) [3, 12]
- **Problem:** Complex custom hardware is difficult, expensive, and time-consuming to design
- **Solution:** Use a balance of software and hardware accelerators (ex: FPU, GPU, Encryption, Video/Image Encoding, etc.)
- **Problem:** Need an architecture that efficiently combines general-purpose processors and accelerators

---

[3] T. Chagnon. Architectural Support for Direct Sparse LU Algorithms.

[12] P. Vachranukunkiet. Power Flow Computation using Field Programmable Gate Arrays.

Goals

- Increase performance of sparse LU over existing software

Goals

- Increase performance of sparse LU over existing software
- Design a flexible, easy to use hardware accelerator that can integrate with multiple platforms

Goals

- Increase performance of sparse LU over existing software
- Design a flexible, easy to use hardware accelerator that can integrate with multiple platforms
- Find an architecture that efficiently uses the accelerator to improve sparse LU

Summary of Results and Contributions

- Designed and implemented a merge hardware accelerator [5]
  - Data rate approaches one element per cycle

[5] Cunningham, Nagvajara. Reconfigurable Stream-Processing Architecture for Sparse Linear Solvers.

[6] Cunningham, Nagvajara, Johnson. Reconfigurable Multicore Architecture for Power Flow Calculation.

Summary of Results and Contributions

- Designed and implemented a merge hardware accelerator [5]
  - Data rate approaches one element per cycle
- Designed and implemented supporting hardware [6]

[5] Cunningham, Nagvajara. Reconfigurable Stream-Processing Architecture for Sparse Linear Solvers.

[6] Cunningham, Nagvajara, Johnson. Reconfigurable Multicore Architecture for Power Flow Calculation.

Summary of Results and Contributions

- Designed and implemented a merge hardware accelerator [5]
  - Data rate approaches one element per cycle
- Designed and implemented supporting hardware [6]
- Implemented a prototype Triple Buffer Architecture [5]
  - Efficiently utilize external transfer bus
  - Not suitable for sparse LU application
  - Advantageous for other applications

---

[5] Cunningham, Nagvajara. Reconfigurable Stream-Processing Architecture for Sparse Linear Solvers.

[6] Cunningham, Nagvajara, Johnson. Reconfigurable Multicore Architecture for Power Flow Calculation.

Summary of Results and Contributions

- Designed and implemented a merge hardware accelerator [5]
  - Data rate approaches one element per cycle
- Designed and implemented supporting hardware [6]
- Implemented a prototype Triple Buffer Architecture [5]
  - Efficiently utilize external transfer bus
  - Not suitable for sparse LU application
  - Advantageous for other applications
- Implemented a prototype heterogeneous multicore architecture [6]
  - Combines general-purpose processor and reconfigurable hardware cores
  - Speedup of 1.3X over sparse LU software with merge accelerator

---

[5] Cunningham, Nagvajara. Reconfigurable Stream-Processing Architecture for Sparse Linear Solvers.

[6] Cunningham, Nagvajara, Johnson. Reconfigurable Multicore Architecture for Power Flow Calculation.

Summary of Results and Contributions

- Designed and implemented a merge hardware accelerator [5]
  - Data rate approaches one element per cycle
- Designed and implemented supporting hardware [6]
- Implemented a prototype Triple Buffer Architecture [5]
  - Efficiently utilize external transfer bus
  - Not suitable for sparse LU application
  - Advantageous for other applications
- Implemented a prototype heterogeneous multicore architecture [6]
  - Combines general-purpose processor and reconfigurable hardware cores
  - Speedup of 1.3X over sparse LU software with merge accelerator
- Modified Data Pump Architecture (DPA) Simulator
  - Added merge instruction and support
  - Implemented sparse LU on the DPA
  - Speedup of 2.3X over sparse LU software with merge accelerator

---

[5] Cunningham, Nagvajara. Reconfigurable Stream-Processing Architecture for Sparse Linear Solvers.

[6] Cunningham, Nagvajara, Johnson. Reconfigurable Multicore Architecture for Power Flow Calculation.

Overview

- Problem and Goals
- Background
- LU Hardware
- Merge Accelerator Design
- Accelerator Architectures
- Performance Results
- Conclusions

Power Systems

- The power grid delivers electrical power from generation stations to distribution stations
- Power grid stability is analyzed with Power Flow (aka Load Flow) calculation [1]
- System nodes and voltages create a system of equations, represented by a sparse matrix [1]

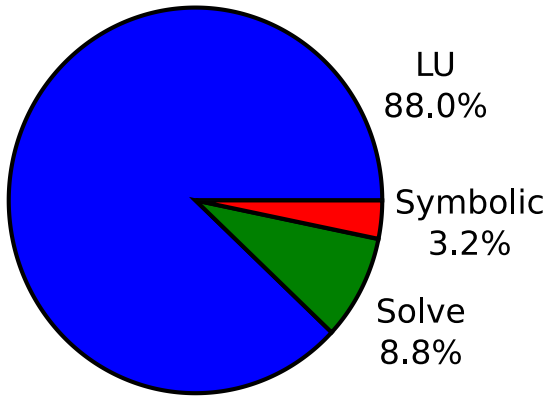
---

[1] A. Albur. Power System State Estimation: Theory and Implementation.

[12] P. Vachranukunkiet. Power Flow Computation using Field Programmable Gate Arrays.

Power Systems

- The power grid delivers electrical power from generation stations to distribution stations
- Power grid stability is analyzed with Power Flow (aka Load Flow) calculation [1]
- System nodes and voltages create a system of equations, represented by a sparse matrix [1]
- Solving the system allows grid stability to be analyzed
- LU decomposition accounts for a large part of the Power Flow calculation [12]
- Power Flow calculation is repeated thousands of times to perform a full grid analysis [1]

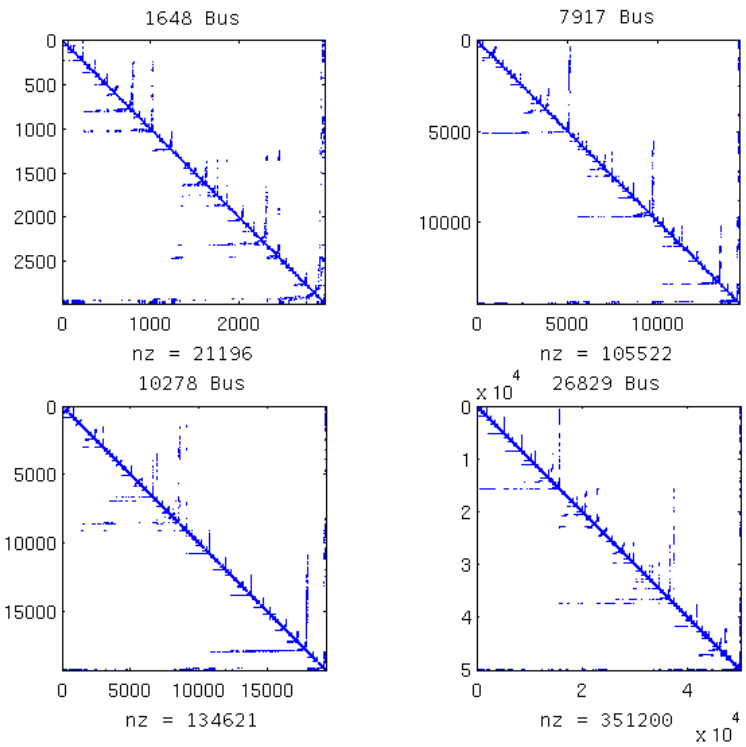


Power Flow Execution Profile [12]

[1] A. Albur. Power System State Estimation: Theory and Implementation.

[12] P. Vachranukunkiet. Power Flow Computation using Field Programmable Gate Arrays.

Power System Matrices



Power System Matrix Properties

System	# Rows/Cols	NNZ	Sparsity
1648 Bus	2,982	21,196	0.238%
7917 Bus	14,508	105,522	0.050%
10278 Bus	19,285	134,621	0.036%
26829 Bus	50,092	351,200	0.014%

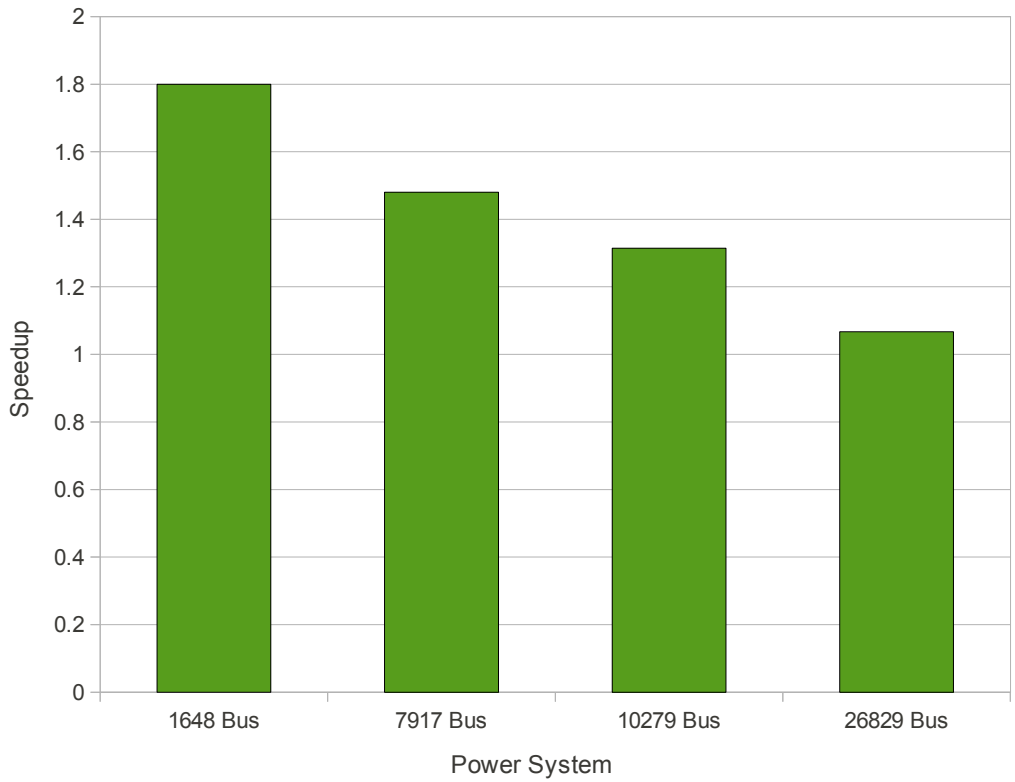
Power System LU Statistics

System	Avg. NNZ per Row	Avg. Num Submatrix Rows	% Merge
1648 Bus	7.1	8.0	65.6%
7917 Bus	7.3	8.5	54.3%
10279 Bus	7.0	8.3	55.8%
26829 Bus	7.0	8.7	62.7%

Power System LU

- Gaussian LU outperforms UMFPACK for power matrices [3]
- As matrix size grows, multi-frontal becomes more effective
- The merge is a bottleneck in Gaussian LU performance [3]
- **Goal:** Design a hardware accelerator for the Merge in Gaussian LU

Gaussian LU vs UMFPACK on Intel Core i7 at 3.2GHz [3]



[3] T. Chagnon. Architectural Support for Direct Sparse LU Algorithms.

