Polysynchronous Stochastic Circuits

M. Hassan Najafi, David J. Lilja, Marc Riedel, and Kia Bazargan

{najaf011, lilja, mriedel, kia}@umn.edu

ASP-DAC 2016
Overview

• Introduction
  • Synchronous systems: CDNs a major design bottleneck
  • Completely asynchronous, GALS
• Our proposed approach
  • Polysynchronous stochastic
• Background
  • Stochastic computation
• Basic Stochastic Operations with polysynchronous inputs
• Stochastic Circuits with Polysynchronous Inputs
• Experimental results
  • High level simulation
  • Circuit level simulation
• Conclusion
Introduction

- Electronic systems are inherently **asynchronous**
  - They can be adapted to behave **synchronously**

- **Synchronism**
  - **brings significant advantages:**
    - Simplifies the design effort, Performance guarantee
  - **But comes at a significant cost:**
    - Must create **clock distribution network (CDN)**
    - A single clock signal arrival time (zero clock skew)

- **CDNs are becoming a major design bottleneck**
  - Accounts for significant **area**,
  - consumes significant **power**,
  - limits the circuit **performance**

Figure 1. A global CDN
Introduction

• Completely asynchronous design methodologies
  • Have been studied for decades
  • have never gained widespread acceptance

• Globally asynchronous, but locally synchronous (GALS)
  • Pros. reduces the cost of the distribution network
    • By Splitting the domains
  • Cons. complex circuitry for handshaking
    • So splitting is only performed at a coarse level.
Our proposed approach

• We propose a radically new approach:

  • **Splitting** clock domains at a very fine level

  • Domains consisting of only a handful of gates each

  • Each domain is synchronized by
    • an inexpensive clock signal, generated locally
      • A ring of inverters.

• This is feasible by adopting
  • a **stochastic representation** for signal values.
Background: Stochastic Computation

- Logical computation is performed on random bit streams.

- The signal values are encoded by
  - the probability of obtaining a one versus a zero.
  - **Unipolar** encoding: $0 \leq x \leq 1$
    - Each bit has probability $x$ of being 1 and $1 - x$ of being 0.
  - **Bipolar** encoding: $-1 \leq x \leq 1$
    - Each bit has probability $\frac{x+1}{2}$ of being 1 and $1 - \frac{x+1}{2}$ of being 0.

- Example. 11010, 00111, 1111001100,...
  - 0.6 in unipolar, 0.2 in bipolar
Background: Stochastic Number Generator

- **Stochastic Number Generator (SNG)**
  - To generate a stochastic bit stream with probability $x$:
    1. Obtain an unbiased random value $r$ from random source
       - physical random sources
       - pseudo-random constructs such as LFSRs.
    2. Compare it to the target value $x$;
    3. Output a 1 if $r \geq x$ and a 0 otherwise.

Fig 6. Stochastic Number Generator
Background: Stochastic Computation

- **Complex operations** can be performed with **very simple logic**.
- **Multiplication:** AND gate  
  **Scaled addition:** MUX

![Figure 2. An example of multiplication using an AND gate](image1)

![Figure 3. An example of scaled addition using a MUX unit](image2)

- A reduction in area of 50x or 100x compared to conventional implementations is common.
Background: Stochastic Computation

- Another advantage over binary radix: Error Tolerance

- In a noisy environment:
  - With a **binary** representation in the worst case:
    - the most significant bit gets flipped => a large error.
  - With a **stochastic** representation:
    - all the bits in the stream have equal weight.
    - A single flip results in a small error.
  - Stochastic representation is high error tolerant.
Polysynchronous Stochastic

- A more compelling advantage:
  - with a **stochastic** representation,
    - computational units can **tolerate skew**
    - **Obviates** the need for a **global clock signal**

- This stems from the fact: the stochastic representation is **uniform**
  - All that matters:
    - **the fraction of time that the signal is high.**

- The correct value is computed even when the inputs are **misaligned temporally**.

- **polysynchronous stochastic**
Polysynchronous Stochastic

- Stochastic operations with polysynchronous inputs:
  - Multiplication
    
    ![Figure 4](image1.png)
    
    Fig 4. Stochastic multiplication using an AND with unsynchronized bit streams.

- Scaled addition
  
  ![Figure 5](image2.png)
  
  Fig 5. Stochastic scaled addition using a MUX with unsynchronized bit streams.
Basic Stochastic Operations with polysynchronous inputs

- Consider an **AND gate**, responsible for **multiplying** two input bit streams, $P_1$ and $P_2$

- $P_1$ and $P_2$ are generated by **SNGs** driven by **two clocks with different periods**, $T_1$ and $T_2$.

- **Simple case.**
  - We connect **two clocks** with periods of $T_1$ and $T_2$ to the inputs
  - The inputs are both $P=0.5$
  - The expected output value is $Y=0.25$.
  - We verify **three different scenarios:**
    1) $T_1=2\text{ns}$, $T_2=3.5\text{ns}$,
    2) $T_1=2\text{ns}$, $T_2=3.2\text{ns}$,
    3) $T_1=1.8\text{ns}$, $T_2=3.2\text{ns}$.
Basic Stochastic Operations with polysynchronous inputs

Fig 7. Input clock signals and the corresponding output from connecting polysynchronous inputs to an AND gate.

