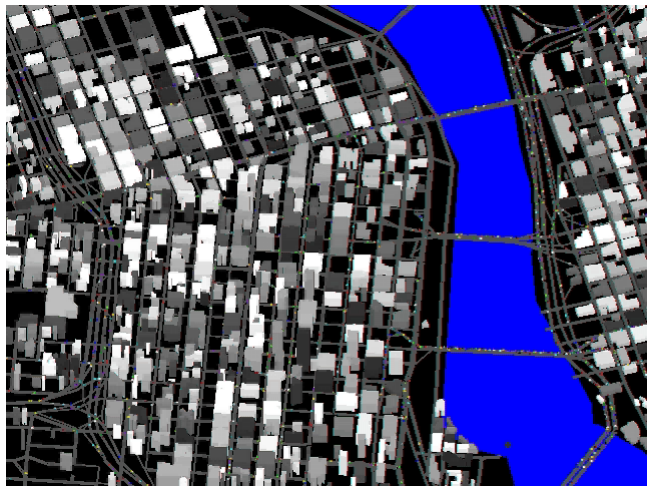
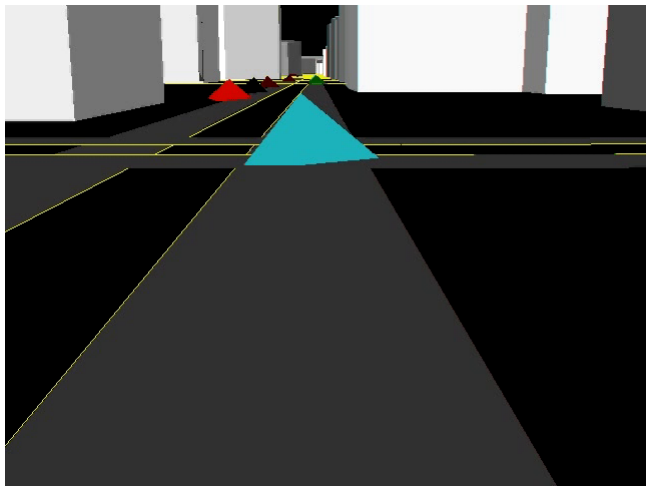
# Metropolitan Road Traffic Simulation on FPGAs





Justin L. Tripp, Henning S. Mortveit, Anders Å. Hansson, Maya Gokhale
Los Alamos National Laboratory
Los Alamos, NM 85745

# Overview

- Background
- Goals
- Using the XD1
- Results
- Conclusion and Future work

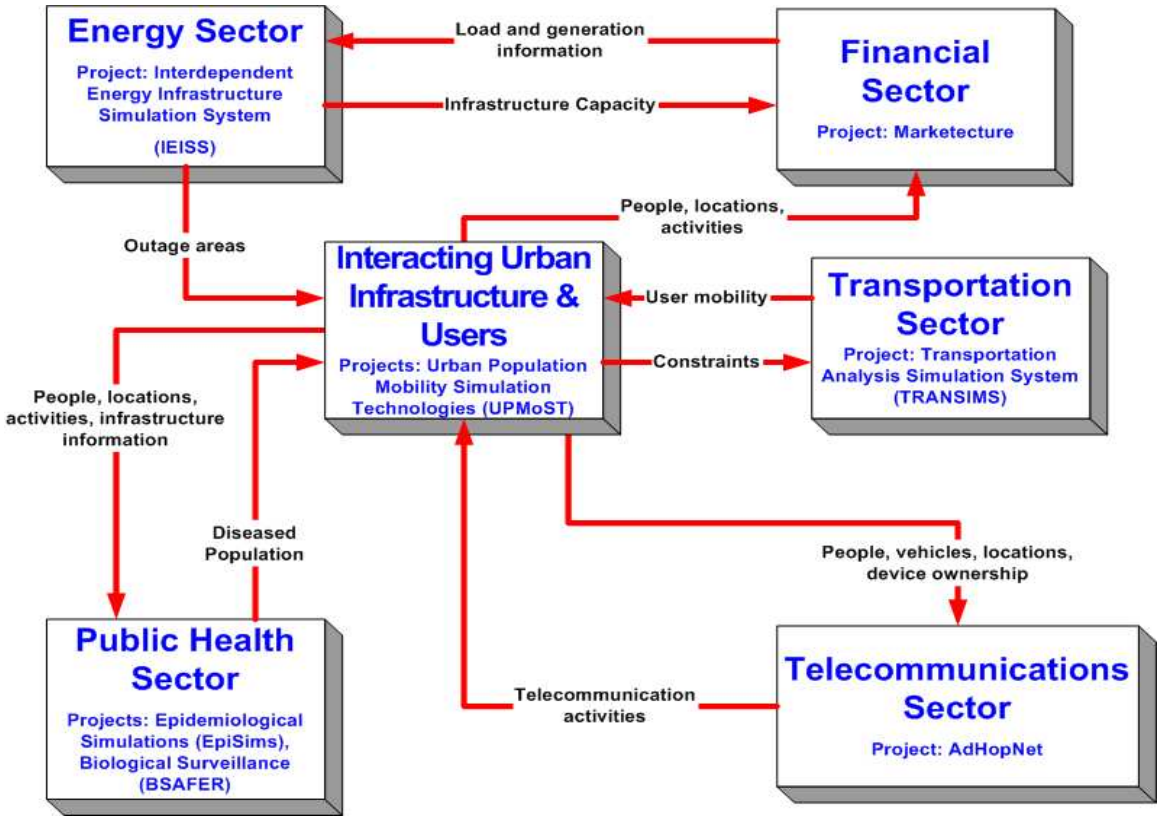
# Hardware acceleration of Large-scale Simulations

