Wave Interference Functions for Neuromorphic Computing

Mostafizur Rahman, Santosh Khasanvis, Jiajun Shi, and Csaba Andras Moritz
Department of Electrical and Computer Engineering, University of Massachusetts Amherst, MA, USA
E-mail: mostafiz@umass.edu, andras@ecs.umass.edu

Abstract— **Neuromorphic computing mimicking the functionalities of mammalian brain holds the promise for cognitive capabilities enabling new intelligent applications. However, research efforts so far mainly focused on using analog and digital CMOS technologies to emulate neural activities, and are yet to achieve expected benefits. They suffer from limited scalability, density overhead, interconnection bottleneck and power consumption related constraints. In this paper, we present a transformative approach for neuromorphic computing with Wave Interference Functions (WIF). This is a framework using emerging non-equilibrium wave phenomenon such as spin waves. WIF leverages inherent wave attributes for multi-dimensional, multi-valued data representation and communication, resulting in reduced connectivity requirements and efficient neural function implementations. It also yields a compact implementation of an artificial neuron. Moreover, since WIF computation and communication are in the spin domain, extremely low-power operation is possible. Our evaluations indicate up to 57x higher density, 775x lower power and 2x better performance when compared to an equivalent 8-bit 45nm CMOS neuron. Our scalability study using arithmetic circuits for higher bit-width neuron implementations indicate up-to 63x density, 884x power and 3x performance benefits in comparison to a 32-bit CMOS equivalent design at 45nm.**

Keywords—Neuromorphic Computing; Wave computation; Multi-valued computation; Spin waves; Wave interference functions.

I. INTRODUCTION

Brain-like computing has always attracted scientists and engineers in a quest for supporting intelligent applications. The new applications can revolutionize the Integrated Circuits (IC) industry and achieve far-reaching socio-economic impact. With the progress in nanoscale manufacturing, device, and circuit technologies the push for neuromorphic computing is reinvigorated.

To-date, research on hardware implementation of neuromorphic architectures has been mostly focused on emulation of neural activities using either analog or digital CMOS technologies. However, none of these technologies are intrinsically suitable for neuromorphic implementations. Analog-CMOS based implementations use analog data representation and analog components to emulate the biophysics of neurons; these designs are fully customized, and suffer from scalability limitations, density and connectivity bottleneck issues [1], On the other hand, digital CMOS implementations use Boolean data representation and digital transistors to emulate the behavior of a neuron at higher level of abstraction; this results in high power consumption and area overheads [1].

This project was supported by DARPA program on Non-volatile Logic, and Center for Hierarchical Manufacturing (CHM, NSF) at UMass Amherst.

In this paper, we present Wave Interference Functions (WIF) framework [2], and show a new transformative approach using WIF towards neuromorphic computing. WIF's core components, multi-valued data representation, communication, and computation allow efficient mapping of a biological neuron's functionalities with ultra-low power operation. WIF uses nanoscale physical components: spin wave bus and magneto-electric (ME) cells. Information is encoded in a combination of spin wave's attributes – amplitude and phase. Information processing is achieved through wave superposition interactions and wave propagation; ME cells are used for wave generation, detection, amplification and non-volatile storage. The information encoding in wave attributes and wave interference allows intrinsic multi-valued data representation, communication and computation. WIF's multi-valued logic constructs for functionality mapping in combination with these intrinsic physical capabilities enable compact logic/functionality implementation with minimum area footprint and reduced connectivity. Moreover, since neuron implementation in WIF is done in a generic neuron architecture using logic, arithmetic, input/output functional units and storage, the design is scalable to higher bit-widths. Furthermore, the power consumption is extremely low due to spin-domain based signal propagation and computation.

This paper presents foundational work towards neuromorphic computing using WIF. Key contributions of this paper are as follows:

- Extensive detail of WIF's core aspects, data representation, communication and computation. Mathematical formulation of wave interactions.
- Introduction of new multi-valued logic constructs, arithmetic units, input/output functions and storage concepts necessary for neuron implementations in WIF.
- Emulation of neuron behavior in WIF targeting scalable designs.
- Evaluation and benchmarking of WIF neuron implementation with respect to equivalent digital CMOS based implementation in 45nm technology.

The paper is organized as follows: Section II presents core fabric aspects, data representation, and details mathematical formulation of wave interactions. Section III presents multi-valued logic constructs, arithmetic unit, input/output functional units, and non-volatile storage. Section IV details neuron implementation in WIF. Section V shows evaluation and benchmarking results. Finally, Section VI draws conclusions.

II. Wave Interference functions

A. *Physical Components*

Primary physical components in Wave Interference Functions (WIF) framework [2] are spin wave bus and ME cell. These components are used to operate on spin wave. Spin waves [4][12][14] are the collective oscillations of electrons spins in an ordered spin lattice around the direction of magnetization in ferromagnetic materials.

Spin Wave Buses (SWB) are ferromagnetic waveguides that facilitate spin wave propagation and superposition (Fig. 1a). SWB is used in between computational junctions for information transmission from one physical location to another. Using SWB, spin waves can be propagated to large distances at room temperature (from tens of micrometers in permalloy films [6] to millimeters in yttrium iron garnet films [7]). SWB also facilitates spin wave superposition. Spin waves interfering at a junction in SWB result in change in magnetization at that point. The magnitude of this local magnetization change is enhanced when the waves are in phase, and is diminished to a minimum when they interfere destructively (if they are out of phase with respect to each other).

The ME cell is a multiferroic heterostructure consisting of a magnetic element with at least two stable states for magnetization. It performs several functions – (i) generates and detects spin waves by converting electric signals into magnetic domain and vice versa, (ii) amplifies spin waves for logic, and for signal restoration in spin wave bus, and (iii) stores encoded information in the state of its magnetization.

When a bias voltage is applied on the top metal electrode (Fig. 1a), a stress generated in the piezoelectric layer causes rotation of the easy axis in the piezomagnetic material through strain-induced anisotropy with two preferred directions (along or opposite to the new easy axis). Propagating spin waves can then control magnetization direction of the cell. In the presence of bias field, when the easy axis rotation changes, it was shown that the incoming spin wave with phase π results in magnetization signal along the positive direction of new easy axis, whereas spin wave with phase 0 results in magnetization signal along the negative direction [10].

Spin wave generation is through applying an alternating voltage at the top metal contact, which results in oscillating strain in the piezoelectric layer that creates magnetoelastic spin wave excitation [16]. Detection of spin waves is through the reverse process from magnetic to electric domain [17].

Similar principle can be applied for spin wave amplification. By applying alternating voltage (required for 90 degree rotation) at a frequency equal to the incoming spin wave frequency, the rotated magnetization component can be amplified to the saturation value of magnetization. The phase of the incoming spin wave determines the direction of rotation (along or opposite to rotated direction) and hence preserves the phase in the output wave[18].

ME cells exhibit nonvolatility, requiring no continuously applied voltage to keep the magnetization in the reoriented state [15][3]. Nonvolatility properties (i.e., endurance, retention) vary based on material choices, dimensions and integration aspects, and are still under active research. According to [15], ME cell's retention and endurance properties can be as high as STTRAM's. Further details about ME cells can be found in [3][8][12][15][18][19].

