UNIVERSITY OF FLORIDA

www.hcs.ufl.edu
High-performance Computing & Simulation Research Lab

RapidIO-based Space System Architectures for Synthetic Aperture Radar and Ground Moving Target Indicator

David Bueno, Chris Conger, Adam Leko, Ian Troxel, and Alan D. George

HCS Research Laboratory
College of Engineering
University of Florida

22 September 2005

Outline

I. Project Overview
II. Background
    I. RapidIO (RIO)
    II. Synthetic Aperture Radar (SAR)
    III. Ground Moving Target Indicator (GMTI)
III. Model Library Overview
IV. RapidIO Experimental Testbed
    I. Overview
    II. Validation results
V. Simulation Experiments and Results
VI. Conclusions

Project Overview

- **Simulative analysis of real-time Space-Based Radar (SBR) systems using RapidIO interconnection networks**
  - **RapidIO is a high-performance, switched interconnect for embedded systems**
- **Experimental validation of simulation models using a RapidIO hardware testbed**
- **Sensitivity analysis of GMTI and SAR to RIO network and algorithm parameters**
  - **Uses discrete-event simulation of RapidIO network, processing elements, and SBR algorithms**
  - **Examine considerations for designing RIO-based systems capable of both SAR and GMTI**

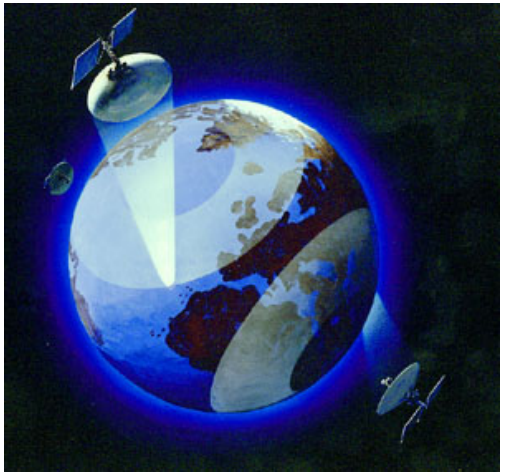


Image courtesy [6]

Background: RapidIO

- Three-layered, embedded system interconnect architecture
  - Logical layer, transport layer, physical layer
- Point-to-point, packet-switched interconnect
- Peak single-link throughput ranging from 2 to 64 Gb/s
- Focus on 16-bit parallel LVDS RIO design for space systems

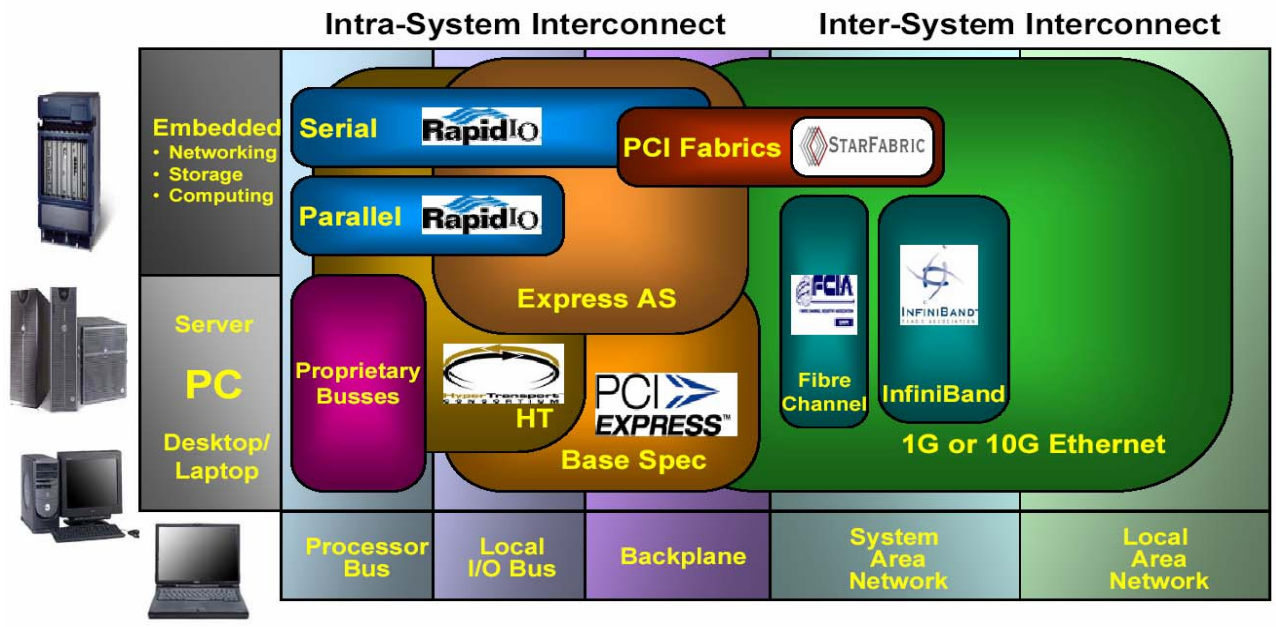
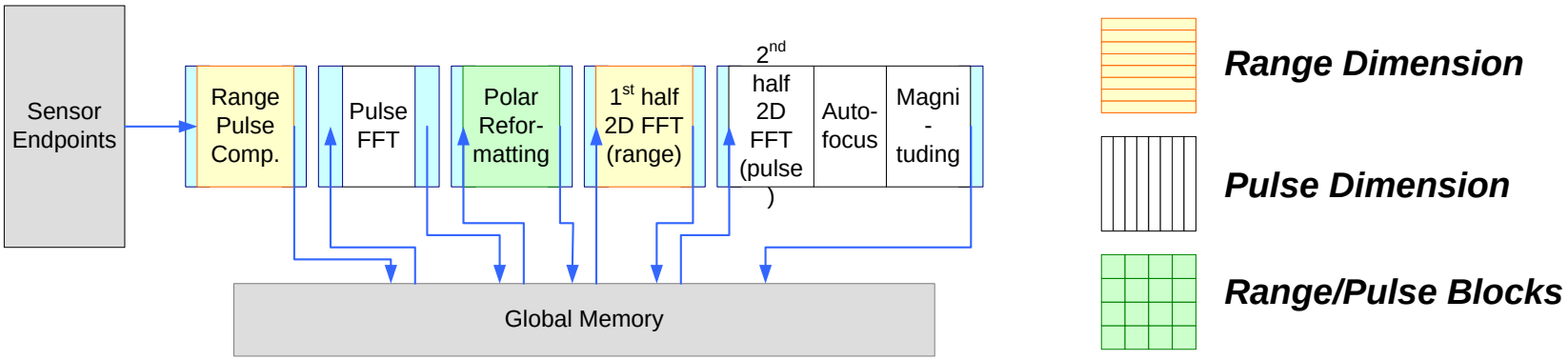