- Simulations have become a standard approach for system analysis.
- The scale of such systems often push computational boundaries.
- Simulations, such as TRANSIMS, have a large component of cellular automata(CA)-like structure with localized information and computation.
- FPGAs or a combination of FPGAs and CPUs seems well-suited to tackle such problems.

→ **TRANSIMS microsimulator is an example/prototype!**

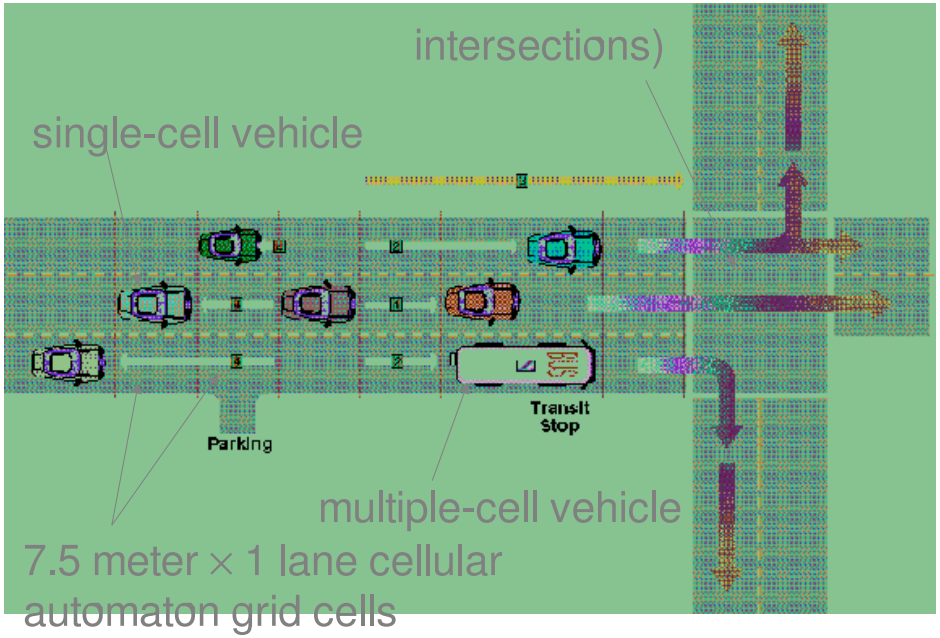
# Simulation Examples: The Unified Infrastructure Suite

- Examples: TRANSIMS, EpiSims, AdHopNet
- Systems are very large: Size: $10^7$ travelers, $10^9$ nodes, $10^9$ transceivers and $10^{12}$ pkts/hr—require HPC based simulations
- Need formal framework for design, analysis and specifications of socio-technical simulations
- Composed of smaller heterogeneous, inter-operable simulations



