October 2017

MAGNETO-ELECTRIC APPROXIMATE COMPUTATIONAL FRAMEWORK FOR BAYESIAN INFERENCE

Sourabh Kulkarni
University of Massachusetts Amherst

Recommended Citation

Kulkarni, Sourabh, "MAGNETO-ELECTRIC APPROXIMATE COMPUTATIONAL FRAMEWORK FOR BAYESIAN INFERENCE" (2017). *Masters Theses*. 558.
https://scholarworks.umass.edu/masters_theses_2/558

MAGNETO-ELECTRIC APPROXIMATE COMPUTATIONAL FRAMEWORK FOR BAYESIAN INFERENCE

A Dissertation Presented

by

SOURABH S. KULKARNI

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL AND COMPUTER ENGINEERING

September 2017

Department of Electrical and Computer Engineering

MAGNETO-ELECTRIC APPROXIMATE COMPUTATIONAL FRAMEWORK FOR BAYESIAN INFERENCE

A Dissertation Presented

by

SOURABH S. KULKARNI

Approved as to style and content by:

____

Csaba Andras Moritz, Chair

____

Zlatan Aksamija, Member

____

Daniel Holcomb, Member

____

Christopher V. Hollot, Department Chair
Electrical and Computer Engineering

ABSTRACT

MAGNETO-ELECTRIC APPROXIMATE COMPUTATIONAL FRAMEWORK FOR BAYESIAN INFERENCE

SEPTEMBER 2017

SOURABH S. KULKARNI,

B.TECH., SHIVAJI UNIVERSITY, INDIA

M.S.E.C.E, UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Csaba Andras Moritz

Probabilistic graphical models like Bayesian Networks (BNs) are powerful artificial-intelligence formalisms, with similarities to cognition and higher order reasoning in the human brain. These models have been, to great success, applied to several challenging real-world applications. Use of these formalisms to a greater set of applications is impeded by the limitations of the currently used software-based implementations. New emerging-technology based circuit paradigms which leverage physical equivalence, i.e., operating directly on probabilities vs. introducing layers of abstraction, promise orders of magnitude increase in performance and efficiency of BN implementations, enabling networks with millions of random variables. While majority of applications with small network size (100s of nodes) require only single digit precision for accurate results, applications with larger size (1000s to millions of nodes) require higher precision computation. We introduce a new BN integrated circuit fabric based on mixed-signal magneto-electric circuits which perform probabilistic computations based on the principle of approximate computation. Precision scaling in this fabric is logarithmic in area vs. linear in prior directions. Results show 33x area benefit for a 0.001 precision compared to prior direction, while maintaining three orders of magnitude performance benefits vs. 100-core processor implementations.

TABLE OF CONTENTS

Page

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

The domain of artificial intelligence (AI) has seen tremendous growth of late, with new mathematical models being proposed and new areas of application being discovered frequently. Progress in this domain has been fueled by the steady growth in computational capabilities and distributed computing paradigms like GPUs. While this progress is projected to continue for several years, the underlying machinery pushing this growth is still implemented in software-based approaches over traditional computational architectures. The fact remains that the architecture over which these AI models are being implemented is largely incompatible to those models at a fundamental level. This fundamental incompatibility introduces inefficiency in the implementation of these models over software. As AI technology becomes more and more central to society, new architectures with fundamental compatibility with the AI models need to be realized to expand the scale of applications to which those models can be used.

1.1 Bayesian Networks

One of the leading approaches in AI research are Bayesian Networks (BNs). BNs are graph theoretical models which work on probabilities to enable reasoning under uncertainty. Several studies in neuroscience [3]-[6] suggest that cognition and higher order reasoning in the human brain may closely resemble Bayesian inference. The Bayesian Model has been successful in explaining several of the brain’s abilities as well as shortcomings [3] [5]. The use of BNs has also proven effective in many important real-world applications [7]-[11] e.g., gene expression, medical diagnosis, text-classification, troubleshooting, macro-finance, etc. The utility of BNs in these application domains is currently limited by the size of the networks which can be modeled in software. For some applications [7], modelling BNs with size in order of 1000s of variables requires super-computer level computational power. It is hence evident that software based BN implementations lack the efficiency and performance required to scale to larger applications. This is because BN implementations in conventional processor architectures are limited by several major issues:

(i) Software solutions involve multiple layers of abstraction to support a non-deterministic framework like BNs;

(ii) The Von-Neumann processor architecture inherently separates memory and computation introducing bottlenecks in accessing data; and

(iii) Non-volatility requirements of cognitive applications are challenging to fulfill efficiently.

It is hence evident that a new approach is required to enable applications of BNs at a larger scale.

1.2 Previous Approaches to architecting with Physical Equivalence

Research in device-level physics has led to the inventions of several emerging device technologies with unique properties. These unique properties. Emerging technology [12] based implementations [13] [14] show great promise to achieve unique benefits that enable large BNs with potentially thousands to millions of nodes, with orders of magnitude efficiency improvements vs. state-of-the-art. For example, by utilizing a new style of mixed-signal magneto-electric computation based on physical equivalence at the level of physical signals, which is a departure from traditional von Neumann computing paradigm, three orders of magnitude efficiency improvement is projected [14]. Although these new fabric architectures could enable large BNs, they do not scale well to higher resolutions. This is because, unlike the radix representation used in conventional digital designs, their flat linear representation (where a single probability value requires multiple physical signals) increases area linearly. For example, adding a digit of precision, i.e., increasing precision by tenfold, would increase area similarly by tenfold. This scaling is prohibitive for very large-scale BNs where precision would need to increase by several digits to support BN inference.

1.3 Proposed Approach for Scalable BN Architecture at Nanoscale

In this thesis, we propose a new magneto-electric BN fabric, which, while still operating on the principle of physical equivalence, provides an efficient framework of scaling computational resolution. The fabric uses non-volatile magneto-electric devices called Straintronic-Magnetic Tunneling Junctions (S-MTJs) [12], which operate at low powers and have low switching delays of ~1ns. The work in this thesis involves introduction of a new hybrid way of representing probabilities and a new approximate circuit style to perform probabilistic computation at higher computational resolution. The approximate circuit style is inspired by the principles of approximate computation [15], wherein the non-critical components of a computation are omitted to obtain better efficiency. The error resilience of BNs toward arithmetic computations performed in Bayesian nodes [16], enable us to use the approximate computation circuit style. Results show that the benefits of achieving higher resolution computation far exceed the loss in accuracy incurred due to the use of approximate techniques. The loss in accuracy is also less of a factor in the quality of the Bayesian inference [16] [17], since the increased precision achieved by scaling enables the implementation of much larger BNs incorporating a larger number of random variables which are known to be the primary factors of determining accuracy at the application level. The new fabric has 5x and 33x area reduction for computational resolutions of 0.01 and 0.001, respectively, as compared with previous emerging-technology paradigms, while it is projected to also maintain at least three orders of magnitude performance benefits over implementations on state of the art 100-core processors.

Key contributions of this thesis include:

(i) A new physically equivalent nanotechnology framework for cognitive computing applications.

(ii) A new hybrid data representation scheme with built-in error resilience for encoding probabilities into magneto-electric devices with exponentially better resolution scalability compared to previous approaches.

(iii) A new mixed-signal approximate computation circuit style to realize arithmetic operations at higher computational resolutions, enabling BN inference operations implemented using novel magneto-electric devices (S-MTJs).

(iv) Design of a reconfigurable parallel architecture based on the new scalable precision computational circuit style, consisting of distributed Bayesian computation cells, which can implement any arbitrary BN.

(v) Exhaustive evaluation of the impact of approximate computation circuit style on the error in arithmetic operations at higher resolutions.

(vi) Comparative study of errors propagating in an example million-node BN structure between low-precision exact computation and higher-precision approximate computation.

The rest of the thesis is organized as follows:

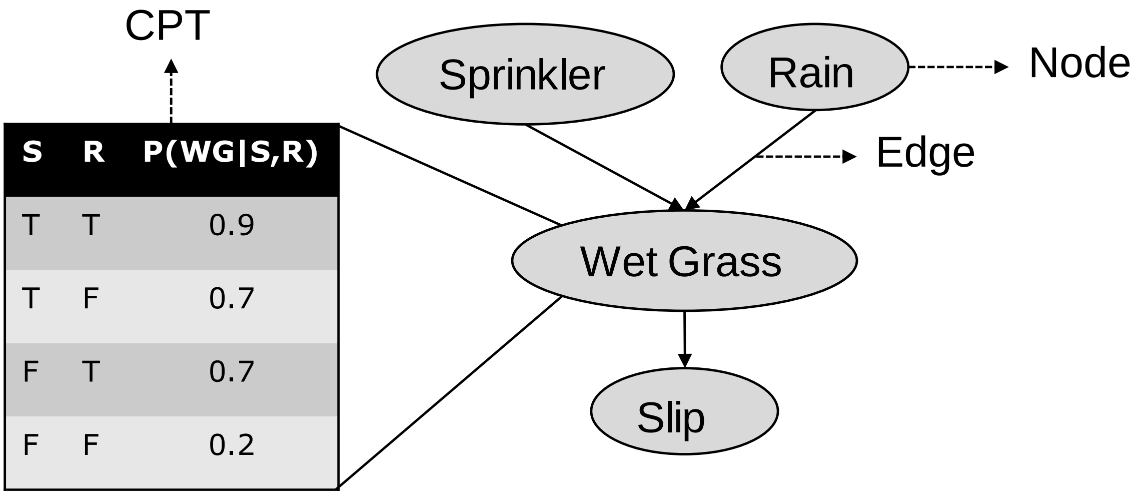
- Chapter 2 presents a brief introduction to BN and provides and overview of previous attempts toward physically equivalent nanoscale cognitive computing architectures.
- Chapter 3 shall discuss the core framework concepts, namely the S-MTJ device characteristics, the information representation scheme and the circuit framework for approximate computation.
- Chapter 4 details the reconfigurable architecture building up on the circuit framework which enables mapping of arbitrary BNs into the fabric.
- Chapter 5 discusses the proposed evaluation methodologies at circuit and architecture levels.

CHAPTER 2

BACKGROUND

2.1 Bayesian Networks overview

BNs are probabilistic graphical models [1] [2], which capture the domain knowledge in a graphical structure. They belong to the set of the modern nature-inspired probabilistic graphical models, which have shown great promise in several critical applications. In BNs, the knowledge of the qualitative relations is encoded as probabilities, which enables these models to provide reasoning under uncertainty. BNs are typically structured as Directed Acyclic Graphs (DAGs), with the nodes representing knowledge about variables in the system. The edges of the graph are directed links which represent the dependencies. A directed link from one node to the other makes the former node the parent of the latter. Each node has several variables and each variable can have multiple states. For every node, the strength of the dependency of the child node on its parents is encoded in the conditional probability table (CPT).



S	R	P(WG|S,R)
T	T	0.9
T	F	0.7
F	T	0.7
F	F	0.2

Figure 1 An intuitive example of a BN demonstrating its various elements.

Figure 1 shows an example BN. The directed arrows in the network encode the causal dependence between the nodes. For example, the probability of someone slipping on the grass is dependent on the grass being wet, which is in turn dependent on the sprinkler being on and on whether it rains.

We shall now briefly discuss the technical terms involved in the inference process of a BN. To come up with a Bayesian model of an application, the hypothesis is expressed as a set of BN variables. The BN variables are assigned initial probabilities which can either be learnt from data or determined from expert knowledge. BN inference is performed by calculating belief (the probability that a hypothesis is true based on observed evidence and state of CPT) through the process of belief update. Belief update is performed via the process of message propagation (likelihood and priors [1] [2]). Of the several algorithms used to perform belief update, we consider Pearl's Belief Propagation algorithm [1] in this work.

