Lecture 6 - LANDAUER: Computing with uncertainty

Igor Neri - NiPS Laboratory, University of Perugia
July 18, 2014

NiPS Summer School 2014
ICT-Energy: Energy management at micro and nanoscales for future ICT
We are used to correct results

- User applications, Algorithms
- Correct logic operations
Physical process underlying computation are not exact
Additional HW and SW to bridge the gap between reliable and not reliable

- User applications, Algorithms
- Correct logic operations
- Redundancy
- Error correction
- Logic gates, memory
- Physical process
Why
Why

- Save time
Why

• Save time

• Save energy
Why

• Save time

• Save energy
Why

- Save time
- Save energy
- We are already doing it!
How many digits of $\pi$ for error lesser then one meter on Earth circumference?

$r = 6,371$ kilometers
How many digits of $\pi$ for error lesser then one meter on Earth circumference?
How many digits of $\pi$ for error lesser then one meter on Earth circumference?
Outline

• History:
  • Probabilistic Logics (Von Neumann) and Stochastic Computing

• Nowadays:
  • Approximate programming
  • Computing with uncertainty
Probabilistic Logics (Von Neumann) and Stochastic Computing
PROBABILISTIC LOGICS AND THE SYNTHESIS OF RELIABLE ORGANISMS FROM UNRELIABLE COMPONENTS

J. von Neumann

1. INTRODUCTION

The paper that follows is based on notes taken by Dr. R. S. Pierce on five lectures given by the author at the California Institute of Technology in January 1952. They have been revised by the author but they reflect, apart from minor changes, the lectures as they were delivered.

The subject-matter, as the title suggests, is the role of error in logics, or in the physical implementation of logics – in automata-synthesis. Error is viewed, therefore, not as an extraneous and misdirected or misleading accident, but as an essential part of the process under consideration – its importance in the synthesis of automata being fully comparable to that of the factor which is normally considered, the intended and correct logical structure.

Our present treatment of error is unsatisfactory and ad hoc. It is the author's conviction, voiced over many years, that error should be treated by thermodynamical methods, and be the subject of a thermodynamical theory, as information has been, by the work of L. Szilard and C. E. Shannon [Cf. 5.2]. The present treatment falls far short of achieving this, but it is hoped some of the building materials, which will have to
## Stochastic computing: timeline

<table>
<thead>
<tr>
<th>Dates</th>
<th>Items</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>1956</td>
<td>Fundamental concepts of probabilistic logic design.</td>
<td>[Von Neumann 1956]</td>
</tr>
<tr>
<td>1960-79</td>
<td>Definition of SC and introduction of basic concepts.</td>
<td>[Gaines 1967; 1969]</td>
</tr>
<tr>
<td></td>
<td>Construction of general-purpose SC computers.</td>
<td>[Poppelbaum 1976]</td>
</tr>
<tr>
<td>1980-99</td>
<td>Advances in the theory of SC.</td>
<td>[Jeavons et al. 1994]</td>
</tr>
<tr>
<td></td>
<td>Studies of specialized applications of SC, including</td>
<td>[Kim and Shanblatt 1995]</td>
</tr>
<tr>
<td></td>
<td>artificial neural networks and hybrid controllers.</td>
<td>[Toral et al. 1999]</td>
</tr>
<tr>
<td>2000-</td>
<td>Application to efficient decoding of error-correcting</td>
<td>[Gaudet and Rapley 2003]</td>
</tr>
<tr>
<td>present</td>
<td>codes. New general-purpose architectures.</td>
<td>[Qian et al. 2011]</td>
</tr>
</tbody>
</table>
Stochastic computing: basic features

- Numbers are represented by bit-streams
- Numbers can be processed by very simple circuits
- Numbers are interpreted as probabilities under both normal and faulty conditions
- A bit-stream $S$ containing 25 percent of 1s and 75 percent of 0s denotes the number $p = 0.25$
  - $(1,0,0,0)$, $(0,1,0,0)$ and $(0,1,0,0,0,1,0,0)$ are all possible representations of 0.25
Stochastic computing: multiplication
Stochastic computing: multiplication