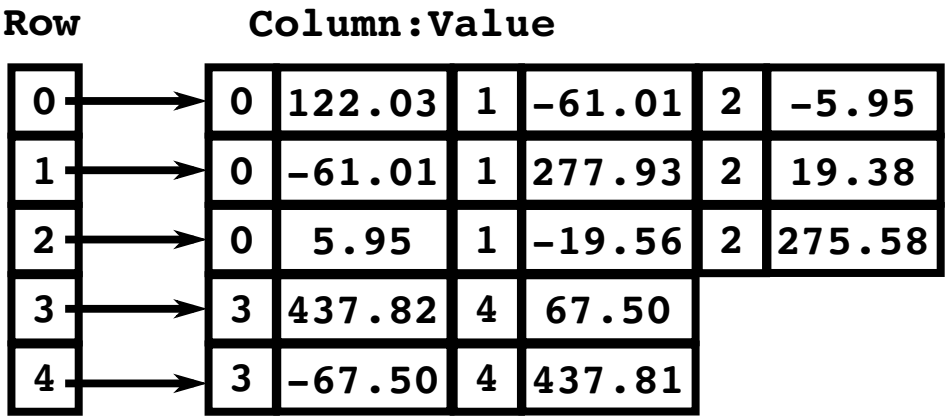
Sparse Matrices

- A matrix is sparse when it has a large number of zero elements
- Below is a small section of the 26K-Bus power system matrix

$$A = \begin{bmatrix} 122.03 & -61.01 & -5.95 & 0 & 0 \\ -61.01 & 277.93 & 19.38 & 0 & 0 \\ 5.95 & -19.56 & 275.58 & 0 & 0 \\ 0 & 0 & 0 & 437.82 & 67.50 \\ 0 & 0 & 0 & -67.50 & 437.81 \end{bmatrix}$$

Sparse Compressed Formats

- Sparse matrices use a compressed format to ignore zero elements
- Saves storage space
- Decreases amount of computation



Sparse LU Decomposition

$$A = \begin{bmatrix} 122.03 & -61.01 & -5.95 & 0 & 0 \\ -61.01 & 277.93 & 19.38 & 0 & 0 \\ 5.95 & -19.56 & 275.58 & 0 & 0 \\ 0 & 0 & 0 & 437.82 & 67.50 \\ 0 & 0 & 0 & -67.50 & 437.81 \end{bmatrix}$$

$$L = \begin{bmatrix} 1.00 & 0 & 0 & 0 & 0 \\ -0.50 & 1.00 & 0 & 0 & 0 \\ 0.05 & -0.07 & 1.00 & 0 & 0 \\ 0 & 0 & 0 & 1.00 & 0 \\ 0 & 0 & 0 & -0.15 & 1.00 \end{bmatrix} \quad U = \begin{bmatrix} 122.03 & -61.01 & -5.95 & 0 & 0 \\ 0 & 247.42 & 16.41 & 0 & 0 \\ 0 & 0 & 276.97 & 0 & 0 \\ 0 & 0 & 0 & 437.82 & 67.50 \\ 0 & 0 & 0 & 0 & 448.22 \end{bmatrix}$$

Sparse LU Methods

- Approximate Minimum Degree (AMD) algorithm pre-orders matrices to reduce fill-in [2]
- Multiple methods for performing sparse LU:
- Multi-frontal
  - Used by UMFPACK [7]
  - Divides matrix into multiple, independent, dense blocks
- Gaussian Elimination
  - Eliminates elements below the diagonal by scaling and adding rows together

---

[2] Amestoy et al. Algorithm 837: AMD, an approximate minimum degree ordering algorithm.

[7] T. Davis. Algorithm 832: UMFPACK V4.3 - An Unsymmetric-Pattern Multi-Frontal Method.

Sparse LU Decomposition

Algorithm 1 Gaussian LU

for $i = 1 \rightarrow N$ **do**
  $pivot\_search()$
  $update\_U()$
  for $j = 1 \rightarrow NUM\_SUBMATRIX\_ROWS$ **do**
    $merge(pivot\_row, j)$
    $update\_L()$
    $update\_colmap()$
  end for
end for

Column Map

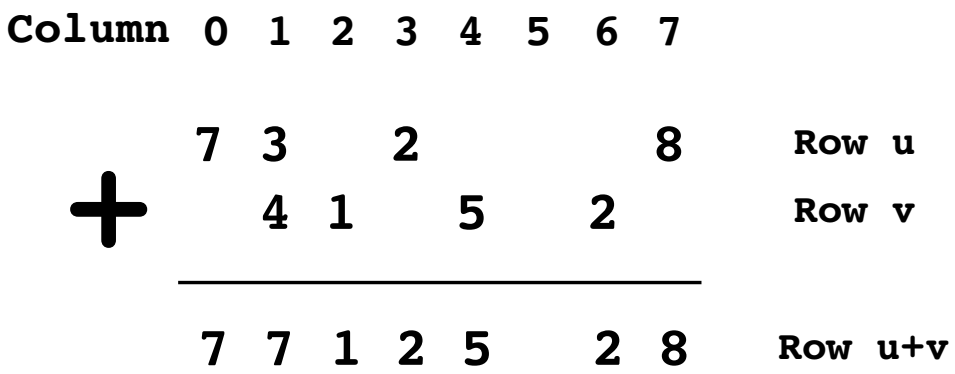
- Column map keeps track of the rows with non-zero elements in each column
- Do not have to search matrix on each iteration
- Fast access to all rows in a column
- Select pivot row from the set of rows
- Column map is updated after each merge
- Fill-in elements add new elements to a row during the merge

Sparse LU Decomposition

Column	0	1	2	3	4	5	6	7	
	7	3		2				8	Row u
+		4	1		5		2		Row v
	7	7	1	2	5		2	8	Row u+v

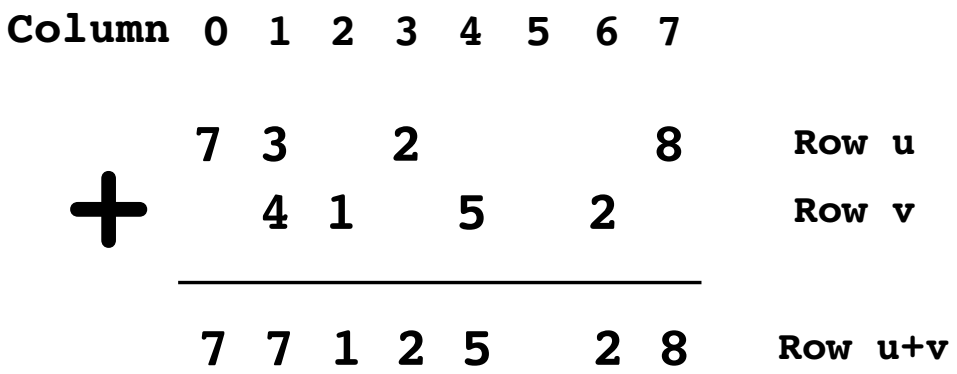
- Challenges with software merging:

Sparse LU Decomposition



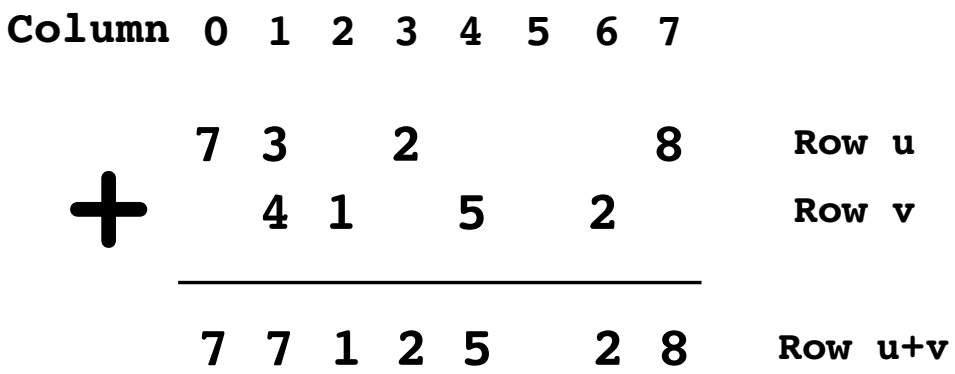
- Challenges with software merging:
  - Indexing overhead

