Wave-based Multi-valued Computation Framework

Santosh Khasanvis, Mostafizur Rahman, Sankara Narayanan Rajapandian*, and Csaba Andras Moritz
Department of Electrical and Computer Engineering, University of Massachusetts Amherst, MA, USA
*Nvidia Corporation, CA, USA
E-mail: skhasanv@umass.edu, andras@ecs.umass.edu

***Abstract*—We present a novel multi-valued computation framework called Wave Interference Functions (WIF), based on emerging non-equilibrium wave phenomenon such as spin waves. WIF offers new features for data representation and computation, which can be game changing for post-CMOS integrated circuits (ICs). Information encoding wave attributes inherently leads to multi-dimensional multi-valued data representation and communication. Multi-valued computation is natively supported with wave interactions, such as wave superposition or interference. We introduce the concept of a multi-valued Interference Function that is more sophisticated than conventional Boolean and Majority functions, leading to compact circuits for logic. We present WIF implementation of multi-valued operators to realize any desired logic/arithmetic function using the Interference Function. We evaluate 2-digit to 16-digit quaternary (radix-4) full adder designs with WIF operators in terms of power, performance and area. Estimates indicate up to 63x higher density, 884x lower power and 3x better performance when compared to equivalent 45nm CMOS adders. WIF features completely change conventional assumptions on circuit design, opening new avenues to implement future nanoscale ICs for general purpose processing and other applications inherently suited to multi-valued computation.**

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

I. INTRODUCTION

Multi-valued logic is defined as the logical calculus that involves more than two logic levels. It allows compact data and functional representation, and is more efficient compared to binary logic. As a result, multi-valued logic implementations have been long sought for general purpose processors, and for applications that are inherently more suited to multi-valued computation such as image processing, big data analytics, many-valued decision diagrams, artificial neural networks etc. However, the transition from binary to multi-valued logic based implementations has been unsuccessful so far, since conventional approaches use digital CMOS technology (which is tailored for binary logic and operates with binary switches) for hardware emulation of multi-valued constructs that is very inefficient.

In this paper, we present a new multi-valued computational paradigm that uses wave physical phenomenon at nanoscale, called Wave Interference Functions (WIF). The vision is to use elementary functional elements that are more sophisticated than switches as the building blocks, to compactly realize multi-valued logic/arithmetic (see Fig. 1). In contrast to CMOS, WIF's features allow inherent data representation, communication and computation in the multi-valued domain. Information is encoded in a combination of wave attributes – amplitude and phase, which is *multi-dimensional*: each of the wave attributes can take multiple values, and when used in conjunction they present a plethora of options for multi-valued data encoding. Information processing is achieved through wave superposition interactions and wave propagation, both of which can affect wave phase and amplitude. The output wave is also multi-valued, encoding the result and information about the inputs in a compressed manner using both its phase and amplitude. We introduce the concept of *Interference Function* that mathematically captures these superposition interactions, and show this is a more sophisticated function than conventional Boolean or Majority. We develop a formalism for multi-valued logic implementation using the Interference Function as the basic element. These features enable efficient multi-valued logic implementation with compressed communication and computation.

We illustrate the WIF framework using spin waves physical phenomenon in this paper. But the ideas presented here are generic and broadly applicable to any phenomenon that exhibits wave-like behavior. Prior research efforts have proposed using spin waves to implement conventional Boolean or Majority logic [1][2]; however these approaches do not harness the full potential and benefits of wave computation. The WIF framework presented here is a new direction where intrinsic wave properties and sophisticated Interference Functions are leveraged for computation and communication, rather than mapping it onto conventional computation paradigms. This could result in significant benefits in terms of power, performance and area compared to CMOS approach.