(a)

\[ S_1: 0,1,1,0,1,0,1,0 (4/8) \]
\[ S_2: 1,0,1,1,1,0,1,1 (6/8) \]
\[ S_3: 0,0,1,0,1,0,1,0 (3/8) \]

(b)

\[ S_1: 0,1,0,1,1,1,0,0 (4/8) \]
\[ S_2: 1,1,1,0,1,0,1,1 (6/8) \]
\[ S_3: 0,1,0,0,1,0,0,0 (2/8) \]
Stochastic computing: sum

\[
p(S_4) = p(S_3)p(S_1) + (1 - p(S_3))p(S_2) = \frac{p(S_1) + p(S_2)}{2}
\]
Stochastic number conversion

Survey of Stochastic Computing - J. Hayes, ACM 2012
Stochastic computing: stochastic numbers

- Fluctuations inherent in random numbers
- Correlations among the stochastic numbers

\[ \sum_{i=1}^{n} S_1(i)S_2(i) = \frac{\sum_{i=1}^{n} S_1(i) \times \sum_{i=1}^{n} S_2(i)}{n} \]

- It is generally not desirable to use truly random number sources to derive or process stochastic numbers

Survey of Stochastic Computing - J. Hayes, ACM 2012
Stochastic Number Generators (SNGs)

- Linear Feedback Shift Register (LFSR)

Gupta and Kumaresan [1988]
Stochastic computing: accuracy

- With m bits of precision, we can distinguish between $2^m$ different numbers.

- The numbers in the interval $[0,1]$, when represented with eight-bit precision reduce to the following 256-member set: $\{0/256, 1/256, 2/256, ..., 255/256, 256/256\}$

- Their stochastic representation requires bit-streams of length 256.

- To increase the precision from eight to nine bits requires doubling the bit-stream length to 512.
Stochastic computing: applications

Figure 10. Error-tolerance comparison between conventional and SC implementations of a gamma correction function. © 2011 IEEE. Reprinted, with permission, from [Qian et al. 2011].

Survey of Stochastic Computing - J. Hayes, ACM 2012
Stochastic computing applications: LDPC

- Linear error correcting code
- Method to transmit a message over a noisy channel
- Rate near to the theoretical maximum
- Difficult to implement in practice
- Largely ignored until 1990s
- Used in WiFi and digital video broadcasting
Stochastic computing applications: LDPC

- Nine-bit single-error detecting code

\[ x_0 \oplus x_1 \oplus x_2 \oplus \ldots \oplus x_7 \oplus x_8 = 0 \]

- \( x_0, x_1, \ldots, x_7 \) are data bits

- \( x_8 \) is the parity bit
Stochastic computing applications: LDPC

- LDPC code of length 6

\[
x_0 \oplus x_2 \oplus x_3 = 0;
x_1 \oplus x_2 = 0;
x_0 \oplus x_4 = 0;
x_1 \oplus x_5 = 0.
\]
Stochastic computing applications: LDPC

• LDPC code of length 6

\[
\begin{align*}
x_0 \oplus x_2 \oplus x_3 &= 0; \\
x_1 \oplus x_2 &= 0; \\
x_0 \oplus x_4 &= 0; \\
x_1 \oplus x_5 &= 0.
\end{align*}
\]

\[
\begin{bmatrix}
1 & 0 & 1 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x_0 \\
x_1 \\
x_2 \\
x_3 \\
x_4 \\
x_5
\end{bmatrix} = 0,
\]
Stochastic computing applications: LDPC

A typical parity matrix $H$ is huge ($n$ and $k$ can be many thousands), but it is sparse in that it has only a few 1s in each row and column.

A relatively small subset of all $2^n$ possible combinations of 0s and 1s on $x_0, x_1, ..., x_{n-1}$ satisfy the code equations.