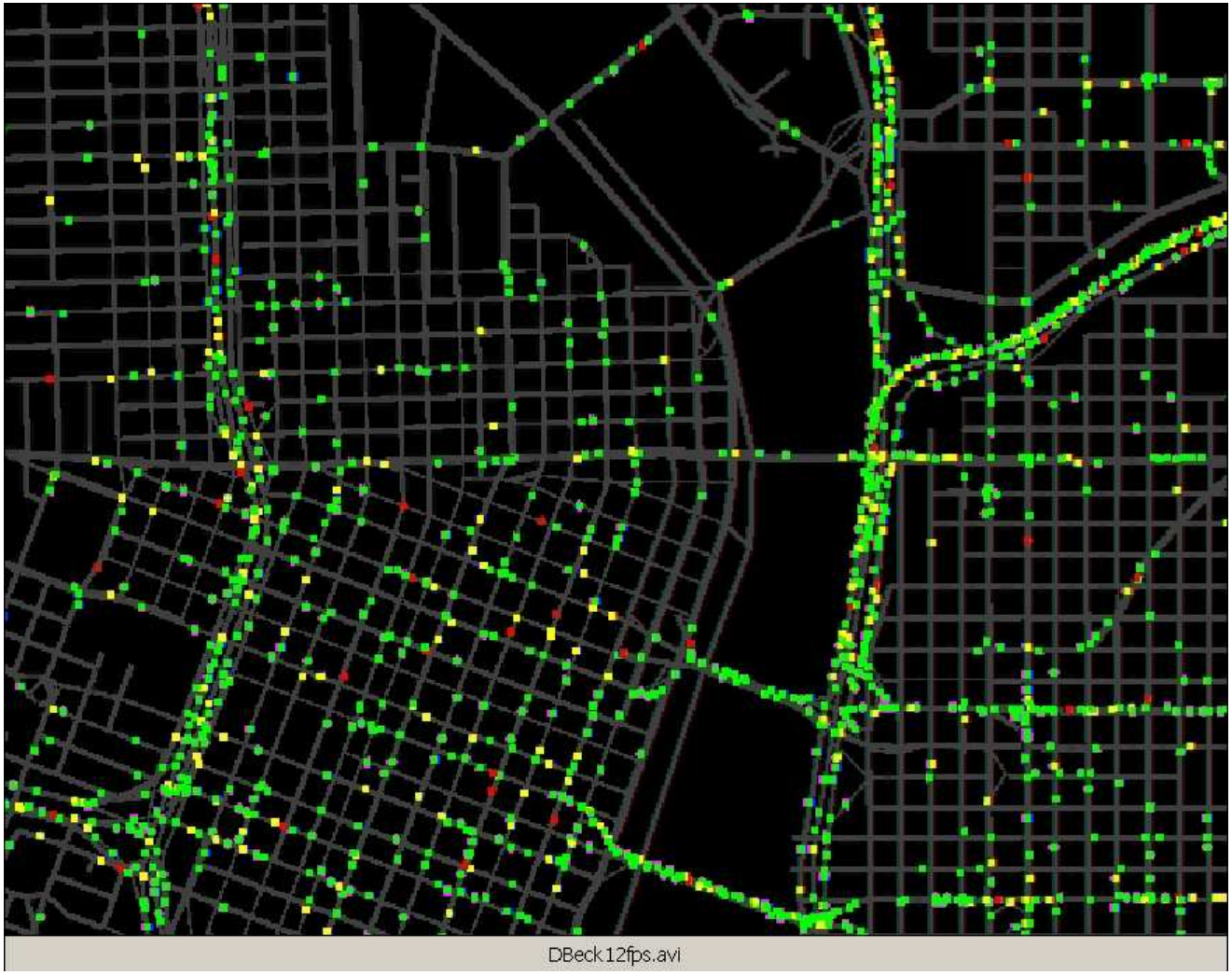
# TRANSIMS - a brief overview

- Realistic traffic on real networks
- People have plans
- Router generates global travel routes from plans
- Micro-simulator moves entities around using plans



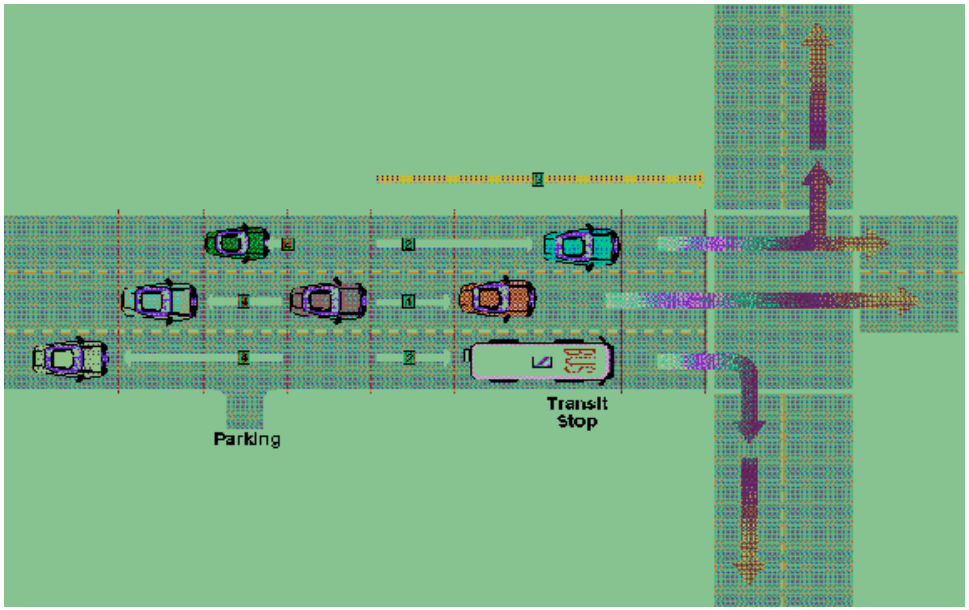

- Portland network has 6.6 million road-cells and about 1500 intersections. Chicago has about 20 million road-cells.
- The traffic micro-simulation time for Portland is about 16 hours.

# Example: Traffic in Portland



# Mathematical Structure of the TRANSIMS micro-simulator

- The micro-simulator consists of four cellular automata:
  - ⋆ Lane-change decision $\Phi_s$
  - ⋆ Lane-change execution (stochastic) $\Phi_l$
  - ⋆ Acceleration (stochastic) $\Phi_v$
  - ⋆ Position update $\Phi_p$
- Driver plans influence the dynamics around intersection and turn-lanes.
- The micro-simulator is the product system: $\Phi = (\Phi_p \circ \Phi_v) \circ (\Phi_l \circ \Phi_s)$




The terms $\Phi_v$ and $\Phi_s$ represent stochastic SDS.

