On the Acceleration of Shortest Path on FPGAs

Zachary K. Baker and 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 Infrastructure Simulations

- Large graph problems must deal with high memory latency (but a high degree of concurrency)
- The scale of such systems often push computational boundaries.
- FPGAs or a combination of FPGAs and CPUs seems well-suited to tackle such problems.
TRANSIMS - a brief overview

- Census data provides (randomized) behavior of real people
  - Realistic traffic on real networks
  - Each person has several travel plans per day
  - Router generates global travel routes from plans
TranSIMs: Route Planning

- Micro-simulation uses daily habits of entire urban area
- Routes are planned for home-work-shopping/activities travel
- Opportunity for parallel execution of route planning
  - 100k-1,000k routes need to be computed
  - Minimal requirement for ordered completion
  - Lat/long of all nodes is available: allows for more efficient directed route-finding
Larger Goals for this Research

- Explore the use of Field Programmable Gate Arrays (FPGA) in accelerating large simulations
- Increase understanding about trade-offs for very large designs
- Use FPGAs for non-traditional FPGA computation
  - Sparse graph structures
  - General sparse data structures
- Explore the use of FPGAs for multi-threaded latency-tolerant computing
  - Especially memory latency-dominated computation (large graphs, low locality)
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)
How to efficiently route?

- Greedy Routing with Backtracking
A* Algorithm

- Point-to-point Shortest Path
  - *A* Algorithm: uses distance to destination to decide which possible paths to pursue. Execution time depends on:
    - *Priority Queue Implementation*: the computational complexity is determined by the implementation of the priority queue
    - *Road Network Graph Bandwidth*: bandwidth to road network graph needs to be sufficient
Architecture Structure

System architecture is optimized for memory bandwidth of Cray XD-1 QDR SRAM banks.

- 3.2 GB to each of 4 SRAM banks
- One 64 bit word per clock period in either direction
- 16 MB total SRAM

For graphs larger than 16 MB (Los Angeles)

- Virtual memory managed in software
- Graph data is paged in to SRAM as required
- Bandwidth - Streaming is bandwidth limited from the host (PCI)
Maximizing performance

- Objective: Maximize performance metric: 
  \[
  \frac{\text{#pathscompleted}}{\text{time}} = \frac{\text{freq} \times \#\text{units}}{\#\text{cycles per path}}
  \]

- The objective is for the system to maximize the usage of the available random access bandwidth:
  \[
  \frac{\text{#pathscompleted}}{\text{time}} = \text{bandwidth (lookups/sec)} \times \#\text{units} \times \text{lookups/path}
  \]

- Kernel design must be space-efficient: If number of units is too small, bandwidth will not be utilized effectively
Architecture for parallel A* system

Edge data output

External Sparse Storage

Block RAM

V

A*

Edge requests

Round-robin service token
Priority Queue Innovations

- Software implementations use Fibonacci Heap to provide $O(\log(n))$ average performance
  - Amortized performance achieved through complicated data structure
  - Heap can grow without bound during execution
  - Data structure not optimal for hardware implementation
Hardware implementation

- Hardware implementation must take advantage of FPGA strengths
  - Queues are efficient
    - Only single memory port necessary: Time-separated pushes and pops
    - If many shortest paths are computed in parallel, the \textit{extract-min} operation does not have to be fast
  - Queue can transparently bubble the minimum distance node to the head
    - Bubble sort requires few hardware resources, has same practical performance as Fibonacci Heap given constrained memory bandwidth
Priority Queue Innovations

• Small, area-efficient bubble sort core provides sorted entries (heap) at low cost
  ◆ Size of buffer can be constrained without affecting results
    ◆ After testing, buffer of size 32 seems to be most effective for Los Angeles graph
## Priority Queue Innovations

### Table

<table>
<thead>
<tr>
<th>ID</th>
<th>Lat</th>
<th>Long</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>490085.4</td>
<td>3653492.5</td>
</tr>
<tr>
<td>4</td>
<td>490083.4</td>
<td>3653498.5</td>
</tr>
<tr>
<td>7</td>
<td>490091.3</td>
<td>3653923.1</td>
</tr>
<tr>
<td>2</td>
<td>490172.7</td>
<td>3653256.9</td>
</tr>
<tr>
<td>5</td>
<td>490123.3</td>
<td>3653923.5</td>
</tr>
<tr>
<td>3</td>
<td>490179.2</td>
<td>3654211.2</td>
</tr>
</tbody>
</table>

Minimum Euclidean distance bubbles to top. Queue wraps around memory.
Buffer Behavior

Effects of Reduced Buffer Size

Legend
- Completed Routes
- Non-optimal Complete

Size of Buffer

0 20 40 60 80 100 120 140 160

1024 128 64 32 16 8

Unclassified: LA-UR-06-7320
Results

- Xilinx 2vp50 -7
- 130 MHz

<table>
<thead>
<tr>
<th>Num Units</th>
<th>Area (slices)</th>
<th>Area (%)</th>
<th>Mult (of 232)</th>
<th>BRAM (of 232)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5277</td>
<td>22</td>
<td>6</td>
<td>20</td>
</tr>
<tr>
<td>8</td>
<td>8562</td>
<td>36</td>
<td>16</td>
<td>40</td>
</tr>
<tr>
<td>16</td>
<td>14628</td>
<td>64</td>
<td>32</td>
<td>72</td>
</tr>
<tr>
<td>24</td>
<td>18165</td>
<td>76</td>
<td>48</td>
<td>112</td>
</tr>
</tbody>
</table>

Unclassified: LA-UR-06-7320
Results

- Not to be trusted!
- Why?
  - Small QDR memories do not hold entire graphs
  - Load/retrieve of origin/destination pairs and results not considered

<table>
<thead>
<tr>
<th>platform</th>
<th>time</th>
<th>loads</th>
<th>rate/sec</th>
<th>speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>cpu</td>
<td>0.25</td>
<td>21594</td>
<td>86376</td>
<td>1</td>
</tr>
<tr>
<td>fpga</td>
<td>1</td>
<td>130000000</td>
<td>130000000</td>
<td>1505.047698</td>
</tr>
</tbody>
</table>
Conclusion

- Multiple point-to-point shortest paths are computed concurrently in hardware
  - Efficient bubble sorting technique allows for more units per device
- SRAM bandwidth is maximized by always having graph requests ready
  - Round-robin memory controller keeps memory pipeline full
    - DRAM will increase latency, and thus will require a higher number of concurrent threads
    - Number of shortest path units must be balanced to available bandwidth
Future Work

• Flow and Dynamic Weighting: Model congestion for more realistic route planning
• Multi-resolution graphs for hierarchical route planning
• Hardware development
  ∗ Virtual memory