The rest of the paper is organized as follows: Section II presents a brief overview on physical layer components to operate on spin waves. Section III presents multi-valued data representation, and elementary WIF operators are introduced in Section IV. Section V discusses the definition and WIF implementation of multi-valued operators for logic. Using

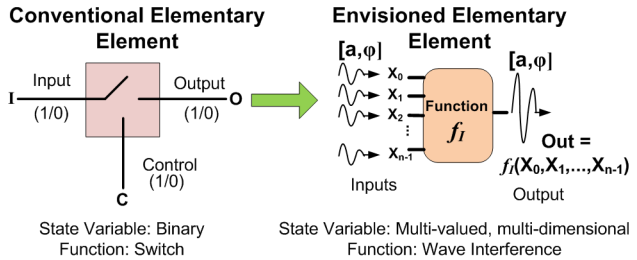

Fig. 1. (Left) Conventional elementary functional element (i.e., binary switch). (Right) Envisioned elementary functional element for Wave Interference Functions computation paradigm.

This project was supported by DARPA program on Non-volatile Logic, and Center for Hierarchical Manufacturing (CHM, NSF DMI-0531171) at UMass Amherst.

these operators, a WIF quaternary (radix-4) full adder design is presented and evaluated in Section VI. Section VII concludes the paper.

II. WIF Physical Layer

Spin waves, also known as magnons, are the collective oscillations of electrons spins in an ordered spin lattice around the direction of magnetization in ferromagnetic materials. The key fabric components [2] required for operating on spin waves are – ferromagnetic waveguides called Spin Wave Bus (SWB) that facilitate spin wave propagation and superposition, and Magneto-Electric (ME) cells (see Fig. 2a). 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) generating and detecting spin waves by converting electric signals into magnetic domain and vice versa, (ii) amplifying spin waves for logic, as well as restoration of wave amplitudes in spin wave bus for signal integrity, and (iii) storing information encoded in the state of its magnetization that can be altered by incoming spin waves.

III. 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. We limit the phase to be either 0 or π in the data encoding used in this paper, since the ME cells used for spin wave detection are designed to differentiate these phases. However, additional phases may be used as per physical component capabilities as well.

Here, we introduce the notations that will be used in subsequent sections for multi-valued logic design. A spin wave is denoted as $\tilde{X}$; and is represented using polar co-ordinates to incorporate both its amplitude (a) and phase (φ) compactly as follows:

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

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. 2b and Table I). 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.

Table I. 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	$\tilde{L}_3 = 3Ae^{i\pi}$	($3A$, π)

IV. Elementary WIF Operators

A. Interference Function

Wave interference is the fundamental operation in the WIF approach. Spin waves interfering at a given point exhibit linear superposition behavior [3]. Elementary spin wave circuit operation has been experimentally demonstrated at room temperature [14]. 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:

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

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

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

In general for n input waves, if the amplitude of any wave

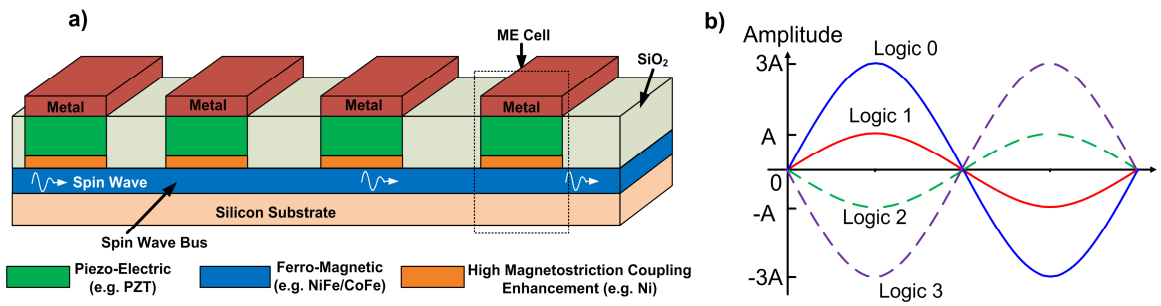

Fig. 2. a) WIF Physical fabric components to operate on spin waves; and b) Illustration of quaternary data representation (radix-4) with waves, encoded in the combination of phase and amplitude.

