Implementing Hirschberg’s PRAM-Algorithm for Connected Components on a Global Cellular Automaton

J. Jendrsczok, R. Hoffmann, J. Keller
Outline

- Cellular Automata
- Global Cellular Automata
- Algorithm of Hirschberg et al.
- Mapping onto the GCA
- Hardware Implementation
- Conclusion & Future Work
Cellular Automata (CA)

Classical Cellular Automata

- each cell is locally connected to its neighbors
- updates only its own state according to a local rule
- all cells update their state in parallel → massively parallel
- optimal model for applications with inherent local neighborhood
Global Cellular Automata (GCA)

**Global Cellular Automata**
- extension of the CA model
- allowing global access to neighbors
- access pattern may change from generation to generation

![Generation t vs Generation t+1 diagram](image)
GCA cell (d, p) and global cell (d*, p*)
## CA vs. GCA

<table>
<thead>
<tr>
<th></th>
<th>CA</th>
<th>GCA</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Ordering in Space</strong></td>
<td>n-dim. grid</td>
<td>ordered set (array)</td>
</tr>
<tr>
<td><strong>Neighbors</strong></td>
<td>local</td>
<td>global</td>
</tr>
<tr>
<td><strong>Links</strong></td>
<td>fixed, regular</td>
<td>variable, irregular</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– time dependent</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– data dependent</td>
</tr>
<tr>
<td><strong>Access to Neighbors</strong></td>
<td>read-only</td>
<td></td>
</tr>
<tr>
<td><strong>Local Rule</strong></td>
<td>Cell changes only it's own state</td>
<td></td>
</tr>
<tr>
<td><strong>Updating</strong></td>
<td>synchronous</td>
<td></td>
</tr>
<tr>
<td><strong>Massively-Parallel</strong></td>
<td>yes (no write conflicts)</td>
<td></td>
</tr>
</tbody>
</table>
Algorithm of Hirschberg et al.

- Calculates connected components of a graph
- Input: Adjacency matrix
- Output: Vector with the connected component to which each vertex belongs
- First described in 1965 for SIMD
- Second adapted for CREW (Concurrent Read exclusive Write) PRAM
- Now adapted for GCA
Algorithm of Hirschberg et al.

1. for all i in parallel do $C(i) \leftarrow i$

2. for all i in parallel do
   $T(i) \leftarrow \min_j \{C(j) \mid A(i,j)=1 \text{ AND } C(j) \neq C(i)\}$
   if none then $C(i)$

3. for all i in parallel do
   $T(i) \leftarrow \min_j \{T(j) \mid C(j) = i \text{ AND } T(j) \neq i\}$
   if none then $C(i)$

4. for all i in parallel do
   $C(i) \leftarrow T(i)$

5. for all i in parallel do $T(i) \leftarrow T(T(i))$

6. for all i in parallel do $C(i) \leftarrow \min\{C(T(i)), T(i)\}$

$\Rightarrow O(\log^2(n))$ with $n(n/\log^2 n)$ processors
Mapping onto the GCA

1. for all \textit{i} in parallel do \( C(i) \leftarrow i \)
2. for all $i$ in parallel do
\[ T(i) \leftarrow \min_j \{ C(j) \mid A(i,j) = 1 \text{ AND } C(j) \neq C(i) \} \text{ if none then } C(i) \]
3. for all i in parallel do
   \[ T(i) \leftarrow \min_j \{ T(j) \mid C(j) = i \text{ AND } T(j) \neq i \} \] if none then C(i)
4. for all i in parallel do
C(i) ← T(i)
Mapping onto the GCA

5. for all i in parallel do
   \[ T(i) \leftarrow T(T(i)) \]

6. for all i in parallel do
   \[ C(i) \leftarrow \min\{C(T(i)), T(i)\} \]
## Time Complexity

<table>
<thead>
<tr>
<th>Step of the algorithm</th>
<th>Generations per step</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1 + log(n) + 1 + 1</td>
</tr>
<tr>
<td>3</td>
<td>1 + log(n) + 1 + 1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>log(n)</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
</tr>
</tbody>
</table>

$=> O(log^2(n))$ with $n(n+1)$ cells
Hardware FPGA Implementation
Hardware FPGA Implementation

<table>
<thead>
<tr>
<th>problem size</th>
<th>cells</th>
<th>register</th>
<th>logic elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>20</td>
<td>88</td>
<td>593</td>
</tr>
<tr>
<td>8</td>
<td>72</td>
<td>392</td>
<td>2742</td>
</tr>
<tr>
<td>16</td>
<td>272</td>
<td>1712</td>
<td>12981</td>
</tr>
<tr>
<td>32</td>
<td>1056</td>
<td>7584</td>
<td>60900</td>
</tr>
</tbody>
</table>

Stratix II FPGA (EP2S180), clock frequency 82-71 MHz
Conclusion

- A GCA algorithm consists of data and pointer operations executed in parallel
- Hirschberg’s algorithm was mapped onto a GCA using \( n(n+1) \) cells
- Time complexity is \( O(\log^2(n)) \)
- Can be executed very fast by a parallel hardware (FPGA implementation)
Future Work

- Mapping Hirschberg’s algorithm using only n cells
- Elaborating other PRAM algorithms
- Language development
- Hardware optimization
Thank you for your attention.