These are the valid codewords.

Survey of Stochastic Computing - J. Hayes, ACM 2012
Stochastic computing applications: LDPC

Survey of Stochastic Computing - J. Hayes, ACM 2012
Stochastic computing applications: LDPC
Approximate programming
Approximate programming

Listing 1: Arithmetic mean in Java

```java
public static double mean(double[] p) {
    double sum = 0; // sum of all the elements
    for (int i=0; i<p.length; i++) {
        sum += p[i];
    }
    return sum / p.length;
}
```
Approximate programming

Listing 1: Arithmetic mean in Java

```java
public static double mean(double[] p) {
    double sum = 0; // sum of all the elements
    for (int i=0; i<p.length; i++) {
        sum += p[i];
    }
    return sum / p.length;
}
```

Listing 2: Approximate arithmetic mean in Java

```java
public static double mean(double[] p) {
    double sum = 0; // sum of all the elements
    for (int i=0; i<p.length; i+=2) {
        sum += p[i];
    }
    return 2 * sum / p.length;
}
```
Approximate programming

```java
@Approx float[] nums;

@Approx float total = 0.0f;
for (@Precise int i = 0;
    i < nums.length;
    ++i)
    total += nums[i];
return total / nums.length;
```
Approximate programming

Disciplined Approximate Computing: From Language to Hardware - L. Ceze - 2013
Approximate programming: HW support

- Approximation-aware ISA
- Approximate operations
- Approximate data
- Dual-voltage microarchitecture

Architecture Support for Disciplined Approximate Programming - H. Esmaeilzadeh et al. - ASPLOS’12
Disciplined Approximate Computing: From Language to Hardware - L. Ceze - 2013
Approximate programming: results
What we did
Sub-Landauer gate

Beating the Landauer's limit by trading energy with uncertainty - L. Gammaitoni - arXiv:1111.2937 [cond-mat.mtrl-sci]
Sub-Landauer gate

Beating the Landauer’s limit by trading energy with uncertainty - L. Gammaitoni - arXiv:1111.2937 [cond-mat.mtrl-sci]
Sub-Landauer gate

Beating the Landauer's limit by trading energy with uncertainty - L. Gammaitoni - arXiv:1111.2937 [cond-mat.mtrl-sci]
Sub-Landauer gate

\[ \Delta S = S_f - S_i = k_B (\ln(1) - \ln(2)) = -k_B \ln(2) \]
Sub-Landauer gate

\[ \Delta S = S_f - S_i \]

Beating the Landauer's limit by trading energy with uncertainty - L. Gammaitoni - arXiv:1111.2937 [cond-mat.mtrl-sci]
Sub-Landauer gate

\[ \Delta S = S_f - S_i \]
\[ S_f(P_e) = -k_B((1 - P_e) \ln(1 - P_e) + P_e \ln(P_e)) \]
Sub-Landauer gate

\[ \Delta S = S_f - S_i \]
\[ S_f(P_e) = -k_B((1 - P_e) \ln(1 - P_e) + P_e \ln(P_e)) \]
\[ Q(P_e) = -k_B T((1 - P_e) \ln(1 - P_e) + P_e \ln(P_e)) + k_B T \ln(2) \]
Sub-Landauer gate

Beating the Landauer's limit by trading energy with uncertainty - L. Gammaitoni - arXiv:1111.2937 [cond-mat.mtrl-sci]
Sub-Landauer gate

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Q</th>
<th>Q error prob.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>$e^2$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>$2(e-e^2)$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$2(e-e^2)$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>$2e-e^2$</td>
</tr>
</tbody>
</table>
Sub-Landauer gate
ALU simulator

- ALU simulator developed in Java
- Based on NAND gates
- Implemented functional units:
  - Adder (ripple-carry)
  - Comparator (based on SN54/74LS85)
Full Adder
Full Adder
Full Adder
Sorting: definition