B. *Multi-valued Data Representation with Waves*

Waves present several attributes to encode data such as phase, amplitude and frequency, thereby providing an opportunity to develop new schemes for multi-dimensional compressed data representation. The choice of using any one or a combination of the wave characteristics is driven by the capabilities of the physical components used to build the computational system. In this paper, we consider only two phases either 0 or π for data encoding, since the ME cells used for spin wave detection are designed to differentiate these phases.

Here, we introduce the notations that will be used in subsequent sections for multi-valued logic design. A spin wave

Table 1. Quaternary (Radix-4) Logic Encoding

Logic Value	**Wave Representation**	**Wave Attributes (Amplitude, Phase)**
0	$\tilde{L}_0 = 3Ae^{i0}$	$(3A, 0)$
1	$\tilde{L}_1 = Ae^{i0}$	$(A, 0)$
2	$\tilde{L}_2 = Ae^{i\pi}$	(A, π)
3	$\tilde{L}_3 = 3Ae^{i\pi}$	$(3A, \pi)$

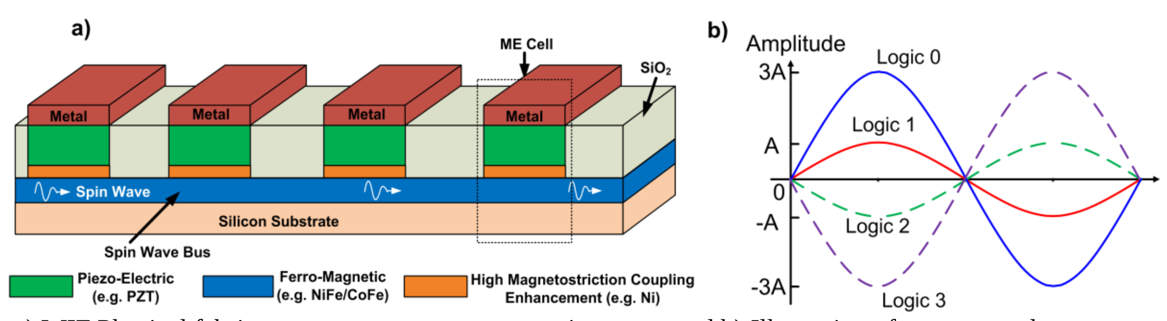


Fig. 1. a) WIF Physical fabric components to operate on spin waves; and b) Illustration of quaternary data representation (radix-4) with waves, encoded in the combination of phase and amplitude.

is denoted as $\tilde{X}$; and is represented using polar co-ordinates to incorporate both its amplitude (a) and phase (φ) compactly as follows:

$$\tilde{X} = ae^{i\varphi} = a(\cos\varphi + i\sin\varphi). \qquad (1)$$

Thus, any wave can be interpreted as having amplitude a when the phase is 0, and $-a$ when the phase is π at the point of interference. When a phase other than 0 and π is employed, either the real or imaginary component of the notation above will need to be used as required.

To represent data in radix-r number system, we need $r/2$ distinct amplitude values if r is even, and $(r+1)/2$ amplitude values if r is odd, in conjunction with aforementioned 2 phase values. For example, for binary data representation (radix-2) we need a single amplitude level A. The phase encodes binary data (1-bit) with logic 0 and logic 1, assigned to waves with initial phase 0 and π respectively. For quaternary data representation (radix-4), we use two amplitude levels (A, $3A$) in conjunction with two phase values (0, π) to get four different combinations. Each combination is assigned to a logic value (see Fig. 1b and Table 1). Alternative combinations for amplitude and phase may also be used. By contrast, conventional charge-based digital computational systems are capable of using only the presence/absence of charge for one-dimensional binary information representation.

C. *Interference Function*

Wave interference is the fundamental operation in the WIF approach. Spin waves interfering at a given point exhibit linear superposition behavior [6][14]. Elementary spin wave circuit operation has been experimentally demonstrated at room temperature[4]. Here, the focus is on a new model of computation with waves departing from conventional Boolean and Majority approach. The *Interference Function* **I** of n input waves $\tilde{X}_0, \tilde{X}_1, \dots, \tilde{X}_{n-1}$ is defined as follows:

$$\begin{aligned}\mathbf{I}(\tilde{X}_0, \tilde{X}_1, \dots, \tilde{X}_{n-1}) &= \tilde{X}_0 + \tilde{X}_1 + \dots + \tilde{X}_{n-1} \\ &= a_0 e^{i\varphi_0} + a_1 e^{i\varphi_1} + \dots + a_{n-1} e^{i\varphi_{n-1}}.\end{aligned} \qquad (2)$$

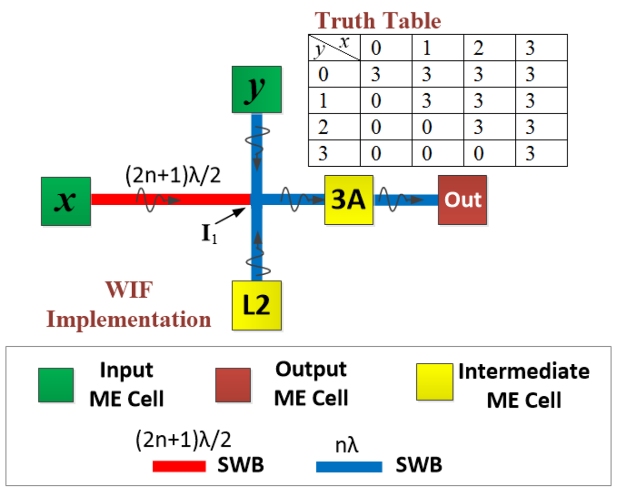
The result of this *Interference Function* is a spin wave $\tilde{Y}$, whose individual wave attributes are denoted as follows:

$$\begin{aligned}&\tilde{Y} = a_y e^{i\varphi_y} = \mathbf{I}(\tilde{X}_0, \tilde{X}_1, \dots, \tilde{X}_{n-1}) \\ \text{where, } & a_y = \mathrm{I}^A(\tilde{X}_0, \tilde{X}_1, \dots, \tilde{X}_{n-1}) \\ & \varphi_y = \mathrm{I}^\varphi(\tilde{X}_0, \tilde{X}_1, \dots, \tilde{X}_{n-1}).\end{aligned} \qquad (3)$$

In general for n input waves, if the amplitude of any wave $\tilde{X}_j$ is $a_j = w_j.A$, where w_j represents a weight in multiples of unit-amplitude A, then the *Interference Function* result encodes the following information:

$$\begin{aligned}&\mathrm{I}^\varphi(\tilde{X}_0, \tilde{X}_1, \dots, \tilde{X}_{n-1}) = \begin{cases}\pi; & \text{if } \sum w_j A e^{i\pi} > \sum w_k A e^{i0} \\ 0; & else\end{cases} \\ &\qquad \to \text{weighted-majority decision} \\ &\mathrm{I}^A(\tilde{X}_0, \tilde{X}_1, \dots, \tilde{X}_{n-1}) = \left|\sum w_k A e^{i0}\right| - \left|\sum w_j A e^{i\pi}\right|.\end{aligned} \qquad (4)$$

The output phase encodes the weighted majority decision of all the input wave phases, and the amplitude represents the weighted difference of the number of input waves that are out-of-phase with respect to each other. Thus the *Interference Function* encodes all the necessary information about the inputs in a compressed manner.



y \ x	0	1	2	3
0	3	3	3	3
1	0	3	3	3
2	0	0	3	3
3	0	0	0	3

Fig. 2. Truth table and physical implementation for *Threshold* Operator. ME cell labeled ‘L2’ and ‘3A’ generate and amplifies waves of logic 2 and amplitude 3A respectively. Here, λ is the spin wavelength and n is an integer. Unless specified explicitly, all SWBs have lengths equal to an integral multiple of λ.