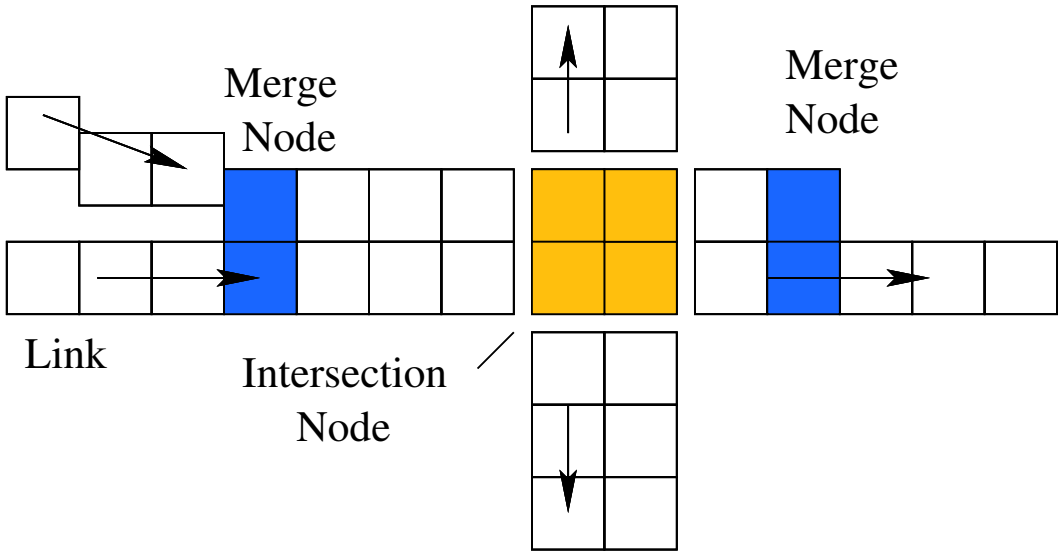
# Goals for Hardware Acceleration with FPGAs

- Accelerate traffic simulation with FPGAs.
- Explore the role of FPGAs in accelerating very large simulations.
- Increase understanding about trade-offs for large designs.
- Extend the use of FPGAs to control dominated computation.

# TRANSIMS: Traffic Simulation

Micro-simulation using cellular automaton computation on an unstructured grid.

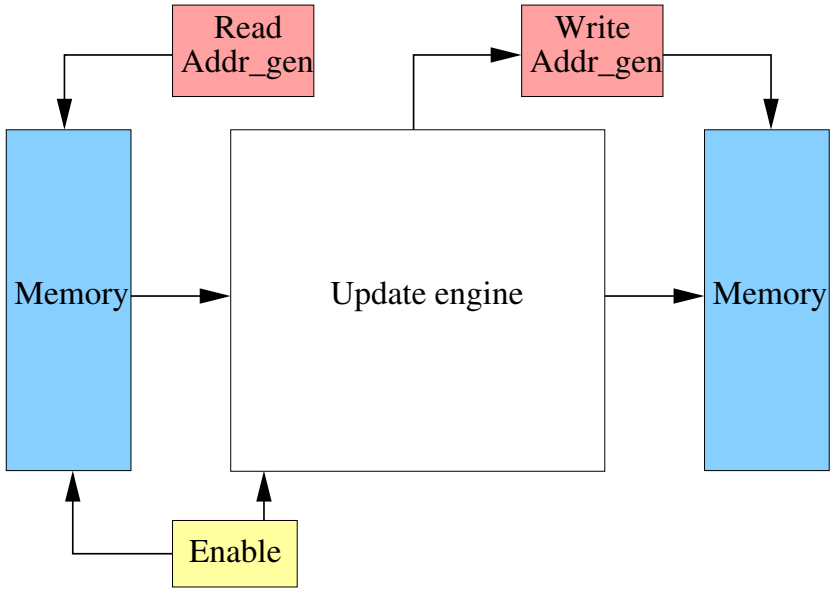
- Road network of nodes and links.
  - ⋆ Nodes - intersections and merge points
  - ⋆ Links - one or more parallel lanes of cells
  - ⋆ cells - hold one car and are 7.5m long.
- Cars
  - ⋆ Four basic rules describe cellular behavior.
  - ⋆ discrete speeds $v \in \{0, 1, 2, 3, 4, 5\}$
  - ⋆ updated once per second.




Example Structure

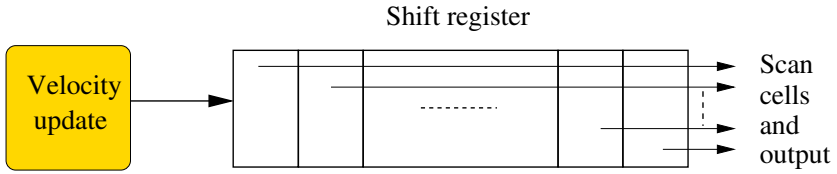
# Scalable Approach

- Use common FPGA streaming data approach.
- Streaming allows for large scale road networks by processing road cells in a continuous stream by trading area for time.
- Road statistics show that 90% of the road cells are single lane roads.
- Design is partitioned between single lane roads on the FPGA, and multi-lane and intersections on the CPU.