Process of taking a list of objects with a linear ordering

\((a_1, a_2, \ldots, a_{n-1}, a_n)\)

and output a permutation of the list

\((a_{k_1}, a_{k_2}, \ldots, a_{kn-1}, a_{kn})\)

such that

\(a_{k_1} \leq a_{k_2} \leq \cdots \leq a_{kn-1} \leq a_{kn}\)

Usually we're interested in sorting a list of “records” in order by some field, however, without loss of generality, this is equal to sorting values
• In general no sort algorithms, based on comparison, can have a run time better than $n \log(n)$

# leafs: $n!$ permutation

tree height: $\log(n!) \sim n \log(n)$

$n \log(n)$ comparison

• However under some circumstance there are algorithms that run in linear time
### Sorting algorithms performances evaluation

<table>
<thead>
<tr>
<th>Name</th>
<th>Best</th>
<th>Average</th>
<th>Worst</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selection</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>1</td>
</tr>
<tr>
<td>Quick</td>
<td>nlogn</td>
<td>nlogn</td>
<td>n</td>
<td>logn</td>
</tr>
<tr>
<td>Merge</td>
<td>nlogn</td>
<td>nlogn</td>
<td>nlogn</td>
<td>n</td>
</tr>
<tr>
<td>Insertion</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>1</td>
</tr>
<tr>
<td>Bubble</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>1</td>
</tr>
</tbody>
</table>

Relevant time operations:
- # of Comparisons
- # of read/write operations
Digital Comparator

• The purpose of a Digital Comparator is to compare a set of variables or unknown numbers, for example A (A1, A2, A3, .... An, etc) against that of a constant or unknown value such as B (B1, B2, B3, .... Bn, etc) and produce an output condition or flag depending upon the result of the comparison.

• **Identity Comparator**: digital comparator with only one output terminal for when $A = B$

• **Magnitude Comparator**: digital comparator that has three output terminals, one each for equality, $A = B$ greater than, $A > B$ and less than $A < B$
1-bit Magnitude Comparator

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A&lt;B</th>
<th>A=B</th>
<th>A&gt;B</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
4-bits Magnitude Comparator

A0  A1  A2  A3
B0  B1  B2  B3

A<B  A=B  A>B
8-bits Magnitude Comparator

A < B in
A = B in
A > B in

Less Significative Bits

A0 → A1 → A2 A3

4-bit Magnitude Comparator

A < B
A = B
A > B

Most Significative Bits

A4 A5 A6 A7

4-bit Magnitude Comparator

A < B
A = B
A > B

8-bit comparison output

A0 → A1 → A2 A3

B0 → B1 → B2 B3

A4 A5 A6 A7

B4 → B5 → B6 B7

8-bit comparison output
4-bits Magnitude Comparator - SN54/74LS85

[Diagram of a 4-bits Magnitude Comparator]
16-bits Magnitude Comparator with NAND gates

Traditional gates replaced with NANDs
Control parameters

- Switch error rate bits 0-3
- Switch error rate bits 4-7
- Switch error rate bits 8-11
- Switch error rate bits 12-15
Monte Carlo simulations

- Varying the switch error rate of each comparator to compute:
  - Comparison error probability given switch error probability of each comparator
  - Computed minimum energy required by each comparison given switch error probability for each comparator in respect to the Landauer limit

- Computed noise level after sorting procedure for different sorting algorithms varying comparison error rate
Comparison error rate vs energy saving

Sorting with uncertainty: Trading accuracy with energy saving, I. Neri et al., NANOENERGY 2013
Sorting performances metrics: output noise level

- Metrics that returns the noise in the output sequence: it is a measure of how the sequence is far from being ordered.

- For each element on the output array check how many pairs of elements are in wrong order in respect to the perfectly ordered sequence.