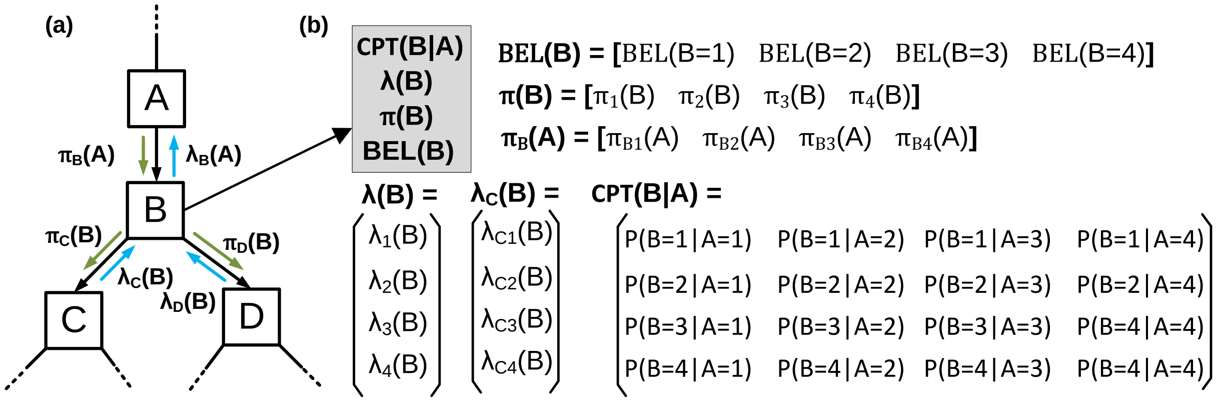

(a)
A
B
C
D
πB(A)
λB(A)
πC(B)
λC(B)
πD(B)
λD(B)
(b)
CPT(B|A)
λ(B)
π(B)
BEL(B)
BEL(B) = [BEL(B=1) BEL(B=2) BEL(B=3) BEL(B=4)]
π(B) = [π1(B) π2(B) π3(B) π4(B)]
πB(A) = [πB1(A) πB2(A) πB3(A) πB4(A)]
λ(B) =
λ1(B)
λ2(B)
λ3(B)
λ4(B)
λC(B) =
λC1(B)
λC2(B)
λC3(B)
λC4(B)
CPT(B|A) =
P(B=1|A=1) P(B=1|A=2) P(B=1|A=3) P(B=1|A=4)
P(B=2|A=1) P(B=2|A=2) P(B=2|A=3) P(B=2|A=4)
P(B=3|A=1) P(B=3|A=2) P(B=3|A=3) P(B=4|A=4)
P(B=4|A=1) P(B=4|A=2) P(B=4|A=3) P(B=4|A=4)

Figure 2. Formalizing BNs. (a) Portion of a typical BN with a node B, its parent node A and child nodes C and D; and (b) Various terminologies involved in obtaining inference for node B.

The Pearl's Belief Propagation algorithm has several operations to be performed in each node – namely, likelihood estimation, prior support and belief update. These operations are now briefly discussed. Consider a typical BN node *B* with a parent node *A* and two children nodes *C* and *D* in Figure 2. The belief of the node *B* is denoted by ***BEL(B)***, the Likelihood and prior vectors are represented as ***λ(B)*** and ***π(B)*** respectively. The ⊗ operator represents matrix multiplication, and asterisk (*) represents element-wise multiplication.

Likelihood and prior estimation for a node *B* are performed using messages from its child nodes (**λ_C(B)**, **λ_D(B)**) and parent node (**π_B(A)**) as follows:

$$\boldsymbol{\lambda}(\boldsymbol{B}) = \boldsymbol{\lambda}_{\boldsymbol{C}}(\boldsymbol{B}) * \boldsymbol{\lambda}_{\boldsymbol{D}}(\boldsymbol{B}) \qquad (1)$$

$$\boldsymbol{\pi}(\boldsymbol{X}) = \boldsymbol{\pi}_{\boldsymbol{X}}(\boldsymbol{A}) \otimes \boldsymbol{CPT}(\boldsymbol{B}|\boldsymbol{A}) \qquad (2)$$

Eq. (2), upon expansion, has a sum-of-product form. The belief update for the node *B* is done as follows:

$$\boldsymbol{BEL}(\boldsymbol{B}) = \alpha\boldsymbol{\pi}(\boldsymbol{B}) * \boldsymbol{\lambda}(\boldsymbol{B}) \qquad (3)$$

Here, *α* is a normalization constant to ensure that the result is a probability. Finally, support messages to parent node (**λ_B(A)**) and child nodes (**π_C(B), π_D(B)**) are calculated as follows:

$$\boldsymbol{\lambda}_{\boldsymbol{B}}(\boldsymbol{A}) = \boldsymbol{CPT}(\boldsymbol{B}|\boldsymbol{A}) \otimes \boldsymbol{\lambda}(\boldsymbol{B}), \qquad (4)$$

$$\boldsymbol{\pi}_{\boldsymbol{C}}(\boldsymbol{B}) = \ \alpha\boldsymbol{\pi}(\boldsymbol{X}) * \boldsymbol{\lambda}_{\boldsymbol{D}}(\boldsymbol{B}), and \qquad (5)$$

$$\boldsymbol{\pi}_{\boldsymbol{D}}(\boldsymbol{B}) = \ \alpha\boldsymbol{\pi}(\boldsymbol{B}) * \boldsymbol{\lambda}_{\boldsymbol{C}}(\boldsymbol{B}).$$

These messages help these nodes to compute their own belief updates. We can observe that all the above operations can be implemented by addition and multiplication operations. The proposed fabric hence encodes probabilities into fundamental circuit elements and implements addition, multiplication and add-multiply operations, enabling implementation of all the above Bayesian operations for Pearl's belief propagation algorithm in the fabric.

2.2 Previous Directions

The direction of using novel magneto-electric device based fabrics to design physically-equivalent architectures for cognitive computing applications is not new. Solid foundations were laid previously in this direction by some works. In [13] [14], a magneto-electric circuit based fabric was proposed, which performed Bayesian inference, by encoding probabilities, without any layers of abstractions, into device states using a flat representation scheme. Mixed-signal circuit elements performed computations with limited computational resolution of 0.1. This limited resolution proved sufficient for several real-world applications [17]. Circuit evaluations and behavioral simulations estimated area, performance and power-efficiency benefits many orders of magnitude greater than inference on state of the art 100 core processors. This speedup and efficiency meant that previously infeasible application sizes could now be implemented on this framework to yield results in a reasonable timeframe.

However, higher order error propagation studies performed on those architectures, where million-node networks were simulated, showed that the results obtained had errors high enough to make them unreliable. It was hence evident that, in million-mode networks, to obtain results within the generally-accepted precision of 0.1, the actual computations would be needed to be done at higher computational resolutions. The flat information representation scheme scaled poorly with computational resolution – for example - while a 0.1 resolution employed 10 devices per variable, a 0.01 resolution would require 100 devices per variable; translating to a 10x increase in area and power. It is hence evident that a new fabric architecture with better scaling of computational precision would greatly increase the scope and utility of these directions in cognitive computing applications.

CHAPTER 3

CORE FRAMEWORK CONCEPTS

The framework builds on the concept of physical equivalence – the probabilities in the BN are encoded in an information representation scheme, which is implemented in a mixed signal magneto-electric circuit style. Efficient scalability of computational resolution is achieved by selective use of approximate computation. The computation circuits are designed such that the inaccuracies incurred by using approximate computing have a much lower impact than the increase in accuracy due to performing computation at a higher resolution.

3.1 Information Representation Scheme

Conventionally, information in digital circuits has been represented in a radix format. This style performs well in all digital circuits, and provides for an efficient way to scale precision. However, when we consider radix representation for mixed-signal architectures, which employ analog domain to perform computations, and rely on devices that are stochastic switches with much lower reliability than MOS counterparts, the lack of error resilience is prohibitive. This is since any single bit error would make an entire computation potentially useless.

An alternative approach, considered in [13] [14], involves a flat information representation scheme, which is more suitable for mixed signal architectures. This approach assigns the same 'weight' to each digit in the representation and the value represented is simply the sum of the values in the individual digits. Although this provides

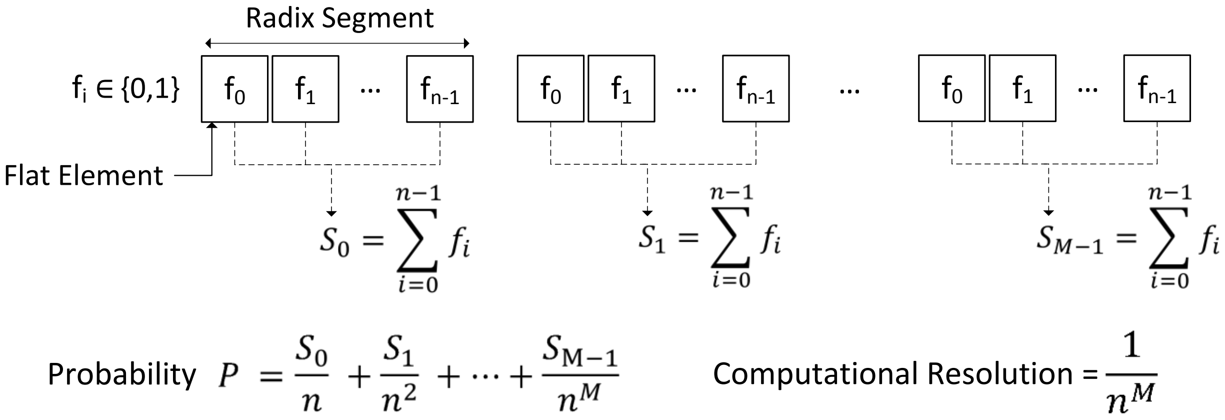

Radix Segment
$f_i \in \{0,1\}$
f_0 f_1 ⋯ f_{n-1}
⋯
Flat Element
$S_0 = \sum_{i=0}^{n-1} f_i$
$S_1 = \sum_{i=0}^{n-1} f_i$
$S_{M-1} = \sum_{i=0}^{n-1} f_i$
Probability $P = \frac{S_0}{n} + \frac{S_1}{n^2} + \cdots + \frac{S_{M-1}}{n^M}$
Computational Resolution = $\frac{1}{n^M}$

Figure 3. Flat-Radix information representation scheme. Probability is encoded in segments which are in a radix arrangement, with each segment containing flat elements. The equivalent probability encoding and the computational resolution obtained are also listed.

error resilience and can perform efficient analog computations, scaling to higher computational resolutions is very inefficient.

We propose a more generalized information representation scheme, which combines the scalability of the radix representation and the error resilience of the flat representation. This new 'Flat-Radix' representation consists of *M* segments in a radix arrangement with *n* elements each in a flat arrangement as shown in Figure 3. Within a single segment, all the flat elements f_i contribute equally toward the value of the segment, making it error resilient. A segment S_J represents a value equal to the sum of all the flat elements f_i in the segment. Because of the radix nature of the segments, a segment S_J has a value *n* times the value of segment S_{J+1}, providing efficient scalability. The probability is represented by the segments arranged in a radix format with base *n*. This new representation opens a spectrum of possible representation states with flat representation at one end (*M*=1) and radix representation at the other (*n*=1). This vast range of possible configurations is attractive as the representation scheme could be potentially tailored to match the application precision requirement.

Consider an example configuration of the scheme with *n*=10 and *M*=2. This corresponds to a representation with two segments in a radix representation with each containing 10 flat elements. The effective computational resolution of this configuration is 0.01. Now if we change *M* to 3, the configuration now has an effective resolution of 0.001. This corresponds to an exponential increase in computational resolution with linear increase in number of devices. It is important to note however, that some of the computations performed are approximate in nature and the increase in computational resolution doesn't directly translate to increased accuracy in computation. This phenomenon is discussed in detail in further sections.

3.2 Technology Overview: Straintronic MTJs

The work utilizes Straintronic Magnetic Tunnel Junctions (S-MTJs) [12] as the underlying physical device for hardware implementation. It is to be noted, though, that the information representation scheme discussed earlier can be realized using any other non-volatile technology. The work also assumes that the devices have two stable states, but the proposed scheme works for multistate devices as well as infinite persistence-state devices like the memristor. In the case of an ideal infinite persistence-state device, the information representation scheme shall collapse into a single element, which can store probability values with arbitrary precision.

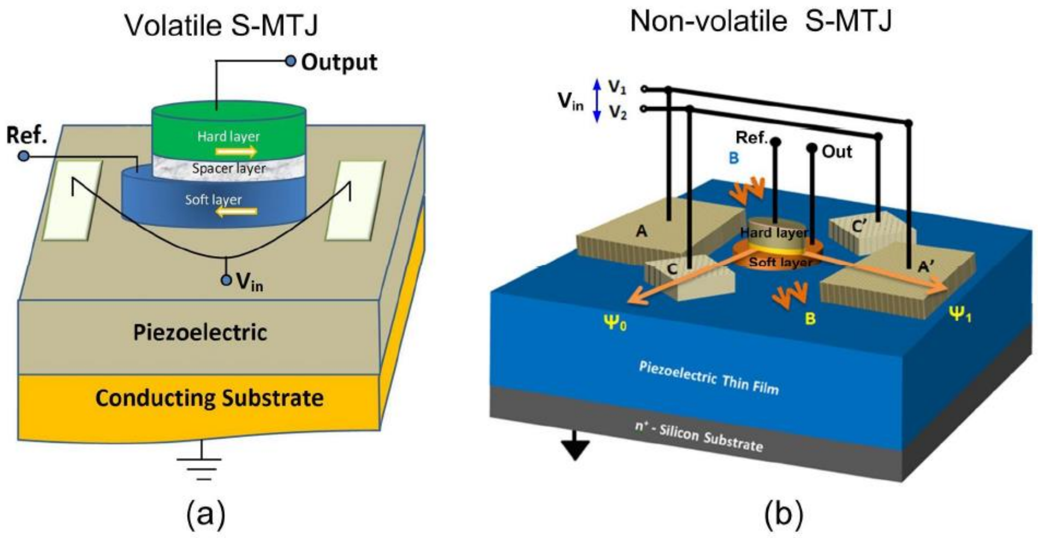


Figure 4. Device configurations (recreated with permission from [13]). (a) Volatile S-MTJ. Application of a voltage generates a strain in the soft-layer, modifying the electrical resistance; (b) Non-Volatile S-MTJ. Functioning similar to Volatile S-MTJ, but has two pairs of electrodes between which the magnetic orientation flips.