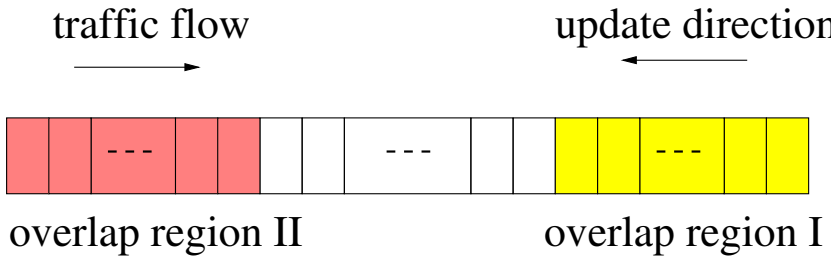


# Update Engine and Overlap Areas

- Compute Engine calculates new velocity based on cars ahead and a pseudo-random slow-down factor.



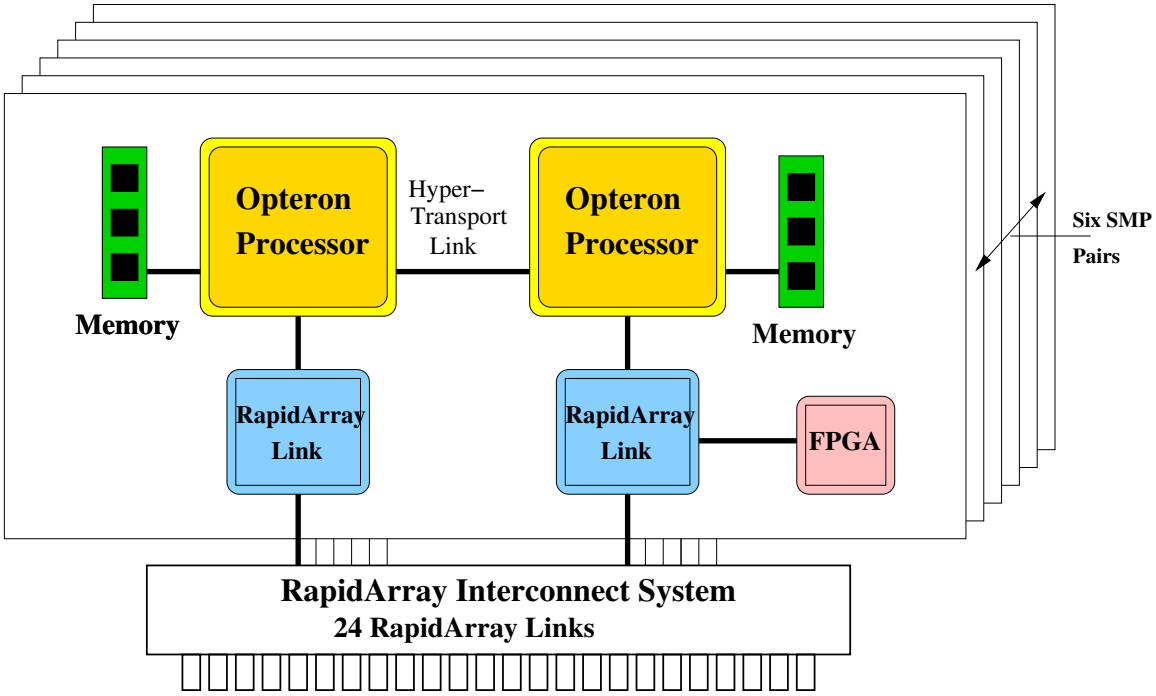

- The results are put into a shift register and pulled out when the cars velocity matches the currently "new" cell.
- Cells that are shared with the software simulation are marked as overlap region I or II.




- The velocity update modifies cars in overlap region II and passes cars in overlap region I.