Sparse LU Decomposition



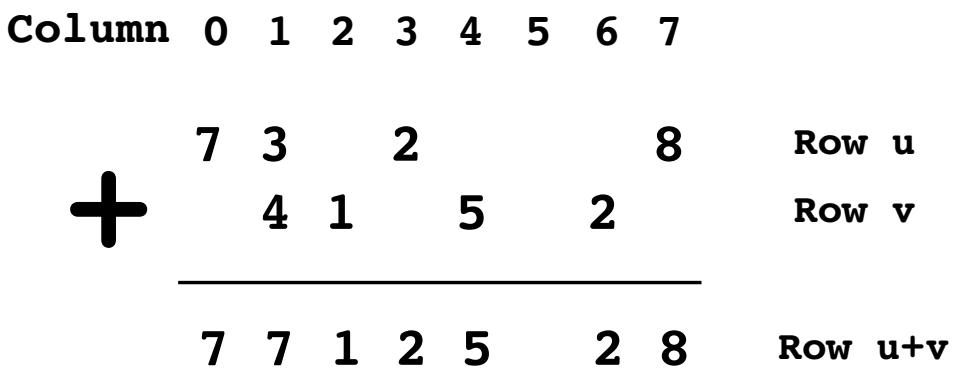
- Challenges with software merging:
  - Indexing overhead
  - Column numbers are different from row array indices

Sparse LU Decomposition



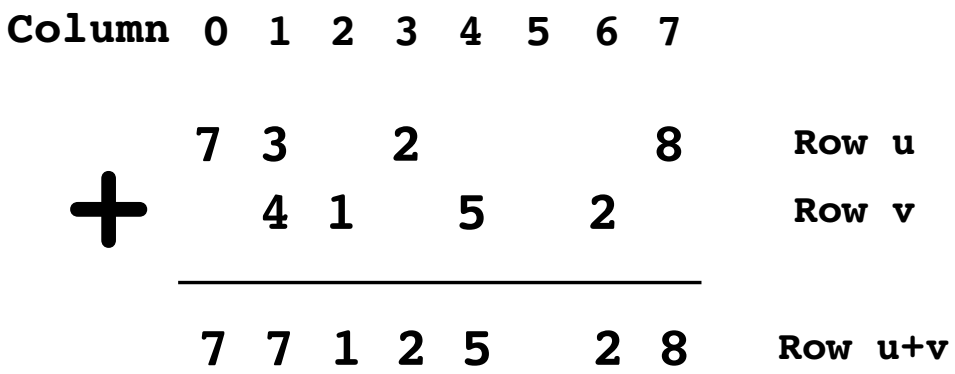
- Challenges with software merging:
  - Indexing overhead
  - Column numbers are different from row array indices
  - Must fetch column numbers from memory to operate on elements

Sparse LU Decomposition



- Challenges with software merging:
  - Indexing overhead
  - Column numbers are different from row array indices
  - Must fetch column numbers from memory to operate on elements
  - Cache misses

Sparse LU Decomposition



- Challenges with software merging:
  - Indexing overhead
  - Column numbers are different from row array indices
  - Must fetch column numbers from memory to operate on elements
  - Cache misses
  - Data-dependent branching

Sparse LU Decomposition

Merge Algorithm

```
int p,s;
int rnz=0, fnz=0;
for(p=1, s=1; p < pivotLen && s < nonpivotLen; ) {
 if(nonpivot[s].col == pivot[p].col) {
  merged[rnz].col = pivot[p].col;
  merged[rnz].val = (nonpivot[s].val - lx*pivot[p].val);
  rnz++; p++; s++;
 } else if(nonpivot[s].col < pivot[p].col) {
  merged[rnz].col = nonpivot[s].col;
  merged[rnz].val = nonpivot[s].val;
  rnz++; s++;
 } else {
  merged[rnz].col = pivot[p].col;
  merged[rnz].val = (-lx * pivot[p].val);
  fillin[fnz] = pivot[p].col;
  rnz++; fnz++; p++;
 }
}
```

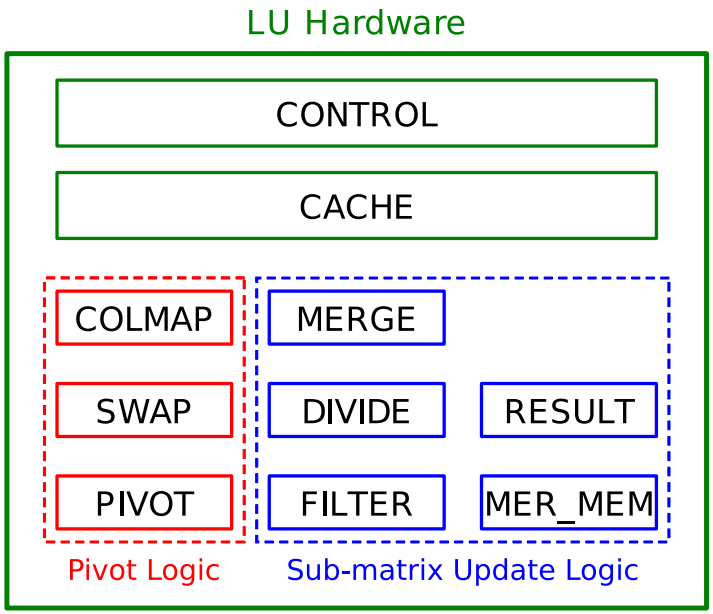
Overview

- Problem and Goals
- Background
- LU Hardware
- Merge Accelerator Design
- Accelerator Architectures
- Performance Results
- Conclusions

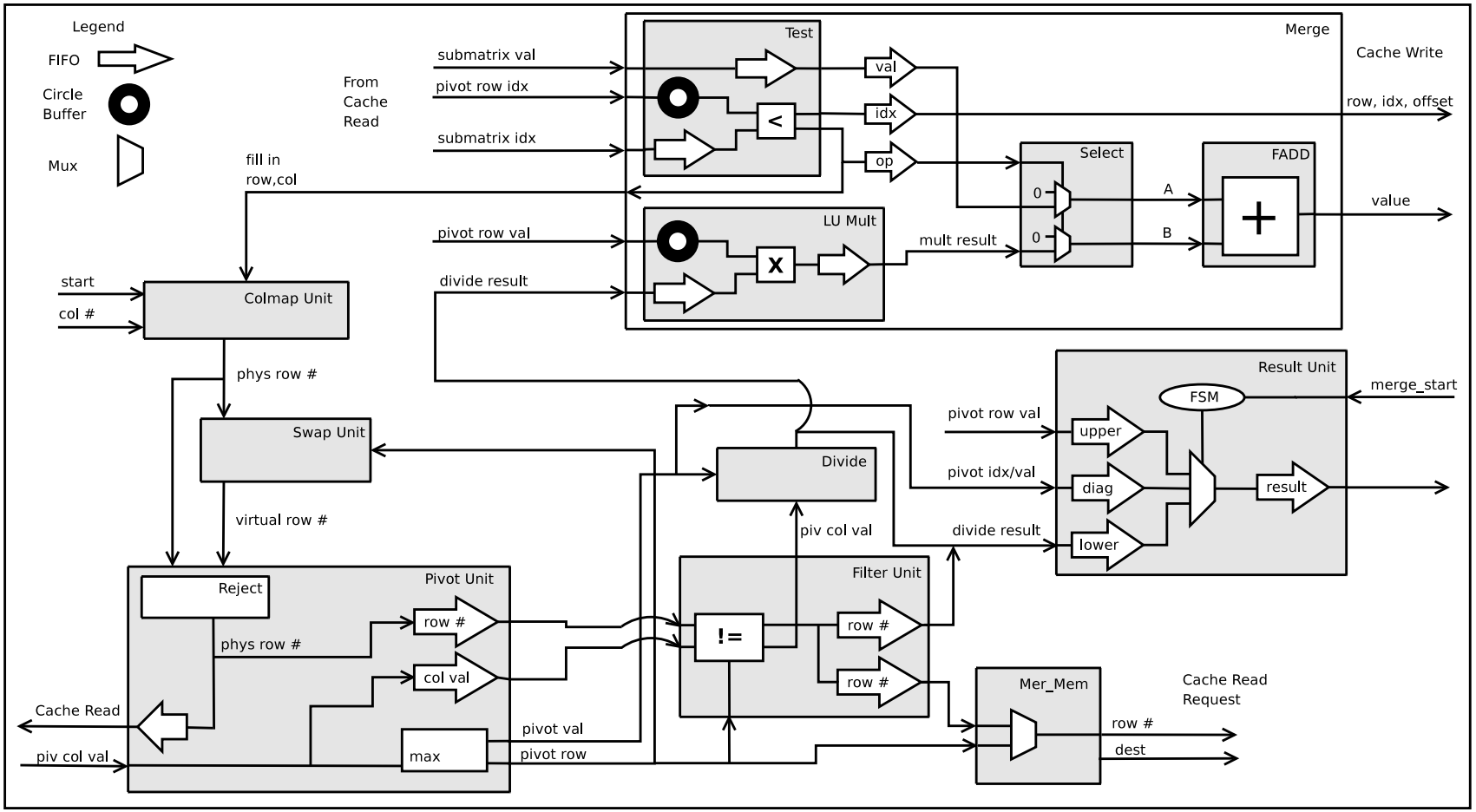
LU Hardware

- Originally designed by Petya Vachranukunkiet[12]
- Straightforward Gaussian Elimination
- Parameterized for power system matrices