Fig. 2 shows an example of multi-valued logic (Threshold function) implementation using Interference Functions in WIF. The multivalued threshold logic implementation in Fig. 2 takes two logical inputs (x,y) and outputs a constant value when one input is greater than the other. The threshold output (x_y^{r-1}) is logically defined as:

$$x_y^{r-1} = \begin{cases} r-1, \text{when } x \geq y \\ 0, \qquad else \end{cases}, x, y \in \{0, 1, \dots, r\text{-}1\}. \qquad (5)$$

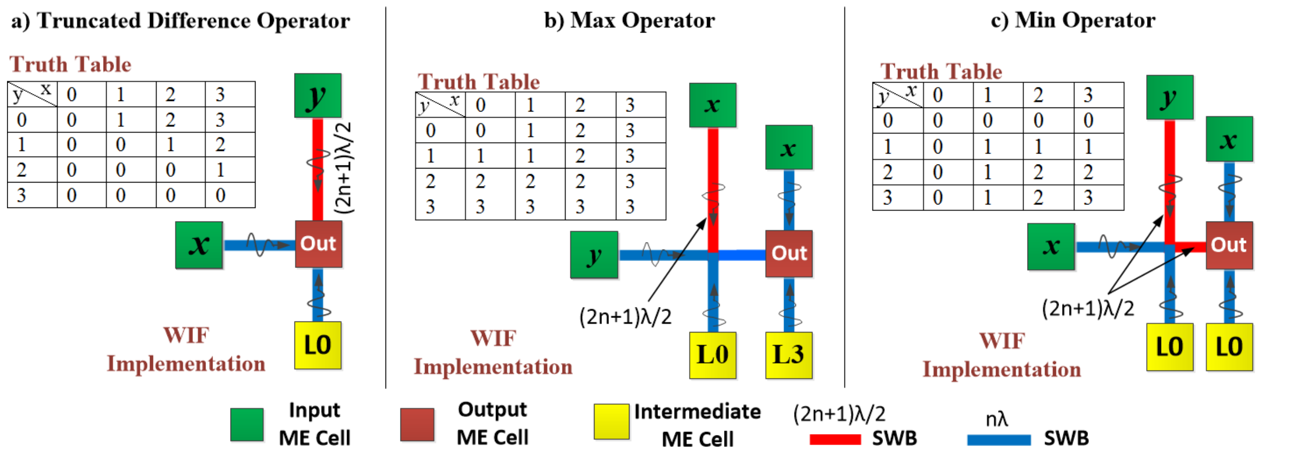
For radix–r it is expressed in terms of *Interference Function* as:

$$\tilde{X}_y^{r-1} = \mathrm{I}[(r-1)\mathrm{I}_1^\varphi(-\tilde{X}, \tilde{Y}, \tilde{L}_{r/2})], \qquad (6)$$

where (r-1) represents either r-1 copies of interference output at $\mathbf{I_1}$ or amplification (using ME cell); $\tilde{X}, \tilde{Y}$ are input waves corresponding to logical inputs x, y respectively; and $\tilde{L}_{r/2}$ is a reference wave corresponding to logic level $r/2$. *Interference Function* $\mathbf{I_1}$ produces an output wave of positive phase when $(x \geq y)$, and generates a negative phase otherwise. To obtain the correct output, here we use an amplification ME cell such that the output wave has a phase equal to the incoming wave, but the amplitude is always pulled up to the highest supported value. Fig. 2 shows the truth table for *Threshold* operation and physical implementation in WIF for quaternary logic (r = 4).

III. WIF Logic Constructs, Arithmetic, Storage and Input/Output Units for Neuromorphic Architectures

In our previous work [2], we have detailed multi-valued algebra, multi-valued operator implementations in WIF, and shown arithmetic functions using them. Quaternary logic encoding can be achieved by using weighted Interference Functions. Details of conversion between binary to quaternary values can be found in [19]. Here, we introduce the multi-



y \ x	0	1	2	3
0	0	1	2	3
1	0	0	1	2
2	0	0	0	1
3	0	0	0	0

y \ x	0	1	2	3
0	0	1	2	3
1	1	1	2	3
2	2	2	2	3
3	3	3	3	3

y \ x	0	1	2	3
0	0	0	0	0
1	0	1	1	1
2	0	1	2	2
3	0	1	2	3

Fig. 3. Truth table and physical implementation for a) *Truncated Difference* Operator; b) *Min* Operator; and c) *Max* Operator. The intermediate ME cell labeled '3A' generates a spin wave with phase equal to input phase and constant amplitude 3A. Other intermediate ME cells labeled 'L0' and 'L3' generate waves corresponding to logic 0 and logic 3 respectively. Here, λ is the spin wavelength and *n* is an integer. Unless specified explicitly, all SWBs have lengths equal to an integral multiple of λ.

valued operators, arithmetic and functional units necessary for neuromorphic architectures in WIF.

A. *Multi-Valued Logic Constructs*

1) *Truncat ed Difference Operator*

This operator outputs the difference between two inputs when a condition is satisfied. The notation is $x \,\Xi\, y$, and the operation is defined as:

$$x \,\Xi\, y = \begin{cases} x - y\,, \text{when } x > y \\ \quad 0, \qquad else \end{cases}, \qquad (7)$$
$$x, y \in \{0, 1, \dots, r-1\}.$$

This can be expressed with *Interference Function* as

$$\tilde{X} \,\Xi\, \tilde{Y} = \mathbf{I}(\tilde{X}, -\tilde{Y}, \tilde{L}_0), \qquad (8)$$

where, $\tilde{X}, \tilde{Y}$ are input waves corresponding to logical inputs *x*, *y* respectively; and $\tilde{L}_0$ is a reference wave corresponding to logic 0. The truth table and the physical implementation for *Truncated Difference* operator are shown in Fig. 3a for quaternary logic. The difference operation is performed at the junction of incoming waves. In order to achieve the correct output, the resultant wave amplitude after interference is always truncated to 3A if it is greater than 3A. This truncation may be achieved by either designing the spin wave bus and ME cells to accommodate this requirement or through external electrical circuits. The same assumption is considered for other multi-valued operators and circuit implementations as well.

