Polysynchronous Stochastic Circuits

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

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

ASP-DAC 2016

UNIVERSITY OF MINNESOTA

Driven to Discover℠

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

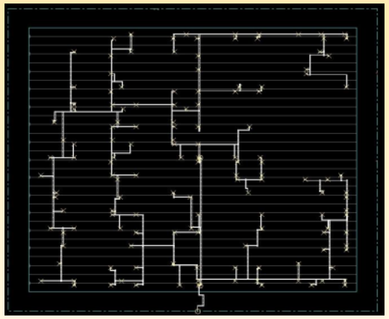


Figure 1. A global CDN

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: $\mathbf{0} \leq \boldsymbol{x} \leq \mathbf{1}$
  - Each bit has probability $\boldsymbol{x}$ of being 1 and $1 - \boldsymbol{x}$ of being 0.
- **Bipolar** encoding: $-\mathbf{1} \leq \boldsymbol{x} \leq \mathbf{1}$
  - Each bit has probability $\frac{\boldsymbol{x}+\mathbf{1}}{\mathbf{2}}$ of being 1 and $1 - \frac{\boldsymbol{x}+\mathbf{1}}{\mathbf{2}}$ of being 0.

- **Example.** **11010, 00111, 1111001100,…**
  0.6 in unipolar, **0.2** in bipolar

Background: Stochastic Computation

- Stochastic representation **is less compact than a binary radix**.
  - **binary radix:** a compressed, **positional** encoding
  - **Stochastic.** an uncompressed, **uniform** encoding.

- **Complex operations** can be performed with **very simple logic.**

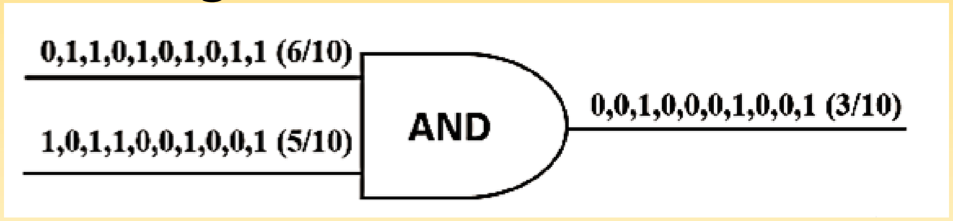


Figure 2. An example of multiplication using an AND gate

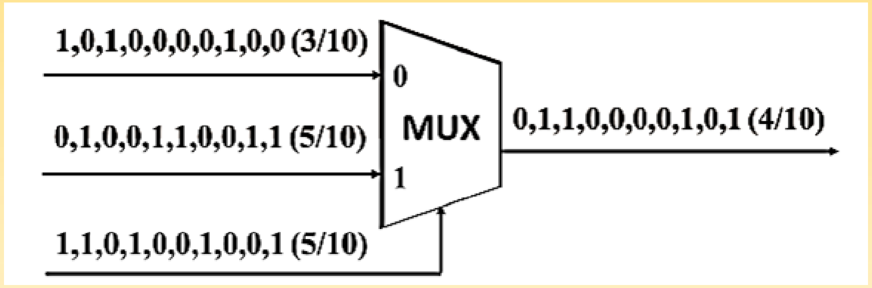


Figure 3. An example of scaled addition using a MUX unit

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 rate tolerant.**

Background: Stochastic Computation

- A more compelling advantage:
  - with a **stochastic** representation,
    - computational units can **tolerate skew** in the arrival time of their inputs.
      - **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**.