The concept of straintronics, where the bi-stable magnetization of a shape anisotropic multiferroic nanomagnet is switched with electrically generated mechanical strain, is attractive due to its extreme low energy of switching. A S-MTJ device consists of three layers – a “hard” ferromagnetic layer with a fixed magnetization orientation, an ultrathin spacer layer, and a "soft" ferromagnetic layer with variable magnetization orientation.

The device configurations are shown in Figure 4. The S-MTJ has two device variants, namely the Volatile and Non-Volatile S-MTJ. In the Volatile S-MTJ, the soft and hard layers are naturally aligned into anti-parallel state. A voltage induced strain changes the magnetization of the soft layer, changing the resistance of the device. In the Non-

Volatile S-MTJ, the soft layer is constrained to be in two states by using two pairs of electrodes. If left in one of these states, the magnetization remains in that orientation indefinitely, which makes the device non-volatile. The orientation is switched by applying a voltage induced strain on the soft layer. Although the work considers two stable states, the number of sable states can be changed by changing the cross-sectional shape of the device.

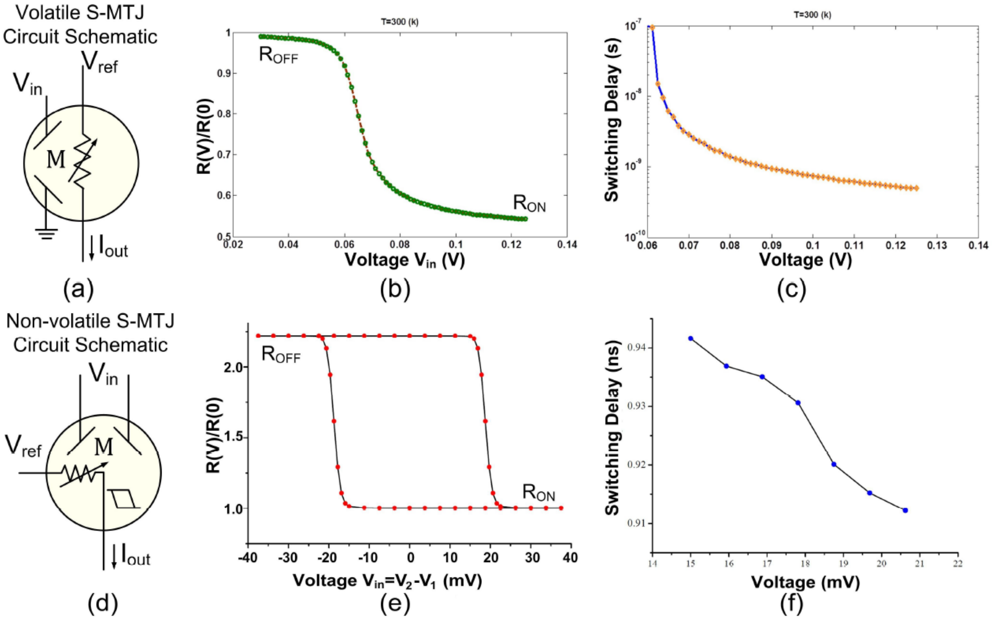


Figure 5. (a) Schematic of Volatile S-MTJ; (b) Simulated DC Characteristics of Volatile S-MTJ; (c) Simulated switching delay of Volatile S-MTJ; (d) Schematic of a Non-Volatile S-MTJ; (e) Simulated DC Characteristics of Non-Volatile S-SMTJs; (f) Simulated switching delay of Non-Volatile S-MTJs. (recreated with permission from[13])

Figure 5 details the schematic, DC charateristics and the switching delay simulations of both the Volatile and Non-Volatile S-MTJs. The hysteresis loop of the Non-volatile S-MTJs is characteristic of its persistence. The switching delay graphs show that both devices are capable of sub-nanosecond switching times with proper applied voltage.

3.3 Probability Representation based on S-MTJs

We shall now discuss how the flat-radix information representation scheme is implemented using the S-MTJ devices. Each flat element in each of the segment of the flat-radix probability representation is encoded in the Non-Volatile S-MTJ. Assuming two stable states, this translates to High resistance state encoding 0 and low resistance state encoding 1.

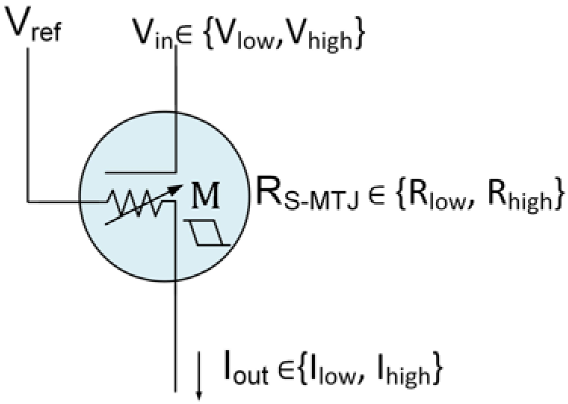

V_{ref}
$V_{in} \in \{V_{low}, V_{high}\}$
M
$R_{S\text{-}MTJ} \in \{R_{low}, R_{high}\}$
$I_{out} \in \{I_{low}, I_{high}\}$

Figure 6. A flat element represented by a Non-Volatile MTJ.

This concept is represented diagrammatically in Figure 6. The voltage V_{in} can control the resistance state of the device. A reference voltage V_{ref} is applied to read out the resistance state R_{S_MTJ} of the device, resulting in output current I_{out}. Building on this concept, the segments can be composed of an array of these flat elements. Each element in the segment can be provided with the same V_{ref} and their individual resistance states are controlled by the individual input voltages. The currents from the flat elements are then summed together the segments can be arranged in a radix format to come up with segment-level representations of the probability. These segments arranged in a radix structure complete the flat-radix probability representation.

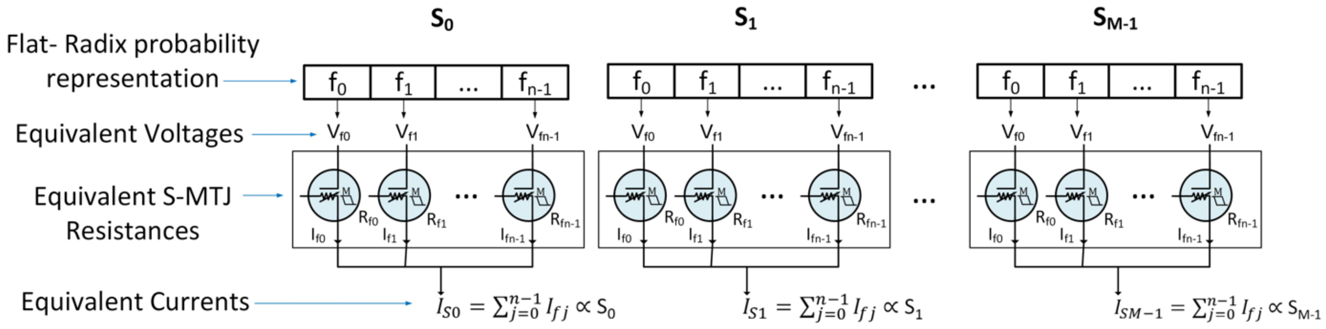

S0
S1
SM-1
Flat- Radix probability representation
Equivalent Voltages
Equivalent S-MTJ Resistances
Equivalent Currents
f0
f1
...
fn-1
Vf0
Vf1
Vfn-1
Rf0
Rf1
Rfn-1
If0
If1
Ifn-1
M
$I_{S0} = \sum_{j=0}^{n-1} I_{fj} \propto S_0$
$I_{S1} = \sum_{j=0}^{n-1} I_{fj} \propto S_1$
$I_{SM-1} = \sum_{j=0}^{n-1} I_{fj} \propto S_{M-1}$

Figure 7. Physically equivalent encoding of probability using the proposed information representation scheme into S-MTJ circuits.

Figure 7 shows how the flat-radix probability encoding process using the S-MTJ devices. The total probability *P* is interpreted by considering the segments in a radix arrangement. The base of this radix structure is based on the number of flat elements in each segment.

Although the representation has radix segments, the computations remain atomic to the flat portions, as there is no notion of radices when computing in analog domain. Using approximate computing when required, we design circuits that bring together the flat computations into a radix representation. The approximate computing techniques (discussed in further subsections) have a lower error bound; the inaccuracies in computation can never be greater than the computations in a representation with a single radix segment (i.e., *M*=1). This case occurs when all the radix segments except the first one have value of 0 and essentially get excluded from the computation.

3.4 Segment-Level Circuit Framework

The flat-radix representation scheme provides a natural way to encode probabilities and in the process, enables unique ways to perform computation. In this architecture, the computation operations like addition and multiplication are consequences of circuit design, in contrast to traditional computation schemes in which there are complex arithmetic units dedicated to perform these computations. The addition operation is performed by the simple summation of currents by the Kirchhoff's current law and the multiplication is performed through the application of the Ohm's law. The computation circuits start from simple adder and multiplier circuits operating on individual flat segments and a hierarchical encapsulation of these circuits performs higher order functionality, i.e., the flat-radix adders and multipliers.

The computational elements operating on flat segments are primarily composed of two components:

(i) a composer which converts input data stored in from of resistances of the S-MTJs in current domain and performs the computation on the current; and

(ii) a decomposer which converts the computed result back into the format which can be stored into the S-MTJs. The composer circuits are discussed first and the decomposer will be described later.

The design of these circuits is based on the arithmetic composer circuits discussed in [13][14]. The segment-level composers, to some extent, resemble the flat composers mentioned in earlier works.

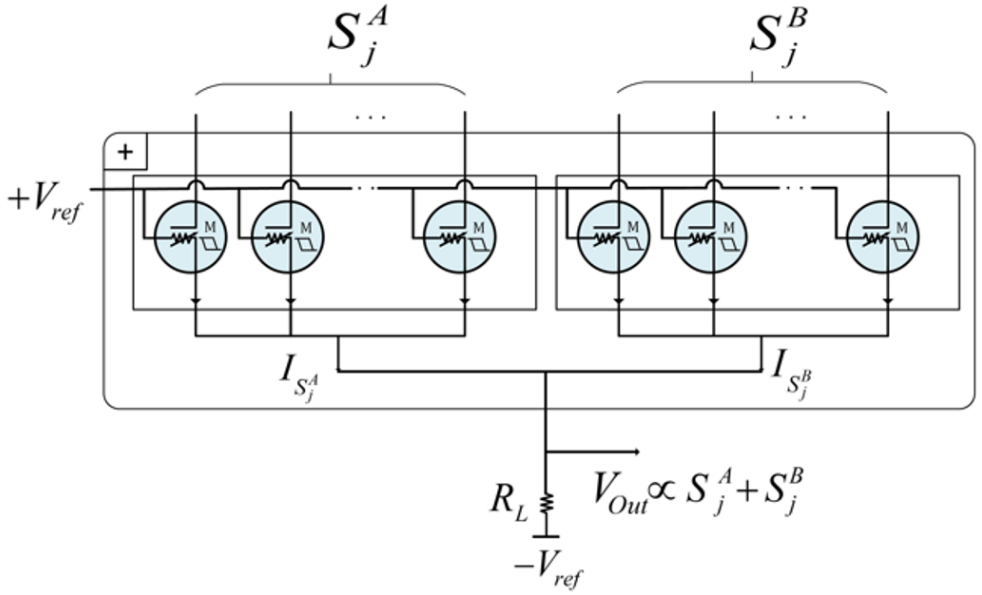


Figure 8. Schematic of a segment-level addition composer. *V_{out}* is proportional to the sum of input currents.