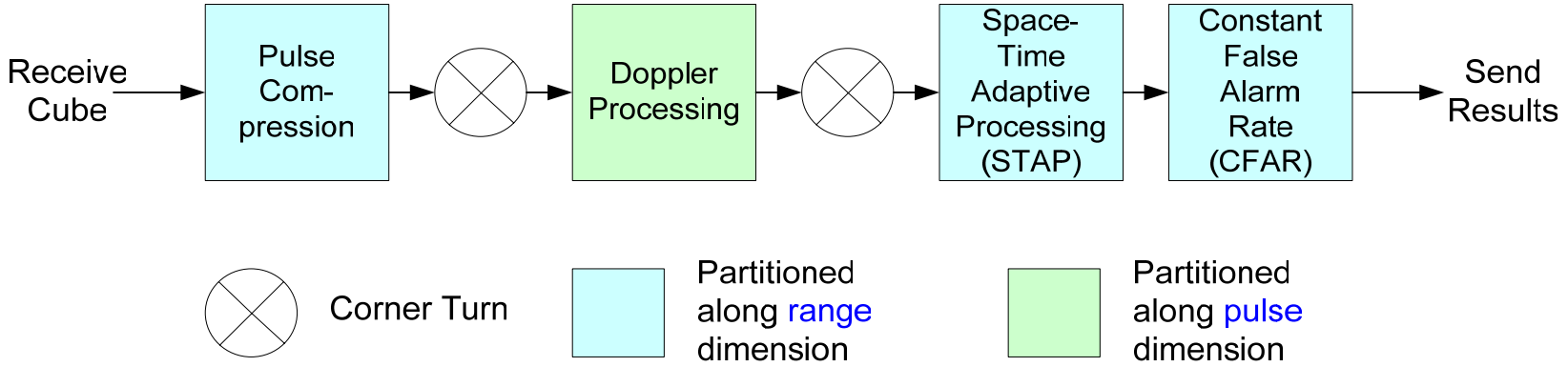
Image courtesy [7]

Background: SAR

- SAR used to take broad-range, high-resolution snap-shots of surface features from satellites, even at night or inclement weather
- Data set is 2D image or matrix, with range and pulse dimension
  - Typical data size is 2GB, each matrix element a 64-bit complex integer
  - Due to large data set size, image processed iteratively in chunks
- Figure below illustrates each stage of algorithm
  - Color denotes partitioning (see legend to right of picture)
  - Blue lines/blocks represent communication events
- Processors write data back to global memory for repartitioning after processing completes in each subtask, if necessary



Background: GMTI

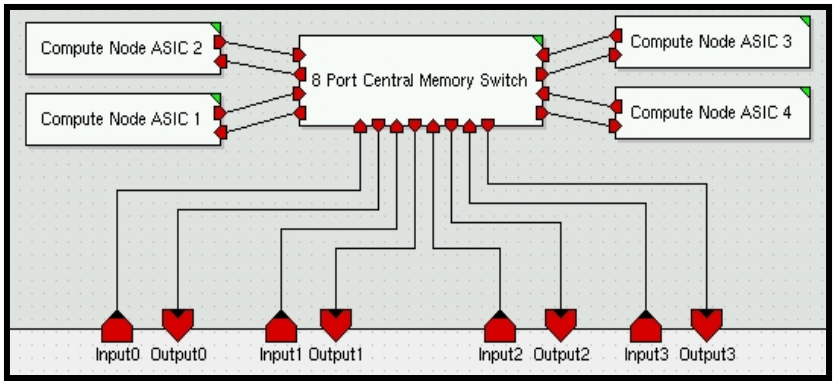


- GMTI used to track moving targets on ground
- Incoming data organized as 3-D matrix (data cube)
  - Data reorganized between stages for processing efficiency
  - Real-time processing deadline for each cube defined as Coherent Processing Interval (CPI)
- Data set size for GMTI much smaller than SAR, however time constraints allow much less time for processing
  - Unlike SAR, entire data set may be stored in processor memories
  - No communication with global memory during processing, inter-processor communication between subtasks (corner turns)
- Previous work in [1] examined various partitionings of GMTI over RIO-based systems

Model Library Overview

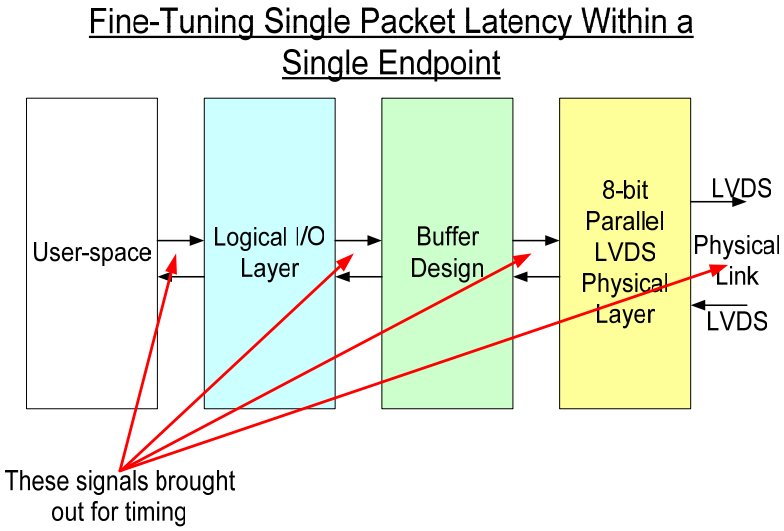
- **Modeling library created using Mission Level Designer (MLD), a commercial discrete-event simulation modeling tool**
  - **C++-based, block-level, hierarchical modeling tool**
- **Our RIO-SBR model library includes:**
  - **Compute node with RIO endpoint**
    - **IO and message passing RIO logical layers**
    - **RIO parallel physical layer**
    - **Script-based processor model**
  - **RIO central-memory switch**
  - **Global memory (GM) board**
  - **External data source**
  - **See [1] for more details on model library**