    - We call this approach **polysynchronous stochastic**

Background: Stochastic Computation

- Stochastic operations with polysynchronous inputs:
- Multiplication

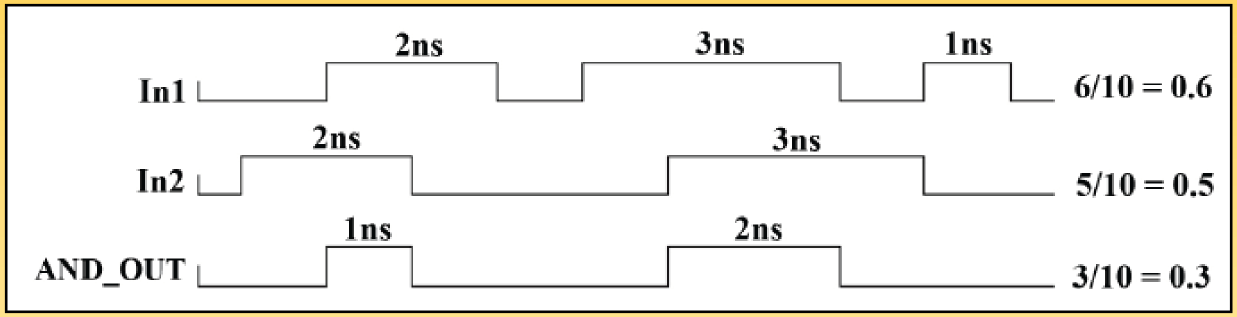


Fig 4. Stochastic multiplication using an AND with unsynchronized bit streams.

- Scaled addition

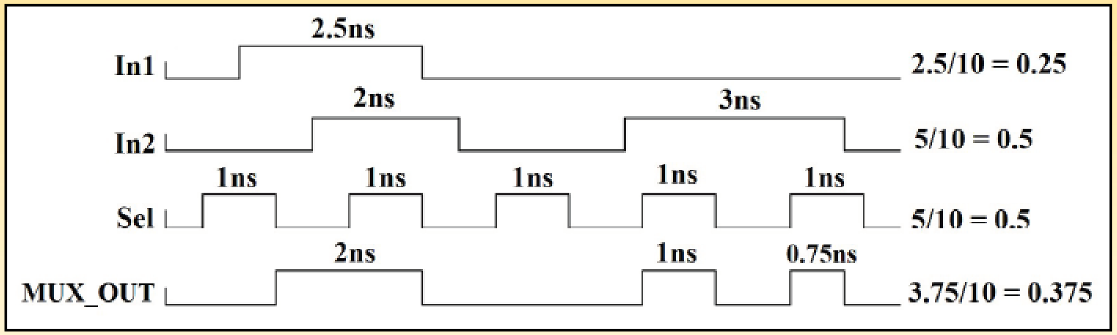


Fig 5. Stochastic scaled addition using a MUX with unsynchronized bit streams.

Background: Stochastic Computation

- **Stochastic Number Generator (SNG)**
  - To generate a stochastic bit stream with probability $\boldsymbol{x}$:
  - 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 $\boldsymbol{x}$;
  - 3. Output a **1** if $\boldsymbol{r} \geq \boldsymbol{x}$ and a **0** otherwise.

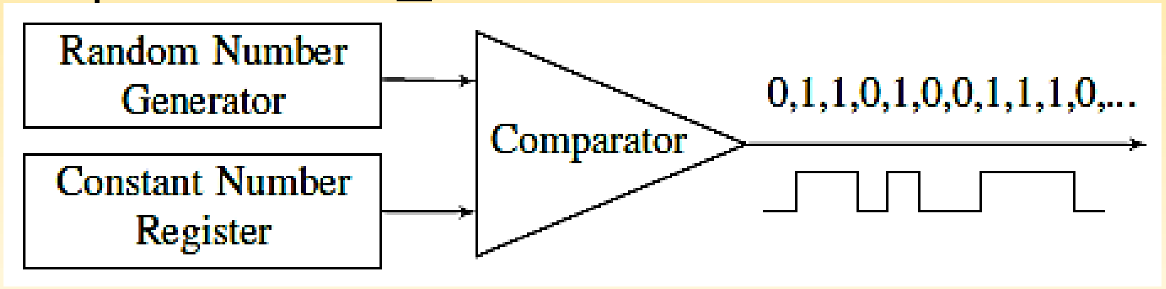


Fig 6. Stochastic Number Generator

Basic Stochastic Operations with polysynchronous inputs

- Consider an **AND gate**, responsible for **multiplying** two input bit streams, **P1** and **P2**
- **P1** and **P2** are generated by **SNGs** driven by **two clocks with different periods**, **T1** and **T2**.
- **Simple case.**
  - If we **connect two clocks** with periods of **T1** and **T2** to the inputs
  - The inputs are both representing **P=0.5**
  - Therefore, the expected output value is **Y=0.25**.
  - We verify **three different scenarios**:

    1) T1=2ns, T2=3.5ns,

    2) T1=2ns, T2=3.2ns,

    3) T1=1.8ns, T2=3.2ns.