2) *Min Operator*

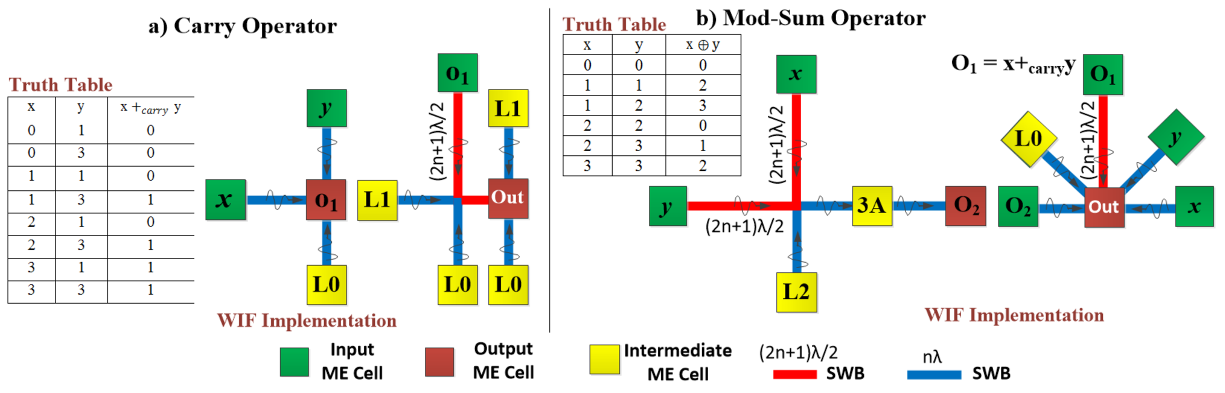
The *Min* operator ($x \cdot y$) in multi-valued logic is analogous to the Boolean AND operator. It is defined as follows:

$$x \cdot y = \begin{cases} x, & x < y \\ x - (x - y), & else \end{cases} \qquad (9)$$
$$x, y \in \{0, 1, \dots, r-1\}.$$

The *Truncated Difference* operator can be used to realize the above output conditions as $x \cdot y = x \,\Xi\, (x \,\Xi\, y)$. Notice that in equation (13), for the condition $x \cdot y = y$, the output is re-expressed as $x \cdot y = x - (x - y)$ to enable implementation with *Truncated Difference* operator. The functional representation in terms of *Interference Function* is:

$$\text{Min}(\tilde{X}, \tilde{Y}) = \tilde{X} \,\Xi\, (\tilde{X} \,\Xi\, \tilde{Y}) = \mathbf{I}[\tilde{X}, -(\tilde{X} \,\Xi\, \tilde{Y}), \tilde{L}_0], \qquad (10)$$

where $\tilde{X}, \tilde{Y}$ are input waves corresponding to logical inputs



x	y	$x +_{carry} y$
0	1	0
0	3	0
1	1	0
1	3	1
2	1	0
2	3	1
3	1	1
3	3	1

x	y	$x \oplus y$
0	0	0
1	1	2
1	2	3
2	2	0
2	3	1
3	3	2

Fig. 4. Truth table and physical implementation for cyclic operators, a) *Carry* Operator; b) *Mod-Sum* Operator. The intermediate ME cells labeled 'L0' and 'L2' generate waves corresponding to logic 0 and logic 2 respectively. Here, λ is the spin wavelength and *n* is an integer. Unless specified explicitly, all SWBs have lengths equal to an integral multiple of λ.

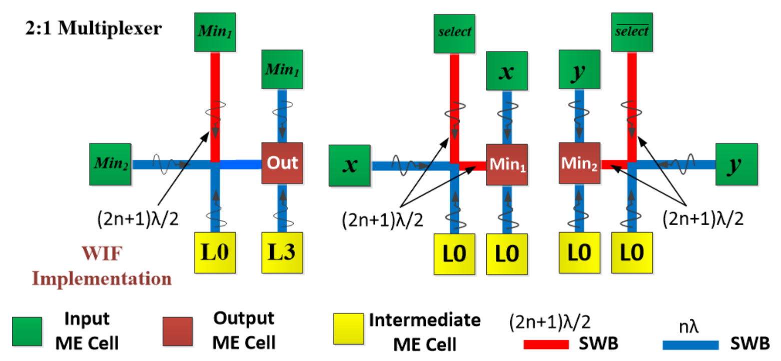


Fig. 5. Quaternary 2:1 Multiplexer implementation in WIF

x, y respectively; and $\tilde{L}_0$ is a reference wave corresponding to logic 0. Fig. 3c shows the truth table and the WIF physical implementation of *Min* operator.

3) Max Operator

The max operator ($x + y$) in multi-valued logic is analogous to the Boolean OR, defined as follows:

$$x + y = \begin{cases} x, & x > y \\ x + (y - x), & else \end{cases} \qquad (11)$$
$$x, y \in \{0, 1, \dots, r-1\}.$$

The functional representation in terms of *Interference Function* is

$$\text{Max}(\tilde{X}, \tilde{Y}) = \tilde{X} + (\tilde{Y} \,\Xi\, \tilde{X}) = \mathbf{I}[\tilde{X}, (\tilde{Y} \,\Xi\, \tilde{X}), \tilde{L}_{r-1}], \qquad (12)$$

where $\tilde{L}_{r-1}$ is a reference wave corresponding to logic value r-1. Fig. 3b shows the physical implementation.

4) Cyclic Operator

The *cyclic* operator is also known as *mod-sum* operator; it performs XOR-like operation in the multi-valued domain. The *mod-sum* operator is defined as:

$$x \oplus y = (x +_{add} y) \bmod r, \qquad (13)$$
$$x, y \in \{0, 1, \dots, r-1\}.$$

Here, '$+_{add}$' represents arithmetic addition of logic inputs. To implement this function, we define a new operator called *Carry* operator (denoted by '$+_{carry}$'):

$$x +_{\text{carry}} y = \begin{cases} 1, & \text{if } x +_{add} y > r-1 \\ 0, & else \end{cases} \qquad (14)$$
$$x, y \in \{0, 1, \dots, r-1\}.$$

The *Carry* operator is implemented using *Min* operator as follows:

$$\tilde{X} +_{\text{carry}} \tilde{Y} = \text{Min}[\mathbf{I}(\tilde{X}, \tilde{Y}, \tilde{L}_0), \tilde{L}_1]. \qquad (15)$$

The output of $\mathrm{I}(\tilde{X}, \tilde{Y}, \tilde{L}_0)$ represents ($x +_{add} y$) – r–1, if $x +_{add} y > r$–1; and 0 otherwise. Therefore, a non-zero output is obtained only when $x +_{add} y > r$–1. The *Min* operation of this output with $\tilde{L}_1$ provides the binary *Carry* output.