$\tilde{X}_j$ 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:

$$I^{\varphi}(\tilde{X}_0, \tilde{X}_1, \dots, \tilde{X}_{n-1}) = \begin{cases} \pi; & if \ \sum w_j A e^{i\pi} > \sum w_k A e^{i0} \\ 0; & else \end{cases} \quad (4)$$

$\rightarrow$ weighted-majority decision

$$I^{A}(\tilde{X}_0, \tilde{X}_1, \dots, \tilde{X}_{n-1}) = \left|\sum w_k A e^{i0}\right| - \left|\sum w_j A e^{i\pi}\right| .$$

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* is much more than a simple Majority, and it results in a spin wave that encodes all the necessary information about the inputs in a compressed manner.

B. Identity Operator

The *identity* operator takes an input wave and provides the same wave at the output. This is analogous to a wire in the electrical domain, which maintains the same voltage at the output as the input.

C. Complement Operator

This is used to perform the inversion function. For radix-*r*, it is represented using the following equation:

$$\bar{x} = (r-1) - x, \text{ where } x \in \{0, 1, \dots, r\text{-}1\}. \quad (5)$$

Here, x represents the logical input value in radix-r number system. This is analogous to a Boolean NOT operation. Given a spin wave $\tilde{X} = ae^{i\varphi}$, corresponding to the logical value x, the inversion operator (with "–" sign) is then defined as follows:

$$-\tilde{X} = -ae^{i(\varphi)} = ae^{i(\varphi+\pi)} = \begin{cases} -a; & \text{if } \varphi = 0 \\ a; & \text{if } \varphi = \pi \end{cases} \quad (6)$$

Physically, this means that the inversion operator introduces a phase shift of π for a given spin wave, as a consequence of the choice of data representation.

D. Physical Implementation of Elementary WIF Operators

The physical implementation of the WIF operators turns out to be quite simple without requiring any active devices (see Fig. 3). The *Interference Function* is simply a junction of spin wave buses. The *Identity* operator is a spin wave bus whose length is an integral multiple of the wavelength (λ). This ensures that the output phase is the same as the input wave phase. The *Complement* operator can be implemented by using a spin wave bus that has a length equal to an odd multiple of the half-wavelength.

V. WIF Operators for Multi-valued Logic

Multi-valued algebra provides the necessary framework for expressing and manipulating multi-valued functions. Similar to Boolean algebra that uses {AND, OR, NOT} operators, multi-valued algebra uses a functionally complete set of {*Min*, *Max*, *Literal*, *Cyclic*} operators for realizing any multi-valued logic function [4][13]. Using these operators, any function in multi-valued domain can be expressed as a sum-of-products (SOP), and reduction techniques have been extensively studied in literature to minimize a multi-valued SOP expression [5][11][12]. To construct these operators *Identity*, *Complement*,

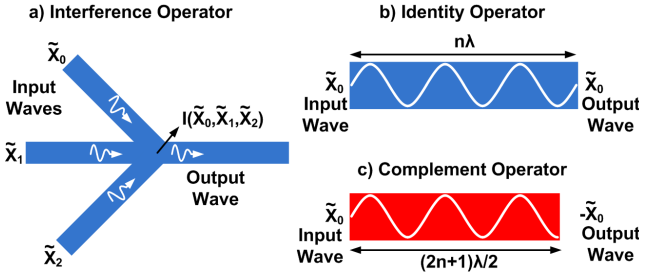

Fig. 3. Physical implementation of WIF elementary operators with spin wave bus for a) *Interference* function; b) *Identity* operator; and c) *Complement* operator. Here n is an integer and λ is the spin wavelength in Fig. 3b-c.

Upper Threshold, *Lower Threshold* and *Truncated Difference* operators are used [4]-[7]. *Identity* and *Complement* operators using WIF were discussed in the previous section. In the following discussion, the rest of the multi-valued logic operators are defined and some of their WIF implementations are presented using quaternary logic as an example. Remaining operators can be implemented following the approach shown based on their *Interference Function* expressions.

