ARCHITECTURE AND IMPLEMENTATION OF AN ASSOCIATIVE MEMORY USING SPARSE CLUSTERED NETWORKS

McGill University
Department of Electrical and Computer Engineering

ISCAS 2012 – Seoul Korea
May 23

Hooman Jarollahi – Ph.D. Candidate
Naoya Onizawa, Vincent Gripon, Warren Gross
Outline

• Motivation

• New class of Associative Memories (AMs) using sparse clustered neural networks

• FPGA implementation and results

• Conclusions and future work
Motivation

- Associative Memories: alternatives to indexed memories
- Contents are linked, links are stored
- No need to input an explicit address
- Part of the content of a message is used to retrieve the full message
- Applications: data mining, set implementation

Indexed Memory

- Index
- 10101101101010101
- Output

Erased Message

- Eg. 101___01101___101

Indexed Memory

- Output

Associative Memory

- Output
- 10101101101010101
Motivation

• Classical AM: Hopfield Neural Networks (HNN)
  • Fast retrieval of a partially erased messages when implemented in hardware
  • Parallelism

• Storing links between contents (learning)
  • Efficiency: Information bits stored / Memory bits used

• Problems of HNN:
  • Efficiency → 0 as learning information is increased
  • A message needs to be as long as the network
    • Long messages + limited capacity → low diversity: Number of different messages the network can learn

Original Message: 001101101110
Erased Message: 001____ 11__
Gripon-Berrou Neural Networks (GBNN)

- Case study: $1.6 \times 10^6$ bits of physical memory
  - Nearly-optimal efficiency ($\sim 1$ and $\sim \times 20$ HNN)
  - Large diversity ($\sim \times 250$ HNN)
- Binary connections and nodes (existence or non-existence)

Simplified Block Diagram

Training Message: $\gamma \cdot \kappa$

# of Iterations: $\log(T) + 1$

Input Message: $g$, $\kappa \cdot 2^\kappa$

Local Decoder T-I

Learning Module

$\gamma \cdot (c-1) \cdot l^2$

Iteration Module

$\gamma \cdot l$

Global Decoder + Local Decoder T-II

Output Encoder

Retrieved Message: $\gamma \cdot \kappa$

$c \cdot l$

$c \cdot l$

$c \cdot l$
GBNN: Learning

- A network consists of
  - $c$ clusters
  - $l$ fanals (neurons) per cluster
  - $n$ neurons
- A message consists of
  - $K$ bits
  - $K/c$ sub-messages
  - $\kappa = K/c = \log_2(l)$
• Need to present erasures somehow.
• Replace 0 with -1, 1 stays 1, 0 will mean erased bit.
• Need to present erasures somehow.
• Replace 0 with -1, 1 stays 1, 0 will mean erased bit.
• **Clique:** activated fanals with fully-interconnected links
GBNN: Message Retrieval

- Local Decoding Type I (LDT-I):
  - Scalar product of a pre-defined matrix $g$ and input bits
  - Followed by maximum detection
    - Compare-and-Select

\[
g = \begin{bmatrix}
-1 & -1 & -1 & -1 \\
-1 & -1 & -1 & 1 \\
-1 & -1 & 1 & -1 \\
-1 & 1 & -1 & -1 \\
-1 & 1 & 1 & -1 \\
1 & -1 & -1 & -1 \\
1 & -1 & 1 & -1 \\
1 & -1 & 1 & 1 \\
1 & 1 & -1 & -1 \\
1 & 1 & 1 & -1 \\
1 & 1 & 1 & 1
\end{bmatrix}
\]

\[I = -1-1111-1111-11111-11\]

\[\kappa = 4\]

\[g = \begin{bmatrix}
-1 & -1 & -1 & -1 \\
-1 & -1 & -1 & 1 \\
-1 & -1 & 1 & -1 \\
-1 & 1 & -1 & -1 \\
-1 & 1 & 1 & -1 \\
1 & -1 & -1 & -1 \\
1 & -1 & 1 & -1 \\
1 & -1 & 1 & 1 \\
1 & 1 & -1 & -1 \\
1 & 1 & 1 & -1 \\
1 & 1 & 1 & 1
\end{bmatrix}
\]

\[2^\kappa\]

\[v(n_j) = \sum_{i=1}^{\kappa} g_{i,j} I(i)\]

\[v(n_j) \leftarrow \begin{cases} 
1, & \text{if } v(n_j) = v_{max} \\
& \text{and } v_{max} \geq \sigma \\
0, & \text{otherwise}
\end{cases}\]
GBNN: Message Retrieval