The *Cyclic* operator is then implemented as:

$$\tilde{X} \oplus \tilde{Y} = \mathbf{I}[\tilde{A}, \tilde{B}, \tilde{L}_0, {}^{r-1}_{\;\;r}(\tilde{X} +_{add} \tilde{Y}), -(\tilde{X} +_{carry} \tilde{Y})]. \qquad (16)$$

Here, ${}^{r-1}_{\;\;r}(\tilde{X} +_{add} \tilde{Y})$ implements the *Lower Threshold* operation, whose output is r–1 if $x +_{add} y \le r$–1, and 0 otherwise. Sample test vectors and outputs from Carry and Mod-sum operator are shown in truth tables in Fig. 4a and Fig. 4b.

5) Multiplexer

A *2:1 Multiplexer* function in WIF follows the following equation:

$$Out = Max\,(Min\,(x, select), Min(y, \overline{select}\,)$$

W here *Out* is the *Max* of two *Min* operations; the Min operations take as inputs *(x, select) and (x,* $\overline{select}$). The multiplexer implementation in WIF is shown in Fig. 5. Similarly, a *r:1* multiplexer can be implemented using *r Min* and *r-1 Max* operators.

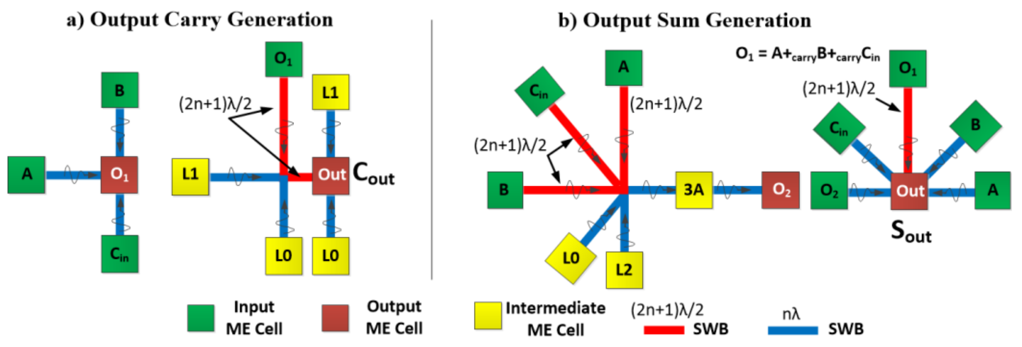


Fig. 6. Quaternary full adder implementation in WIF for: a) Carry function (C_{out}); and b) Sum function (S_{out}).

B. *Quaternary Full Adder Design*

As mentioned earlier and in [2][5], the use of multi-valued operators for circuit design reduces complexity significantly and provides a framework for arbitrary logic/arithmetic implementation. The quaternary full adder circuit operates on two quaternary operands (*A*, *B*) and a binary carry-in (C_{in}). It has two outputs representing the result of the addition – the quaternary least significant digit (S_{out}) and the binary carry-out (C_{out}). The same full adder design can be extended to implement high bit-width adders. The conditions for binary carry generation are:

$$C_{out} = \begin{cases} 1, & \text{if } A +_{add} B +_{add} C_{in} \geq r \\ 0, & else \end{cases} \tag{17}$$

$$A, B \in \{0, 1, \dots, r-1\} \text{ and } C_{in} \in \{0, 1\}.$$

Here $r = 4$ for quaternary logic, and '$+_{add}$' represents arithmetic addition of logic inputs. The above operation is realized using 3-input *Carry* operator as:

$$\tilde{C}_{out} = \tilde{A} +_{\text{carry}} \tilde{B} +_{\text{carry}} \tilde{C}_{in} = \text{Min}[\mathbf{I}(\tilde{A}, \tilde{B}, \tilde{C}_{in}), \tilde{L}_1], \tag{18}$$

where $\tilde{A}, \tilde{B}, \tilde{C}_{in}$ are input waves corresponding to logical inputs *A*, *B*, C_{in} respectively; $\tilde{C}_{out}$ is the output wave corresponding to output C_{out}; and $\tilde{L}_1$ is a reference wave corresponding to logic 1.

The quaternary full adder sum output (S_{out}) conditions are:

$$\begin{cases} \mathrm{A} +_{add} \mathrm{B} +_{add} \mathrm{C_{in}} - r, & \text{if } \mathrm{A} +_{add} \mathrm{B} +_{add} \mathrm{C_{in}} > r - 1 \\ \mathrm{A} +_{add} \mathrm{B} +_{add} \mathrm{C_{in}}, & else \end{cases} \tag{19}$$

Here *A*, *B* ϵ {0,1,2,3} and C_{in} ϵ {0, 1} for quaternary adder. This is expressed using 3-input *Cyclic* operator as follows:

$$\tilde{S}_{out} = \tilde{A} \oplus \tilde{B} \oplus \tilde{C}_{in} = \mathbf{I}\left(\tilde{A}, \tilde{B}, \tilde{C}_{in}, \tfrac{\mathrm{r}-1}{\mathrm{r}}\tilde{X}, -\tilde{C}_{out}\right),$$

$$\text{whe re } \tilde{X} = (\tilde{A} +_{add} \tilde{B} +_{add} \tilde{C}_{in}). \tag{20}$$

The WIF implementation of equations (24) and (26) are shown in Fig. 6.

C. *Input/Output Logic for Data Conversion Between Binary and r-ary Domains & Non-Volatile Storage*

In addition to computational logic, WIF's intrinsic properties can be utilized for data conversion between binary and multi-valued domains. This data conversion technique along with non-volatile ME cells are used for latch implementation as discussed in Section V.

1) *Binary to Quaternary Conversion:*

Binary to multi-valued conversion is achieved by using weighted *interference functions*. For binary to *r*-ary (i.e. radix-*r*) conversion, each binary digit is weighted according to its' least significant bit position. For binary inputs (A_{n-1}, ..., A_1, A_0), the weighted *interference function* to convert to *r*-ary output Y is:

$\widetilde{\boldsymbol{Y}} = \mathbf{I}\left(2^0\widetilde{\boldsymbol{A}}_0, 2^1\widetilde{\boldsymbol{A}}_1, 2^2\widetilde{\boldsymbol{A}}_2, \dots, 2^{n-1}\widetilde{\boldsymbol{A}}_{n-1}\right)$, where *n* is the number of bits**.**

Here, $\widetilde{\boldsymbol{A}}_i$ is the input wave corresponding to bit A_i. The weights can be implemented either with amplification ME cells or by replicating the particular input wave. The same principle can be applied to convert binary data into quaternary. All possible combinations for two-bit binary inputs, and their corresponding quaternary output is shown in Table 2. The WIF

Table 2 . Binary and quaternary logic states and data representations

Binary Value A_1A_0	Binary Spin Wave	Equivalent Quaternary Logic State	Quaternary Spin Wave Representation
00	Ae^{i0}, Ae^{i0}	0	$3Ae^{i0}$
01	$Ae^{i0}, Ae^{i\pi}$	1	Ae^{i0}
10	$Ae^{i\pi}, Ae^{i0}$	2	$Ae^{i\pi}$
11	$Ae^{i\pi}, Ae^{i\pi}$	3	$3Ae^{i\pi}$

implementation of binary to quaternary conversion logic is shown in Fig. 7a, where the weight for A_1 is implemented by replication.

