Non-Boolean Associative Architectures Based on Nano-Oscillators

Yan Fang,
Advisor: Steven P. Levitan,
Department of Electrical & Computer Engineering, University of Pittsburgh
Collaborators: Denver H. Dash, Tadashi Shibata, Dmitri E. Nikonov, George I. Bourianoff

Overview

Motivation
- Compete and complement “end of life” CMOS for low power high density applications
- Exploit non-charge based state variables and non-Boolean operations
- Explore systems based on new devices for compute intensive tasks such as pattern recognition and computer vision

Goals
- Implement non-Boolean associative memory using coupled oscillator network
- Represent states by frequency/phase relationship between oscillators.
- Develop a hierarchical architecture capable of storing large amounts of data and processing queries at high speed

Approach & Method

Nano-Oscillator Associative Memory

Two types of Nano-Oscillator: STO, RBO.
Use synchronization of nano-scale non-linear oscillators to perform pattern matching operation.

N-Tree Structure

Face recognition problem: Retrieve the nearest neighbor of an input photo
Use hierarchical k-means clustering to organize data into a tree structure
Branch and bound search algorithm provides lists of possible nodes that might contain nearest neighbor

Simulation & Performance

Average Hit Rate

Recognition performance with different distance metrics, using FERET data set (2015 photos, 724 subjects)

Simulation of a 3-oscillator cluster, three input vector generated different degrees of match

The number of nodes visited during Branch and Bound search and the size of N-Tree structure

<table>
<thead>
<tr>
<th>B&amp;B Search</th>
<th>Total Nodes</th>
<th>Visited Nodes</th>
<th>Visited Leaf Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max</td>
<td>1199</td>
<td>454</td>
<td>293</td>
</tr>
<tr>
<td>Min</td>
<td>795</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>Mean</td>
<td>956.7</td>
<td>159.9</td>
<td>59.6</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>34.3</td>
<td>86.28</td>
<td>47.4</td>
</tr>
<tr>
<td>Average N-Tree Size</td>
<td>Total Nodes</td>
<td>Depth</td>
<td>Leaf Nodes</td>
</tr>
<tr>
<td></td>
<td>956.7</td>
<td>9.3</td>
<td>846.7</td>
</tr>
</tbody>
</table>

A single AM module has to compare every stored pattern. N-tree hierarchy can achieve the same performance with much higher efficiency

Acknowledgements: Youssry Botros, Wolfgang Porod, Mircea Stan, Dan Hammerstrom, Dana Weinstein, Mathew Puffal, Tamas Roska, Valeriu Beiu
This work was partially supported by the Intel Labs University Research Office