- Noise level: \( \frac{\text{#pairs in wrong order}}{\text{#total number of pairs}} \)

<table>
<thead>
<tr>
<th>Noise Level</th>
<th>12</th>
<th>18</th>
<th>21</th>
<th>3</th>
<th>25</th>
<th>23</th>
<th>30</th>
</tr>
</thead>
</table>
Results: input sequence #500 - uniform 0-2

Sorting with uncertainty: Trading accuracy with energy saving, I. Neri et al., NANOENERGY 2013
Results: input sequence #10000 - uniform 0-2^{16}

Sorting with uncertainty: Trading accuracy with energy saving, I. Neri et al., NANOENERGY 2013
Results: input sequence #20000 - uniform 0-2^{16}

Sorting with uncertainty: Trading accuracy with energy saving, I. Neri et al., NANOENERGY 2013
Results: input sequence #20000 - uniform 0-2^{16}

Sorting with uncertainty: Trading accuracy with energy saving, I. Neri et al., NANOENERGY 2013
Results: input sequence #20000 - uniform 0-2^{16}

>40% energy saving on comparison

Sorting with uncertainty: Trading accuracy with energy saving, I. Neri et al., NANOENERGY 2013
Results: seq. #10000 - number of comparisons

Bubble sort

Insertion sort

Quick sort

Merge sort

Comparison with uncertainty: Trading accuracy with energy saving, I. Neri et al., NANOENERGY 2013
Results: seq. #10000 - number of I/O operations

Bubble sort

Insertion sort

Quick sort

Merge sort

Comparison error rate

Sorting with uncertainty: Trading accuracy with energy saving, I. Neri et al., NANOENERGY 2013
Energy/Error ratio optimisation through Genetic Algorithms

- Starting from truth table evolve logic network
- Initial population randomly generated
- Each generation random selected individuals breed and mutate
- Fitness evaluated on:
  - maximising the number of correct outputs given the truth table
  - minimising the number of logic gates used
  - minimising the energy consumption
Individual

INPUTS

OUTPUTS

PE1
PE2
PE3
PE4
PE5
PE6
PE7
PE8
PE9
PE10
PE11
PE12
PE13
PE14
PE15
PE16
Individual: Processing Element

INPUTS

INPUTS SELECTOR

FUNCTION SELECTOR

Switch error rate

Input A

Input B

AND

NAND

OR

...

NOT

OUTPUT
Crossover

Parents

INPUTS

OUTPUTS

INPUTS

OUTPUTS

Parents
Crossover

Parents

Sons
Mutation

Switch error rate

Input A

Input B

AND
NAND
OR
...
NOT

INPUTS
FUNCTION SELECTION
INPUTS SELECTION
OUTPUT
Mutation

INPUTS

Switch error rate

Input A

Input B

AND
NAND
OR
...
NOT

OUTPUT

INPUTS SELECTOR

FUNCTION SELECTOR
Mutation

Switch error rate
Input A
Input B
AND
NAND
OR
...
NOT

INPUTS
OUTPUT
INPUTS SELECTOR
FUNCTION SELECTOR
Mutation

INPUTS

Switch error rate

Input A

Input B

AND

NAND

OR

...

NOT

OUTPUT

INPUTS SELECTOR

FUNCTION SELECTOR
Unreliable logic circuit simulator: genetic evolution

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Unreliable logic circuit simulator: genetic evolution
Unreliable logic circuit simulator: genetic evolution
Unreliable logic circuit simulator: genetic evolution
Unreliable logic circuit simulator: genetic evolution
Unreliable logic circuit simulator: genetic evolution

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Diagram:

- I0
- I1
- XOR8
- AND9
- NOT15
- AND41
- O0
- O1
- O2
Unreliable logic circuit simulator: genetic evolution
Unreliable logic circuit simulator: genetic evolution

![Graph showing energy and fitness over generations.](image)
Unreliable logic circuit simulator: genetic evolution
Unreliable logic circuit simulator: genetic evolution
Thank you for your attention!

igor.neri@nipslab.org