2) *Quaternary to Binary Conversion:*

The following principle is used for converting *r*-ary logic state to equivalent binary using WIF. By implementing majority function based on the phase of the multi-valued logic state, an *r*-ary input (A) can be decomposed to binary outputs (O_{n-1} ....O_1O_0), where *n* represents number of bits and $2^n = r$. The LSB (O_0) is computed first using an output ME cell and external circuitry, which generates constant amplitude with

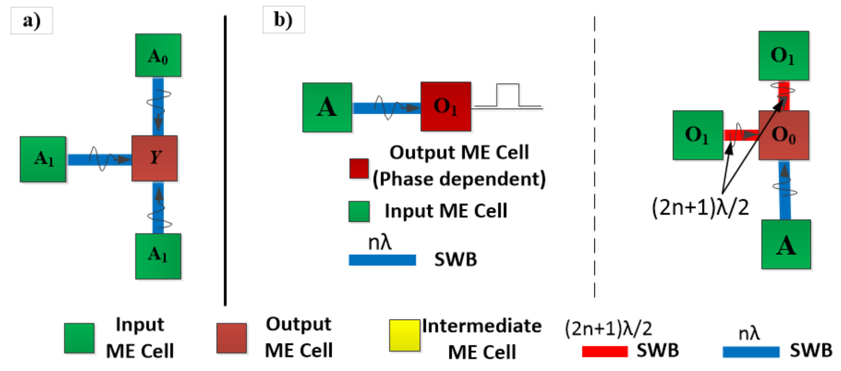


Fig. 7. a) Binary to Quaternary conversion. b) Quaternary to binary conversion logic; Binary most significant bit generation from phase information of the quaternary state (left), binary least significant bit generation by interference logic (right).

either positive or negative phase. The remaining output bits ($O_{d-1....}O_1$) are generated with similar constant amplitude generating ME cells. The *interference function* is:

$\tilde{\boldsymbol{Y}} = \mathbf{I}(\tilde{\boldsymbol{A}}, -(2\mathrm{n}-1)\tilde{\boldsymbol{O}}_{n-1}, \ldots, -(2\mathrm{n}-\mathrm{i}+1)\tilde{\boldsymbol{O}}_{n-i+1})$, where *n* is the number of bits.

Here, $\tilde{\boldsymbol{O}}_i$ represents the output wave corresponding to output bit O_i, and $\tilde{\boldsymbol{A}}$ is the input quaternary wave. Using these rules we can convert quaternary logic to binary. A single quaternary input (A) will have two binary outputs (O_1O_0). The binary MSB output (O_1) is 1 only for quaternary input states 2 and 3 (Table 2), and 0 for quaternary input states 0 and 1. The phase dependent ME cell along with external CMOS circuitry generates spin wave with positive or negative phase and constant amplitude (Fig. 7b (left)), which is the MSB bit (O_1) for binary representation. The LSB (O_0) is generated by subtracting the weighted MSB (O_1) from the quaternary input (A) as shown in Fig. 7b (right).

3) *Non-Volatile Storage*

As discussed earlier in Section II and in [2][12][5][3], ME cells can be used as non-volatile storage elements. Upon switching, depending on the energy barrier between two magnetization states, ME cells can preserve their state (magnetization state 0 or π) for very long time without requiring any external power supply. This property of ME cell is utilized along with data conversion units to implement latch for weight update in neuron circuit, as discussed in next section.

IV. Neuron Implementation in WIF

Key characteristics of biological neuron's computation include event-based processing on analog data, parallel computations, redundancy, connectivity, adaptation and learning, under extreme energy constraints. To map these characteristics, research direction so far has primarily focused on implementations using analog and digital CMOS. Analog CMOS implementations, while being compact, suffer from scalability and flexibility issues since analog CMOS components (e.g., capacitors, analog transistors) scale poorly, and are usually fully customized for each technology/design. In addition, they also face density and connectivity challenges [1]. On the other hand, digital CMOS components scale aggressively, and digital CMOS neuromorphic architectures are flexible in terms of design (support modular design, reconfiguration, memory augmentation, etc.). However, digital CMOS implementations use Boolean data representation. As a result, various levels of abstraction are used to represent equivalent behavior of a neuron, which is very inefficient resulting in density and power overhead.

In this section we present a neuron implementation in the WIF framework, which closely maps some of the key computational characteristics of biological neurons. By harnessing intrinsic properties of nanoscale components for multi-valued data representation, communication and computation, and by designing core logical constructs to achieve arithmetic operations efficiently, the WIF based neuron implementation solves CMOS challenges and achieves orders of magnitude benefits in comparison. Moreover, since all computations are performed in the spin domain, extreme low power operations are achieved for neuron implementation in WIF compared to CMOS. Analog implementation benefits are matched by using multi-valued data representation and computation in a compact form, while digital implementation benefits are matched by using emerging nanoscale components in a flexible framework that uses logic, arithmetic, input/output and storage units. Connectivity requirements are significantly lowered in WIF-based implementation, since multi-valued communication is achieved through spin waves, and multi-valued logic and arithmetic functions are achieved in a compact form through wave interference functions.

The neuron implementation in WIF is shown in Fig. 8; it performs the integrate-and-fire operation. Similar to biological neuron, the weighted excitatory/inhibitory inputs are integrated over time and a spike is fired if the integrated sum is greater than spike threshold. The input and output of the neuron circuit shown is in quaternary multi-valued WIF, and all computation is done with quaternary data. The neuron implementation in Fig. 8 uses a multiplexer for input selection, adder block for integration, threshold comparator for comparison, and input/output conversion units together for latching and feedback. Inter-block communication is achieved

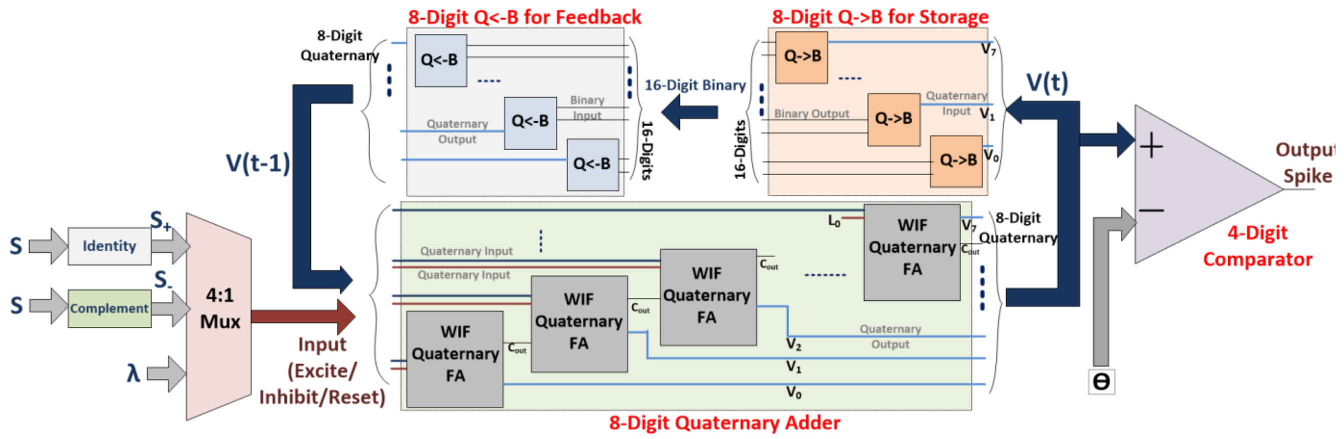


Fig. 8. Neuron implementation in WIF. The block diagram shows implementation details and interconnection. All input/output communication is through SWBs, data representation and computations are done in Quaternary logic. Summation results are stored in ME cells by Quaternary to Binary conversion and storage.

through spin-wave buses. This quaternary data based neuron implementation can be extended for any *r-ary* data based neuron implementation in WIF.