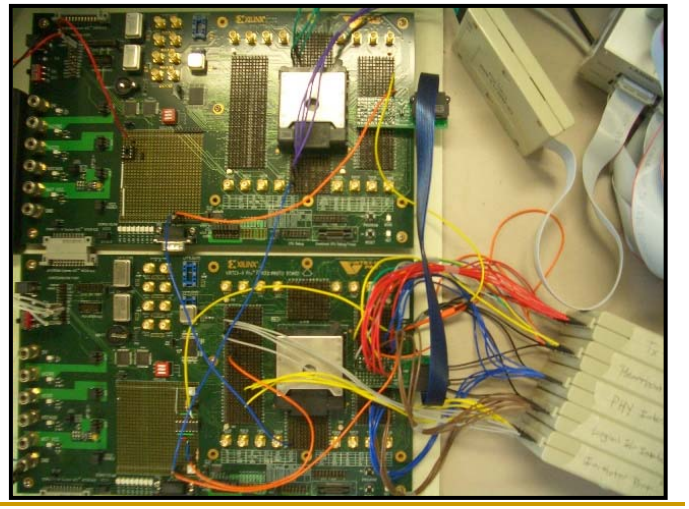
Model of Four-Processor Board



RapidIO Testbed and Model Validation

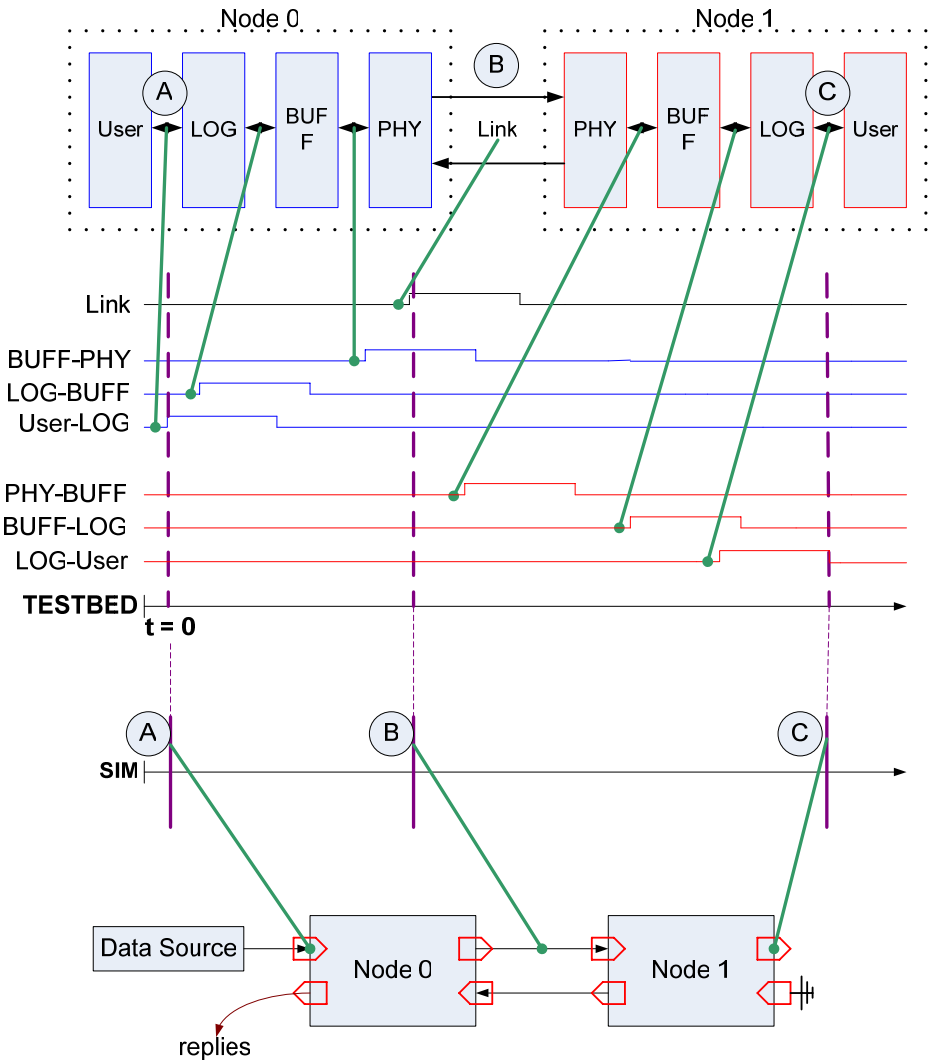
- 2-node RapidIO testbed constructed
  - Xilinx Virtex-II Pro FPGAs
  - Xilinx RapidIO Core
    - 250MHz link clock, 8-bit parallel
- Block diagram of single endpoint depicted in upper figure
  - Layer interface signals brought out for logic analyzer visibility
  - Timing probes inserted into simulation model in equivalent positions, used to validate timing
- Using same signals described above, transaction may be viewed across both endpoints
- Equivalent system constructed in MLD for validation