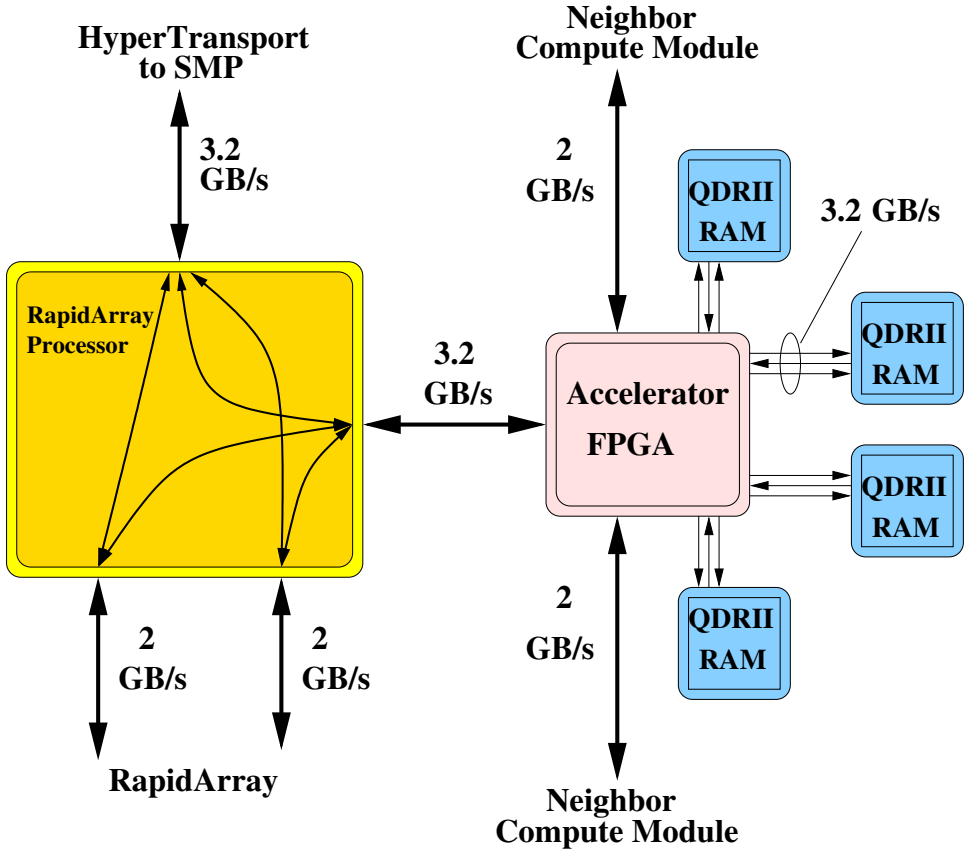
# Cray XD1 - Reconfigurable Supercomputing

- Single Chassis: 12 AMD Opterons, up to 8GB/processor
- 48 or 96 GB/s non blocking RapidArray Fabric
- 1 V2Pro30 or V2Pro50 for each SMP Pair
- 3.2 GB/s Link to RapidArray and FPGA



# Cray XD1 - FPGA Module

- V2Pro30 or V2Pro50
- 3.2 GB/s RapidArray Link
- Four 4 MB QDR SRAMs with 3.2 GB/s bandwidth
- Links to Neighbor FPGAs (not yet supported)

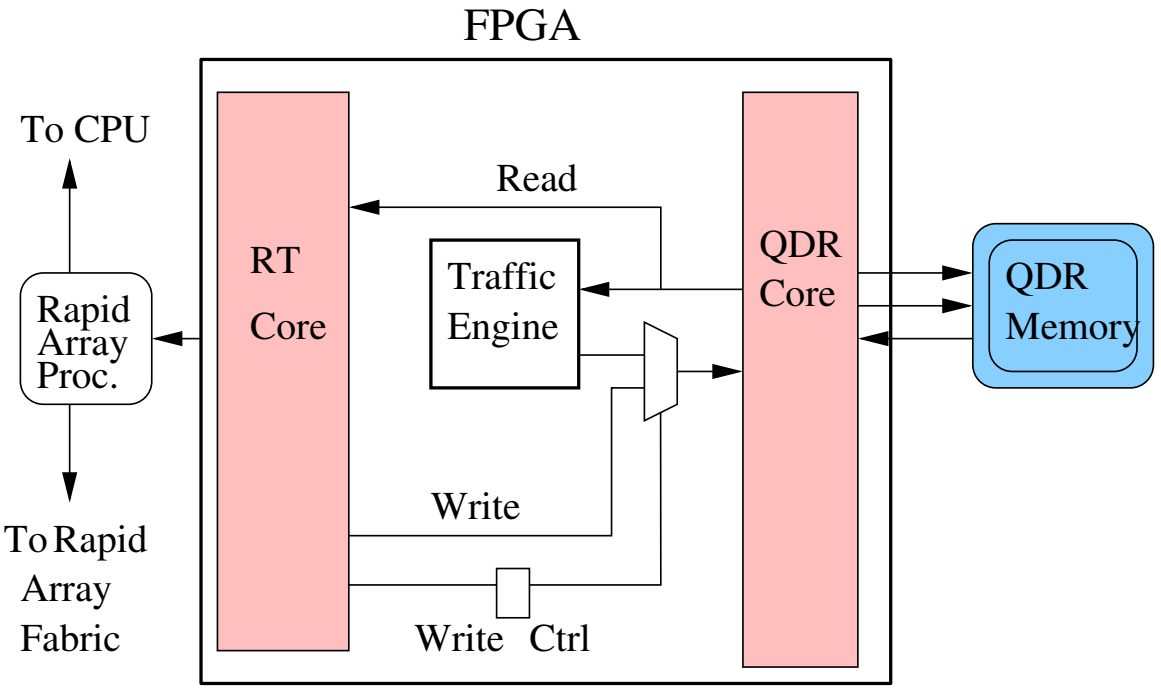


# Cray XD1 - Software Support

- Opteron SMP nodes run Cray/SuSE Linux
- MPI is provided for internode communication
- Linux device drivers and command-line tools for FPGA interaction.
- HW/SW designs are accomplished by **manual partitioning** of the HW and the SW.
- FPGA API for loading, executing, resetting, reading/writing memories, reading/writing registers, and mapping memory for FPGA's use.
- VHDL models provided for fabric, memories and provided cores (rt and qdr).