At a functional level the neuron implementation follows the equation:

$$V_{present} = V_{previous} + S_{+} - S_{-} - \lambda$$

where *V* represents membrane potential (integration result). V_{prsent} is the potential at time *t* and $V_{previous}$ is the potential at time *t-1*. S_{+} and S_{-} are weighted excitatory and inhibitory inputs, and λ is the reset input. If $V_{present}$ exceeds a threshold Θ, a spike is generated and $V_{present}$ is set to an initializing value.

The input and output of neuron implementation in Fig. 8 are through spin-waves, and data encoding follows quaternary representation discussed in Section II. A *4:1 Multiplexer* is used to select from three inputs S_{+}, S_{-}, and λ. Synaptic Inputs (S) from other neurons are passed through the multiplexer using *Identity* and *Complementary* operators [2][5] to represent excitatory (S_{+}) or inhibitory (S_{-}) inputs. Each of the S_{+}, S_{-}, and λ inputs are 4-digit wide. An 8-digit adder is used to add two 4-digit quaternary operands, and to handle overflow under worst-case conditions. Although the full width 8-digit adder is not necessary for this design, it was incorporated for equivalent comparison with an industrial design [9]. Storing the summation results for integration over time is a key requirement in a neuron circuit. We use the ME cell's non-volatility to meet this requirement. Since ME cells can only retain the magnetization state (phase 0 or π), the quaternary data needs to be converted to Binary. The 8-digit Quaternary to Binary conversion unit shown in Fig. 8 is used for this purpose. Whenever, a new synaptic input is available the binary data is converted back to quaternary using the 8-digit Binary to Quaternary conversion unit, for summation purposes. The integration result is compared with the threshold Θ using 4-digit *Threshold* comparator. In this digit-wise comparison, most significant digit is compared with that of Θ. The same *Threshold* comparator can be extended with *Min*, *Max* operators to do any data comparison.

The *4:1 Multiplexer* used in Fig. 8, is extended from the based 2:1 Multiplexer design presented in Section IIIA. The 8-digit adder uses 8 quaternary Full-Adders (Section IIIB) in a ripple carry design. The data conversion units used are extended to for 8-digits, and follow the same design principle as presented in Section IIIC. The threshold comparator is an extension of the 1-digit comparator discussed in Section IIC. Since the WIF-framework based neuron design is component centric and each component can be extended to support higher-digits and any *r*-radix based implementations, this design is very flexible. In addition, the core neuron circuit can be extended with ME cell based memory components or a spin-wave based memory grid [5] to support weight adaptation and learning.

V. BENCHMARKING VS. CMOS NEURONS

To evaluate the potential of this Neuromorphic cell architecture implementation in WIF, extensive benchmarking was done with respect to binary CMOS. The neuron circuit presented in the previous section that operates on 4-digit inputs was compared against equivalent Digital CMOS based industrial design [9] with 8-bit inputs at 45nm. In addition, to study the scalability potential of neuromorphic architecture implementation in WIF, we have studied the scalability aspects of core arithmetic circuits, and benchmarked with equivalent CMOS; 4-, 8-, 16- and 32-bit ripple carry adders were designed and compared against equivalent CMOS at 45nm.

Table 3. Comparison of Neuron Implementation in WIF vs. CMOS

	Area (μm^2)		Delay (ps)		Power (μW)	
	CMOS	WIF	CMOS	WIF	CMOS	WIF
16-bit Neuron Core	1785	30.9	1480	726	18000	23.2

Table 4. Scalability Study of High-Bit Width Arithmetic Circuits For Neuromorphic Computation

	Area (μm^2)		Delay (ps)		Power (μW)	
	CMOS	WIF	CMOS	WIF	CMOS	WIF
16-bit Adder	1700	27	1400	515	14600	17
32-bit Adder	3410	54	2800	915	29200	33

WIF Parameters: [λ=100nm, ME cell area = λxλ, ME delay = 100ps, Wave velocity = 10^4 m/s, ME switching power =100nW]

For WIF evaluation the parameters were based on experimental evidence and numerical simulations [12][7][3]: The wavelength of the spin wave was taken to be 100nm. Accordingly, ME cell dimensions of 100nmx100nm were used, and the spin wave bus length was considered in multiples of 100nm. The group velocity of the spin waves was assumed to be 10^4 m/s. The switching delay of the ME cell was taken to be 100ps. The total delay of a WIF circuit was calculated as the sum of ME cell switching delay and propagation delay of the spin waves along the longest path (critical path delay).

Energy consumption in WIF is mainly attributed to ME cell switching for generating new waves, amplification and latching. Spin wave propagation does not involve any physical movement of charge particles. Therefore, total energy of circuits is calculated based on the number of ME cells (N_{ME}) and the energy consumption per ME cell (E_{ME}):

$$E = N_{ME} \times E_{ME}$$

The ME cell structure represents a parallel plate capacitor consisting of a non-magnetic metallic layer (e.g. Al), a layer of piezoelectric material (e.g. PZT), and a conducting magnetostrictive material (e.g. Ni). The total energy consumed by ME cell per switch can be calculated as follows [4][14]:

$$E_{ME} = \frac{CV^2}{2} = \frac{\epsilon_0 \epsilon_r AV_{\pi/6}^2}{2d}$$

Where ϵ_0 is the vacuum permittivity, ϵ_r is the relative permittivity of the piezo-electric, A is the surface area of the ME cell, d is the thickness of the dielectric layer, $V_{\pi/6}$ is the voltage required for 30 degree magnetization rotation. In order to provide high-frequency spin wave excitation, the thickness of the piezoelectric layer should be adjusted to the spin wave frequency (e.g. d = 0.8μm for resonance frequency of 1GHz). Taking the following data: ϵ_0 = 8.854 × 10^{-12}F /m, ϵ_r = 1700 for PZT, A = 100nm ∗ 100nm, d = 0.8μm, $V_{\pi/6}$ = 0.4MV/m × 0.8μm = 0.32V, we would require approximately 10aJ of energy for ME cell switching. The energy per switch scales proportional to the size of the ME cell and may be reduced further by optimizing the material structure and switching dynamics.