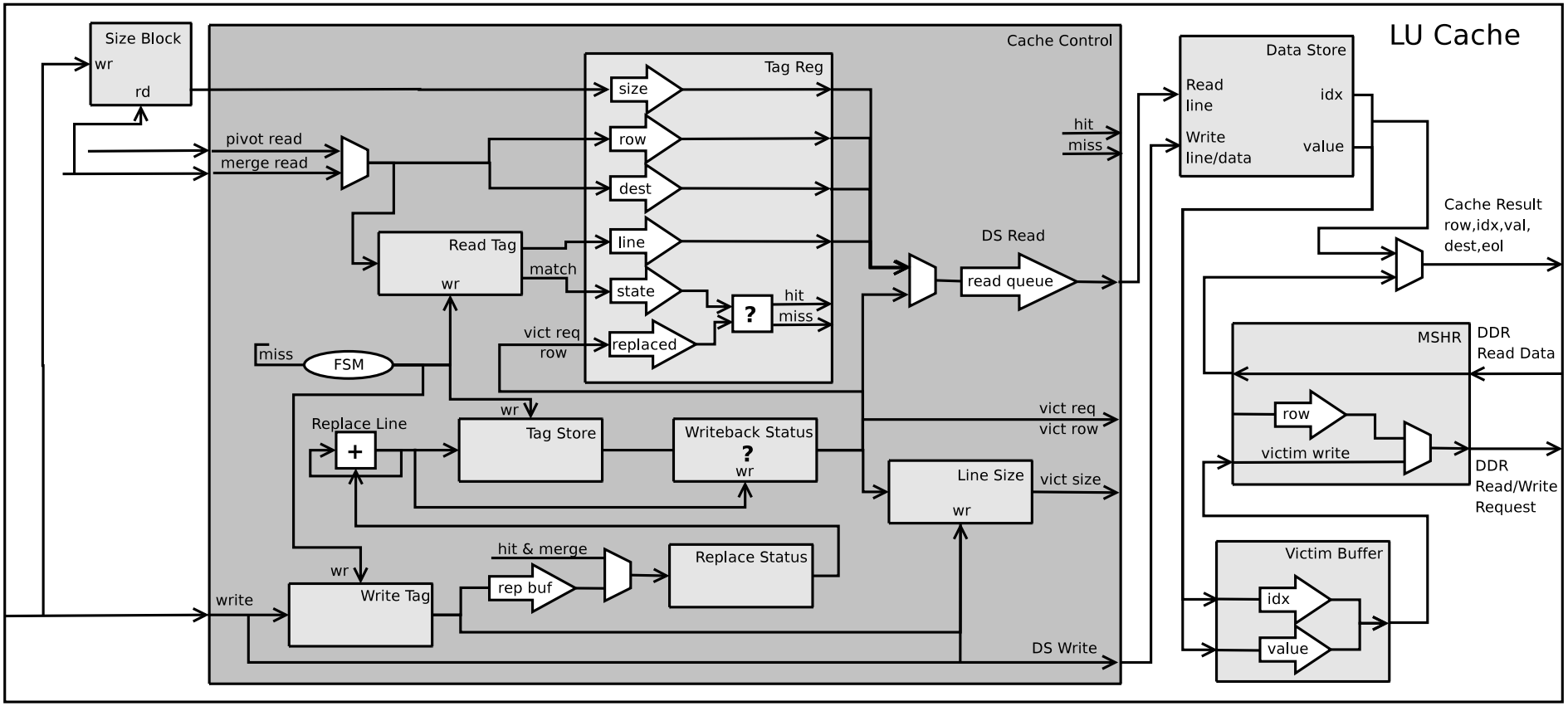
- Streaming operations hide latencies
- Computation pipelined for 1 Flop / cycle
- Memory and FIFOs used for cache and bookeeping
- Multiple merge units possible



Pivot and Submatrix Update Logic



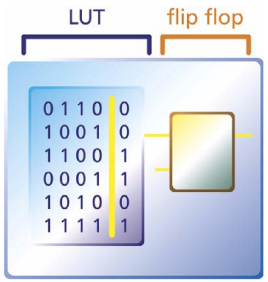
Custom Cache Logic



- Cache line is a whole row
- Fully-associative, write-back, FIFO replacement

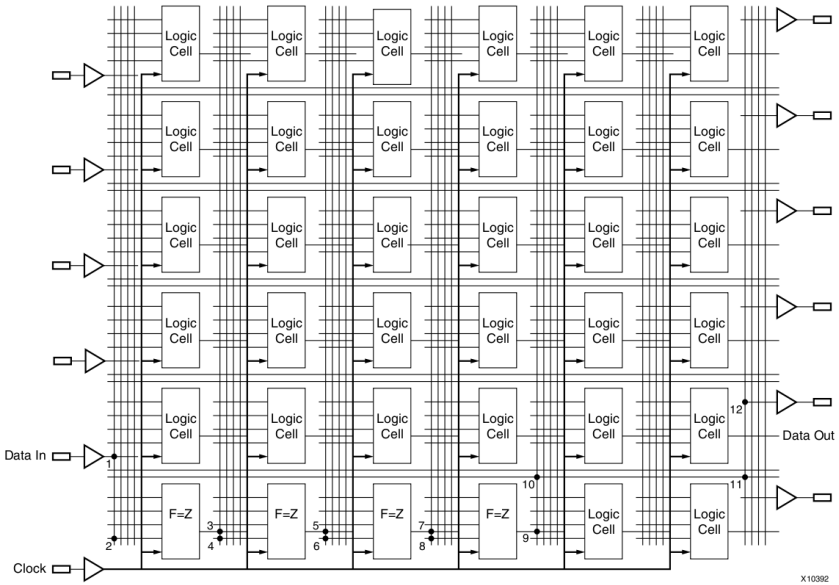
Reconfigurable Hardware

- Field Programmable Gate Array (FPGA)
- Advantages:
  - Low power consumption
  - Reconfigurable
  - Low cost
  - Pipelining and parallelism



SRAM Logic Cell [13]

- Disadvantages:
  - Difficult to program
  - Long design compile times
  - Lower clock frequencies



FPGA Architecture [13]

[13] Xilinx, Inc. Programmable Logic Design Quick Start Handbook.

ASIC vs FPGA

Application-Specific Integrated Circuit (ASIC)

- Higher clock frequency
- Lower power consumption
- More customization

Field Programmable Gate Array (FPGA)

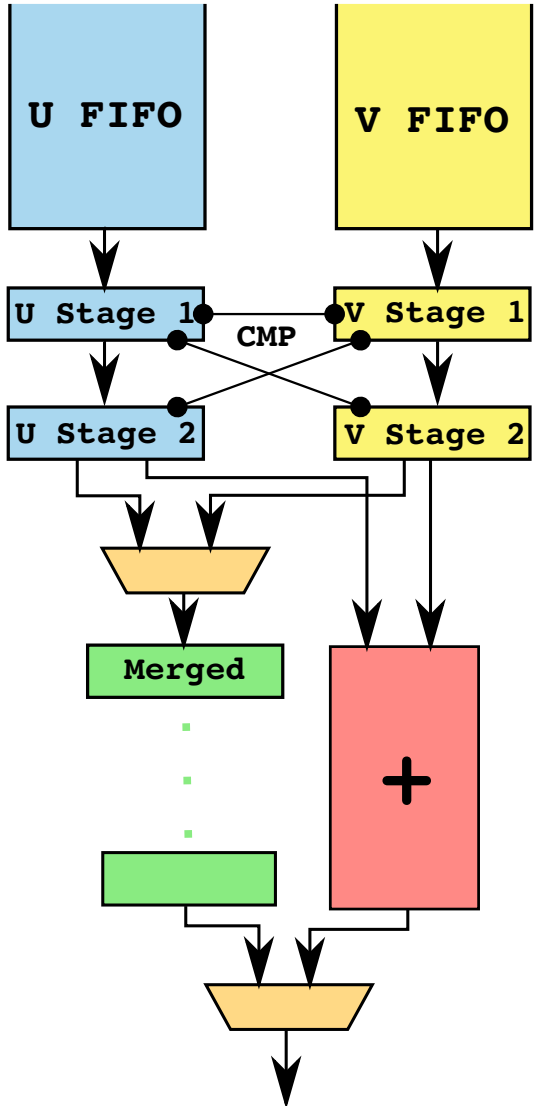
- Reconfigurable
- Design updates and corrections
- Multiple designs in same area
- Lower cost
- Less implementation time

Overview

- Problem and Goals
- Background
- LU Hardware
- Merge Accelerator Design
- Accelerator Architectures
- Performance Results
- Conclusions

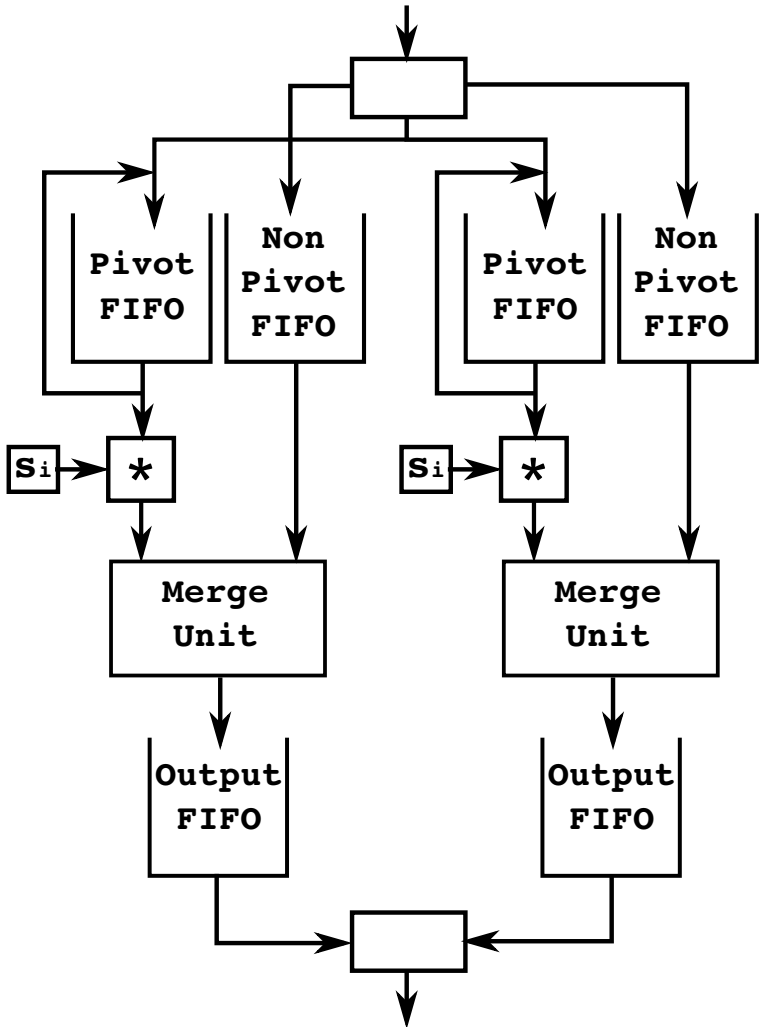
Merge Hardware Accelerator

- Input and Output BlockRAM FIFOs
- Two compare stages allow look-ahead comparison
- Pipelined floating point adder
- Merge unit is pipelined and outputs one element per cycle
- Latency is dependent on the structure and length of input rows