Table 1. Different observed lengths of high pulses at the output of the AND gate and the number of occurrences of each one for 1000ns.

<table>
<thead>
<tr>
<th>Length</th>
<th>#</th>
<th>Length</th>
<th>#</th>
<th>Length</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>72</td>
<td>0.2</td>
<td>63</td>
<td>0.1</td>
<td>35</td>
</tr>
<tr>
<td>0.50</td>
<td>72</td>
<td>0.4</td>
<td>63</td>
<td>0.2</td>
<td>35</td>
</tr>
<tr>
<td>0.75</td>
<td>71</td>
<td>0.6</td>
<td>62</td>
<td>0.3</td>
<td>35</td>
</tr>
<tr>
<td>1.00</td>
<td>142</td>
<td>0.8</td>
<td>62</td>
<td>0.4</td>
<td>35</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>1.0</td>
<td>125</td>
<td>0.5</td>
<td>35</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.6</td>
<td>35</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.7</td>
<td>35</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.8</td>
<td>34</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.9</td>
<td>138</td>
</tr>
<tr>
<td>Total High</td>
<td>249.25</td>
<td>249.60</td>
<td>249.40</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Temporal misalignment of input values does not affect the accuracy of the computation.
Basic Stochastic Operations with polysynchronous inputs

• Functionality of a MUX unit performing scaled addition with temporally misaligned inputs.

Table 2. The measured output of the MUX when three polysynchronous clocks with distinct periods are connected to its inputs for 1000ns.

<table>
<thead>
<tr>
<th>T1(ns)</th>
<th>T2(ns)</th>
<th>T3(ns)</th>
<th>Total High Time</th>
<th>Measured Output</th>
<th>Expected Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.00</td>
<td>1.80</td>
<td>3.75</td>
<td>499.43</td>
<td>0.499</td>
<td>0.500</td>
</tr>
<tr>
<td>1.90</td>
<td>2.63</td>
<td>2.12</td>
<td>500.21</td>
<td>0.500</td>
<td>0.500</td>
</tr>
<tr>
<td>3.20</td>
<td>1.60</td>
<td>2.00</td>
<td>498.80</td>
<td>0.499</td>
<td>0.500</td>
</tr>
<tr>
<td>2.87</td>
<td>2.43</td>
<td>2.10</td>
<td>499.23</td>
<td>0.499</td>
<td>0.500</td>
</tr>
</tbody>
</table>

• The measured output values are essentially equal to the expected output value of 0.5

\[(0.5 + 0.5)/2 = 0.5\]
Basic Stochastic Operations with polysynchronous inputs

- General case.

Table 3. Stochastic \textit{multiplication} and \textit{scaled addition}, using an AND gate and a MUX, with inputs generated by unsynchronized SNGs.

<table>
<thead>
<tr>
<th>In1</th>
<th>T1(ns)</th>
<th>In2</th>
<th>T2(ns)</th>
<th>T3(ns)</th>
<th>AND Output Meas.</th>
<th>Expec.</th>
<th>MUX Output Meas.</th>
<th>Expec.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.50</td>
<td>2.10</td>
<td>0.50</td>
<td>2.30</td>
<td>2.00</td>
<td>0.247</td>
<td>0.250</td>
<td>0.502</td>
<td>0.500</td>
</tr>
<tr>
<td>0.35</td>
<td>2.82</td>
<td>0.66</td>
<td>3.11</td>
<td>3.68</td>
<td>0.237</td>
<td>0.231</td>
<td>0.498</td>
<td>0.505</td>
</tr>
<tr>
<td>0.27</td>
<td>2.81</td>
<td>0.48</td>
<td>2.36</td>
<td>3.61</td>
<td>0.128</td>
<td>0.129</td>
<td>0.372</td>
<td>0.375</td>
</tr>
<tr>
<td>0.18</td>
<td>1.60</td>
<td>0.53</td>
<td>3.70</td>
<td>2.20</td>
<td>0.096</td>
<td>0.095</td>
<td>0.350</td>
<td>0.355</td>
</tr>
</tbody>
</table>

- Results are based on bit streams of length 1024, generated with 32-bit LFSRs.

- In spite of the polysynchronous clocking, the results are accurate to within the error bound expected for stochastic computation.
Stochastic Circuits with Polysynchronous Inputs

\[
S_{i,j} = \frac{1}{2} \times \left( \frac{1}{2} |r_{i,j} - r_{i+1,j+1}| + \frac{1}{2} |r_{i,j+1} - r_{i+1,j}| \right)
\]

Fig. 8. Stochastic implementation of the Robert's cross edge detection algorithm [1].

Fig. 9. Stochastic implementation of the Gamma Correction function [2].


Stochastic Circuits with Polysynchronous Inputs

Fig. 10.a. Basic Sorting Unit [1]

Fig. 10.b. Stochastic implementation of the basic sorting unit [1].