Area of WIF circuits was calculated based on ME cell dimensions and patterning area required for SWB with the above assumptions. All quaternary adders were designed using multi-valued operators, and followed the design principles illustrated previously. CMOS designs for the adders were defined in Verilog and synthesized using Synopsys Design Compiler in 45nm node using North Carolina State University (NCSU) Product Development Kit (PDK). Performance and power for CMOS were calculated using HSPICE simulations. The benchmarking results are shown in Table 3 and Table 4.

Promising benefits are achieved across all metrics for WIF-based implementations. As shown in Table 3, WIF based 4-digit Quaternary neuron implementation achieves 57x density, 775x power and 2x performance benefits over equivalent 8-bit Boolean Digital CMOS-based design at 45nm technology node. Our scalability to higher bit-width study (see Table 4) indicates that even more substantial benefits can be attained if the neuron implementation is extended to 16-digits. The 16-digit quaternary full adder shows 63x density, 884x lower power and 3x performance improvement vs. 32-bit CMOS.

The substantial improvement in power consumption is due to the low ME cell switching power, and overall low-energy computation and communication in WIF without charge transfer. The density benefits are primarily due to WIF's inherent support for multi-valued logic. The implementation through multi-valued operators leads to compact circuits. Reduced communication requirements are achieved through multi-valued wave propagation.

VI. Conclusion

We presented a new approach towards neuromorphic computing using a spin-domain multi-valued WIF framework. The neuron circuit presented, leverages intrinsic properties of WIF's nanoscale components for multi-valued data representation, communication and computation, as well as its core logical constructs to achieve functionality equivalent of a biological neuron. Significant benefits were projected across all aspects; the WIF framework based neuron implementation showed 57x, 775x and 2x benefits in terms of area, power, and performance compared to an equivalent 45nm CMOS design. The scalability studies for higher bit-width neuromorphic architectures indicated that additional benefits can be attained with bit-width scaling; the 16-digit quaternary adder design in WIF showed 63x density, 884x power and 3x performance benefits compared to a 32-bit CMOS adder at 45nm technology. The transformative new approach for neuromorphic computing presented in this paper, can provide the basis for novel neuromorphic architectures.

References

[1] A. S. Cassidy, J. Georgiou, and A. G. Andreou, "Design of silicon brains in the nano-CMOS era: Spiking neurons, learning synapses and neural architecture optimization", *Neural Networks*, vol. 45, pp. 4-26, Sept. 2013.

[2] S. Khasanvis, M. Rahman, S. N. Rajapandian, and C. A. Moritz, "Wave-based Multi-valued Computation Framework", in *Proceedings of IEEE/ACM International Symposium on Nanoscale Architectures (NanoArch)* , pp. 171-176 , 2014

[3] A. Khitun, and K. L. Wang,. Non-volatile magnonic logic circuits engineering. *Arxiv.org*, 2010

<http://arxiv.org/abs/1012.4768>

[4] P. Shabadi, S. N. Rajapandian, S. Khasanvis, and C. A. Moritz, "Design of spin wave functions-based logic circuits," *SPIN*, vol. 2, no. 3, Article 1240006, World Scientific Publishing Company, 2012.

[5] S. Khasanvis, M. Rahman**,** P. Shabadi and C. A. Moritz, "Unconventional Computation with Spin Wave Functions", Nanomagnetic and Spintronic Devices for Energy Efficient Computing. *Wiley Publishers* (New York 2014) (Book Chapter- In Press)

[6] M. Covington, T. M. Crawford, and G. J. Parker, "Time-resolved measurement of propagating spin waves in ferromagnetic thin films," *Phys. Rev. Lett.*, vol. 89, pp. 237202-1–237202-4, 2002.

[7] M. P. Kostylev, A. A. Serga, T. Schneider, B. Leven, and B. Hillebrands, "Spin-wave logical gates," *Appl. Phys. Lett.*, vol. 87, pp. 153501-1– 153501-3, 2005.

[8] P. Shabadi, A. Khitun, P. Narayanan, M. Bao, I. Koren, K. L. Wang, and C. A. Moritz, "Towards logic functions as the device," in *proceedings of IEEE/ACM International Symposium on Nanoscale Architectures (NanoArch)*, pp.11,16, 17-18 June 2010.

[9] J. Seo, et. al., "A 45nm CMOS neuromorphic chip with a scalable architecture for learning in networks of spiking neurons," IEEE *Custom Integrated Circuits Conference (CICC),*, vol. 1, no. 4, pp. 19-21 Sept. 2011

[10] A. Khitun, M. Bao, and K. L. Wang, "Spin wave magnetic nanofabric: A new approach to spin-based logic circuitry," *IEEE Transactions on Magnetics*, vol. 44, no. 9, pp.2141-2152, Sept. 2008.

[11] D. M. Miller, and M. A. Thornton, "Multiple valued logic: Concepts and representations," *Synthesis Lectures on Digital Circuits and Systems Series*, Morgan & Claypool, 2008.

[12] P. Shabadi, A. Khitun, K. Wong, P. K. Amiri, K. L. Wang, and C. A. Moritz, "Spin wave functions nanofabric update," in proc. of *IEEE/ACM International Symposium on Nanoscale Architectures (NanoArch)*, pp.107-113, 8-9 June 2011.

[13] T. Schneider, A. A. Serga, B. Leven, B. Hillebrands, R. L. Stamps, and M. P. Kostylev, "Realization of Spin Logic Gates," *Applied Physics Letters*. vol 92, 022505, 2008.

[14] P. Shabadi and C. A. Moritz, "Post-CMOS hybrid spin-charge nanofabrics," in proceedings of *IEEE Conference on Nanotechnology (IEEE-NANO)*, pp.1399-1402, 15-18 Aug. 2011.

[15] K. L. Wang and P. K. Amiri, "Nonvolatile Spintronics: Perspectives on Instant-On Nonvolatile Nanoelectronic Systems," *SPIN*, vol. 2, no. 2, Article 1250009, World Scientific Publishing Company, 2012.

[16] W. Yu, J. A. Bain, W. C. Uhlig and J. Unguris, "The effect of stress-induced anisotropy in patterned FeCo thin-film structures," *J. Appl. Phys.*, vol. 99, no. 8, p. 8B706, Apr. 2006.

[17] S. Cherepov et al., "Electric-field-induced spin wave generation using multiferroic magnetoelectric cells," *Appl. Phys. Lett.*, vol. 104, no. 8, pp. 082403, Feb. 2014.

[18] A. Khitun, D. E. Nikonov and K. L. Wang, "Magnetoelectric spin wave amplifier for spin wave logic circuits," *J. Appl. Phy.*, vol. 106, no. 12, p. 123909, Dec. 2009.

[19] S. Khasanvis, M. Rahman, C. A. Moritz, "Unconventional Nanocomputing with Physical Wave Interference Functions," *Nanomagnetic and Spintronic Devices for Energy Efficient Memory and Computing*, Eds. S. Bandyopadhyay and J. Atulasimha, Wiley, 2015. In Press