Basic Stochastic Operations with polysynchronous inputs

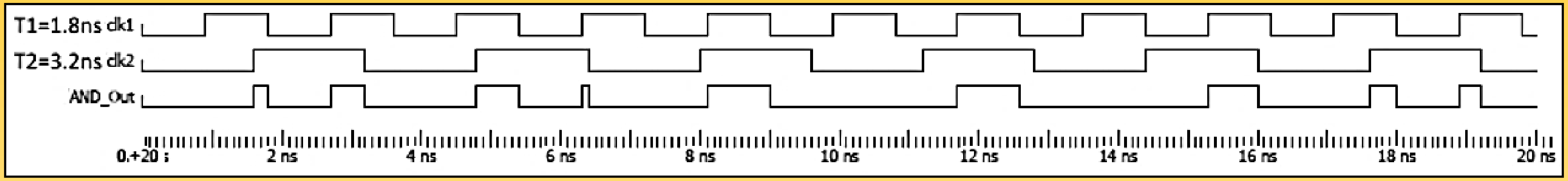
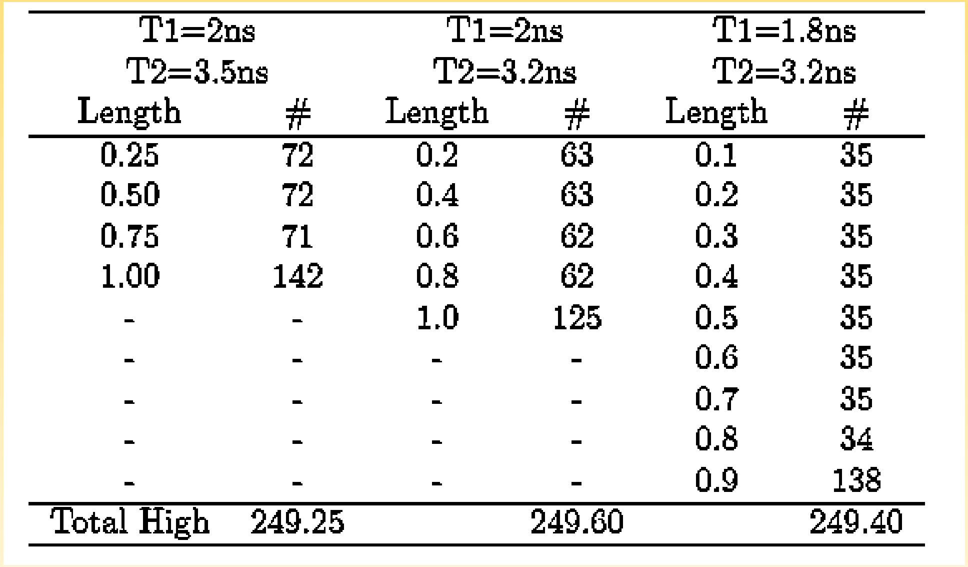


Fig 7. Input clock signals and the corresponding output from connecting polysynchronous inputs to an AND gate.

Table 1. **Different observed lengths of high pulses** at the output of the **AND gate** and the **number of occurrences of each one** for three pairs of clock periods when executing the multiplication operation for **1000ns**.



T1=2ns T2=3.5ns		T1=2ns T2=3.2ns		T1=1.8ns T2=3.2ns	
Length	#	Length	#	Length	#
0.25	72	0.2	63	0.1	35
0.50	72	0.4	63	0.2	35
0.75	71	0.6	62	0.3	35
1.00	142	0.8	62	0.4	35
-	-	1.0	125	0.5	35
-	-	-	-	0.6	35
-	-	-	-	0.7	35
-	-	-	-	0.8	34
-	-	-	-	0.9	138
Total High	249.25		249.60		249.40

- **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.**

T1(ns)	T2(ns)	T3(ns)	Total High Time	Measured Output	Expected Output
2.00	1.80	3.75	499.43	0.499	0.500
1.90	2.63	2.12	500.21	0.500	0.500
3.20	1.60	2.00	498.80	0.499	0.500
2.87	2.43	2.10	499.23	0.499	0.500