Merge Hardware Manager

- Must manage rows going through merge units
- Pivot row is recycled
- Floating point multiplier scales pivot row before entering merge unit
- Can support multiple merge units
- Input and output rotate among merge units after each full row
- Rows enter and leave in the same order



Merge Performance

Software Merge Performance on Core i7 at 3.2GHz

Power System	Average Data Rate (Millions of Elements per Second)	Average Cycles per Element
1648 Bus	90.77	35.3
7917 Bus	99.54	32.2
10279 Bus	100.75	31.8
26829 Bus	97.46	32.8

- Merge accelerator:
  - Outputs one element per cycle
  - Data rate approaches hardware clock rate
  - FPGA clock frequencies provide better merge performance than processor
- **Goal:** Find an architecture that efficiently delivers data to the accelerator

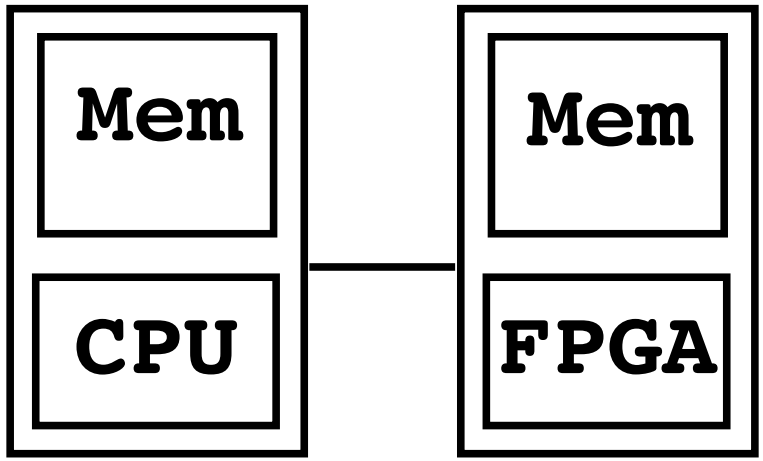
Overview

- Problem and Goals
- Background
- LU Hardware
- Merge Accelerator Design
- Accelerator Architectures
- Performance Results
- Conclusions

Summary of Design Methods

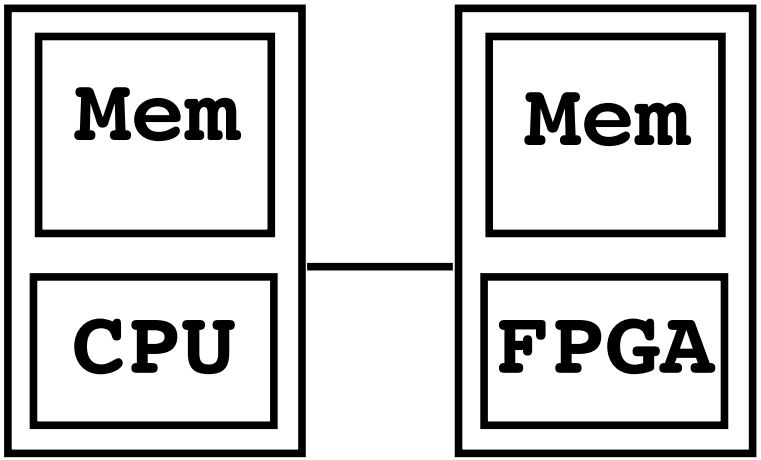
- Completely Software (UMFPACK, Gaussian LU)
  - Inefficient performance
- Completely Hardware (LUHW on FPGA)
  - Large design time and effort
  - Less scalable
- Investigate three accelerator architectures:
  - Triple Buffer Architecture
    - Efficiently utilize external transfer bus to a reconfigurable accelerator
  - Heterogeneous Multicore Architecture
    - Combine general-purpose and reconfigurable cores in a single package
  - Data Pump Architecture
    - Take advantage of programmable data/memory management to efficiently feed an accelerator

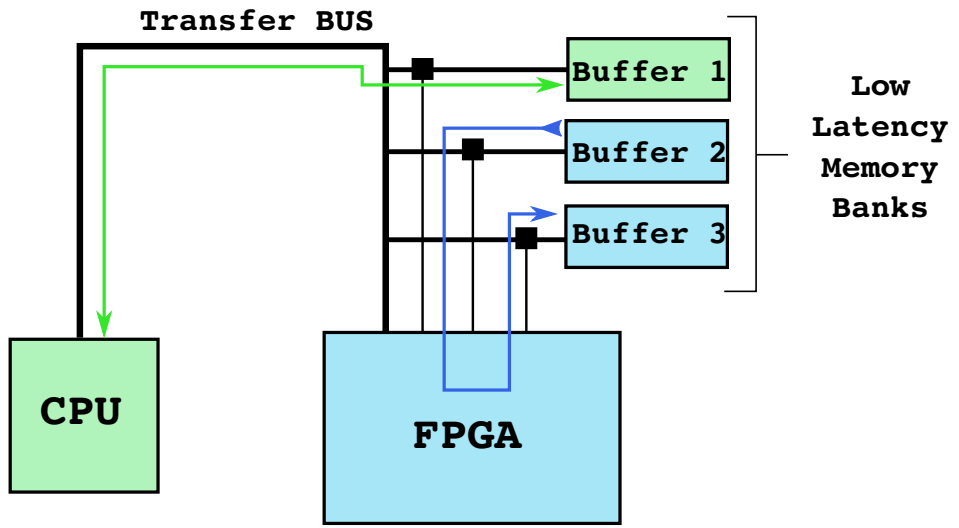
Accelerator Architectures



- Common setup for processor and FPGA
- Each device has its own external memory
- Communicate by USB, Ethernet, PCI-Express, HyperTransport, ...
- Transfer bus can be bottleneck between processor and FPGA

Triple Buffer Architecture

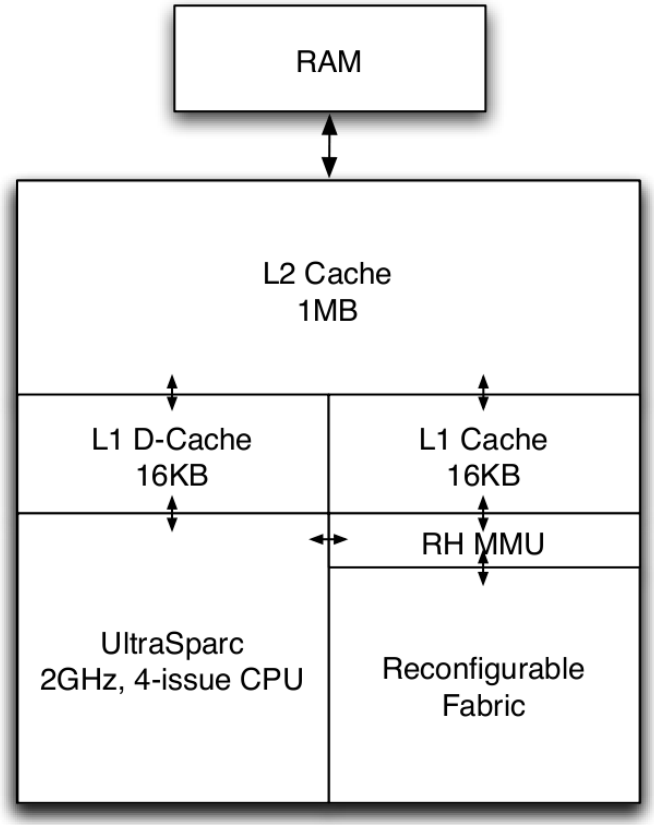




- How to efficiently use transfer bus?
- Break data into blocks and buffer
- Three buffers eliminate competition on memory ports
- Allows continuous stream of data through FPGA accelerator
- Processor can transfer data at bus/memory rate, not limited by FPGA rate

Heterogeneous Multicore Architecture

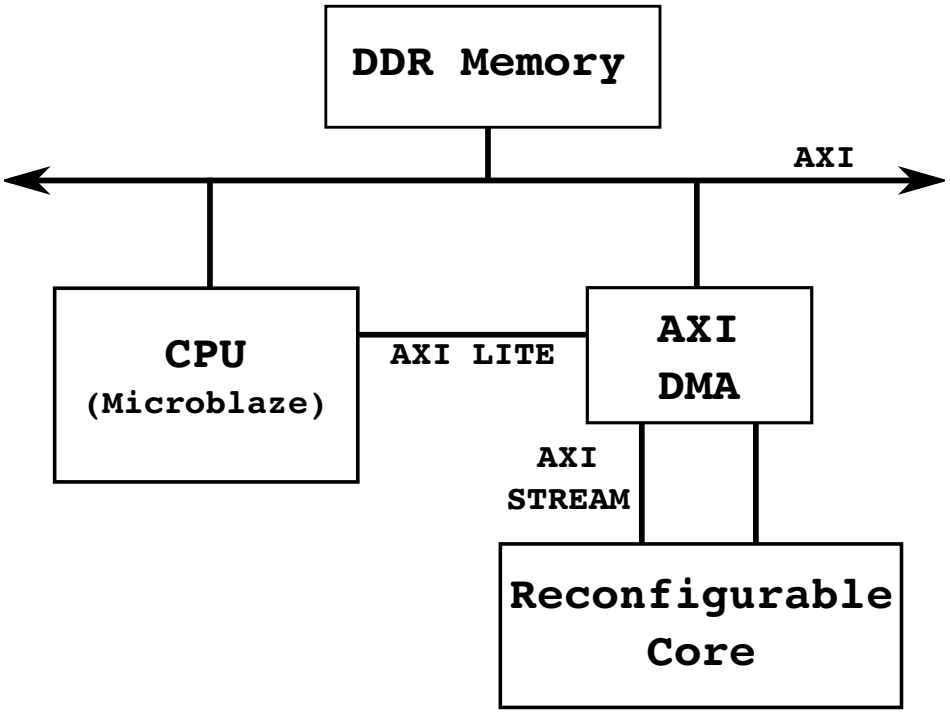
- Communication with an external transfer bus requires additional overhead
- Merge accelerator does not require a large external FPGA
- Closely integrate processor and FPGA
- Share same memory system