# Cray XD1 - Communication Costs

- Rapid Array Transport (RT) Core provides the interface to the RapidArray Fabric
- QDR Core provides the interface to each of the four QDR SRAM memories
- Data transfer arrives in the RT core and must be routed via the user's design to the QDR core.
- The user's design must also provide arbitration, if necessary.



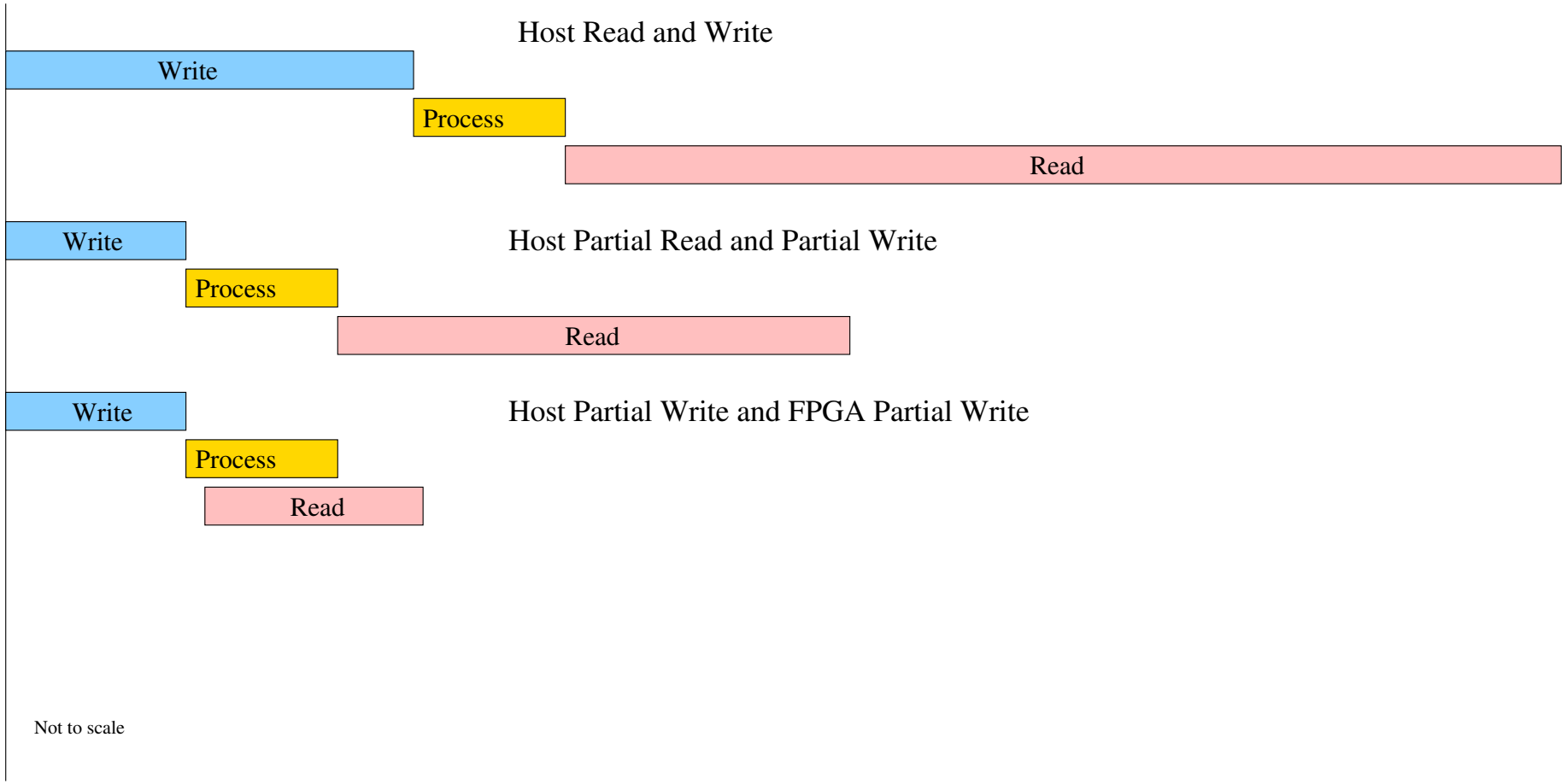
# Cray XD1 - Read and Write Bandwidth

- Cray's Manual suggests "write-only" designs.
- Measured the available bandwidth for host read and write.

| Bandwidth: Host to FPGA (MB/s) | | | |
|---|---|---|---|
| | Array | Pointer | Memcpy |
| Read | 5.94 | 5.95 | 6.01 |
| Write | 1260 | 1320 | 1320 |