- 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.**

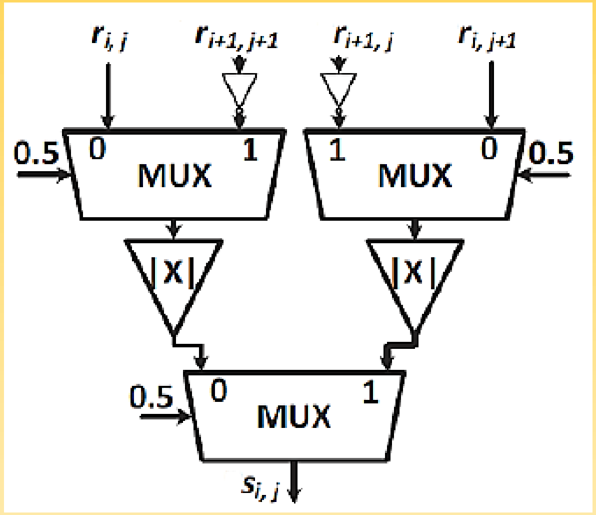
Table 3. Stochastic **multiplication** and **scaled addition**, using an AND gate and a MUX, with **inputs generated by unsynchronized** SNGs.

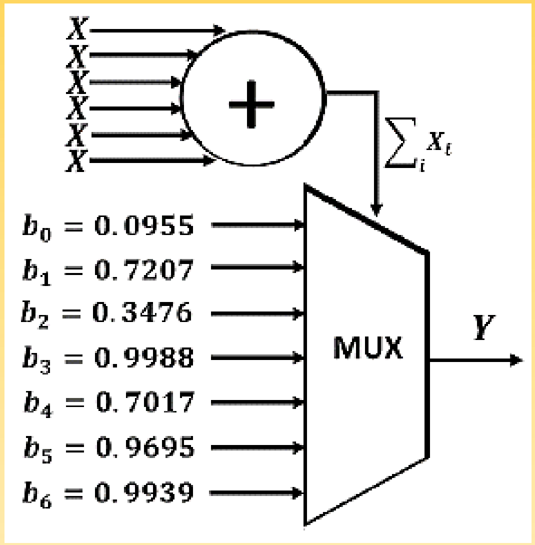
					AND Output		MUX Output	
In1	**T1(ns)**	**In2**	**T2(ns)**	**T3(ns)**	**Meas.**	**Expec.**	**Meas.**	**Expec.**
0.50	**2.10**	**0.50**	**2.30**	**2.00**	**0.247**	**0.250**	**0.502**	**0.500**
0.35	**2.82**	**0.66**	**3.11**	**3.68**	**0.237**	**0.231**	**0.498**	**0.505**
0.27	**2.81**	**0.48**	**2.36**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

- 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

- Extending the analysis to more complex stochastic circuits





$$S_{i,j} = \frac{1}{2} \times (\frac{1}{2}|r_{i,j} - r_{i+1,j+1}| + \frac{1}{2}|r_{i,j+1} - r_{i+1,j}|)$$

$$f(x) = x^{0.45}$$

Fig. 8. Stochastic implementation of the **Robert's cross edge detection** algorithm.

Fig. 9. Stochastic implementation of the **Gamma Correction** function

[Peng Li, D.J. Lilja, Weikang Qian, K. Bazargan, and M.D. Riedel. **Computation on stochastic bit streams digital image processing case studies**. TVLSI 2014.]

Stochastic Circuits with Polysynchronous Inputs

- Extending the analysis to more complex stochastic circuits

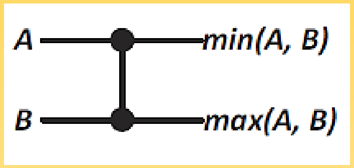


Fig. 10.a. **Basic Sorting Unit**

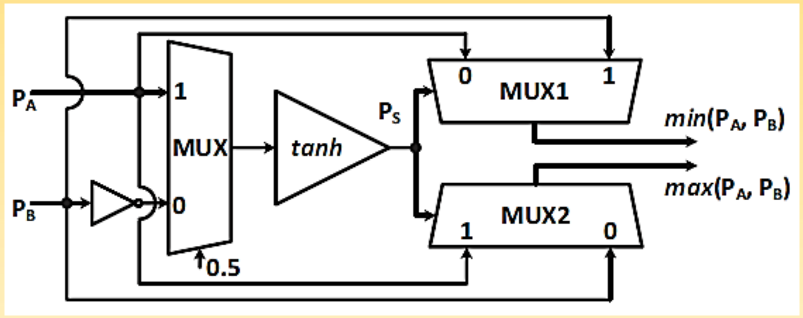


Fig. 10.b. Stochastic implementation of the basic sorting unit.