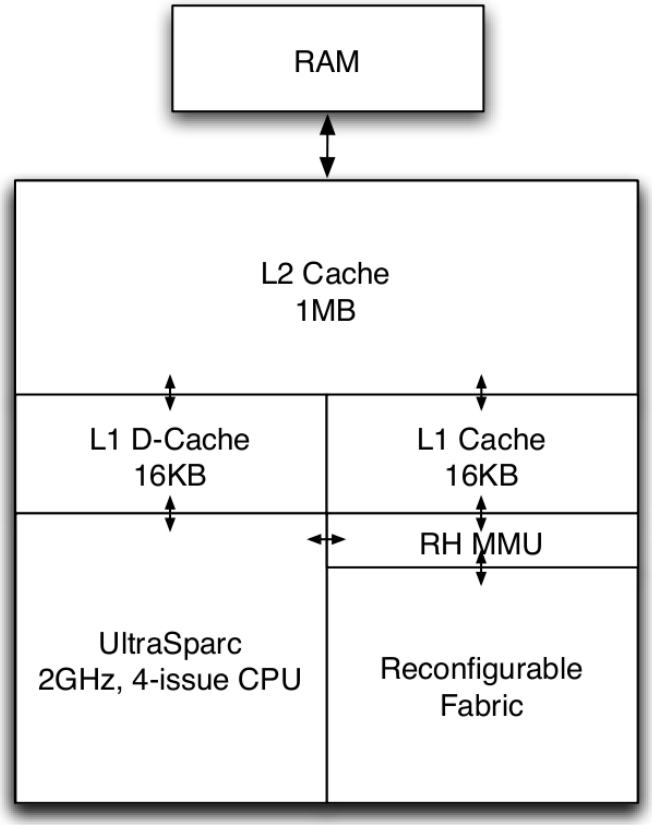


Reconfigurable Processor Architecture [8]

[8] Garcia and Compton. A Reconfigurable Hardware Interface for a Modern Computing System.

Heterogeneous Multicore Architecture

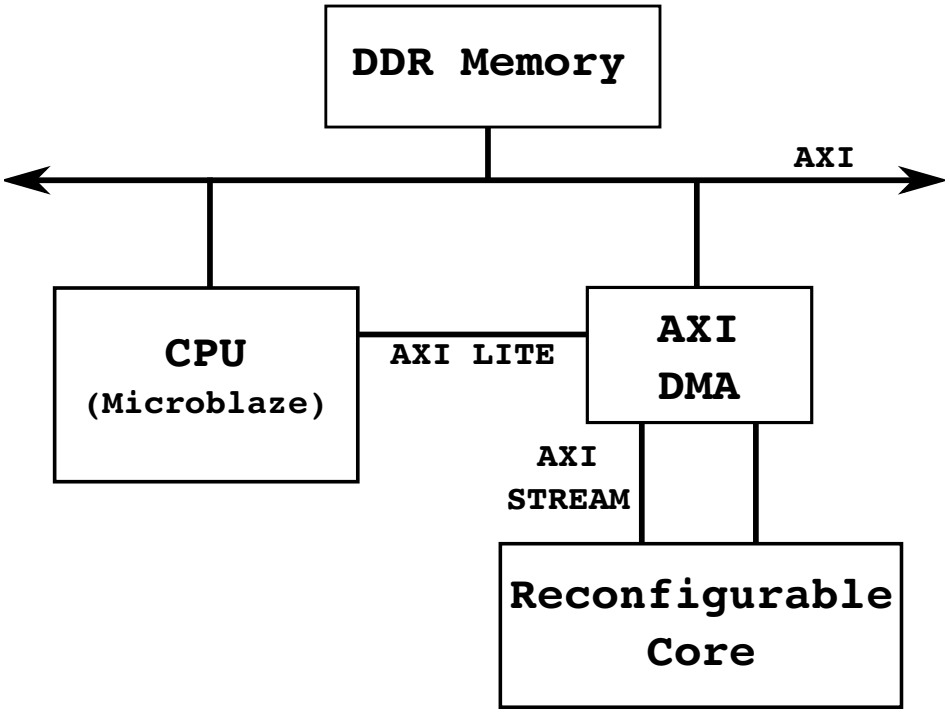




Reconfigurable Processor Architecture [8]

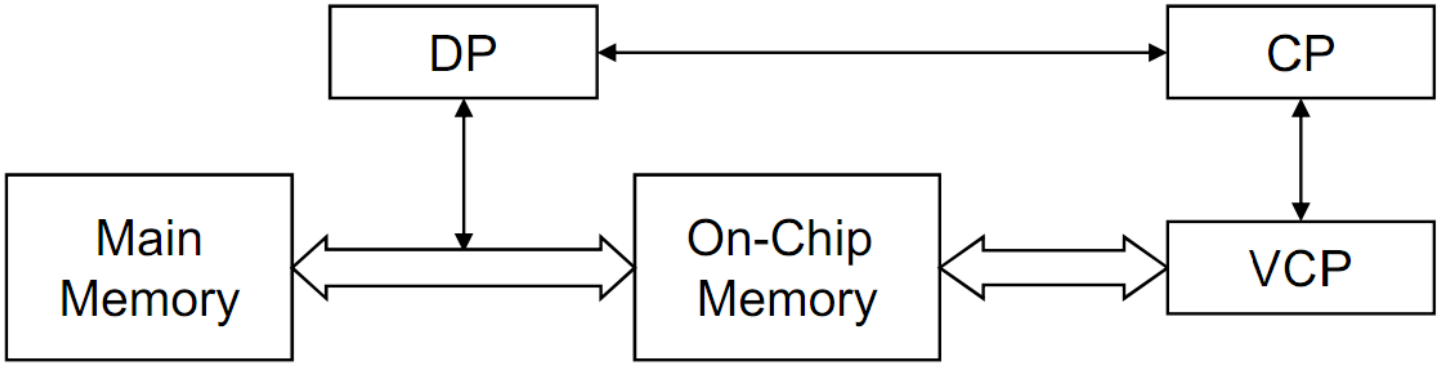
[8] Garcia and Compton. A Reconfigurable Hardware Interface for a Modern Computing System.

Heterogeneous Multicore Architecture



- Processor sends DMA requests to DMA module
- DMA module fetches data from memory and streams to accelerator
- Accelerator streams outputs back to DMA module to store to memory

Data Pump Architecture

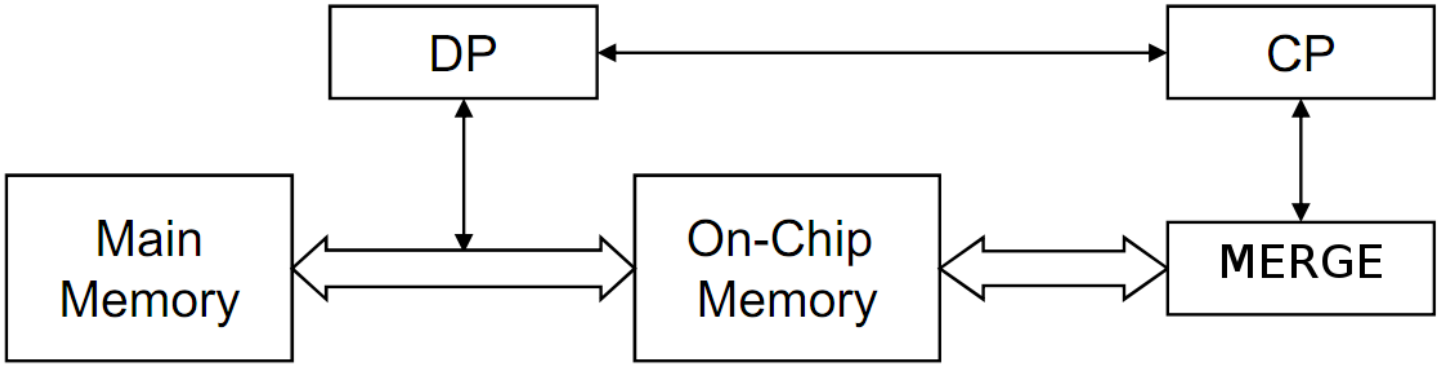


Data Pump Architecture [11]

- Developed as part of the SPIRAL project
- Intended as a signal processing platform
- Processors explicitly control data movement
- No automatic data caches
- Data Processor (DP) moves data between external and local memory
- Compute Processor (CP) moves data between local memory and vector processors

[11] SPIRAL. DPA Instruction Set Architecture V0.2 Basic Configuration.

