Efficient Implementation of Memristor Cellular Nonlinear Networks using Stochastic Computing

Oscar Camps, Rodrigo Picos,
Universitat de les Illes Balears
and
Stavros G. Stavrinides
International Hellenic University
Outline

• Introduction
  • Stochastic Logic
  • Stochastic Computing
  • Stochastic Logic Revisited
  • Cellular Nonlinear Networks

• Stochastic Implementation of a Memristor Emulator

• Stochastic Logic CNN Implementation

• Implementation and Results

• Conclusions
Stochastic logic was introduced by Von Neumann in 1956

- The basic idea is the use of probabilistic calculations.
- A stochastic logic number (SLN) represents the probability ($p$) to find “1” in any given position in a string of random binary numbers.
- The values to be used are encoded in this probability $p$.
- Stochastic Logic Numbers mimic analog numbers: $p$ belongs to (0..1) or (-1..1).
Basic Operations between two probabilities $p_1$ and $p_2$ utilizing digital functions/gates

- Multiplication, $z = p_1 \cdot p_2$ is executed as: $Z = P_1 \text{ AND } P_2$

  - Example:
    
    \[
    \begin{array}{cccccccc}
    0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\
    1 & 0 & 1 & 1 & 0 & 1 & 1 & 1
    \end{array}
    \begin{array}{cccc}
    (4/8 = 1/2) \\
    (6/8 = 3/4)
    \end{array}
    \rightarrow
    \begin{array}{cccc}
    0 & 0 & 1 & 0 & 1 & 0 & 1 & 0
    \end{array}
    

- Addition, $z = p_1 + p_2$ is executed as: $(Z/2) = (P_1 \text{ AND } 0.5) \text{ OR } (P_2 \text{ AND } 0.5)$

  0.5 is needed to circumvent overflow.

Mapping in the case of these basic operations from the real to the digital domain

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
<th>$+/2$</th>
<th>$*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>+/2</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>+/2</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

a. The non-integer value is due to the fact it represents the probability of 0 or 1
Introduction – Stochastic Computing

**Advantages**
- SL allows to perform complicated mathematical functions with simple logic gates, thus with just a few transistors.
- SL calculations use simple digital functions/gates, providing SL schemes with energy efficiency.
- SL offers the capability to perform calculations with *progressive precision*. Acts as computing the most significant bits, providing a kind of early approximation.

\[
\begin{array}{ccccccc}
1 & 0 & 1 & 1 & 1 & 0 & 0 \quad &= 4/8 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \quad &= 7/16 \\
\end{array}
\]

Allowing for ending computations earlier, if the result is satisfactory (enough bits appeared).

**Disadvantages**
- SL has the inherent problem of low precision.
- It needs enormous strings for representing large numbers, with good accuracy.
Introduction – Stochastic Logic Revisited

**Existing modern era applications:**

- The Low Density Parity Check (LDPC) coding used from 2005 and on.
  - It includes probabilistic calculations efficiently implemented by stochastic logic.
  - It is used in communications, including wi-fi.

**In the future:**

- SL is compatible with biological systems.
  - It shares similarities with neural activity.
  - Initial encoding of neural activity by the frequency of the occurring “spikes”
  - It is resilient to various kinds of disturbances.
- SL energy consumption efficiency could be proved critical.
  - In applications of the IoT
  - In applications of AI, e.g. being combined with CNNs

*Look at*
Introduction – Cellular Nonlinear Networks

- It is an idea mixing Artificial Neural Networks and Cellular Networks.
- CNN’s are able to implement a parallel processing universal computer.
- They are analog.
- Memristor’s are very beneficial in implementing CNNs.
A proposed memristor emulator (equations)

• Within a Stochastic Logic Environment:

The simplest model for a memristor

\[ i(t) = G(Q) \cdot v(t) \]

\[ G_1(Q) = G_0 + G_1 \cdot Q \]

\[ G_2(Q) = \min(G_{\text{max}}, G_1(Q)) \]

\[ G(Q) = \max(G_{\text{min}}, G_2(Q)) \]

Then \( Q \) through the memristor is calculated by integrating:

\[ Q = \int^t i(t) dt \approx \Delta t \cdot \sum_j i(t = j \cdot \Delta t) \]

And addition can be rewritten as:

\[ G_1(Q) = G_0 + G_1 \Delta t \cdot \sum_j i(t = j \cdot \Delta t) S \]
A proposed memristor emulator (block-diagram)
Stochastic Implementation of a Memristor Emulator

A proposed memristor emulator (block-diagram)
Stochastic Implementation of a Memristor Emulator

A proposed memristor emulator (performance)
Stochastic Implementation of a Memristor Emulator

A proposed memristor emulator (performance)
Stochastic Logic CNN Implementation


• The output is a function of $v_x$, which is determined by:

$$\frac{dv_x}{dt} = \frac{1}{C} \left[ \sum_i (i_i + o_i) - i_M \right]$$

for 8 (closest) neighboring cells.
Stochastic Logic CNN Implementation

• Such an approach can be easily implemented in SC: $\Delta v_x = \frac{\Delta t}{C} \left[ \sum_i (i_i + o_i) - i_M \right]$
For a bit-wise approach it is demanded:

- for the adders: 16 OR-gates, 16 (1-bit) multiplexers,
- for the multipliers: 2 AND-gates,
- 1 inverter,
- 1 accumulator, and
- the memristor emulator

For the cell circuit <60 gates

- According to O. Camps et al., IEEE MOCAST 2018, for 8-bit greyscale images the needed words for properly representing the information, are 14-bits long.
- In this case there is 95% probability that the error will be in the last 6 bits.
- Regarding the speed, for an FPGA @ 80MHz: ~10000fps (parallel processing).
Implementation and Results

Implementation
• The method was verified by implementing the circuit into a DE2-70 Altera FPGA.
• Compilation was achieved by using Quartus-II.
• The circuit included less that 600 gates (<2% of available gates).
• Communication was implemented by JTAG interface.
• The system was controlled by Matlab.

Results
• Border detection through the EDGE routine was selected for the proof of concept demonstration.
• 8-bit grayscale images, converted to 14-bits (by zero padding) were utilized.
• The method performed correctly in all cases.
• The reverse process (stochastic images to conventional ones) properly performed.
• Several images of various size were tested.

Some of them appear in the next slides
Results

Real Picture

Processed Image
Results

Real Picture

Processed Image
Results

Real Picture

Processed Image

ECCDT2020 – September 7th – 9th, Sofia, Bulgaria
Results

Real Picture

Processed Image
Results

Real Picture

Processed Image
Results

Real Picture

Processed Image
Conclusions

• A fully digital implementation of a CNN based on Stochastic Logic.
• The basic cell of the CNN architecture includes a memristor emulator.
• The memristor emulator is implemented by utilizing a digital implementation.
• The memristor emulator cell was implemented onto an FPGA.
• Matlab was utilized to implement edge detection.

Next steps

• Use of a full FPGA implementation, resulting to real time image processing within the demands for the IoT.

Thank you!