A. Upper Threshold and Lower Threshold Operators

These operators use two-inputs (x, y), and realize multi-valued threshold functions. In these operations, when one input is above or below the other input in terms of logic value, a constant output is selected. The *Upper Threshold* operator (x_y^{r-1}) is defined as:

$$x_y^{r-1} = \begin{cases} r-1, & \text{when } x \geq y \\ 0, & else \end{cases}, x, y \in \{0, 1, \dots, r\text{-}1\}. \quad (7)$$

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

$$\tilde{X}_y^{r-1} = I\left[(r-1) I_1^{\varphi}(-\tilde{X}, \tilde{Y}, \tilde{L}_{r/2})\right], \quad (8)$$

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. 4a shows the truth table for *Upper Threshold* operator and physical implementation in WIF for quaternary logic (r = 4).

The *Lower Threshold* operator $({}_y^{r-1}x)$ is defined as

$${}_y^{r-1}x = \begin{cases} r-1, & \text{if } x \leq y \\ 0, & else \end{cases}. \quad (9)$$

It is implemented the same way as the *Upper Threshold* operator, but inputs are interchanged. The *Interference Function* to express *Lower Threshold* operation is:

$${}_y^{r-1}\tilde{X} = \mathbf{I}\left[(r-1) I_1^{\varphi}(\tilde{X}, -\tilde{Y}, \tilde{L}_{r/2})\right]. \quad (10)$$

B. *Truncated Difference Operator*

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

$$x \,\Xi\, y = \begin{cases} x - y\,, \text{when } x > y \\ 0, \quad else \end{cases}, \qquad x, y \in \{0, 1, \dots, r-1\}. \tag{11}$$

This can be expressed with *Interference Function* as

$$\tilde{X} \,\Xi\, \tilde{Y} = \mathbf{I}(\tilde{X}, -\tilde{Y}, \tilde{L}_0), \tag{12}$$

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. 4b 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.

Using these operators, we discuss WIF implementation of *Min*, *Max*, *Literal* and *Cyclic* operators to enable any arbitrary multi-valued logic function realization.

C. *Min Operator*

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

$$x \cdot y = \begin{cases} x, & x < y \\ x - (x - y), & else \end{cases}, \qquad x, y \in \{0, 1, \dots, r-1\}. \tag{13}$$

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:

$$\text{Min}(\tilde{X}, \tilde{Y}) = \tilde{X} \,\Xi\, (\tilde{X} \,\Xi\, \tilde{Y}) = \mathbf{I}[\tilde{X}, -(\tilde{X} \,\Xi\, \tilde{Y}), \tilde{L}_0], \tag{14}$$

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. Fig. 4c shows the truth table and the WIF physical implementation of *Min* operator.

D. *Max Operator*

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

$$x + y = \begin{cases} x, & x > y \\ x + (y - x), & else \end{cases}, \qquad x, y \in \{0, 1, \dots, r-1\}. \tag{15}$$

The functional representation in terms of *Interference Function* is

$$\text{Max}(\tilde{X}, \tilde{Y}) = \tilde{X} + (\tilde{Y} \,\Xi\, \tilde{X}) = \mathbf{I}[\tilde{X}, (\tilde{Y} \,\Xi\, \tilde{X}), \tilde{L}_{r-1}], \tag{16}$$

where $\tilde{L}_{r-1}$ is a reference wave corresponding to logic value r-1.

E. *Literal Operator*

This operator combines both *Upper Threshold* and *Lower Threshold* operators, and provides more flexibility for conditional operations. The notation for *Literal* operator is ${}^{p}x^{q}$. The output conditions are defined as:

$${}^{p}x^{q} = \begin{cases} r-1, & p \le x \le q \\ 0, & else \end{cases}, \qquad p, q, x, y \in \{0, 1, \dots, r-1\}. \tag{17}$$

The *Interference Function* implementing literal operator using *Upper Threshold* and *Lower Threshold* operators is

$${}^{p}\tilde{X}^{q} = \mathbf{I}(\tilde{X}_p^{r-1}, {}_{q}^{r-1}\tilde{X}, \tilde{L}_0). \tag{18}$$

F. *Cyclic Operator*

This is analogous to Boolean XOR, and is defined as

$$x \oplus y = (x +_{add} y) \bmod r, \qquad x, y \in \{0, 1, \dots, r-1\}. \tag{19}$$