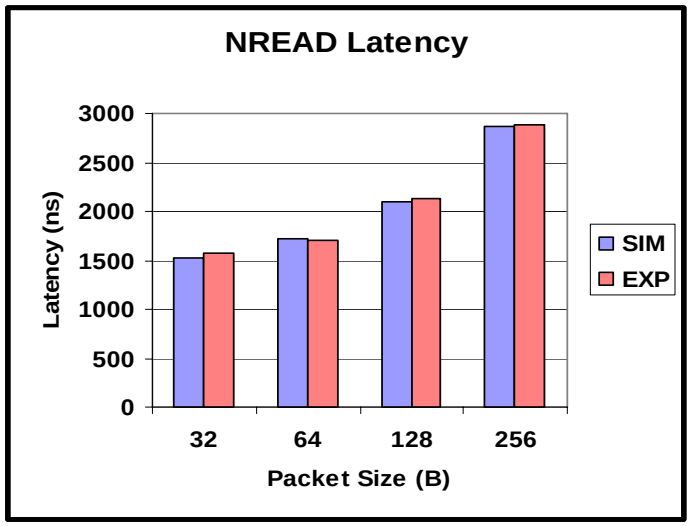
Calibration Experiments

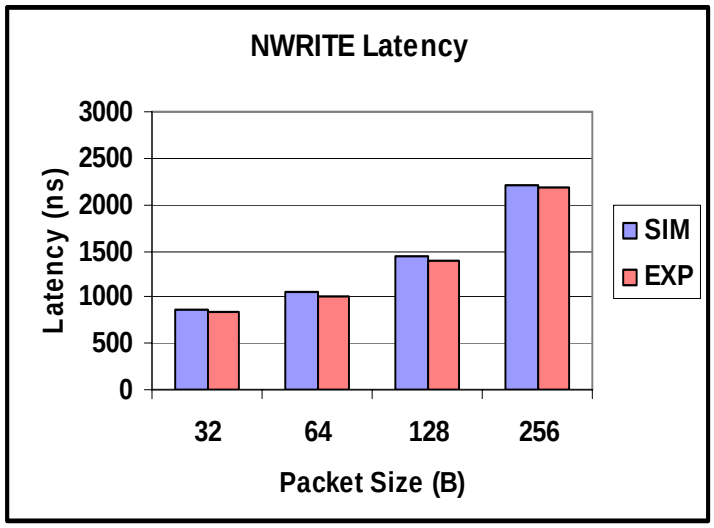
- Calibration experiments designed to match transaction latencies
  - Single-packet experiments measure internal endpoint latencies
    - 32B, 64B, 128B, and 256B
    - NREAD, NWRITE, NWRITE_R
  - Multi-packet experiments calibrate realistic transaction sizes, endpoint operation overhead
    - 1KB, 4KB, 16KB, 64KB, 256KB, 1MB, and 4MB transactions
    - NREAD, SWRITE, NWRITE_R
- Figure depicts SWRITE trans.



Validation Results: Latency

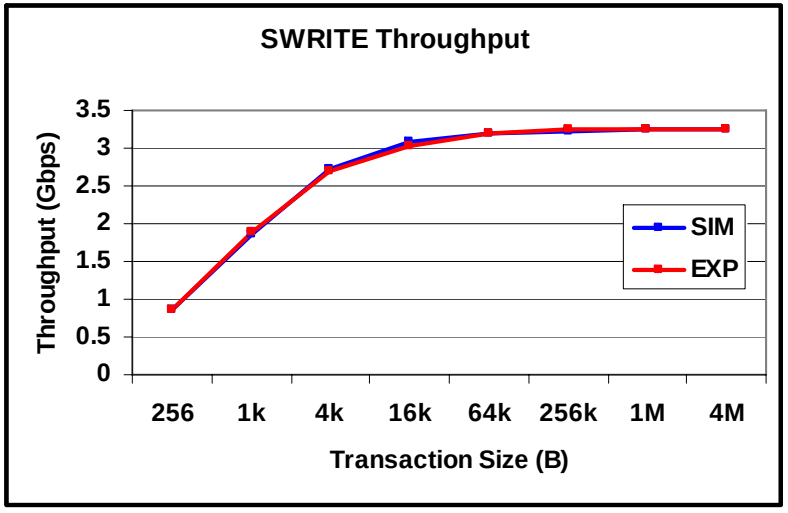
- **Single-packet latency results shown below for read and response-less write transactions**
  - **Simulation models match hardware to less than 5% error in all cases**
  - **NWRITE_R not shown, but results are consistent with those shown**
- **Maximum absolute difference between simulated and measured time is less than 40ns**
  - **Difference has two components, model error (up to 32ns) and measurement error (4-8ns)**

