Acceleration of Traffic Simulation on Reconfigurable Hardware

Justin L. Tripp, Henning S. Mortveit, Maya Gokhale
Los Alamos National Labs
Los Alamos, NM 87545
Email: {jtripp, henning, maya}@lanl.gov

I. Introduction

Modern society relies on a set of complex, inter-related and inter-dependent infrastructures. The Los Alamos National Infrastructure Simulation Center (NISAC) has, over the past ten years developed a sophisticated simulation suite to simulate various infrastructure components such as road networks, communication networks, and even the spread of disease in human populations. These powerful simulation tools can help policymakers understand the inter-related systems and support decision-making for better planning, monitoring, and proper response to disruptions. Recently, infrastructure simulations have been scaled to entire cities, with people traveling in cars on road networks and talking over cellular communications networks. These coupled simulations, consisting of thousands of simple interacting automata, require the use of supercomputers and execute at a rate greater than real-time.

In this work, we investigate the possibility of accelerating a road network simulation by building a traffic simulation Cellular Automata (CA) on FPGAs. Since the simulation is inherently parallel, with independent agents making decisions based on local knowledge, it would seem fairly naturally to map to the large-scale spatial parallelism offered by FPGAs.

In the next section we describe the TRANSIMS software-based road network simulator as well as other efforts of hardware-based simulation of traffic. We then describe our approach, based on the TRANSIMS rule set, and its mapping to hardware. Performance results for one- and two-lane CAs on two different Xilinx FPGAs are presented. Finally, we end with conclusions and opportunities for future work.

II. Related Work

TRANSIMS [1] is a tool for large-scale traffic simulation and analysis of entire cities. A short description of how it operates is in order. First, a synthetic population is created based on survey data for the given city. The synthetic *population* is created in a such a way that all statistical quantities and averages considered are consistent with the survey data. Example of such quantities are age distributions, household sizes, income distributions, car ownership distributions and so on. In the next stage, realistic *plans* are made for all the individuals for a twenty-four hour day. An example plan could be 'bring kids to school – go to work – pick up kids – stop at the grocery store – drive home'. The *router* coordinates the plans of all individuals to produce realistic travel routes with realistic travel times for all of them. The router operates together with the *micro-simulator* which is the system unit responsible for moving entities around. TRANSIMS uses the actual transportation infrastructure of the city, so a route could look like 'start at A – drive to B – walk to C – take shuttle to D – etc'. Further information can be found at [1] along with descriptions of recent study of the Portland metro area. Our FPGA implementation will only be concerned with the micro-simulator and will be limited to car transportation. The details are given in the next section.

Two earlier works exist where FPGAs were used to accelerate traffic simulation. The earliest work [2] using Xilinx FPGAs, implements a limited length of single-lane road with intersections. This is used to simulate the load characteristics of a road, based on the available throughput. This approach uses a constructive approach to link together different types of road blocks. A more recent FPGA based traffic simulator by Bumble [3], uses a streaming approach based on discrete event simulation. Bumble implements a discrete event simulator in FPGAs and uses this to simulate the traffic interactions. His road models are limited to single-lanes with simple intersections. To implement a single four-way intersection using this approach requires 30-32 Altera Apex FPGAs.

III. Implementation

A. Traffic Modeling

The TRANSIMS micro-simulator can best be described as a cellular automaton computation on an unstructured grid or cell network: The city road network is split into nodes and links. Nodes correspond to places where there is a change in the road network. (e.g., intersections and lane merging points.) Nodes are connected by links which consist of one or more parallel lanes. Each lane is divided into road cells. The TRANSIMS road cell is 7.5 meters long. One cell can hold at most one car. A car travels with velocity $v \in \{0, 1, 2, 3, 4, 5\}$ cells per iteration step. The positions of the cars are updated once every second using a synchronous update. The maximal speed of a car is cell dependent, but it is at most 5.

The basic driving rules for multi-lane traffic consists of four steps. In each step we consider a single cell i in a given lane and link. Note that our model allows passing on the left and the right. To avoid cars merging into the same lane cars may only change lane to left on odd time steps and only change lane to the right on even time steps. We let $\Delta(i)$ and $\delta(i)$ denote the cell gap in front of cell i and behind cell i, respectively.