Fig. 10.c. A stochastic implementation of a 3*3 Noise reduction median filter [1]

Stochastic Circuits with Polysynchronous Inputs

- Parallel processing of a 4x4 input image

The input bit streams arriving from neighboring are generated by SNGs driven by their local clocks and so are all potentially misaligned temporally.

Fig. 11. 16 Robert's Cross Cells processing a 4x4 input image concurrently.
Experimental Results

• Circuits are implemented in Verilog,
• Simulation in Modelsim
• a 256*256 sample input image

• For the Robert's cross circuit,
  • 3 out of 4 streams are received asynchronously

• For the Median Filtering circuit,
  • 8 out of 9 streams are received asynchronously

• For the Gamma correction circuit,
  • Bernstein coefficients streams are generated with SNGs driven by local clocks.

• For the SNGs: 32-bit maximal period LFSR, seeded with a random initial value for each trial, Bit streams of length 1024
Experimental Results

• Generating a golden case
  • All local clocks are synchronized (period of 2ns).
  • Even with synchronized clocks there are some inaccuracy when comparing with the conventional binary outputs.

Fig. 12. The original sample image, and the outputs of processing the sample image using stochastic circuits working with synchronized local clocks.
Experimental Results

- We compare 6 different clocking schemes

- **Scheme 1.** The period of the local clock in all processing cells
  - Is fixed at 2ns (the golden case).

- **Scheme 2.** The period of the local clock in each cell
  - a random real value between 2-3 ns
  - So 50% variation between the clock periods.

- **Scheme 3.** The period of the local clock in each cell
  - a random real value between 2-4ns
  - so 100% variation.
Experimental Results

• We compare 6 different clocking schemes

• **Scheme 4.** The periods of the local clocks
  • random values between 2-4ns,
  • but high output pulses that are less than 10% of the 2ns clock period (0.2ns) are filtered out (i.e., they are set to 0).

• **Scheme 5.** Same as Scheme 4, but we filter out high output pulses that are less than 15% of the 2ns clock period.

• **Scheme 6.** Same as Scheme 4, but we filter out high output pulses that are less than 20% of the 2ns clock period
Experimental Results

• To produce more reliable results
  • **Simulate** each circuit based on **the six clocking schemes**
    • **10 times**, each time with **different initial conditions**:
      • **10 different LFSR seed values** for each SNGs
      • **10 different sets of values for the periods** of the local clocks.

• We calculate the average output error rate as follows:

\[
E = \frac{\sum_{i=1}^{256} \sum_{j=1}^{256} |T_{i,j} - S_{i,j}|}{255 \times (256 \times 256)} \times 100
\]
Experimental Results

Table 4. The **mean of the output error rates** for the **three implemented stochastic circuits**, simulated in **six different clocking schemes**.

<table>
<thead>
<tr>
<th>Clocking Schemes</th>
<th>S.1</th>
<th>S.2</th>
<th>S.3</th>
<th>S.4</th>
<th>S.5</th>
<th>S.6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robert</td>
<td>2.88%</td>
<td>2.89%</td>
<td>2.94%</td>
<td>2.89%</td>
<td>2.92%</td>
<td>2.88%</td>
</tr>
<tr>
<td>Gamma.</td>
<td>2.56%</td>
<td>2.50%</td>
<td>2.49%</td>
<td>2.51%</td>
<td>2.59%</td>
<td>2.64%</td>
</tr>
<tr>
<td>Median.</td>
<td>3.15%</td>
<td>3.19%</td>
<td>3.31%</td>
<td>3.28%</td>
<td>3.39%</td>
<td>3.31%</td>
</tr>
</tbody>
</table>

- The **accuracy of the computations** is essentially **independent** of how synchronized the local clocks are.

- **Polysynchrony** might actually **help** instead of **hurting**!
Experimental Results

SPICE-Level Verification

- **SPICE netlist** of the Robert's cross circuit
- Simulation using 45-nm gate library in HSPICE
- 500 sets of random input values,

- For the conventional **synchronous**: clock periods => 1ns.
- For the **polysynchronous**: clock periods => random (1ns-2ns)

- On 500 trials, the mean of the output error rates were
  - 4.91% for the **synchronous** approach.
  - 4.42% for the **polysynchronous** approach.

- Polysynchronous stochastic circuits are essentially as accurate as conventional synchronous circuits.
Conclusion

• We presented “Polysynchronous Stochastic Computation”
  • As a proof of concept

• It is predicated on the paradigm of stochastic computing.
  • Low cost, low power consumption, high resiliency

• Another important benefit of the stochastic paradigm
  • the flexibility it provides for the clocking mechanism.
  • Computation is accurate irrespective of the temporal alignment of input values,
    • So it can tolerate arbitrary amounts of clock skew.
    • We can remove the CDN and instead use
      • Locally inexpensive generated clocks
  • Improving area, power, and design complexity
Thank you

Questions?

M. Hassan Najafi
Najaf011@umn.edu