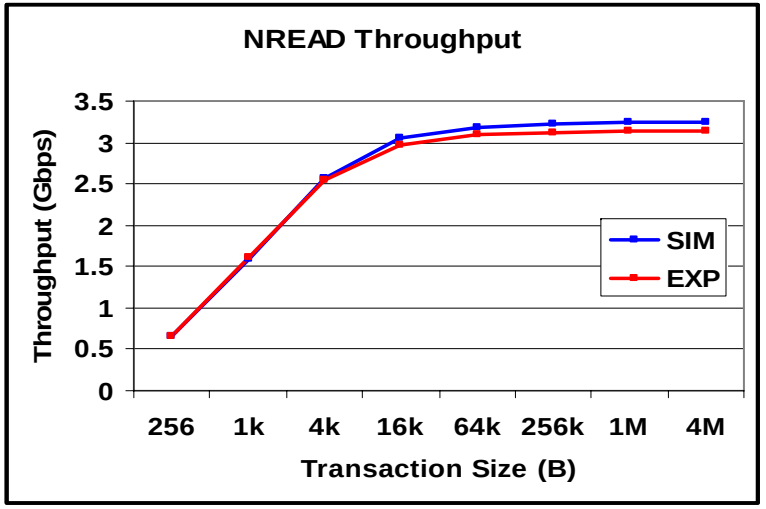




Validation Results: Throughput

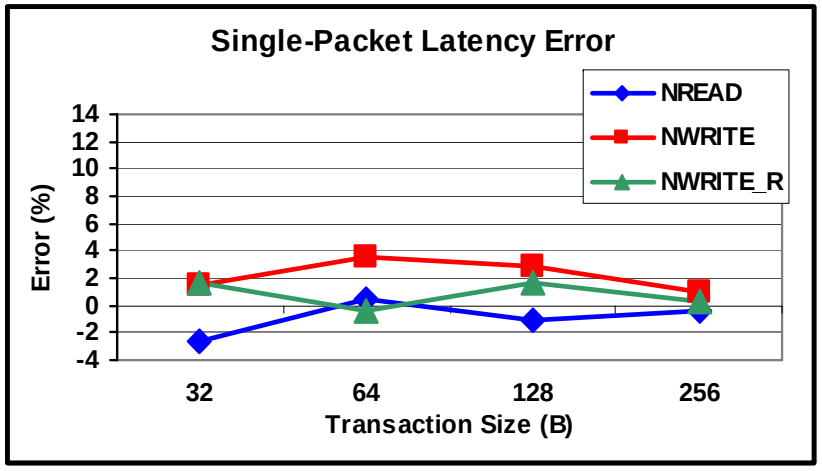
- Throughput results calculated based on measured latencies and amount of data transferred
- Maximum actual throughput of 3.25Gbps for write transactions, 3.14Gbps for read transactions
  - 4Gbps ideal throughput (250MHz DDR × 8 bits)
  - 3.76Gbps ideal effective throughput, considering header/CRC overhead
- Simulation models match hardware well (< 5% error) for small, medium, and large transactions

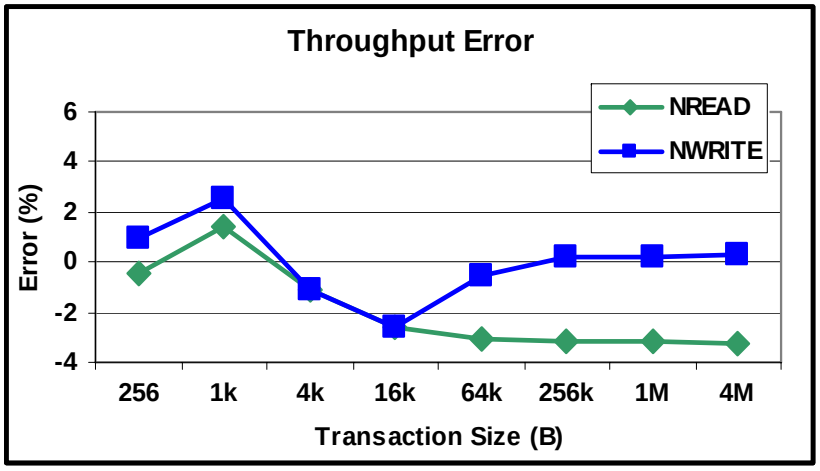




Validation Results: Error Analysis

- **Single-packet latency error**
  - Simulated packets of smaller-than-max size experience varying levels of error, depending upon transaction type and size
  - All transaction types simulate most accurately for max-sized packets
- **Throughput error**
  - Endpoints driven by user-space state machine, which contributes overhead between packet transfers not currently modeled
  - Error in simulation levels out for larger transaction sizes
- **In all cases, simulation models match hardware to within 4%**