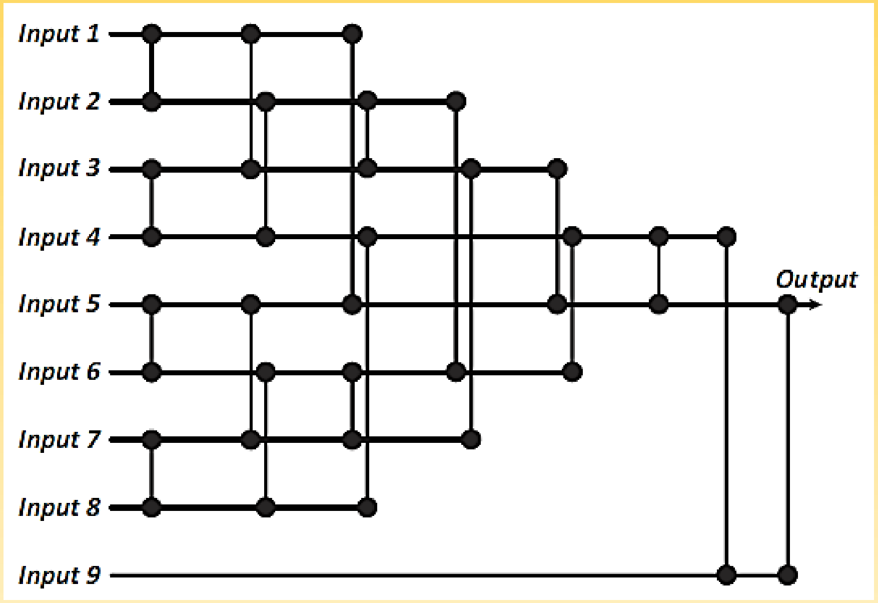
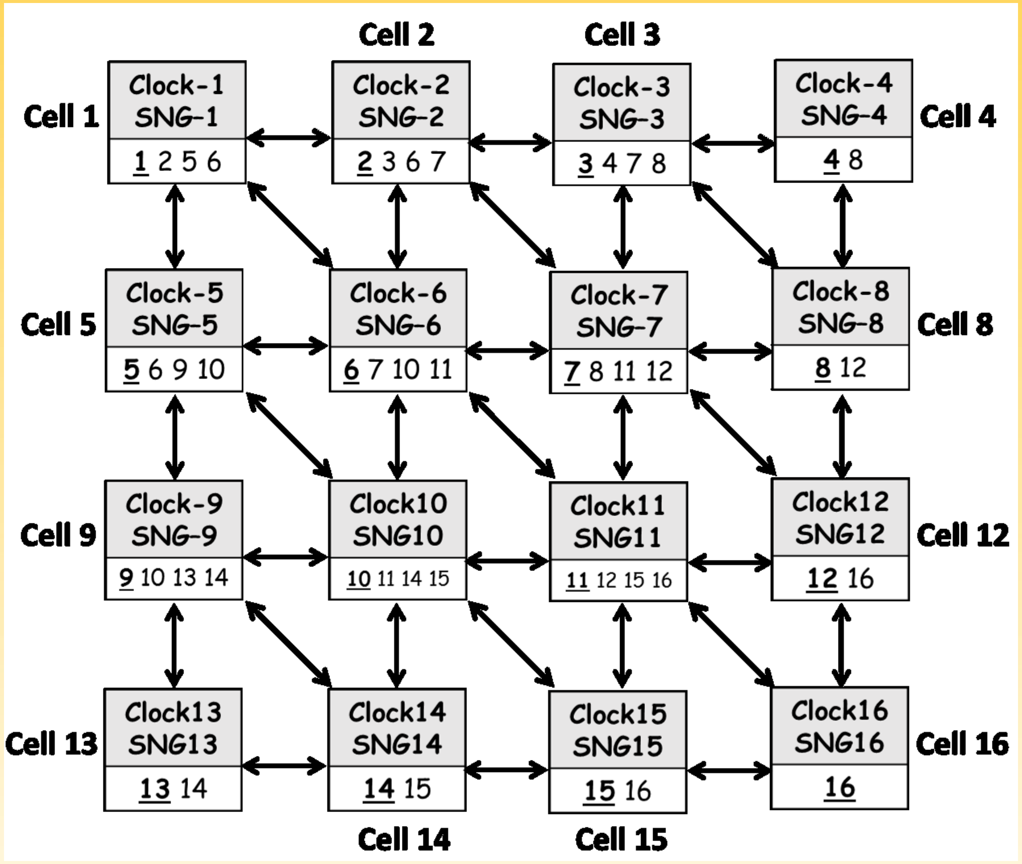