1) *Lane Change Decision:* Odd time t: If cell i has a car and a left lane change is *desirable* (car can go faster in target lane) and *permissible* (there is space for a safe lane change) flag the car/cell for a left lane change. Similar for even time t.
2) *Lane Change:* Odd time t: If the car in cell i is flagged for a left lane change clear cell i. Otherwise, if the right neighbor of cell i is flagged for a left lane change then move the car from the neighbor cell to cell i with probability p_α. Analogously for even times t.
3) *Velocity Update:* Each cell i that has a car, updates the velocity using the two-step sequence:
   - $v := \min(v + 1, v_{\max}(i), \Delta(i))$ (acceleration)
   - If $[\mathrm{UniformRandom}() < p_{\mathrm{break}}]$ and $v > 0]$ then $v := v - 1$ (stochastic deceleration).
4) *Position Update:* If there is a car in cell i with velocity $v = 0$, do nothing — if $v > 0$ then clear cell i. Otherwise, if there is a car $\delta(i) + 1$ cells behind cell i and the velocity of this car is $\delta(i) + 1$ then move this car to cell i. The velocity update pass (3) guarantees that there will be no collisions.

All cells in a road network are updated simultaneously. The steps 1 – 4 are performed for each road cell in the sequence they appear. Step i is thus a cellular automaton Φ_i in the classical sense. The whole update pass is a product CA

$$\Phi_4 \circ \Phi_3 \circ \Phi_2 \circ \Phi_1,$$

where product is function composition.

B. FPGA Implementation

The current implementation of FPGA acceleration for TRANSIMS uses a constructive approach. Several kinds of road elements: lane elements, stop signs, intersections, exits, entrances are combined together as directed by the road topology. Figure 1 outlines the structure of a basic road cell.

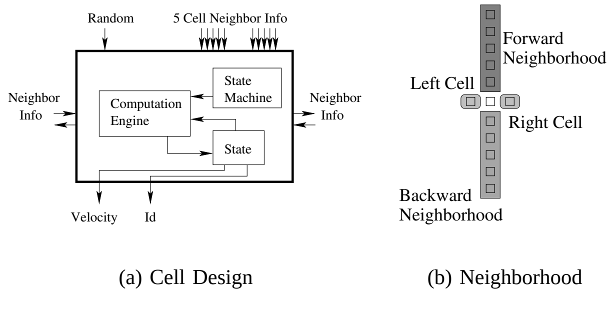

Fig. 1. Road Cell Design and Cell Neighborhoods

The rules outlined in Section III-A are used to calculate the state of every road segment. Each rule requires a single cycle to calculate, except for the the third rule, Velocity Update. Velocity Update takes three clock cycles to accomplish its two step sequence. Shown in Figure 1(a), the computation engine, driven by the state machine, calculates a new velocity and car id every six cycles.

The computation engine is an implementation of all four the rules described in Section III-A. The engine uses a neighborhood of 5 cells back to determine what the new values of the road segment should be. A smaller neighborhood of right and left cells and the 5 cells in front are used during when calculating lane changes. An example of the neighborhoods is shown in Figure 1(b). This neighborhood can be reduced by road topology (e.g., a reduction of lanes, dead-end, etc.).

The FPGA implementation is written in VHDL and synthesis was performed by Synplify. The resultant EDIF description was passed into the Xilinx tools to produce the results reported in the next section.

IV. RESULTS

The results for circular multi- and single-lane circular traffic are described in Table I. The hardware implementation of single-lane traffic has only four states, since single-lanes do not require the extra hardware for lane changes. The two-lane implementation which includes the hardware to perform lane changes, is 63% larger in area.