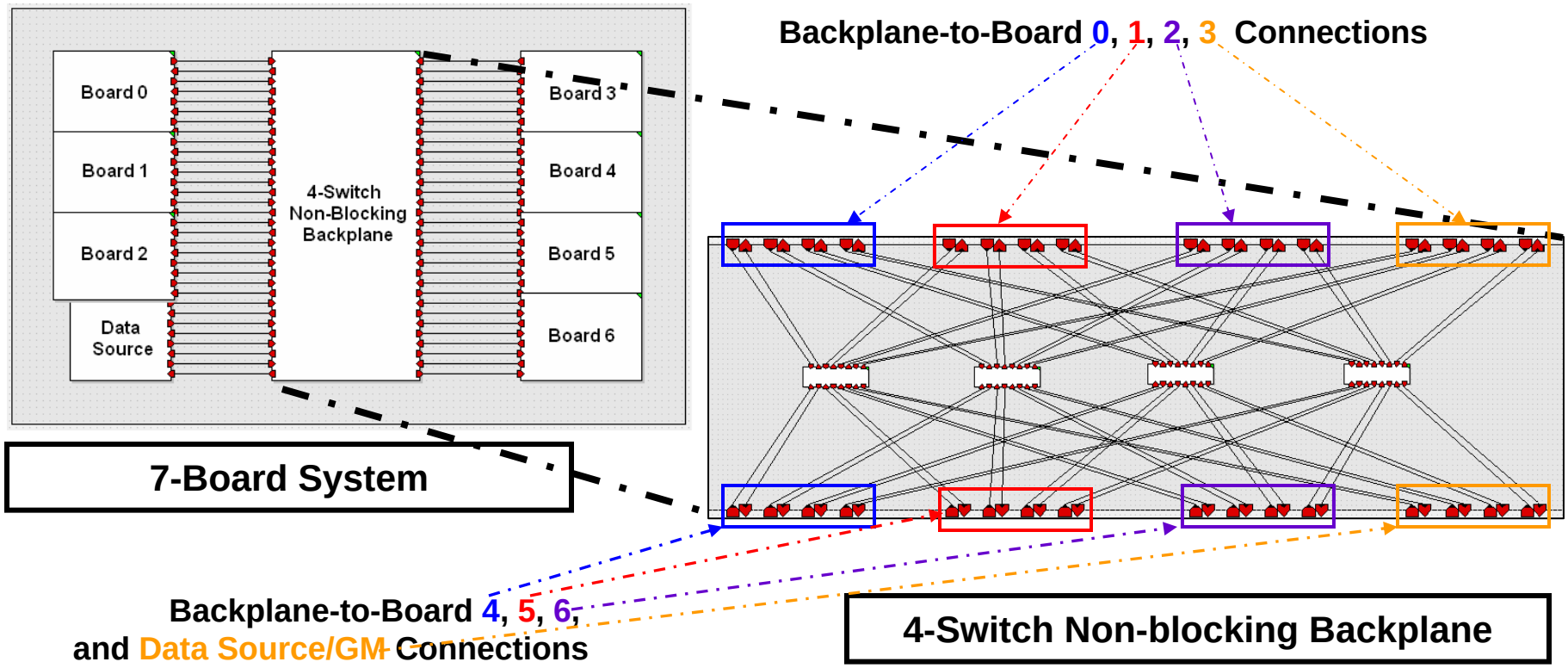


Simulation System Design Constraints

- 16-bit parallel 250MHz DDR RapidIO links (1 GB/s)
- Systems composed of processor boards interconnected by RIO backplane
  - 4 processors per board, 8 Floating-Point Units (FPUs) per processor
- Baseline SAR algorithm parameters:
  - Chunk-based algorithm performed out of global memory
  - 16k ranges, 16k pulses, 16s CPI (~2GB)
  - Requires less processing power and network throughput, more memory (global memory required)
- Baseline GMTI algorithm parameters:
  - "Straightforward" partitioning used for lowest latency
    - Divides each data cube up across all processing elements
  - Data cube: 64k ranges, 256 pulses, 6 beams, 256ms CPI (~750 MB)
  - Requires more processing power and network throughput, less memory (global memory not required)

Backplane and System Models

- System architecture mainly dictated by computational and network requirements of GMTI and its small CPI (256 ms)
  - Requires ~3GB/s from source to sink to meet real-time deadlines

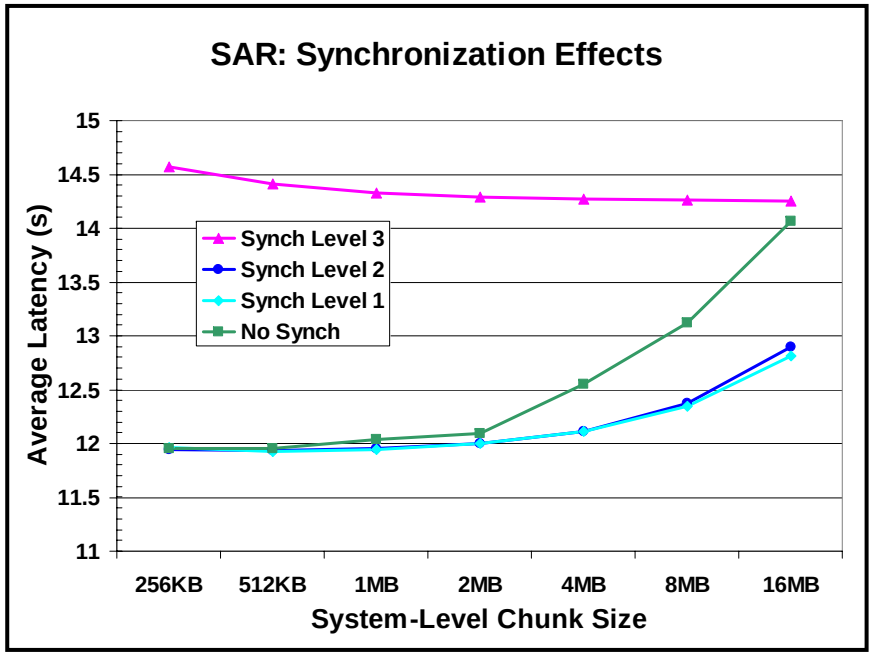


Experiments: Overview

- SAR experiments use four active processor boards
  - Remaining boards available for redundancy purposes, etc.
  - Chunk size varied for each system/algorithm configuration
- Baseline GMTI partitioning uses seven active processor boards
  - Explicit inter-processor communication for data redistribution (rather than global memory)
- Average CPI completion latency metric of choice to evaluate each algorithm/configuration
  - SAR completion time must be less than CPI to allow algorithm to be performed in real-time
  - Double-buffering allows “straightforward” GMTI completion deadline to be relaxed to 2x CPI

SAR: RIO Logical IO Optimization (1)

- No Synch: “free for all” access to 4 GM ports by 16 processors
- Level 1: access to each global memory (GM) port controlled by read token
- Level 2: access to GM controlled by write token and read token
- Level 3: access to GM controlled by single read + write token



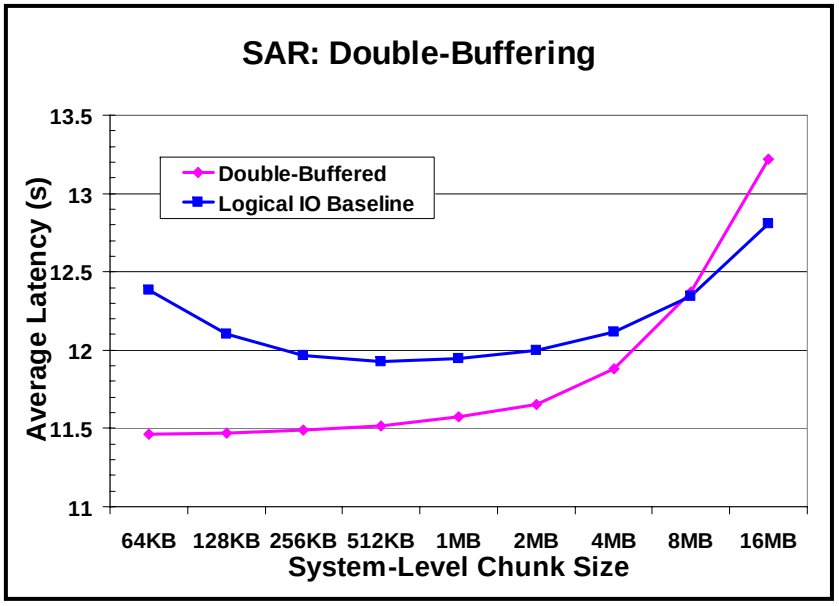
- *Chunk size per-processor = system-level chunk size/number of processors in system*
  - 4 boards × 4 processors per board = 16 processors in system

SAR: RIO Logical IO Optimization (2)