Data Pump Architecture



Data Pump Architecture [11]

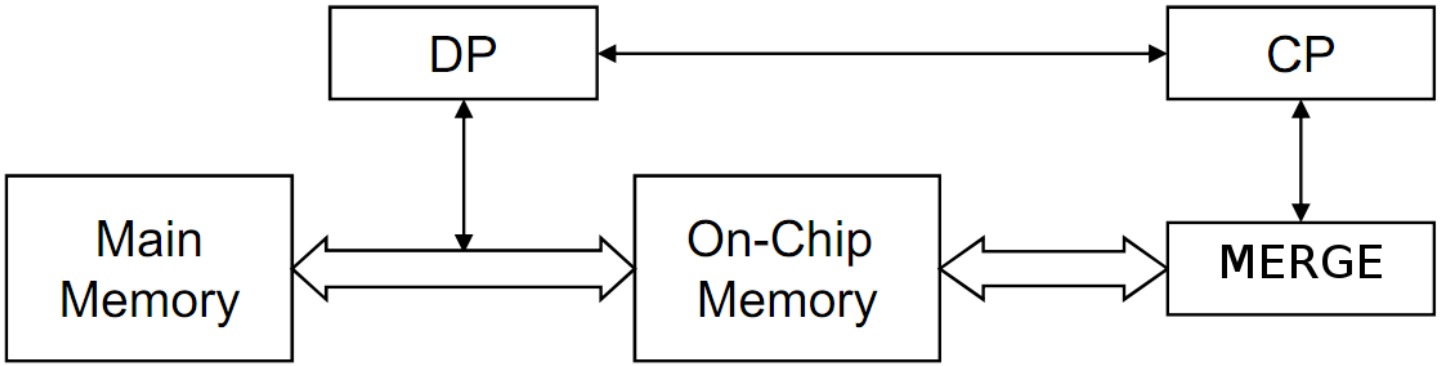
- Replace vector processors with merge accelerator
- DP brings rows from external memory into local memory
- CP sends rows to merge accelerator and writes results in local memory
- DP stores results back to external memory

[11] SPIRAL. DPA Instruction Set Architecture V0.2 Basic Configuration.

Overview

- Problem and Goals
- Background
- LU Hardware
- Merge Accelerator Design
- Accelerator Architectures
- Performance Results
- Conclusions

DPA Implementation



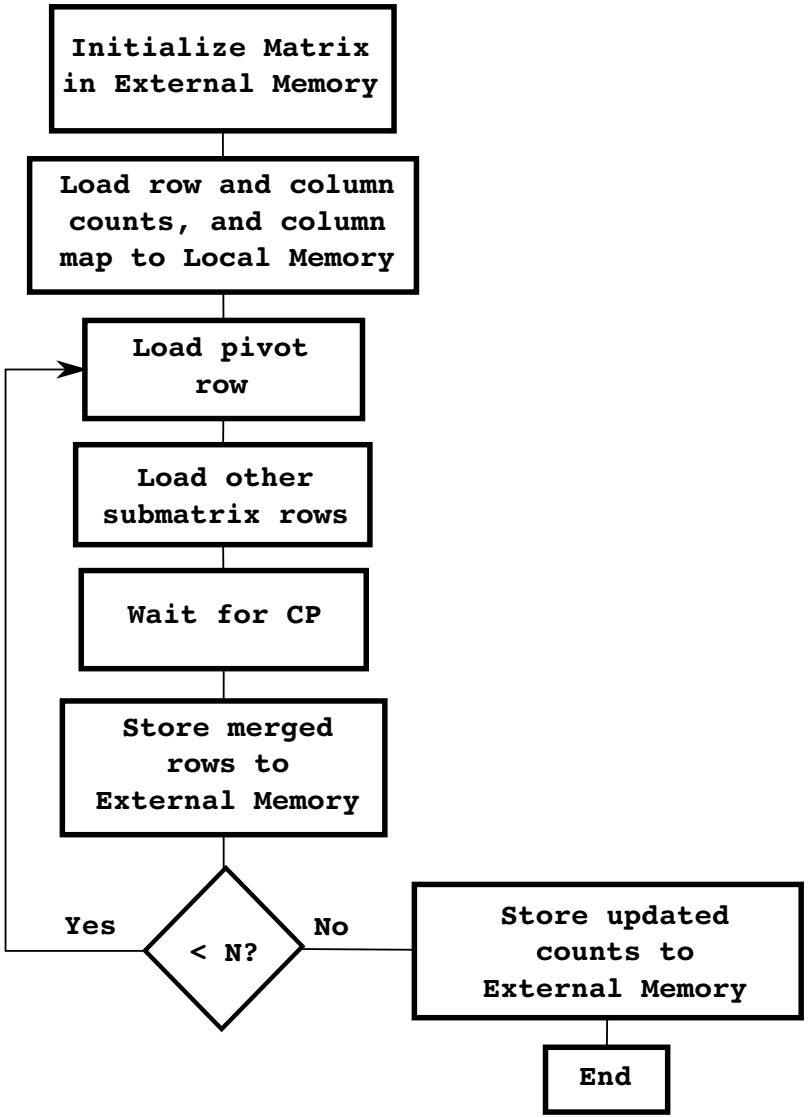
Data Pump Architecture [11]

- Implemented Sparse LU on DPA Simulator [10]
- DP and CP synchronize with local memory read/write counters
- Double buffer row inputs and outputs in local memory
- DP and CP at 2GHz, DDR at 1066MHz
- Single merge unit

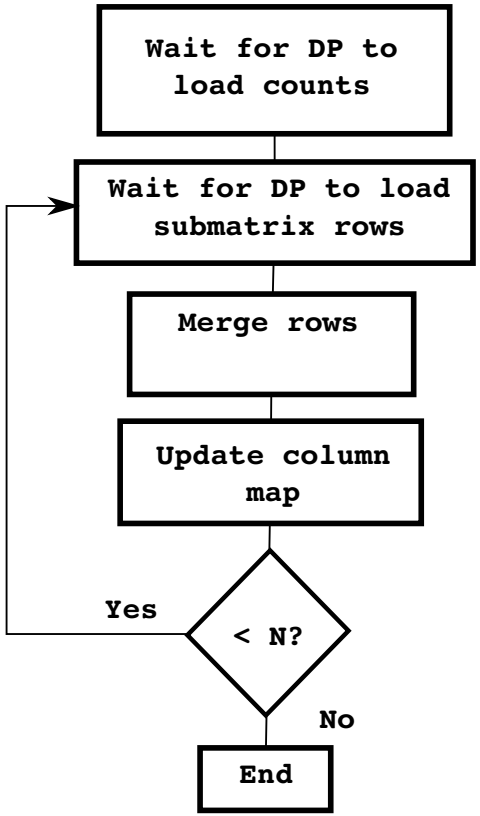
[11] SPIRAL. DPA Instruction Set Architecture V0.2 Basic Configuration.
[10] D. Jones. Data Pump Architecture Simulator and Performance Model.