	One-lane		Two-lane	
	XC2V6000	XC2VP100	XC2V6000	XC2VP100
Cells	650	650	400	400
LUTs/Cell	104	97	169	175
Clock(MHz)	48.68	64.17	35.53	54.43
Slices	33790	31576	33790	34999
(% of Slices)	(99%)	(71%)	(99%)	(79%)

TABLE I
FPGA DESIGN RESULTS

Table II compares the results for the two-lane traffic implementation achieved by the Xilinx XC2V6000 and the XC2VP100 to a software implementation running on two different Xeon processors. The speed up reported is relative to the 1.7GHz Xeon. The XC2V6000 simulates the road cells at a rate $303.1\times$ faster than the Xeon. This speed up comes primarily from the fact that the FPGA implementation is executing all cells concurrently, and the software implementation, which may have instruction level parallelism, calculates each cell individually.

	XC2V6000	XC2VP100	1.7GHz Xeon	3.0GHz Xeon
Cells	400	400	600	600
Cells/sec	2.37×10^9	3.64×10^9	7.82×10^6	1.35×10^7
Speed Up	303.1	651.1	1.0	1.73

TABLE II
RESULTS COMPARISON FOR TWO LANE IMPLEMENTATIONS

V. CONCLUSION

Although the results are currently limited to circular loops, the current speed up of $303\times$ demonstrates the potential acceleration when combining FPGAs with TRANSIMS. Two problems still need to be addressed before FPGAs can be used are part of a “real-data” simulation: scaling and integration.

Using the constructive approach, an FPGA can only handle a relatively small number of road cells. The following data from the Portland [4] TRANSIMS study, explain the requirements of a large traffic micro-simulation. The Portland road network has 124904 links, with the average link length close to 250 meters. Assuming 1.5 lanes/link on the average and using the TRANSIMS standard 7.5m cell length, there are roughly 6.25 million road cells. Studies of larger cities seek to expand this amount by $10\times$.

Scaling issues could be overcome by modifying the designs to support a streaming architecture as opposed to a constructive architecture. This would support virtual road cells and may be able to more easily handle the scaling demands. We are currently pursuing this approach.

The other problem to overcome is the integration of the traffic simulation accelerator into the larger TRANSIMS framework. In its current form, the FPGA implementation handles multi-lane traffic with some roadway elements such as stop signs. With this, there are two alternatives for TRANSIMS integration.

In the first approach the FPGA implementation would take over the computation on links only. This could provide a large benefit on freeways, but would not work as well with shorter links. There is also a large amount of per-update communication with the TRANSIMS module that handles nodes and link end-points. Overcoming these communication requirements could be difficult.

The second approach will require an extension of the existing FPGA implementation such that it can handle travel plans or routes. Within TRANSIMS, travel routes are used to determine where different entities will travel within the city. The travel routes could be programmed into the FPGAs and the FPGA would calculate the results of the entire simulation. We are currently pursuing this approach, by first determining the cost of incorporating the plans into the hardware design.

Acceleration of TRANSIMS opens the door to a whole range of infrastructure simulation where FPGAs may be able to lend computational speed up. TRANSIMS is part of a much larger city infrastructure simulation system, in which other elements have similar computations and structures. Accelerating TRANSIMS will help demonstrate how FPGAs can aid in infrastructure simulation. This implementation is also valuable in that it provides a starting point for identifying simulation problem characteristics and modeling techniques for the efficient use of FPGAs.

References

[1] L. L. Smith, “Transims home page,” 2002. [Online]. Available: http://transims.tsasa.lanl.gov/

[2] G. Russell, P. Shaw, and J. McInnes, “Rapid simulation of urban traffic using fpgas,” 1994. [Online]. Available: http://citeseer.ist.psu.edu/russell94rapid.html

[3] M. D. Bumble, “A parallel architecture for non-deterministic discrete event simulation,” Ph.D. dissertation, Pennsylvania State University, 2001.

[4] C. L. Barrett, R. J. Beckman, K. P. Berkbigler, K. R. Bisset, B. W. Bush, K. Campbell, S. Eubank, K. M. Henson, J. M. Hurford, D. A. Kubicek, M. V. Marathe, P. R. Romero, J. P. Smith, L. L. Smith, P. E. Stretz, G. L. Thayer, E. Van Eeckhout, and M. D. Williams, “TRansportation ANalysis SIMulation system (TRANSIMS) portland study reports,” December 2002. [Online]. Available: http://maynard.tsasa.lanl.gov/PortlandStudyReports.html