- As a general rule, contention in network increases as chunk sizes increase, causing slower CPI completion times
  - Model does not account for processing inefficiencies that may result from using chunks that are “too small”
  - “Medium-sized” chunks likely provide good compromise
- No synch case relies heavily on RIO flow control which bogs down network as many processors contend for access to global memory
- Synch level 1 (read token) most simple and effective method of synchronizing access to global memory
  - Define as baseline for Logical IO SAR
- Synch level 3 adds too much synchronization, completely removing contention but serializing all memory accesses in the process
  - Trend of decreasing performance with increasing chunk size is reversed in this case

SAR: Double-Buffering

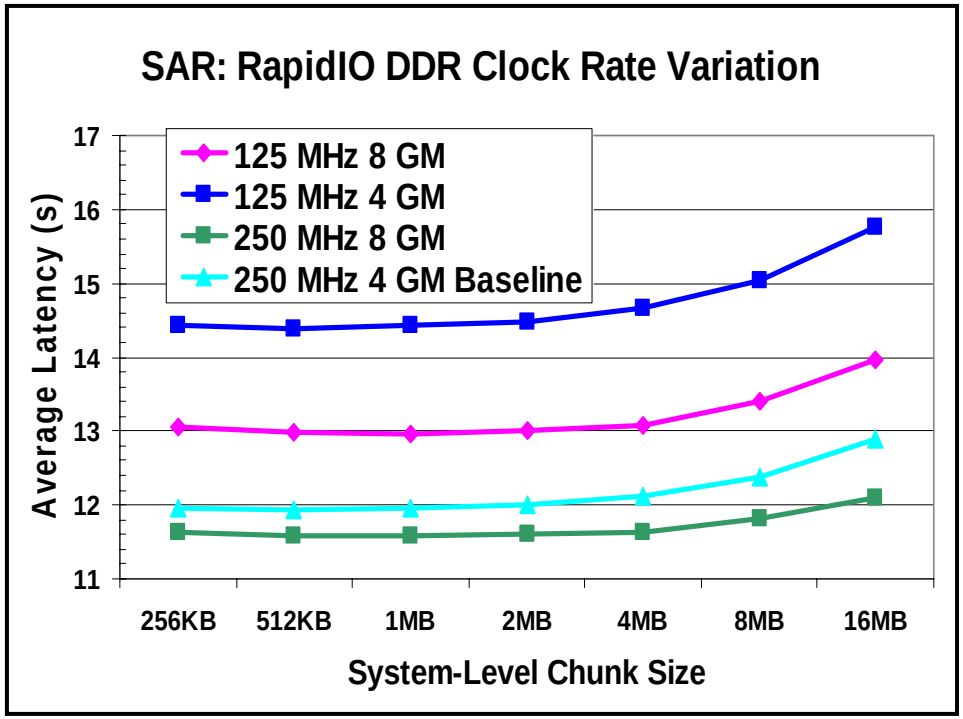
- Double-buffered version of SAR allows processors to process “current” chunk while reading “next” chunk
  - Requires 2x-3x more on-board memory for buffering each chunk
  - If system also going to perform GMTI in a “straightforward” fashion (not out of GM), processors will already have significantly more than enough memory for SAR double-buffering



- Significantly increases performance for small chunk sizes, but increases contention at larger sizes
  - Synchronized version of 2x-buffered alg. removes benefits

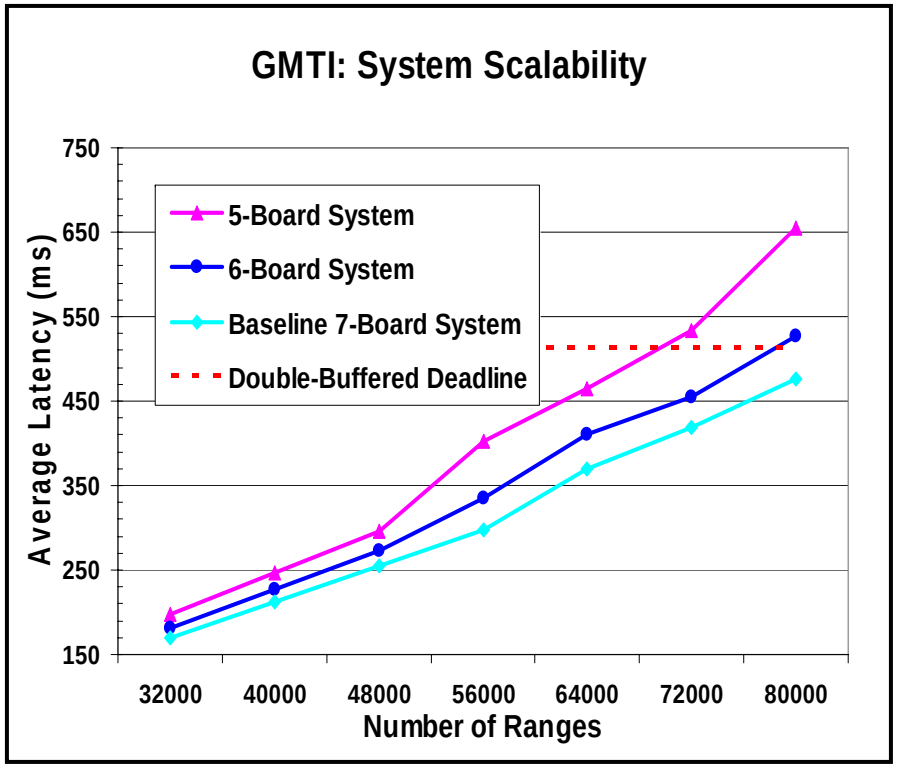
SAR: RIO Clock Rate

- **Compares 125 MHz RIO system with 250 MHz baseline**
  - **Examine both systems with 4 GM ports and with 8 GM ports**
    - **Uses 4 processor boards and 2 GM boards ⇒ 2 backplane slots free**
- **125 MHz systems significantly slower but still well inside 16 sec deadline**
- **Doubled number of GM ports significantly increases system performance, especially for 125 MHz system**
  - **Additional 4 GM ports help to provide processors with lots of data for computation**