- Local Decoding Type I (LDT-I):
  - Scalar product of a pre-defined matrix $g$ and input bits
  - Followed by maximum detection
    - Compare-and-Select

\[
g = \begin{bmatrix}
  -1 & -1 & -1 & -1 \\
  -1 & -1 & -1 & 1 \\
  -1 & -1 & 1 & -1 \\
  -1 & 1 & -1 & 1 \\
  1 & -1 & 1 & -1 \\
  1 & 1 & -1 & 1 \\
  1 & 1 & 1 & -1 \\
  1 & 1 & 1 & 1
\end{bmatrix}
\]

\[
l = \begin{bmatrix}
  1 & 1 & 1 & 1 \\
  -1 & 1 & 1 & 1 \\
  1 & -1 & 1 & 1 \\
  1 & 1 & -1 & 1 \\
  1 & 1 & 1 & -1 \\
  1 & 1 & 1 & 1
\end{bmatrix}
\]

\[
I = g^\top l = \begin{bmatrix}
  -1 & 1 & 1 & 1 & 1 \\
  -1 & -1 & -1 & -1 & -1 \\
  -1 & -1 & -1 & -1 & -1 \\
  -1 & -1 & -1 & -1 & -1 \\
  -1 & -1 & -1 & -1 & -1 \\
  -1 & -1 & -1 & -1 & -1 \\
  -1 & -1 & -1 & -1 & -1 \\
  -1 & -1 & -1 & -1 & -1
\end{bmatrix}
\]

\[
v(n_j) = \sum_{i=1}^{\kappa} g_{i,j} I(i)
\]

\[
v(n_j) \leftarrow \begin{cases} 
1, & \text{if } v(n_j) = v_{\text{max}} \\
& \text{and } v_{\text{max}} \geq \sigma \\
0, & \text{otherwise}
\end{cases}
\]

Winner-Take-All
GBNN: Message Retrieval

- Global Decoding followed by LDT-II
- Iterative process

\[
\forall i, j, v(n_{i,j}) \leftarrow \sum_{i' = 1}^{c} \sum_{j' = 1}^{l} w(i', j')(i, j)v(n(i', j')) + \gamma v(n(i, j))
\]

\[
v(n_{j}) \leftarrow \begin{cases} 
1, & \text{if } v(n_{j}) = v_{\text{max}} \\
& \text{and } v_{\text{max}} \geq \sigma \\
0, & \text{otherwise}
\end{cases}
\]
Hardware Architecture: Learning Module

• Learning Module

• Memory used: $c(c-1)l^2$ bits
Local Decoding Type I (LDT-I)

- Input messages are antipodal → 2 bits per input bit required (+1 → 01), -1 → 10, erasure → 00)
- Fanal Activation Module: Max-Function
- Threshold value (σ) is set to 0 for LDT-I
Global Decoding + LDT-II

- Threshold value is set to $c$ for LDT-II
- Fanal Activation Module : Max-function (Compare-and-Select)
Proof-of-Concept Design Parameters

<table>
<thead>
<tr>
<th>No. of Neurons ((n))</th>
<th>128</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Clusters ((c))</td>
<td>8</td>
</tr>
<tr>
<td>No. of Fanals per Cluster ((l))</td>
<td>16</td>
</tr>
<tr>
<td>Message Length (bits)</td>
<td>32</td>
</tr>
<tr>
<td>(\kappa)</td>
<td>4</td>
</tr>
<tr>
<td>Maximum Diversity (Upper bound) ((M_{max}))</td>
<td>224</td>
</tr>
</tbody>
</table>

Maximum Diversity: 

\[
M_{max} = \frac{(c - 1)n^2}{2c^2 \log_2(n/c)}
\]
FPGA results

• Software delay measured on AMD Opteron 8387 (2.8 GHz)
• Hardware delay calculated using Altera Stratix IV

<table>
<thead>
<tr>
<th>Memory usage dedicated to $w$ (bits)</th>
<th>14,336</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dedicated Logic Registers</td>
<td>15,783/182, 400(9%)</td>
</tr>
<tr>
<td>Combinational Look-up Tables (LUT)</td>
<td>35,224/182, 400(19%)</td>
</tr>
<tr>
<td>Total pins</td>
<td>169/888(19%)</td>
</tr>
<tr>
<td>Slow 900mv 85C maximum frequency (Mhz)</td>
<td>107.15</td>
</tr>
<tr>
<td>Training, Retrieving delay (per message)</td>
<td>10 ns, 50 ns</td>
</tr>
<tr>
<td>Software to hardware delay ratio</td>
<td>$\approx 2000$</td>
</tr>
</tbody>
</table>
Software vs. FPGA Implementation

50% Cluster Erasure

10% Bit Erasure
Conclusions and Future Work

• Proof-of-concept architecture and implementation of a new class of associative memories based on sparse clustered neural networks

• Applications: data mining, set implementation

• Advantage to conventional HNN:
  • Large diversity, nearly-optimal efficiency, lower complexity (binary vs. Integer connections)

• Architecture and Implementation on FPGA (Altera Stratix IV)
Conclusions and Future Work

• ~2000 times faster than software when implemented on hardware ($n=128$, $c=8$)
• Verified H/W using S/W, compared BER and MER
• Maximum-Function is expensive and resource hungry: investigation of efficient implementation
• Large scale implementation utilizing external memory resources
Thank you!

Q/A
GBNN: Learning

- A message is partitioned into equally-sized sub-messages
- Each sub-message is mapped to a neuron in a cluster using a scalar product of a pre-defined matrix with the sub-message
- Clique is formed

\[ \kappa = K/c = \log_2(l) \]