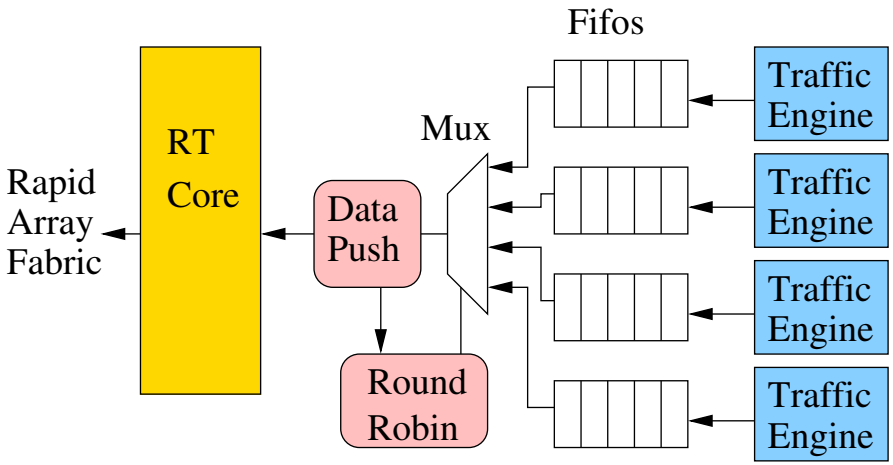
- $200\times$ bandwidth gap between reads and writes.
- Writes have all the advantages (posted, combined in kernel)
- Memory access method was not significant.

# Communication Approaches



# Data Push to Host

- Only cars in overlap areas are sent to host.
- FIFOs are required since cars in overlap can be bursty.
- Round-robin selection for each FIFO.
- Overlaps push with execution to eliminate the cost of data transfer.



# Design Results for FPGA

Raw Speed Results

| | V2p50 | 2.2GHz Opteron |
|---|---|---|
| Slices | 1857 | |
| Clock(MHz) | 180 | 2199 |
| Cells/sec | $7.2 \times 10^8$ | $5.7 \times 10^6$ |
| Speedup | 126.3 | 1.0 |

Software was timed using the time step counter register on the Opteron. The FPGA design was synthesized from VHDL using Xilinx ISE v6.2.

# Design Results for FPGA

Comparisons of streaming the data including communication to and from the host.

| | w/o Push V2p50 | Push V2p50 | 2.2GHz Opteron |
|---|---|---|---|
| Cells/sec | $2.56 \times 10^7$ | $1.96 \times 10^8$ | $5.7 \times 10^6$ |
| Speedup | 4.5 | 34.4 | 1.0 |

Without push is the speedup when the FPGA's memories are read by the host microprocessor. Using push, sends back the incremental updates using the FPGA to shared memory on the host.

# System Integration

Overlapping the processing and data read steps allows us to effectively reduce communication to nearly zero.

| | Without push | With push |
|---|---|---|
| Network state update | 8.3% | 9.7% |
| Software to hardware | 0.2% | 0.18% |
| Intersections | 14.0% | 13.8% |
| Lane change update | 13.3% | 16.8% |
| Velocity update | 21.9% | 25.5% |
| Position update | 30.0% | 33.8% |
| **Hardware to software** | **12.3%** | **0.14%** |

However, not all processing of the road straightaways could happen on the FPGA due to the size of the memories.

# Strategies to Reduce Memory usage

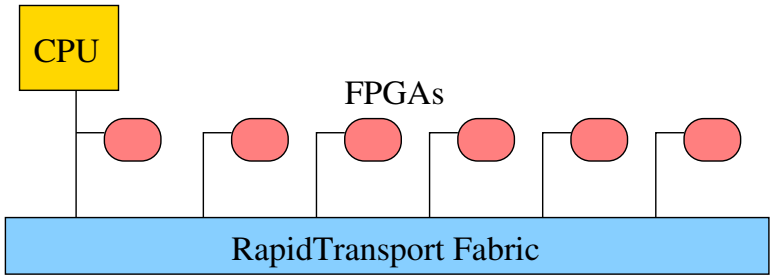
- Smaller road cell representation 64bits to 32bits.
  - ⋆ Remove car ids.
  - ⋆ Make assumptions about sequentially numbered road cells.
- Compression of "empty" road segments.
  - ⋆ Run-length encoding.
  - ⋆ traffic data semantics.

# Conclusions

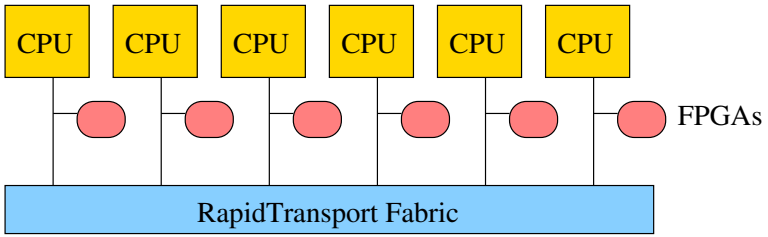
- FPGAs can scale with large simulation requirements (millions of simulation elements).
- Using the Cray XD1 with a realistic road network, we achieved a $34.4\times$ speedup.
- FPGAs can accelerate large traffic simulations using custom calculations.
- These results are for a single FPGA and single processor, and higher level cluster partitioning can be achieved similar to the PC cluster of the original TRANSIMS.

# Future Work

- Using additional FPGAs via the RapidTransport Fabric to add more computation power for a single CPU.




- Produce a *clustered* version using the XD1 Cray machine and MPI.



# Future Work

- Reduce data size by using either compression or smaller semantic representation of the data.