Figure 8 shows a schematic of a segment-level addition composer. Current addition can be implemented by using a parallel configuration of individual segment composers. In a manner, similar to how the currents in individual segments is summed together, the current outputs of the two segments is summed together. The individual current summations of the two segments are shorted together to obtain the current addition between segments. To convert the output from current domain to voltage domain, a load resistor is used. The obtained voltage V_{out} is hence proportional to the addition of the input two segments. This obtained voltage is passed on further to either further segment-level composer circuits. For the sake of brevity, the addition segment-level composer is henceforth referred to as the addition SLC.

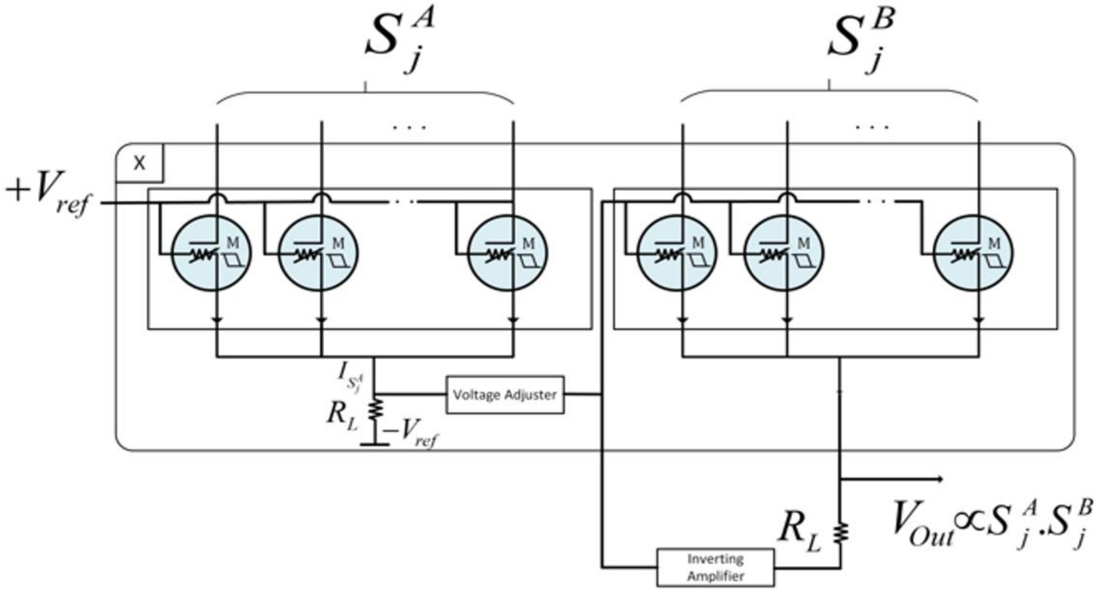


Figure 9. Schematic of the multiplication segment-level composer. V_{out} is proportional to the product of input segments.

Figure 9 shows a schematic of the multiplication segment-level composer. In this setup, the current output from one of the input segment is converted to voltage domain using load resistance as discussed in the addition SLC circuit. This voltage is then fed to a voltage adjustment circuit, which is principally an amplifier. This circuit is configured to provide a voltage output equal to V_{ref} when it receives a voltage input corresponding to the maximum value of the segment, i.e., when all elements in the segment are in state corresponding to 1 or logic ‘high’. All other segment configurations have a proportionately lower voltage outputs. This amplified (or adjusted) voltage is then applied as the reference voltage to the second input segment. This setup results in a final output current which is proportional to the product of the two input segments. This current is converted into voltage domain by another load resistor. For the sake of brevity, the multiplication segment-level composer is abbreviated as multiplication SLC.

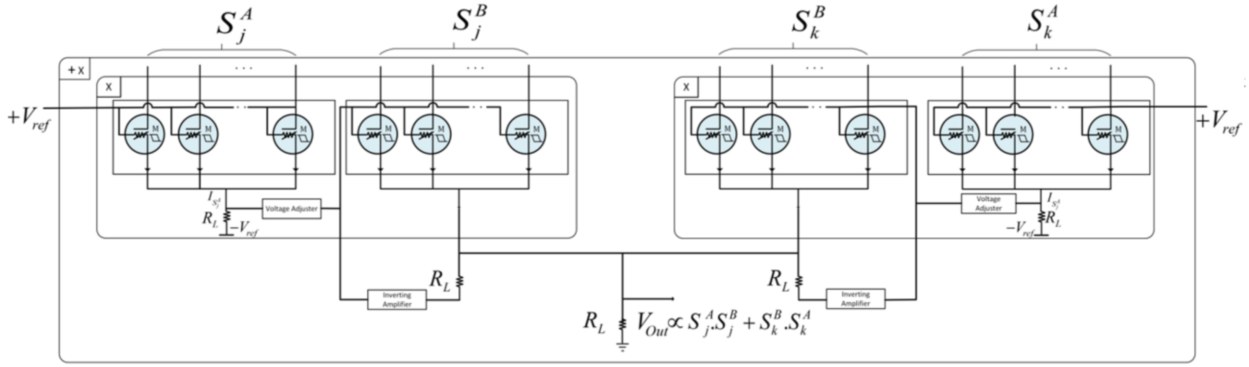


Figure 10. Schematic of the add-multiply segment-level composer, a hierarchical combination of the addition and multiplication SLCs which implements a sum-of-products operation.

Figure 10 shows the schematic of the add-multiply segment-level composer. Each product term implemented with a multiplication SLC is arranged in a topology of addition SLC. It is a hierarchical combination of the addition and multiplication SLCs. The multiplication SLCs are the ‘internal’ composer, i.e., the two multiplications occur in the first stage. The outputs of these are then summed together by the ‘external’ addition composer. The resulting output is proportional to the sum-of-product of the inputs.

These segment-level composers form the building blocks of flat-radix composer which perform the higher-resolution computations. This follows the convolutional hierarchical trend of increasing complexity, while the fundamental computation still occurs on probability values. The addition flat-radix composer and the multiplication flat-radix composer are hierarchical combination of the addition SLC, multiplication SLC and the add-multiply SLC. The add-multiply flat-radix composer, going higher up the level of hierarchy, builds on multiplication flat-radix composers and addition flat-radix composers.

To summarize:

(i) Segment level composer circuits are the fundamental computation units; higher order operations are performed by cascaded version of these circuits.

(ii) Segment-level information is condensed to currents to perform computations

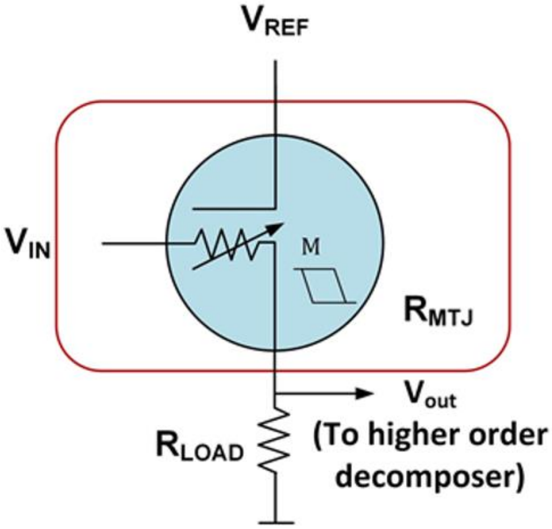


Figure 11. Carry circuit which detects carry generation in a lower order SLC and transfers it to higher order SLC.

Figure 11 shows a schematic of the carry circuit used in flat-radix composers. In any radix-based information representation scheme, there is a notion of an order of individual segments, and that of carry propagation between them. The flat-radix composers are made using a combination of multiple SLCs and the carry circuits communicate the carry generated in lower order SLC to higher order SLC. With all the required components detailed, we shall now discuss the flat-radix composer circuits.

3.5 Flat-Radix Addition Circuit Design

Consider two probabilities A and B, represented in the flat-radix scheme as follows:

$$A = \left(\frac{1}{n}\right) S_0^A + \left(\frac{1}{n^2}\right) S_1^A + \cdots + \left(\frac{1}{n^M}\right) S_{M-1}^A \qquad (6)$$

$$B = \left(\frac{1}{n}\right) S_0^B + \left(\frac{1}{n^2}\right) S_1^B + \cdots + \left(\frac{1}{n^M}\right) S_{M-1}^B \qquad (7)$$

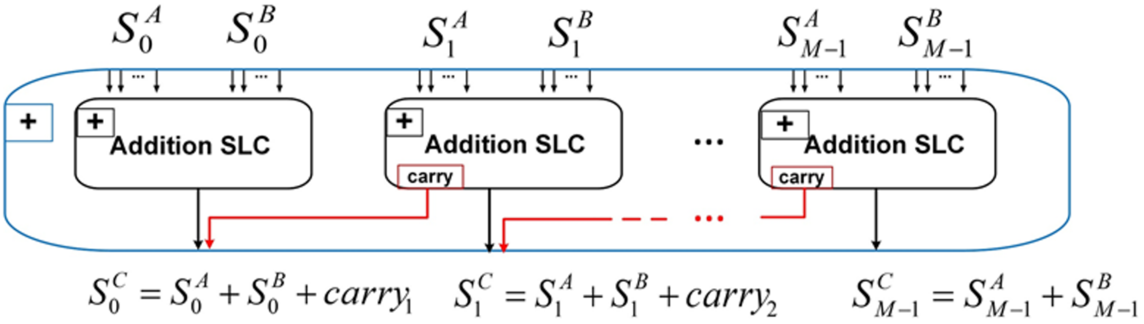
The addition operation in the new representation scheme is done using the following formula:

$$A + B =$$
$$\left(\frac{1}{n}\right) (S_0^A + S_0^B + carry_1) + \left(\frac{1}{n^2}\right) (S_1^A + S_1^B + carry_2) + \cdots$$
$$+ \left(\frac{1}{n^M}\right) (S_{M-1}^A + S_{M-1}^B) \qquad (8)$$

In implementing the above formula in-circuit, each individual addition term is computed in its own segment using the addition SLC, while the carry generated by a segment is propagated to the segment one order higher, using the previously described carry circuit.

Figure 12 shows the circuit schematic of the addition flat-radix composer. This design makes it very efficient to scale to higher resolution just by adding more segments to the hybrid representation. This implementation is an exact expression of addition formula hence the calculations have no approximation involved. This accuracy is achieved because in an addition operation, the causality flows strictly in one direction (lower-order to higher-order) in the form of carry; i.e., a lower order term may affect the result of a higher order calculation (through carry), but a higher order term cannot affect the result of a lower order calculation. For brevity, the flat-radix composers shall henceforth be referred to as FRCs.

3.6 Flat-Radix Approximate Multiplication Circuit Design



S_0^A
S_0^B
S_1^A
S_1^B
S_{M-1}^A
S_{M-1}^B
+
+
Addition SLC
+
Addition SLC
carry
+
Addition SLC
carry
$S_0^C = S_0^A + S_0^B + carry_1$
$S_1^C = S_1^A + S_1^B + carry_2$
$S_{M-1}^C = S_{M-1}^A + S_{M-1}^B$

Figure 12. Addition flat-radix composer, comprising of several addition SLCs and carry circuits.