DPA Implementation

Sparse LU DP

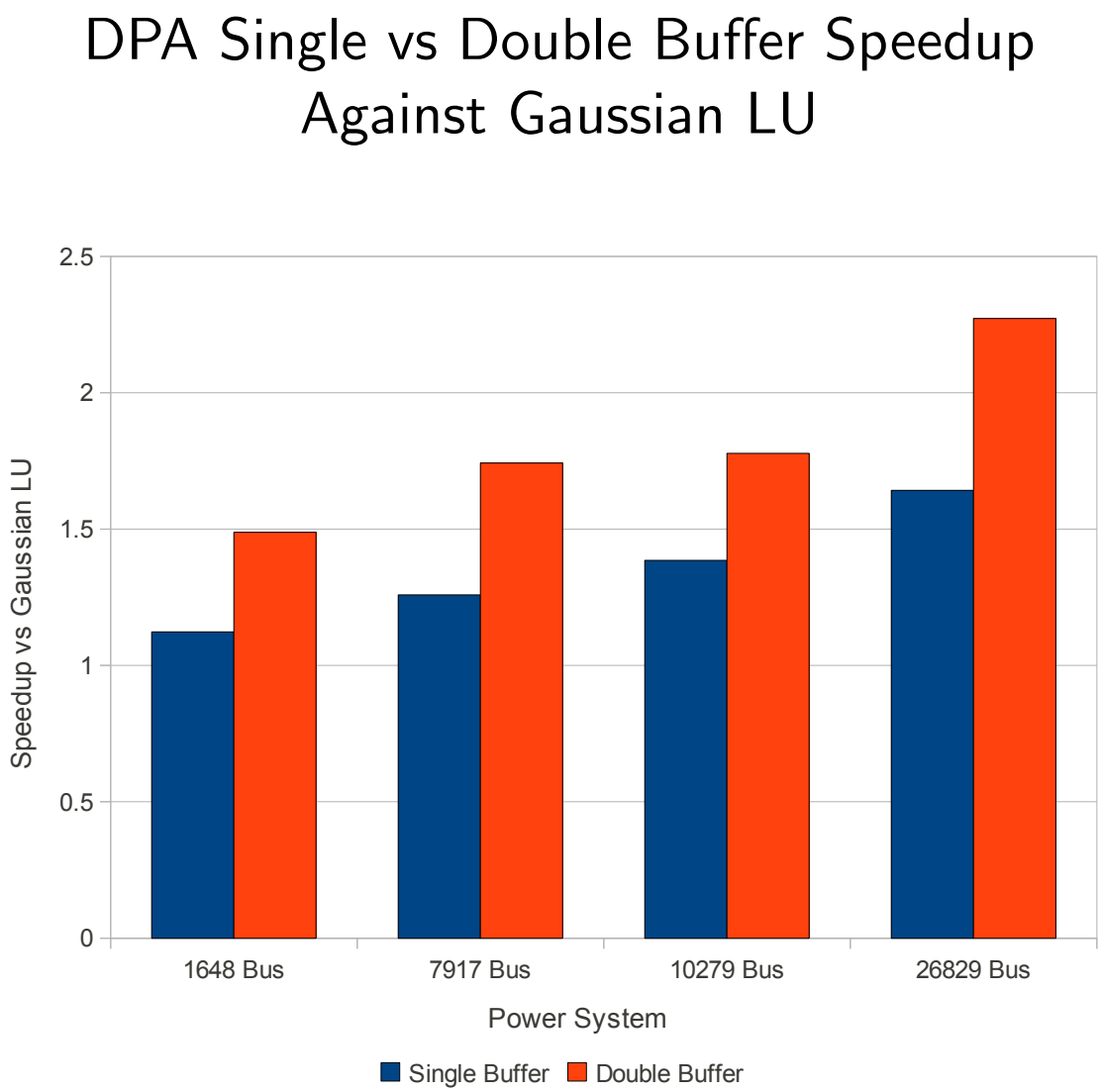


Sparse LU CP

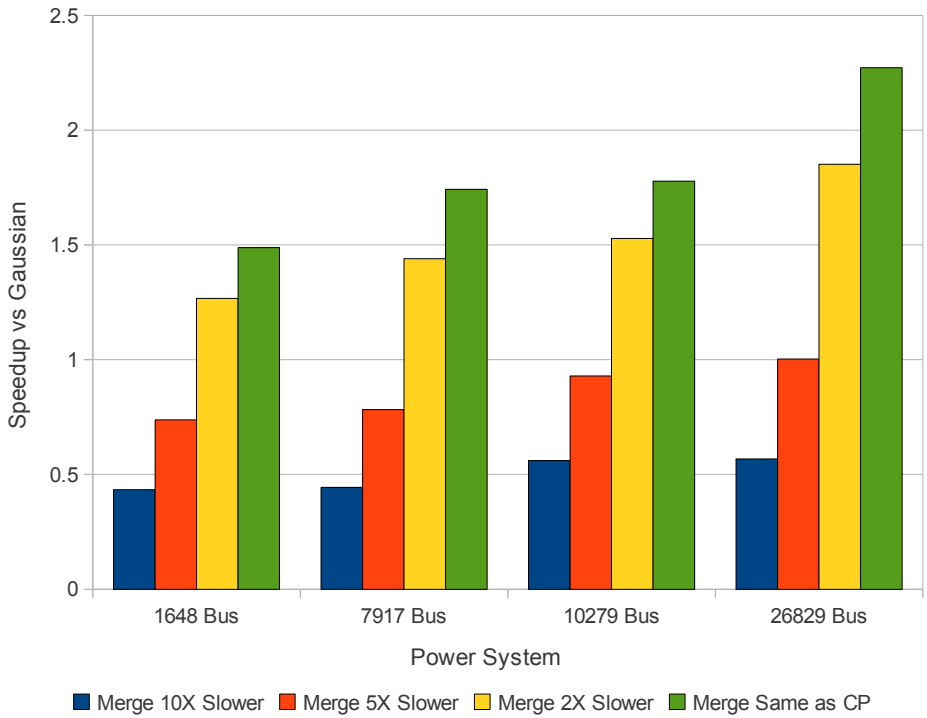


DPA Performance

- Increase performance by double buffering rows
- DP alternates loading to two input buffers
- CP alternates outputs to two output buffers
- DP and CP at 2GHz, DDR at 1066MHz
- Merge accelerator at same frequency as CP

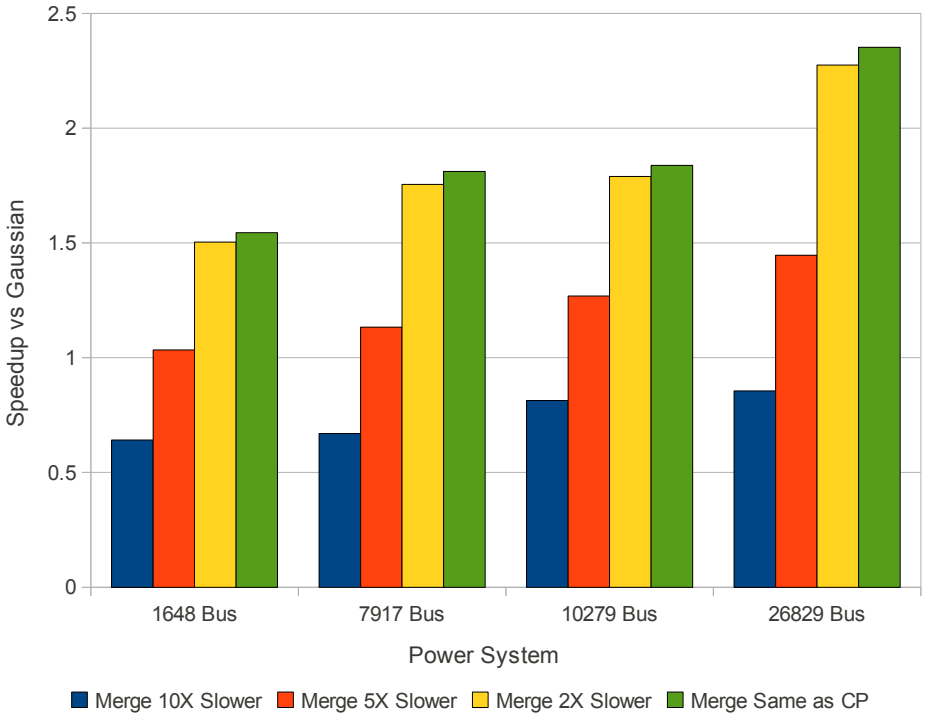


DPA Performance



- Measured with different merge frequencies
- Slower merges do not provide speedup over software
- Faster merges require ASIC implementation
- Larger matrices see most benefit

DPA Performance



- DP and CP at 3.2GHz
- DDR at 1066MHz

DPA LU Speedup vs Gaussian LU for 26K-Bus Power System

CP/DP Freq	10 times slower	5 times slower	2 times slower	Same as CP
2.0GHz	0.57X	1.00X	1.85X	2.27X
3.2GHz	0.86X	1.45X	2.28X	2.35X

Overview

- Problem and Goals
- Background
- LU Hardware
- Merge Accelerator Design
- Accelerator Architectures
- Performance Results
- Conclusions

Conclusions and Future Work

- Merge accelerator reduces indexing cost and improves sparse LU performance (2.3X improvement)

Conclusions and Future Work

- Merge accelerator reduces indexing cost and improves sparse LU performance (2.3X improvement)
- DPA architecture provides a memory framework to utilize merge accelerator and outperforms triple buffering and heterogeneous multicore schemes investigated

Conclusions and Future Work

- Merge accelerator reduces indexing cost and improves sparse LU performance (2.3X improvement)
- DPA architecture provides a memory framework to utilize merge accelerator and outperforms triple buffering and heterogeneous multicore schemes investigated
- Software control of memory hierarchy allows alternate protocols such as double buffering to be investigated

Conclusions and Future Work

- Merge accelerator reduces indexing cost and improves sparse LU performance (2.3X improvement)
- DPA architecture provides a memory framework to utilize merge accelerator and outperforms triple buffering and heterogeneous multicore schemes investigated
- Software control of memory hierarchy allows alternate protocols such as double buffering to be investigated
- Additional modifications such as multiple merge units, row caching and prefetching will be required for further improvement

Conclusions and Future Work

- Merge accelerator reduces indexing cost and improves sparse LU performance (2.3X improvement)
- DPA architecture provides a memory framework to utilize merge accelerator and outperforms triple buffering and heterogeneous multicore schemes investigated
- Software control of memory hierarchy allows alternate protocols such as double buffering to be investigated
- Additional modifications such as multiple merge units, row caching and prefetching will be required for further improvement
- With accelerator, high performance load flow calculation is possible with low power device
- Suggesting a distributed network of such devices

Conclusions and Future Work

- Merge accelerator reduces indexing cost and improves sparse LU performance (2.3X improvement)
- DPA architecture provides a memory framework to utilize merge accelerator and outperforms triple buffering and heterogeneous multicore schemes investigated
- Software control of memory hierarchy allows alternate protocols such as double buffering to be investigated
- Additional modifications such as multiple merge units, row caching and prefetching will be required for further improvement
- With accelerator, high performance load flow calculation is possible with low power device
- Suggesting a distributed network of such devices

References

[1] A. Albur, A., Expsito.
Power System State Estimation: Theory and Implementation.
Marcel Dekker, New York, New York, USA, 2004.

[2] P R Amestoy, Timothy Davis, and I S Duff.
Algorithm 837: AMD, an approximate minimum degree ordering algorithm, 2004.

[3] T. Chagnon.
Architectural Support for Direct Sparse LU Algorithms.
Masters thesis, Drexel University, 2010.

[4] DRC Computer.
DRC Coprocessor System User's Guide, 2007.

[5] Kevin Cunningham and Prawat Nagvajara.
Reconfigurable Stream-Processing Architecture for Sparse Linear Solvers.
Reconfigurable Computing: Architectures, Tools and Applications, 2011.

[6] Kevin Cunningham, Prawat Nagvajara, and Jeremy Johnson.
Reconfigurable Multicore Architecture for Power Flow Calculation.
In *Review for North American Power Symposium 2011*, 2011.

[7] Timothy Davis.
Algorithm 832:UMFPACK V4.3 - An Unsymmetric-Pattern Multi-Frontal Method.
ACM Trans. Math. Softw, 2004.

[8] Philip Garcia and Katherine Compton.
A Reconfigurable Hardware Interface for a Modern Computing System.
15th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM 2007), pages 73–84, April 2007.

[9] Intel.
Intel Atom Processor E6x5C Series-Based Platform for Embedded Computing.
Technical report, 2011.

[10] Douglas Jones.
Data Pump Architecture Simulator and Performance Model.
Masters thesis, 2010.

[11] SPIRAL.
DPA Instruction Set Architecture V0.2 Basic Configuration (SPARC V8 Extension), 2008.

[12] Petya Vachranukunkiet.
Power flow computation using field programmable gate arrays.
PhD thesis, 2007.

[13] Xilinx Inc.
Programmable Logic Design Quick Start Handbook, 2006.