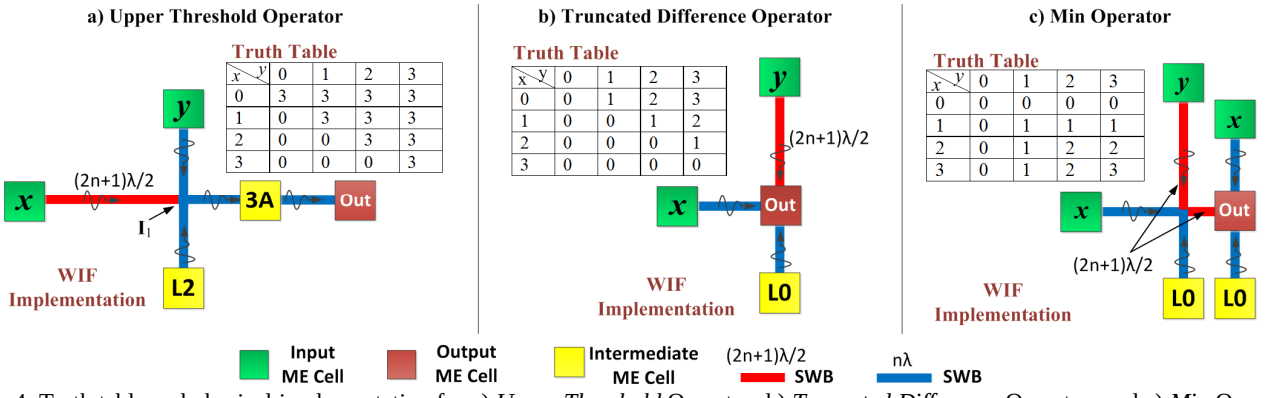

Fig. 4. Truth table and physical implementation for a) *Upper Threshold* Operator; b) *Truncated Difference* Operator; and c) *Min* Operator. The intermediate ME cell labeled '3A' generates a spin wave with phase equal to input phase and constant amplitude 3A. Other intermediate ME cells labeled 'L0' and '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.

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

$$x +_{\text{carry}} y = \begin{cases} 1, & \text{if } x +_{add} y > r-1 \\ 0, & else \end{cases} \qquad (20)$$
$$x, y \in \{0, 1, \dots, r-1\}.$$

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

$$\tilde{X} +_{\text{carry}} \tilde{Y} = \text{Min}\left[\mathbf{I}\left(\tilde{X}, \tilde{Y}, \tilde{L}_0\right), \tilde{L}_1\right]. \qquad (21)$$

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

The *Cyclic* operator is then implemented as:

$$\tilde{X} \oplus \tilde{Y} = \mathbf{I}\left[\tilde{A}, \tilde{B}, \tilde{L}_0, {}^{r-1}_{r}(\tilde{X} +_{\text{add}} \tilde{Y}), -(\tilde{X} +_{\text{carry}} \tilde{Y})\right]. \qquad (22)$$

Here, ${}^{r-1}_{r}(\tilde{X} +_{\text{add}} \tilde{Y})$ implements the *Lower Threshold* operation, whose output is r–1 if $x +_{\text{add}} y \leq r$–1, and 0 otherwise.

VI. WIF Example: Quaternary Full Adder

A. Quaternary Full Adder Design

The use of multi-valued operators for circuit design reduces complexity significantly and provides a framework for arbitrary logic/arithmetic implementation. In this section, we present a quaternary full adder as an example using WIF multi-valued constructs described earlier. 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:

$$C_{out} = \begin{cases} 1, & \text{if } A +_{add} B +_{add} C_{in} \geq r \\ 0, & else \end{cases} \qquad (23)$$
$$A, B \in \{0, 1, \dots, r-1\} \text{ and } C_{in} \in \{0, 1\}.$$

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:

$$\tilde{C}_{out} = \tilde{A} +_{\text{carry}} \tilde{B} +_{\text{carry}} \tilde{C}_{in} = \text{Min}\left[\mathbf{I}\left(\tilde{A}, \tilde{B}, \tilde{C}_{in}\right), \tilde{L}_1\right], \qquad (24)$$