The flat-radix computation scheme utilizes approximate computation techniques in the FRC multiplier. Consider the sum-of-segments form of probabilities A and B from equations (6) and (7). A multiplication operation between these would be:

$$A.B =$$

$$\left(\left(\frac{1}{n}\right)S_0^A + \left(\frac{1}{n^2}\right)S_1^A + \cdots + \left(\frac{1}{n^M}\right)S_{M-1}^A\right).\left(\left(\frac{1}{n}\right)S_0^B + \left(\frac{1}{n^2}\right)S_1^B + \cdots + \left(\frac{1}{n^M}\right)S_{M-1}^B\right) \quad (9)$$

Upon expanding equation (9), we end up with a much higher number of terms; in this case, a multiplication of two probabilities with *m* terms each would result in m^2 terms. The error resilient nature of BNs [17], whose quality of inference is dependent more on the graph structure and random variable selection than the accuracy of the arithmetic operations, enable us to optimize the design of the multiplier circuit. To optimize the circuit, we include only the top *m* contributing terms of the expansion. The effect of performing this optimization on the accuracy of the multiplier circuit is discussed in the evaluation chapter. Using this optimization process, we design an approximate multiplication formula which maximizes efficiency while attempting to minimize approximation error:

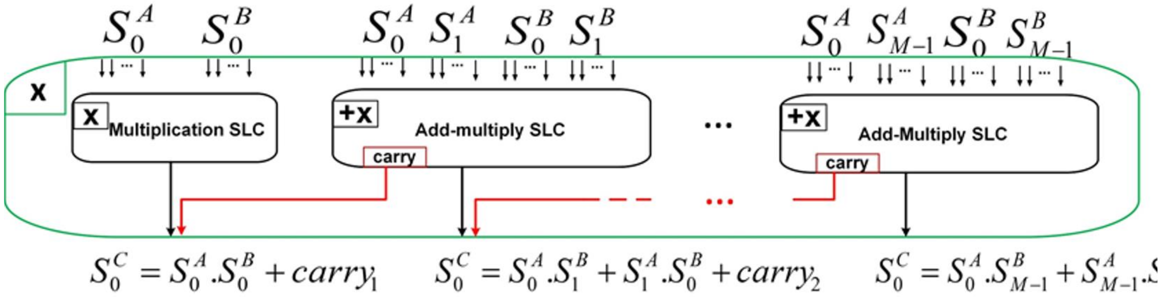

S_0^A S_0^B
S_0^A S_1^A S_0^B S_1^B
S_0^A S_{M-1}^A S_0^B S_{M-1}^B
X
X Multiplication SLC
+X Add-multiply SLC
carry
+X Add-Multiply SLC
carry
$S_0^C = S_0^A.S_0^B + carry_1$
$S_0^C = S_0^A.S_1^B + S_1^A.S_0^B + carry_2$
$S_0^C = S_0^A.S_{M-1}^B + S_{M-1}^A.$

Figure 13. Multiplication FRC made up of one multiplication SLC and several Add-Multiply SLCs and carry circuits

$$A.B = \left(\frac{1}{n}\right)(S_0^A.S_0^B + carry_1) + \left(\frac{1}{n^2}\right)(S_0^A.S_1^B + S_1^A.S_0^B + carry_2) + \cdots +$$

$$\left(\frac{1}{n^2}\right)(S_0^A.S_{M-1}^B +$$

$S^A_{M-1}.S^B_0$)PA.PB=PAH+PAL.PBH+PBL=(PAHPBH)+(PAHPBL+PBHPAL)+(PALPBL

) (10)

Figure 13 shows the circuit diagram implementation of equation (10). This formulation is approximate, as it omits the equation terms, which contribute the least to the end-result. These terms are the results of partial multiplications of lower order segments. The omission of these terms is an optimization, resulting from the application of the concept of approximate computing in this case. This approximate multiplier has the worst-case performance when all the segments except the highest order one are zero. In this worst-case, the FRC performs equally to a flat representation with an equivalent of a single SLC.

Exhaustive MATLAB simulations, discussed in evaluation section suggest that including additional segment-level computation terms to equation 10 yield no significant improvement to the accuracy of the approximate computation scheme.

3.7 Flat-Radix Approximate Add-Multiply Circuit Design

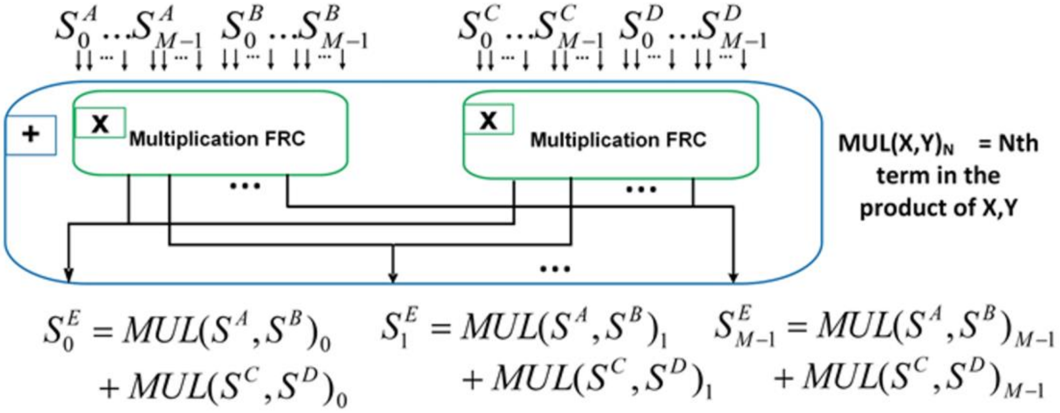

$S_0^A \ldots S_{M-1}^A$
$S_0^B \ldots S_{M-1}^B$
$S_0^C \ldots S_{M-1}^C$
$S_0^D \ldots S_{M-1}^D$
+
X
Multiplication FRC
X
Multiplication FRC
MUL(X,Y)$_N$ = Nth term in the product of X,Y
$S_0^E = MUL(S^A,S^B)_0 + MUL(S^C,S^D)_0$
$S_1^E = MUL(S^A,S^B)_1 + MUL(S^C,S^D)_1$
$S_{M-1}^E = MUL(S^A,S^B)_{M-1} + MUL(S^C,S^D)_{M-1}$

Figure 14. Add-Multiply FRC made from a hierarchical combination of Multiplication FRC and Addition FRC

Following the trend of hierarchical arrangement to make more complicated circuits, the add-multiply FRC is composed of two Multiplication FRCs and a Addition FRC. The individual segments from each of the multiplication FRCs corresponding to the same radix order are added together by ‘plugging in’ to the input of the FRC adder’s corresponding order of radix segment. The equations corresponding to these computations are shown in Figure 14.

These circuits perform two parts of the computation process, viz., taking in digital signals and composing them into analog currents, and performing computation on those currents to achieve basic probability arithmetic. To complete the architecture, we need decomposers to convert the obtained analog result into digital, non-volatile S-MTJ states. This is done using the decomposer circuits.

3.8 Decomposer Circuit Design

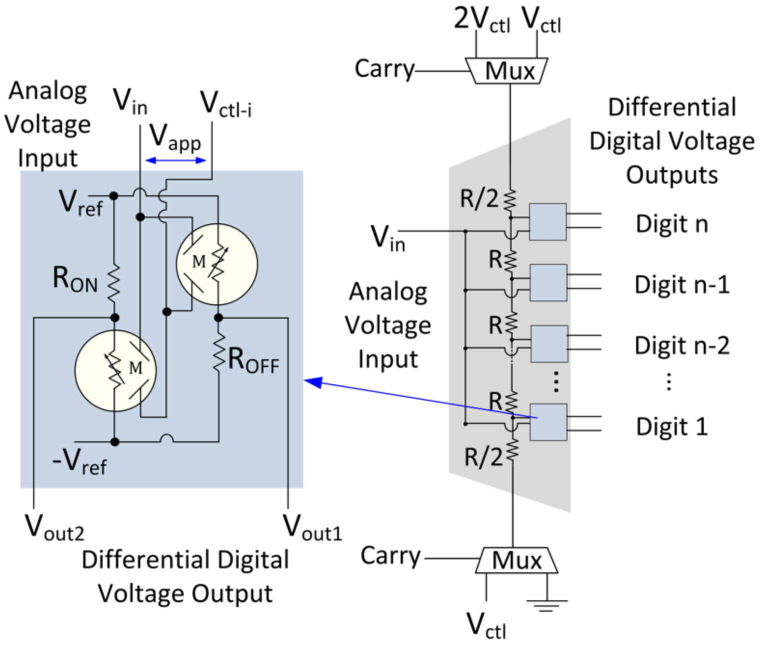


Figure 15. Segment-level Decomposer schematic.

The final aspect of the circuit-level design of this fabric is the decomposer circuit, which converts the computed results back to the flat-radix format at segment level for further computations. This circuit is designed for devices with two states. The basic design of the segment-level decomposer uses a R-2R ladder circuit commonly used to convert continuous voltages in discrete states. The voltage V_{IN} is compared in several steps to V_{ctl} using MTJ-based comparators shown in Figure 15, where each comparator from top to bottom compares V_{IN} with increasingly smaller fractions of V_{ctl}, which is set to be equal to the voltage that corresponds to all the flat elements of a segment being high. Hence, the number of comparators, which output ‘high’ will be proportional to the ratio of V_{IN} to V_{ctl}. All the segments sans the first can have a maximum value equal to twice the max capacity of a single segment. To correctly decompose values greater than V_{ctl} we design a carry-based mechanism to switch the comparator voltage levels as shown in Figure 15. The carry signal is generated by the carry circuits mentioned earlier. The carry signal, if high, changes the comparator voltage levels appropriately.

To summarize, we have now designed a scalable precision, non-volatile, S-MTJ based, flat-radix, mixed-signal computation framework with approximate computing. The creation of a reconfigurable BN architecture using this computation framework shall now be discussed in the next chapter.

CHAPTER 4

SCALABLE PRECISION MAGNETO-ELECTRIC ARCHITECTURE FOR BAYESIAN INFERENCE

In this chapter, we discuss the realization of the Bayesian inference procedure based on Pearl's belief propagation algorithm was described in the chapter 2, using the scalable precision magneto-electric computation framework discussed in chapter 3. The proposed architecture supports up to 4 random variables per Bayesian node. The architecture will be structured as follows:

(i) 5 Bayesian computation units are designed using FRC composer circuits.

(ii) The Bayesian cell, which is a hierarchical combination of the 5 Bayesian computation units, is designed, which functions as a programmable node in a BN.

(iii) An FPGA-style programmable switchbox cell is designed to perform as the edges of connectivity in a BN.

(iv) A uniform array of the Bayesian Cells and the switchboxes forms the overall architecture.

These architectural components are now discussed in detail, starting from the bottom-most all the way up to the top.

4.1 Bayesian Computation Units

Continuing the path of hierarchical combination of circuits, we design 5 computation units, each implementing equations (1) -(5), discussed in chapter 2, on the probabilities in Flat-Radix representation using combination of several FRCs, discussed in chapter 3.

$$\boldsymbol{BEL}(\boldsymbol{X}) = \alpha\boldsymbol{\pi}(\boldsymbol{X}) * \boldsymbol{\lambda}(\boldsymbol{X}).$$

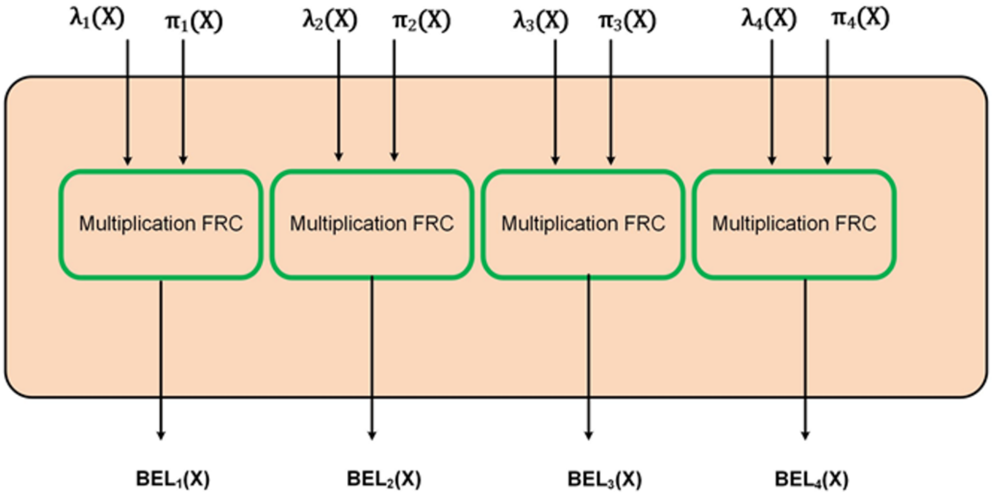


Figure 16. One of the 5 computation units of a Bayesian Cell - the 'Belief Update' computation unit, comprising of 4 Multiplication FRCs

Consider, for example, the belief update operation in Bayesian inference. It comprises of 4 multiplication operations, one for each of the 4 supported random variables per node. The computations are performed by Multiplication FRC circuits. Figure 16 shows a simplistic overview of the arrangement. Similar computation units are designed for all the remaining 4 computation operations required by the Bayesian node using the FRC circuits.

4.2 Bayesian Cell and Programmable Switchboxes

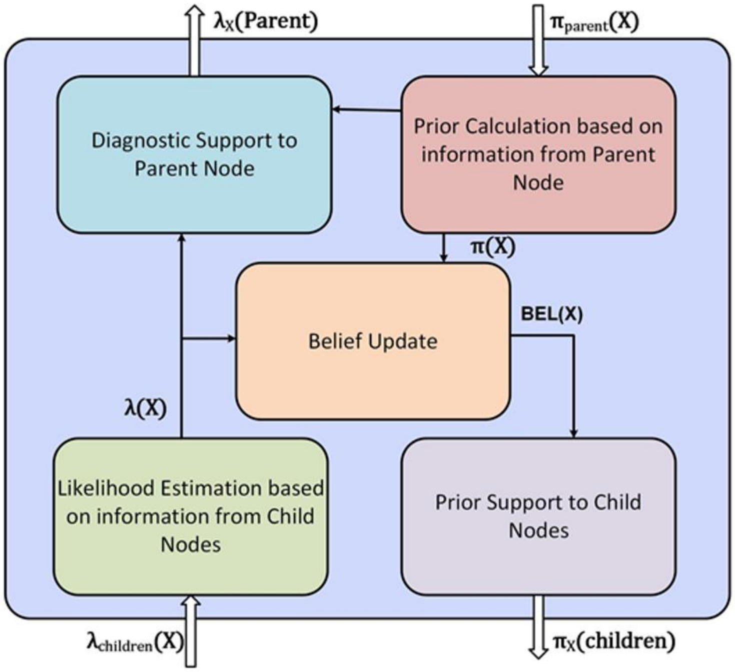

λ_X(Parent)
π_{parent}(X)
Diagnostic Support to Parent Node
Prior Calculation based on information from Parent Node
π(X)
Belief Update
BEL(X)
λ(X)
Likelihood Estimation based on information from Child Nodes
Prior Support to Child Nodes
$\lambda_{children}$(X)
π_X(children)

Figure 17. Block diagram view of a Bayesian Cell, a hierarchical combination of the 5 computation units which perform all the calculations required in a Bayesian node for inference.