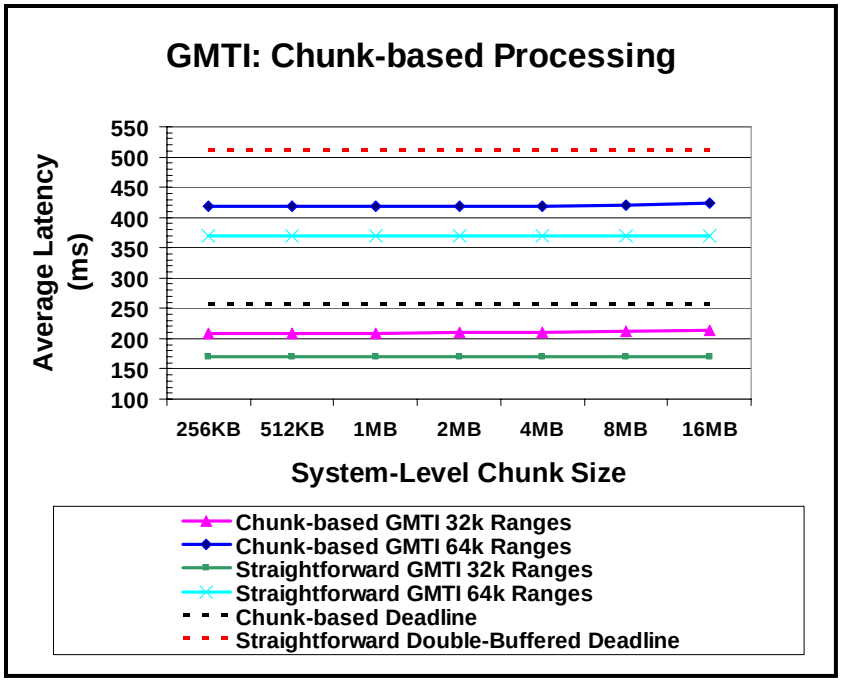
GMTI: Scalability

- Purpose of experiment to stretch data cube sizes beyond baseline and explore systems with 5, 6, and 7 active processor boards
- 7-board system scales almost up to 80k ranges with full double-buffering
  - "Double-buffering" implies storage of "current" cube while receiving "next" cube
- All systems able to handle 64k ranges baseline cube size
  - 6-board system fails on 80k ranges cube, 5-board system fails on 72k and 80k ranges cubes



GMTI: Global Memory-based GMTI

- **Experiment examines GMTI performed out of global memory in "chunks" similar to SAR**
  - **Performance much worse than baseline "straightforward" GMTI due to redundant transfer of data to/from global memory for each chunk**
  - **Problems compounded by inability to double-buffer entire data cube when using chunks (creates strict 256ms deadline)**
- **Advantage is that individual processing elements need much less local memory**
  - **~1-2 chunks vs. entire 1/*N* of data cube (*N* = # of processors)**



Conclusions (1)

- Developed and validated suite of RapidIO models
  - Minimal error experienced in simulation vs. testbed results
  - Sources of small error being investigated with help from Xilinx due to lack of visibility with internals of RapidIO core
  - Future work will integrate RIO switches into testbed
- Used models to study performance of variations of GMTI and SAR algorithms on RapidIO-based system
  - Results emphasize importance of carefully scheduling communication rather than letting RapidIO network be solely responsible for managing contention
  - Double-buffering of chunks provides mechanism for decreasing SAR CPI completion latencies (or chunk-based GMTI latencies)
  - Double-buffering of entire data cube for "straightforward" GMTI enables handling of larger cube sizes with fewer processing resources

Conclusions (2)

- **Several important considerations for building systems capable of both GMTI and SAR**
  - **Wide disparity in CPI completion deadlines**
  - **SAR memory requirements much higher than GMTI**
  - **GMTI processing and network requirements higher than SAR**
- **Systems capable of both GMTI and SAR must be built to "greatest common denominator"**
  - **Can sometimes be wasteful of system resources when performing one algorithm or the other**
  - **Compromise may be obtained by performing GMTI in a "chunk-based" manner similar to SAR**
    - **Evens out usage of system resources at expense of GMTI CPI completion latencies and data cube-size capabilities**

Bibliography

[1] D. Bueno, C. Conger, A. Leko, I. Troxel, and A. George, "Virtual Prototyping and Performance Analysis of RapidIO-based System Architectures for Space-Based Radar," *Proc. High-Performance Embedded Computing (HPEC) Workshop*, MIT Lincoln Lab, Lexington, MA, Sep. 28-30, 2004.

[2] C. Cho, "Performance Analysis for Synthetic Aperture Radar Target Classification," *MSEE Thesis*, Mass. Institute of Technology, Feb. 2001.

[3] P. Meisl, M. Ito, and I. Cumming, "Parallel Synthetic Aperture Radar Processing on Workstation Networks," *Proc. 10th International Parallel Processing Symposium*, Apr. 15-29, 1996, pp. 716-723.

[4] S. Plimpton, G. Mastin, and D. Ghiglia, "Synthetic Aperture Radar Image Processing on Parallel Supercomputers," *Proc. 1991 ACM/IEEE Conf. on Supercomputing*, Albuquerque, NM, 1991, pp. 446-452.

[5] D. Bueno, A. Leko, C. Conger, I. Troxel, and A. George, "Simulative Analysis of the RapidIO Embedded Interconnect Architecture for Real-Time, Network-Intensive Applications," *Proc. 29th IEEE Conf. on Local Computer Networks (LCN)* via *IEEE Workshop on High-Speed Local Networks (HSLN)*, Tampa, FL, Nov. 16-18, 2004.

[6] http://www.afa.org/magazine/aug2002/0802radar.asp

[7] G. Shippen, "RapidIO Technical Deep Dive 1: Architecture & Protocol," Motorola Smart Network Developers Forum, 2003.

Acknowledgements

- **We wish to thank Honeywell Defense and Space Electronic Systems (DSES) in Clearwater, FL for support of this research.**

- **We also extend thanks to Xilinx for their generous donation of hardware and IP cores, as well as MLDesign Technologies for their donation of simulation software.**