Fig. 10.c. A stochastic implementation of a **3*3 Noise reduction median filter**

[Peng Li, D.J. Lilja, Weikang Qian, K. Bazargan, and M.D. Riedel. **Computation on stochastic bit streams digital image processing case studies**. TVLSI 2014.]

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

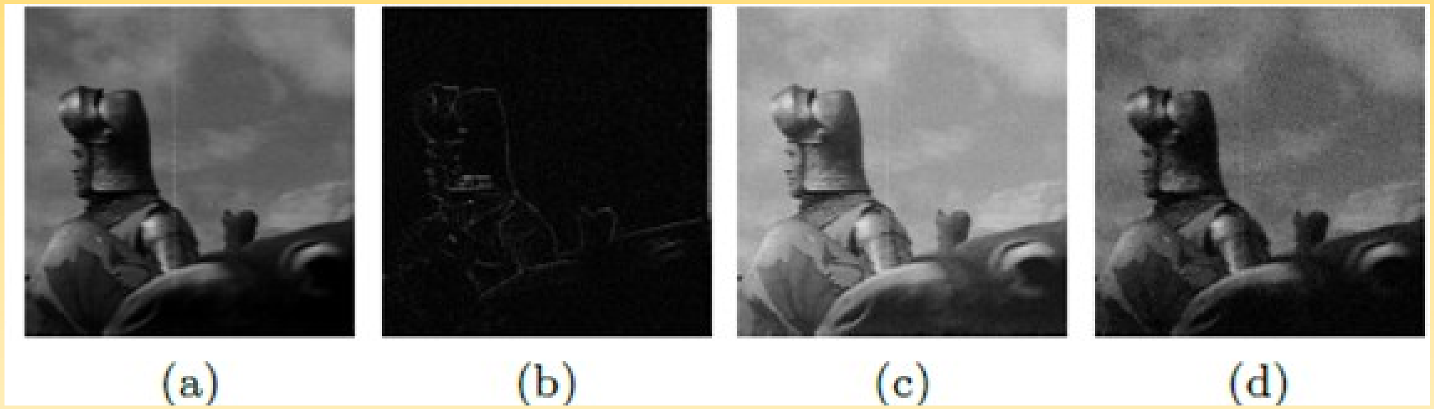
- Circuits are implemented in **Verilog,**
- Simulation in **Modelsim**
- **a 256*256 sample input image**

- 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.**



Original image **Robert's cross 2.88%** **Gamma 2.56%** **Median Filter 3.15%**

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:

$$E = \frac{\sum_{i=1}^{256}\sum_{j=1}^{256}\left|T_{i,j} - S_{i,j}\right|}{255.(256 \times 256)} \times 100$$

Experimental Results

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

	Clocking Schemes					
	S.1	**S.2**	**S.3**	**S.4**	**S.5**	**S.6**
Robert	**2.88%**	**2.89%**	**2.94%**	**2.89%**	**2.92%**	**2.88%**
Gamma..	**2.56%**	**2.50%**	**2.49%**	**2.51%**	**2.59%**	**2.64%**
Median.	**3.15%**	**3.19%**	**3.31%**	**3.28%**	**3.39%**	**3.31%**

- 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 <u>**remove the CDN**</u> and instead use
      - Locally inexpensive generated clocks
    - **Improving area, power, and design complexity**

Thank you

Questions?

M. Hassan Najafi
Najaf011@umn.edu