Figure 17 shows how all the 5 computation units operate together to function as a Bayesian Cell. Briefly, the computations occurring in the Bayesian cell are as follows:

(i) Likelihood computation is using the diagnostic support from children nodes;

(ii) Priors are computed using prior support from parent node;

(iii) Computed Likelihood and Priors are used to perform belief update;

(iv) Computed Likelihood is used to provide diagnostic support to Parent node; and

(v) Computed Belief is used to provide prior support to children nodes.

Each of these operations takes place in each Bayesian cell, while the messages required for these computations are passed between nodes through programmable switchboxes.

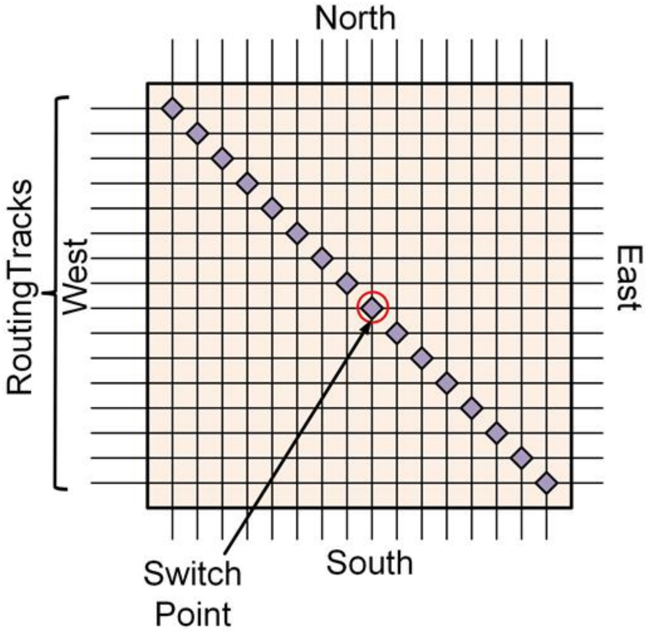


Figure 18. Schematic of a programmable switchbox

Figure 18 shows the schematic of a programmable switchbox. The design of this switchbox is similar to the ones commonly used in FPGA architectures. The messages through the network are sets of probability vectors associated with diagnostic support for bottom-up and prior support for top-down messages and the propagation supported is through switch-boxes. In our example, if each node supports 4 states, then each of these messages contains 4 sets of probability vectors. Thus, each switch-box has to accommodate sufficient switch-points to allow transmission of all the elements of probability vector sets in parallel.

4.3 Hierarchical Summary of Overall Architecture

With all the elements of the architecture described, we shall now provide a top-down summary to visualize all the architectural elements into a single framework.

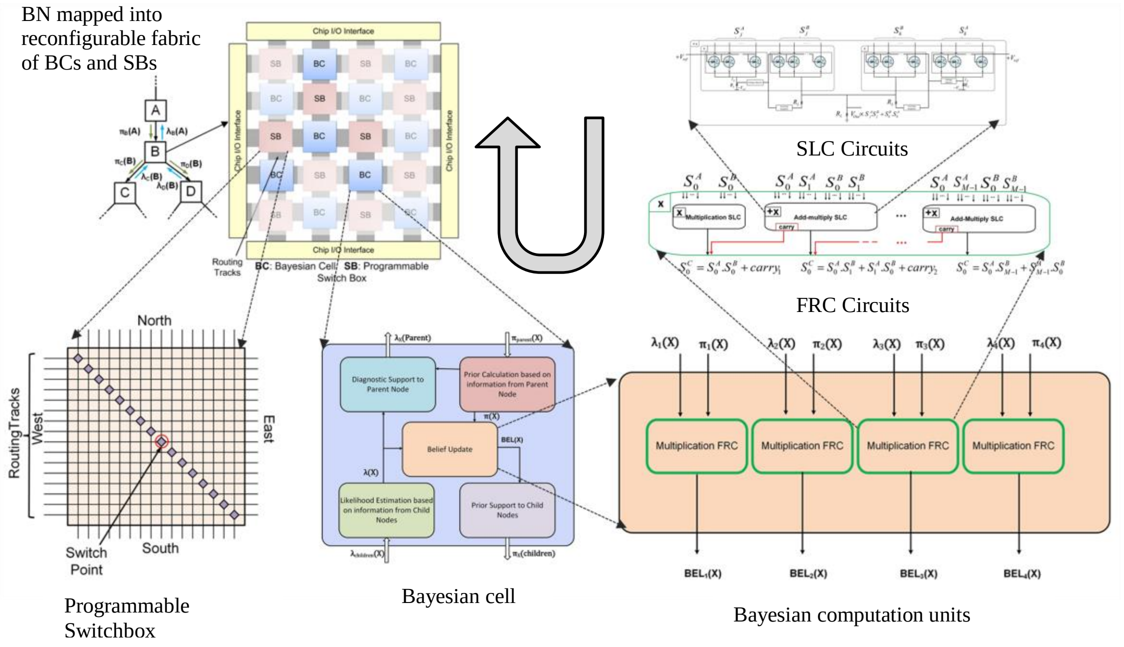


Figure 19. All the architectural elements summarized into a single framework.

BNs are mapped into the reconfigurable fabric made up of Bayesian Cells as nodes and Switchboxes as edges. Switchboxes are FPGA-styled connection mechanisms facilitating connectivity between Bayesian Cells. The connections are made possible by cross points between the north-south and east-west wires, with programmable switch points. Bayesian Cells are made from 5 Bayesian computation units, each performing one of the 5 computations essential for the Pearl's belief propagation algorithm. Figure 19 shows one such Bayesian computation unit, which is the one performing belief update computation. Bayesian computation units are combinations of the flat-radix composer circuits. The flat-radix composer circuits are combinations of segment-level composer and decomposer circuits. The belief update computation unit comprises of multiplication FRCs as the belief update computation is a multiplication operation. The FRC are hierarchical combinations of the SLCs. The multiplication FRC is made up of a combination of multiplication SLC and add-multiply SLC. SLC are based on the S-MTJ devices, which compose probabilities encoded in resistance states to currents which are then used for computation.

While there are several levels of hierarchy, the computations that occur are still directly on probabilities, tied to the physical level. The hierarchy introduces complexity of design, without introducing the performance loss which is commonly associated with digital systems which have layers of abstraction accompanying the layers of hierarchy. Hence, the proposed architecture manages the same level of level of complexity in operation (although only for a single application) as the digital, von Neumann counterpart, but does that without the overhead associated with adding layers of abstraction.

CHAPTER 5

EVALUATION

We shall now discuss the evaluation methodologies used to validate the circuit designs of all the elements of the architecture. The MTJ devices central to this computational framework are known to exhibit certain switching error. The impact of the stochastic nature of the switching of the MTJs on the result of the computations was studied in [13]. It is shown that, due to the error resilient nature of the flat representation scheme, which is utilized by the flat-radix scheme used in this work within each flat segment, the inaccuracies related to errors in the arithmetic operations far exceed the errors due to the switching errors due to the MTJs. In both cases the errors are gracefully tolerated, due to carefully designed information representation schemes. The chapter is arranged as follows:

(i) S-MTJ HSPICE Macromodel

(ii) Segment level circuit simulation in HSPICE

(iii) Area, power and performance estimation of proposed architecture along with comparisons to previous works.

(iv) Exhaustive error evaluation for FRC circuits

(v) Higher-level simulation for error propagation comparison between previous work and proposed work.

5.1 S-MTJ HSPICE Macromodel

To evaluate the proposed circuit framework in HSPICE, we develop HSPICE behavioral device macromodel of the S-MTJ. This model was developed in conjunction with the VCU group[12].

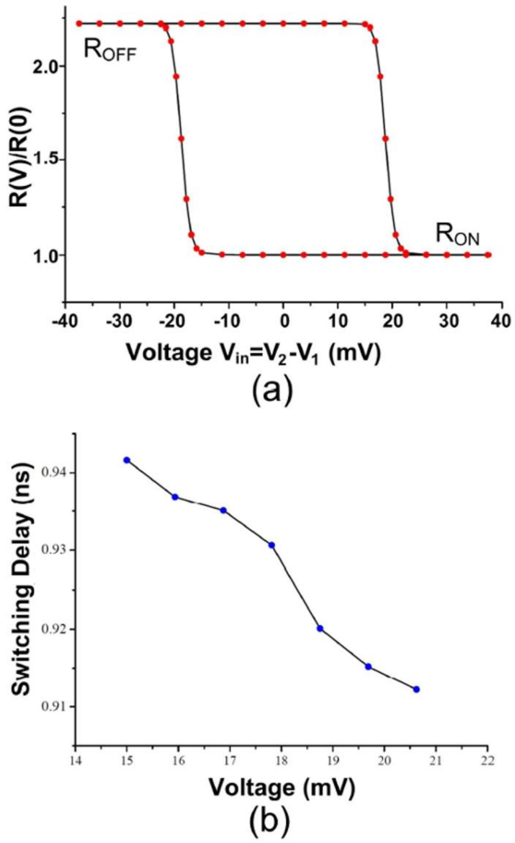


Figure 20. Simulated DC characteristics for S-MTJ[12]. (a) Resistance vs. input voltage showing two stable resistance states and switching threshold voltages; and (b) Switching delay vs. input voltage.

The S-MTJ characteristics are shown in Figure 20. HSPICE offer several behavioral constructs to model these characteristics such as voltage/current controlled sources. For modelling S-MTJs, voltage controlled resistors (VCRs) are used. Two such VCRs were used, one to model switching behavior from low resistance to high resistance state, and another for modelling switching behavior from high resistance state to low resistance state. Each VCR is connected in series with ideal switches; only one of them is active at a time, and the active switch selects the VCR for the given operating condition. The decision logic takes the current inputs and previous state of device (stored in flip-flop) and determines the new state of the device. To model the switching delay, custom voltage-

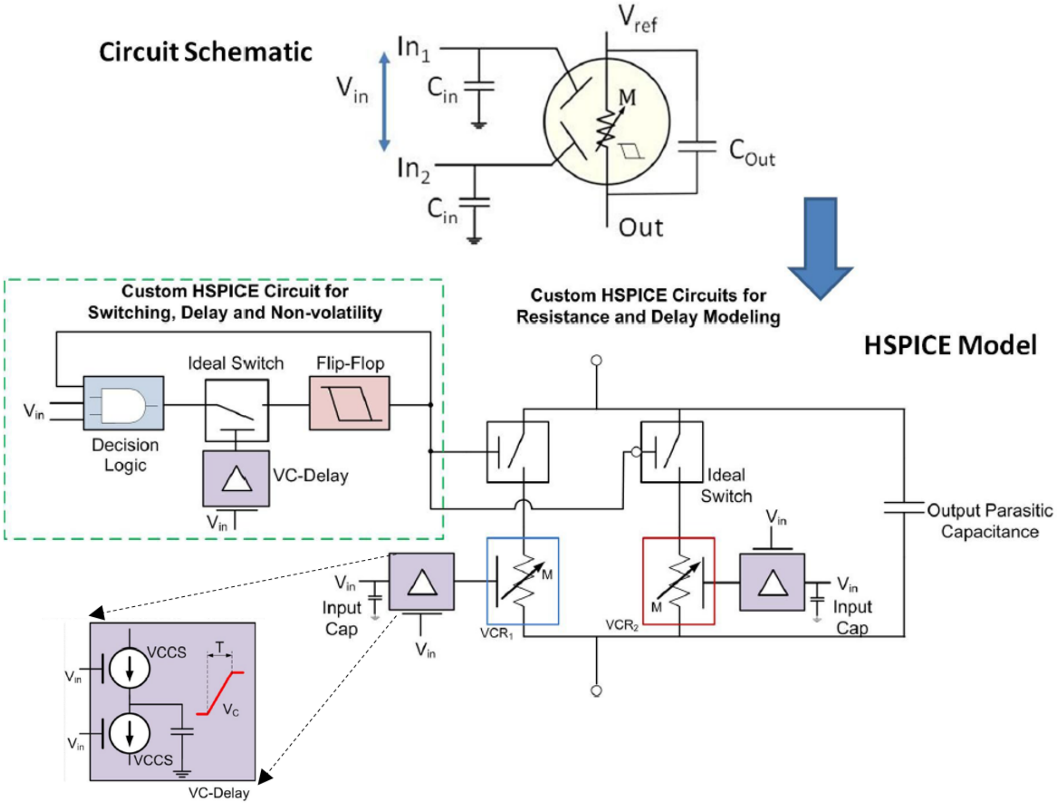


Figure 21. HSPICE behavioral macromodel describing S-MTJ device characteristics for circuit simulation.

controlled delay elements were inserted, which comprised of voltage-controlled current sources(VCCS) and capacitances.

