Efficient Traffic Simulation Using the GCA Model

Christian Schäck, Wolfgang Heenes, Rolf Hoffmann
Outline

- Introduction
- Application, model and architecture hierarchy
  - Applications
  - Agent System
  - Global Cellular Automata (GCA) Model
  - Multiprocessor Architecture
- Random numbers in the GCA Model
- Traffic Simulation
  - Nagel-Schreckenberg Algorithm
  - CA Model with Searching
  - GCA Model with Linked Agents
- Results
- Summary & Outlook
Introduction

- dedicated system for Agent Simulation

- agents
  - moving entities on a n-dimensional field
  - homogeneous/heterogeneous behavior

- traffic simulation
  - Nagel-Schreckenberg algorithm
    - cars are represented as agents

- accelerate simulation using GCA model instead of CA model
System Overview
Application, Model and Architecture Hierarchy

- **Application**: Agent applications (traffic simulation, biology systems)
- **Model**: Agent system, GCA model
- **Architecture**: GCA architecture, MPA, DPA, ...

**Object Representation**: Application objects (car, bus, truck), agent 1, agent 2, agent 3, current cell state, binary.
Applications

- general applications
  - graph algorithms
  - hypercube algorithms
  - numerical algorithms
  - graphics
  - ...

- agent based applications
  - multi-agent simulation
  - traffic simulation (Nagel-Schreckenberg)
  - biology systems (honey bees)
  - ...

APDCM 2010 | Atlanta | Computer Architecture Group, TU Darmstadt | Christian Schäck
Agent System

- generalization layer
- adds additional functions for agent simulations
- allows adjustment of layers underneath
- common interface regarding agent simulation
- accelerate simulation by defining hardware functions
- “library” for common used functions
  - defines function interfaces

- Agent System layer currently under investigation
Global Cellular Automata Model

- massively parallel computational model
- extension of the classical cellular automata
- dynamic, global neighborhood (read only)
Multiprocessor Architecture - General Structure

- P NIOSII/f softcore processors + Custom Instructions
- internal memory (program, data)
- Interconnection network
Random Numbers

- necessary for individual agent behavior
- can not be generated within the processor
  - synchronization issues (delete + copy)

- random numbers generated during write
  - allows all PU’s to access same random number
  - different random number for each cell and generation, but same random number while execution a generation
Traffic Simulation
Nagel-Schreckenberg Algorithm

- theoretical traffic simulation model
- street consists of cells

- cell update rules:
  1. if $V_{\text{max}}$ not reached then $V=V+1$ (accelerate)
  2. if gap $< V$ then $V=gap$ (collision free)
  3. $V=V-1$ with probability $p$ (dally)
  4. move all vehicles
CA Model with Searching
agent cell checks

When the agent cell checks,

$t=0$

$t=1$

stopping checking if another agent is found or $V$ is reached
CA Model with Searching
empty cell checks

X stops checking if an agent is found or V_max is reached

X checks two front cells 1. gap size 2. speed reduction
Calculating speed for generation \( g+1 \) for cell \( i \):

\[
\begin{cases}
- & \text{not considering} \ V_{\text{max}} \\
\max[\min[z(L(i)) + 1, L(L(i)) + z(L(L(i))) - i - 1] - R, 0], & E \rightarrow E \\
\max[\min[1, L(i) + z(L(i)) - i - 1] - R, 0], & E \rightarrow A \\
- & A \rightarrow E \\
- & A \rightarrow A
\end{cases}
\]
## Results

2048 cells, $p=0.5$, $V_{\text{max}}=5$

$$gain = \frac{\text{execution time CA} - \text{Alg orithm}}{\text{execution time GCA} - \text{Alg orithm}}$$

<table>
<thead>
<tr>
<th>$P$</th>
<th>gain (10%) 204 agents</th>
<th>gain (50%) 1024 agents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.00</td>
<td>1.11</td>
</tr>
<tr>
<td>2</td>
<td>2.00</td>
<td>1.13</td>
</tr>
<tr>
<td>4</td>
<td>2.01</td>
<td>1.18</td>
</tr>
<tr>
<td>8</td>
<td>1.97</td>
<td>1.29</td>
</tr>
<tr>
<td>16</td>
<td>2.00</td>
<td>1.25</td>
</tr>
</tbody>
</table>

### Scalability (50% agents):

<table>
<thead>
<tr>
<th>$P$</th>
<th>execution time</th>
<th>speed-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>153.7 ms</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>84.2 ms</td>
<td>1.83</td>
</tr>
<tr>
<td>4</td>
<td>49.2 ms</td>
<td>3.12</td>
</tr>
<tr>
<td>8</td>
<td>28.8 ms</td>
<td>5.33</td>
</tr>
<tr>
<td>16</td>
<td>17.8 ms</td>
<td>8.66</td>
</tr>
</tbody>
</table>
Additional Results
2048 cells, p=0.5

- GCA-Algorithm performs very well for high speeds and low densities

- Low density of 1% agents (20)

<table>
<thead>
<tr>
<th>$V_{\text{max}}$</th>
<th>gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>2.3</td>
</tr>
<tr>
<td>10</td>
<td>4.1</td>
</tr>
<tr>
<td>20</td>
<td>7.9</td>
</tr>
<tr>
<td>40</td>
<td>14.9</td>
</tr>
<tr>
<td>80</td>
<td>29.3</td>
</tr>
</tbody>
</table>

$\text{gain} = \frac{\text{execution time CA} - \text{Alg orithm}}{\text{execution time GCA} - \text{Alg orithm}}$
Summary

- design and FPGA implementation of a multiprocessor architecture with NIOSII processors for the GCA model
- Agent System layer extension for agent simulation
- GCA-Algorithm performs faster compared to the CA-Algorithm
  - ~2x for 10% Agents
- GCA-Algorithm performs very well for
  1. low densities
  2. high maximum speeds
Outlook

- Agent System layer
  - further definition
  - evaluating HW/SW-functions for agents

- new architectures avoiding empty cells
  - based on hash functions
  - using dedicated agent memories
Thank you very much for your attention!