Figure 21 shows the S-MTJ device micromodel schematic. The macromodel is used to validate the functionality of the computation circuits and to estimate the area, power and performance metrics of the proposed architecture.

5.2 Segment Level Circuit Simulation in HSPICE

We discuss the functionality of the segment-level composer circuits and validate segment-level computation and decomposer circuits.

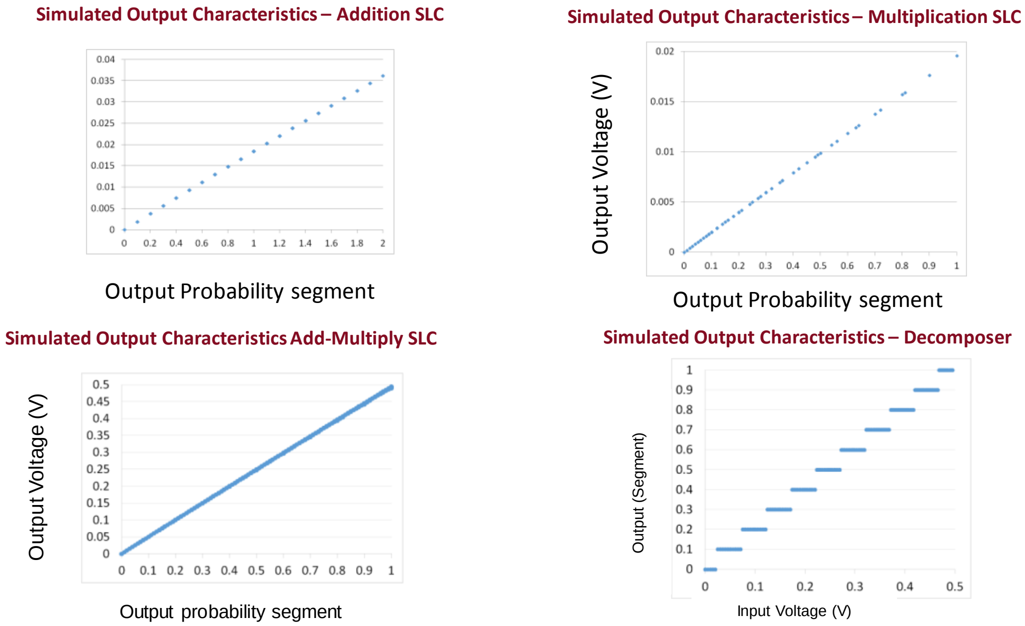


Figure 22. Segment-level circuit validation with HSPICE for Addition. Multiplication, Add-multiply computation, and decomposer circuits.

In Figure 22, the linear trend of the voltage which follows the expected probability segment value it is meant to represent indicates expected computational behavior of the circuits. The simulations validate all possible segment-level input combinations to all the circuits. The CMOS support circuitry is designed assuming 45nm technology node. The simulations also yield the power, delay and area data which is used to estimate these parameters for larger circuits like the Bayesian Cell. These metrics are discussed in the next subsection.

5.3 Area, Power and Performance

Metrics (Worst case)	Resolution	MUL	ADDMUL	OPAMPs	Decomposers
Area(μm^2)	0.1	5	17	95.4	240
	0.01	21	42	190.8	480
	0.001	39	78	286.2	720
	0.0001	56	112	381.6	960
Power(μW)	0.1	1.15	2.81	89.32	11.37
	0.01	3.96	7.92	178.64	22.74
	0.001	6.77	13.54	267.96	34.11
	0.0001	9.58	19.16	357.28	45.48
Delay(ns)	0.1	144	137	100	132.9
	0.01	144	144	100	132.9
	0.001	144	144	100	132.9
	0.0001	144	144	100	132.9

Table 1. The area, power and delay of FRCs and other circuit components for various computational resolutions.

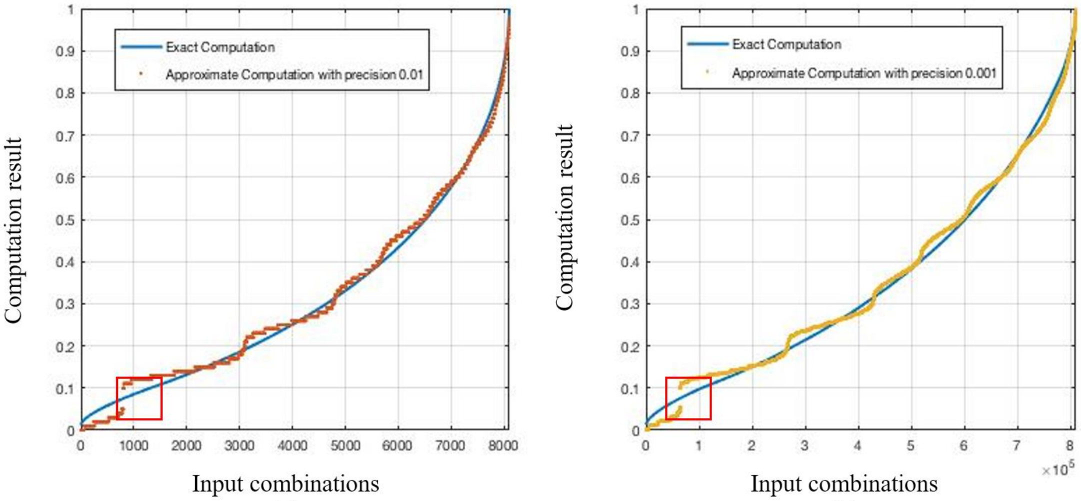
Table 1 shows the area, power and delay values for flat-radix architecture schemes with computational resolutions from 0.1 (1 segment) to 0.0001 (4 segments). The results suggest up to 30x area and power benefits of this architecture over previous approaches[14] for a computational resolution of 0.01. The area and power benefits vs. these approaches grow exponentially with resolution.

In [14], the performance of a physically equivalent magneto-electric Bayesian inference system (with a computational resolution of 0.1) was compared with a 100-core processor assuming best-case idealized processor performance. This comparison yielded a 6,000x performance benefit for Bayesian inference over the 100-core processor. For the same resolution, assuming a flat organization with one segment, the delay characteristics of the circuit framework described in this paper matches that of [14], and hence similar performance relative to 100-core processors is expected.

For a precision of 0.001, on the other hand, combining three flat segments into a 10-3 flat-radix magneto-electric framework (10 MTJs for encoding a probability, in each of the three radix segments), this architecture could still maintain a 2,000x benefit while the original architecture would have lost most of its performance, power, and area advantages. A further 1,000x improvement (i.e., a 0.000001) in precision would require 6 segments arrangement and would maintain a 1,000x improvement. Any single error in a segment would affect that segment by a tenth of a precision. A higher error resiliency may combine 100 MTJs in each segment: e.g., a 100-6 encoding would yield an even more graceful degradation would an error occur in any segment due to say an MTJ not switching correctly, for a 10x additional area impact (and 10x additional precision). The overall computational error would depend on the segment position in a radix, but highest error would occur when highest segment (in radix order) is affected. This calculation does not yet account for any approximate calculations that will be discussed below.

5.4 Approximation Errors in FRCs

We shall now observe the accuracy tradeoffs of the flat-radix approach vs. the flat scaling implemented in [14] by comparing the accuracy performance. These comparisons are done by generating behavioral models of the multiplication composers in MATLAB, as exhaustive hardware simulations in HSPICE are infeasible to be done in reasonable time and have convergence issues due to large number of devices and input combinations involved. These exhaustive simulations considered all possible input combinations in each case.



(a) (b)

Figure 23. The computational accuracy of Multiplication FRCs with precision 0.01 (10-2 flat-radix with 20 devices per probability) in (a) and 0.001 (10-3 flat-radix with 30 devices per probability) in (b) as compared to the accuracy of 0.01 precision flat-only scheme with 100 devices per probability, and 0.001 flat-only scheme with 1000 devices per probability, respectively. The regions highlighted in red indicate the maximum error of 0.1 due to the approximate nature of the computations. The plot displays the outputs for all possible input combinations sorted in ascending order.

Figure 23 contains two plots that compare the multiplication operation with scaled flat[13][14] and flat-radix approaches, with 0.01 and 0.001 resolution respectively. The plots were generated by calculating the results for all possible input combinations. From the plots, it is evident that the substantial savings in area and power (5x for 0.01 and 30x for 0.001) are obtained by the flat-radix approach.

Computational Resolution	No. of devices	Mean error	Error variance	Max. Error	% Input combinations with max error
0.1(Previous work)	10	0.065	0.006	0.1	10%
0.01	20	0.027	0.000097	0.1	0.01%
0.001	30	0.0037	0.000023	0.1	0.00001%

Table 2. Precision comparison between low resolution accurate computation and higher resolution approximate computation.

With a 2x and 3x increase in number of devices, computations at higher resolutions, although approximate, yield lower mean errors (~2.4x and ~17x) and orders of magnitude lower error variance (~61x and ~260x) as shown in Table II. The plot also shows where the new computation system has the highest error of 0.1 (highlighted in red) for precision levels. Although the approximate computation scheme has a worst-case error of 0.1, at higher precision levels, the percentage of input combinations that lead to the maximum error is very low (0.01% and 0.00001%). MATLAB simulations suggest accuracy benefit is obtained by including additional intermediate terms to the computation. In case of 10-5 configuration (5 segments with 10 elements each), with effective resolution of 0.00001, the mean error without intermediate terms is 0.00039 and the mean error without those terms is 0.00037.

Furthermore, in the context of Bayesian inference, it has been shown that the quality of inference of a BN depends primarily on the structure of the graph and the number of random variables captured accurately, while the numerical precision required in the arithmetic computations plays a secondary role [17]. Although the computation scheme described in this paper has a maximum error of 0.1, even for higher resolutions, the infrequent occurrence of the high error case is unlikely to affect the outcome of the BN inference; also at the application level accuracy is considered as the likelihood of correct prediction in a belief across a large input measurement set vs. individual inference. On the other hand, the increased precision obtained, along with significantly lower mean error and error variance, will allow for much larger BNs to be implemented in this architecture at a low area cost.

5.5 High-level Error-propagation simulation

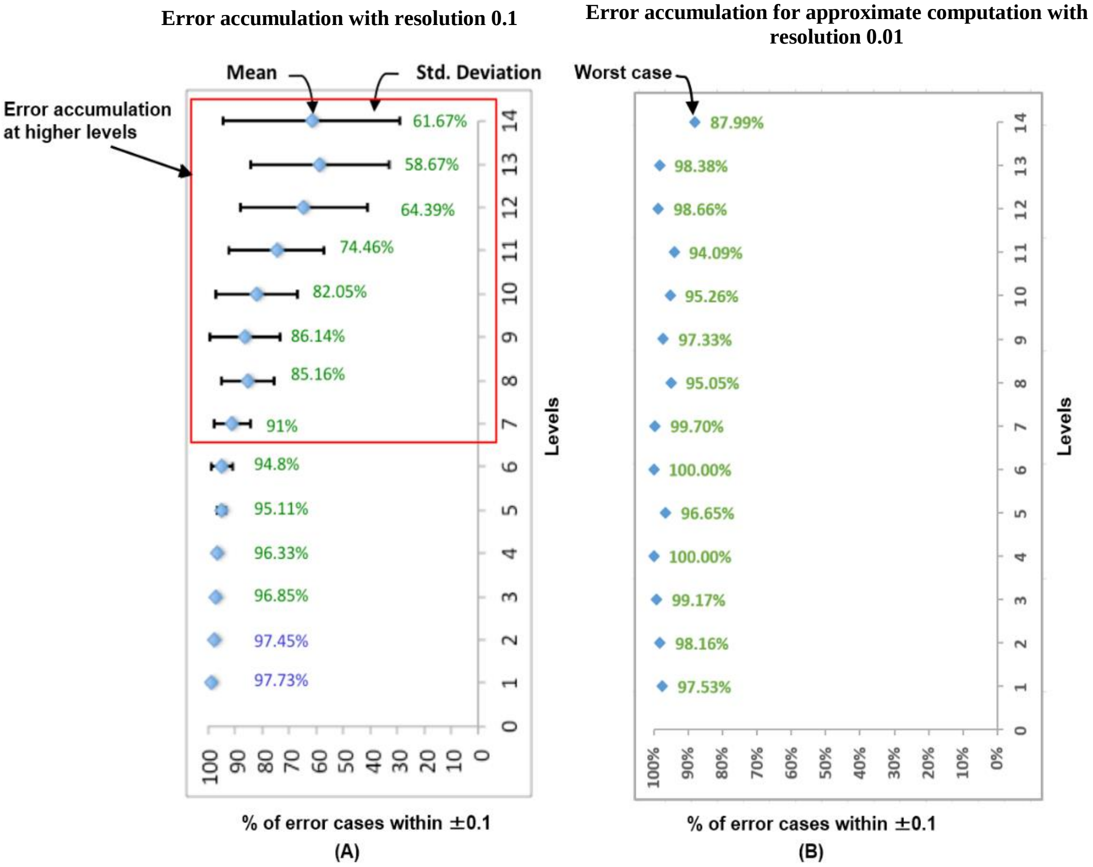
We observe that the approximate nature of computation provides with results accurate enough for BN applications at a circuit level, while achieving significant area benefits. The error resilience of the proposed fabric needs to be validated over large networks, wherein small errors rend to accumulate over several levels.

To validate high-level error propagation resilience of proposed fabric, we perform exhaustive simulations of the fabric by developing a behavioral simulation of the fabric using C++, simulating BNs with sizes up to a million random variables.

This simulation could be performed exhaustively for the flat-only architecture[13][14], but as the computational resolution increases the number of possible test cases increases super exponentially – it becomes increasingly time consuming to perform exhaustive simulations for flat-radix based architecture. Hence to validate error propagation for flat-radix based architecture we only consider one case which is the one with resolution 0.01 (two segments).

This simulation is not exhaustive from level 2 onwards because of the exponential growth of possible test cases. Instead we randomly sample 10^6 output combinations from the level below to calculate errors. Due to this limitation, the simulation does not capture all the corner cases that might lead to errors accumulation in higher levels. Nevertheless, the random sampling of input combination makes the simulation closer to actual Bayesian networks and hence the results could be expected to reflect the real-world performance to a high degree. The metric used to measure the error accumulation is the percentage of cases with error greater than 0.1. This metric is chosen for two reasons. Many of the applications in which probabilistic inference is used a factor of ±10% is acceptable[16][17].

The other reason is to inspire a design choice wherein the resolution of the FRCs can be selected to be one resolution-scale higher than the required resolution to obtain satisfactory results.


Error accumulation with resolution 0.1

Error accumulation for approximate computation with resolution 0.01

Figure 24. Comparison of simulation results for error accumulation in large networks. Both plots denote % of error cases within ±0.1 (considered acceptable in majority of applications) with increasing depth or 'Levels' of the network; (a) The flat-only composers' low computational resolution (0.1) causes large amount of error accumulation at higher levels as indicated in red; (b) Even though it implements approximate computation, the worst-case performance of the FRCs with computational resolution of 0.01 is significantly better, at just 2x area cost.

Figure 24 demonstrates this design choice. Figure 21(a) shows the error accumulation in flat-only composer based framework, and Figure 21(b) shows the worst-case performance in error accumulation for FRC based framework with resolution just one resolution-scale higher than the flat-only composers. As seen the worst-case error accumulation in (b) is better than the best-case error accumulation in (a). The error resilience in Bayesian applications allows for a robust inference, even if ~90% of the computations performed are accurate and the rest have errors. The randomized sampling performed in the higher levels of the simulation makes sure that the results are representative of real-life applications.

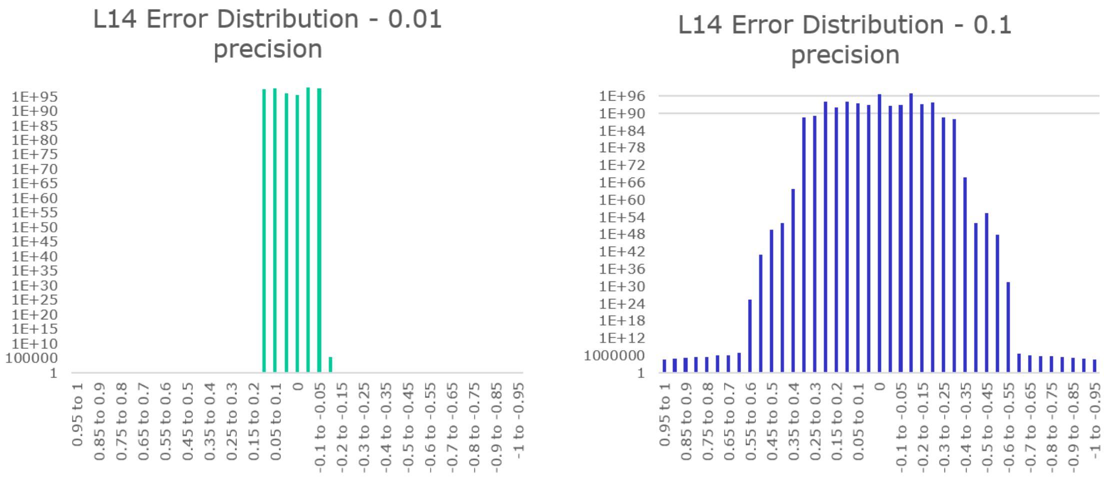


Figure 25. Error distribution comparison at level 14 of the simulated binary BN

Figure 25 shows the error distribution at level 14 of the simulated binary BN in both the lower and higher computational precision cases. The higher precision computation results in significantly lower cases (note that the y-axis is logarithmic) than the lower resolution case. This provides with a visual representation of how tightly the errors are bound due to the higher resolution computation. The approximation involved in the higher precision computation is not significant enough for large error buildups, the likes of which we observe in the lower precision computation.

CHAPTER 6

CONCLUSION

Probabilistic reasoning frameworks enable many important applications like gene expression, threat detection, text classification and macroeconomics [7]-[11]. As more disciplines of science incorporate probabilistic reasoning into their research process, the list of applications which could benefit from BNs is increasing. The fundamental incompatibility of these probabilistic frameworks with the conventional computing paradigm demands new fabric architecture approaches, which perform probabilistic computations much more efficiently. The magneto-electric circuit framework proposed in this paper performs high resolution probabilistic computations with high efficiency and provides with an easily scalable information representation scheme for analog computations with probabilities. The ability to scale efficiently while maintaining error resiliency will enable accurate representation of very large BNs with sizes up to a million random variables, which could potentially be used in applications like personalized gene-expression networks for cancer treatments [7], large-scale threat detection in computer networks [8], and others currently not feasible in software-only computing paradigms.

APPENDIX

PSEUDO-CODE OF THE BEHAVIORAL SIMULATION IN C++

Pseudo-code of the behavioral simulation performed in C++ to evaluate accumulation of errors at higher levels of BN due to low computational resolution.

```
Algorithm to simulate error propagation over large binary tree Bayesian Networks:
Each Bayesian node B has children C, D and parent A.
The subscript $f$ denotes full precision computation while the subscript $l$ denotes limited precision computation.
level 1
**for** all possible combinations of C, D:
        {
        estimate likelihood $\lambda(B)$;
        calculate error $e_{\lambda}(B)$;
        increment counter of corresponding error interval;
        perform belief update $BEL(B)$;
        calculate error $e_{BEL}(B)$;
        increment counter of corresponding error interval;
        provide diagnostic support $\lambda_{B}(A)$;
        add output combination to file;
        }
**for** level n from 2 to 15:
{
randomly sample $10^{6}$ output combinations from level(n-1);
**for** all sampled combinations:
        {
        estimate likelihood $\lambda(B)$;
        calculate error $e_{\lambda}(B)$;
        increment counter of corresponding error interval;
        compute prior $\pi(B)$;
        calculate error $e_{\pi}(B)$;
        increment counter of corresponding error interval;
        perform belief update $BEL(B)$;
        calculate error $e_{BEL}(B)$;
        increment counter of corresponding error interval;
        provide diagnostic support to parent $\lambda_{X}(A)$;
        provide predictive support to children          $\pi_{C}(B)$, $\pi_{D}(B)$;
        add output combination to file;
        }
}
```

BIBLIOGRAPHY

[1] J. Pearl, Probabilistic reasoning in intelligent systems: Networks of plausible inference, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1988.

[2] A. Darwiche, *Modeling and reasoning with Bayesian networks*, Cambridge University Press, 2009.

[3] D. C. Knill and A. Pouget, “The Bayesian brain: The role of uncertainty in neural coding and computation,” *Trends Neurosci.*, vol. 27, no. 12, pp. 712–719, 2004.

[4] E. Téglás, E. Vul, V. Girotto, M. Gonzalez, J. B. Tenenbaum, and L. L. Bonatti, “Pure reasoning in 12-month-old infants as probabilistic inference.,” *Science (80-.).*, vol. 332, no. 6033, pp. 1054–9, 2011.

[5] D. De Ridder, S. Vanneste, and W. Freeman, “The Bayesian brain: Phantom percepts resolve sensory uncertainty,” *Neurosci. Biobehav. Rev.*, vol. 44, pp. 4–15, 2014.

[6] K. Friston, “The free-energy principle: a unified brain theory?,” *Nat. Rev. Neurosci.*, vol. 11, no. 2, pp. 127–138, 2010.

[7] C. Su, A. Andrew, M. R. Karagas, and M. E. Borsuk, “Using Bayesian networks to discover relations between genes, environment, and disease,” *BioData Mining*, vol.6, no. 6, 2013.

[8] Valdes and K. Skinner, “Adaptive, model-based monitoring for cyber attack detection,” in *Recent Advances in Intrusion Detection (RAID)*, pp. 80–92, 2000.

[9] G.J. Brostow, R. Cipolla, "Unsupervised bayesian detection of independent motion in crowds," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol.1, pp.594-601, 17-22 June 2006.

[10] P. Lucas, "Bayesian networks in medicine: A model-based approach to medical decision making," *in Proceedings of the EUNITE workshop on intelligent systems in patient care,* 2001.

[11] M. A. Hooker, "Macroeconomic factors and emerging market equity returns: a Bayesian model selection approach" *Emerging Markets Review, Elsevier,* 2004.

[12] A. K. Biswas, S. Bandyopadhyay and J. Atulasimha, "Energy-efficient magnetoelastic non-volatile memory," Appl. Phys. Lett., 104, 232403, 2014

[13] S. Khasanvis, M. Li, M. Rahman, M. Salehi-Fashami, A. K. Biswas, J. Atulasimha, S. Bandyopadhyay, and C. A. Moritz. "Self-similar magneto-electric nanocircuit technology for probabilistic inference engines." *IEEE Transactions on Nanotechnology*, vol. 14, no. 6, pp980-991. 2015.

[14] S. Khasanvis, M. Li, M. Rahman, A. K. Biswas, M. Salehi-Fashami, J. Atulasimha, ... & C. A Moritz, "Architecting for Causal Intelligence at Nanoscale". *IEEE Computer*, vol.48, no.12, pp54-64, 2015.

[15] Han, Jie, and Michael Orshansky. "Approximate computing: An emerging paradigm for energy-efficient design." *2013 18th IEEE European Test Symposium (ETS)*. IEEE, 2013.

[16] A. Onisko, and M. J. Druzdzela, "Impact of precision of Bayesian network parameters on accuracy of medical diagnostic systems,"*Artificial Intelligence in Medicine, Elsevier*, vol. 57, no. 3, pp. 197 – 206, 2013.

[17] S. Tschiatschek, and F. Pernkopf, "On Bayesian Network classifiers with reduced precision parameters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 4, pp.774-785, 2015.

[18] J. A. Currivan, Y. Jang, M. D. Mascaro, M. A. Baldo, and C. A. Ross, “Low energy magnetic domain wall logic in short, narrow, ferromagnetic wires,” *IEEE Magn. Lett.*, vol. 3, 2012

[19] M. Sharad, D. Fan, K. Aitken, and K. Roy, “Energy efficient non-Boolean computing with spin neurons and resistive memory,” *IEEE Trans. Nanotechnol.*, vol. 13, no. 1, pp. 23–34, Jan. 2014.