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:

$$\begin{cases} A +_{add} B +_{add} C_{in} - r, & \text{if } A +_{add} B +_{add} C_{in} > r-1 \\ A +_{add} B +_{add} C_{in}, & else \end{cases} \qquad (25)$$

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

$$\tilde{S}_{out} = \tilde{A} \oplus \tilde{B} \oplus \tilde{C}_{in} = \mathbf{I}\left(\tilde{A}, \tilde{B}, \tilde{C}_{in}, {}^{r-1}_{r}\tilde{X}, -\tilde{C}_{out}\right), \qquad (26)$$
$$\text{where } \tilde{X} = (\tilde{A} +_{add} \tilde{B} +_{add} \tilde{C}_{in}).$$

The WIF implementation of equations (24) and (26) are shown in Fig. 5.

B. Benchmarking vs. CMOS

To evaluate the potential of multi-valued logic implementation in WIF, extensive benchmarking was done with respect to binary CMOS for equivalent 4-, 8-, 16- and 32-bit ripple carry adder designs. For WIF evaluation the parameters were as follows based on experimental evidence and numerical simulations [9][10]: 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. ME cell switching energy was presumed to be 10aJ.

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). The propagation of the spin waves does not involve any movement of charge and hence there is no energy consumed for the propagation and the interference of waves. Accordingly, the total energy consumed by the circuit depends on the total number of ME cells that are switching. 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

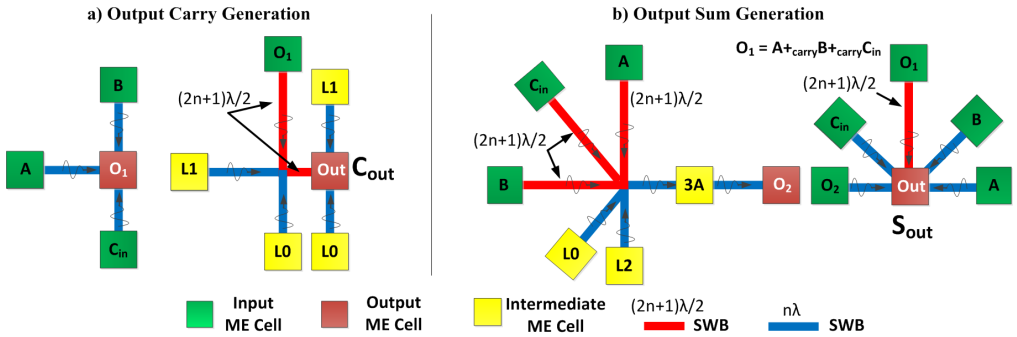

Fig. 5. Quaternary full adder implementation in WIF for: a) Carry function (C_{out}); and b) Sum function (S_{out}).

TABLE II. COMPARISON BETWEEN QUATERNARY WIF AND BINARY 45NM CMOS FULL ADDER DESIGNS

Adder Bit-Width	Area (μm^2)		Delay (ps)		Power (μW)	
	CMOS	WIF	CMOS	WIF	CMOS	WIF
4-bit	430	7	550	225	3200	5
8-bit	850	14	750	315	7300	9
16-bit	1700	27	1400	515	14600	17
32-bit	3410	54	2800	915	29200	33

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

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 II.

Tremendous benefits are achieved across all metrics for quaternary full adder designs using WIF vs. binary CMOS. The results also indicate increase in benefits with higher bit-width implementations, suggesting WIF's scalability potential. The 2-digit quaternary full adder design showed 61x density, 640x lower power and 2.2x performance advantage vs. CMOS binary 4-bit adder, whereas the 16-digit quaternary full adder showed 63x density, 884x lower power and 3x performance improvement vs. 32-bit CMOS. The improvement in power consumption is due to low ME cell switching power, and low energy computation and communication without charge transfer. The density benefits are primarily due to WIF's inherent support for multi-valued logic, compressed functional implementation through multi-valued operators leading to compact circuits, and reduced communication requirements through multi-valued wave propagation. These factors also contribute significantly towards performance improvements. Estimation results showed up to 3x performance improvement vs. CMOS, despite the fact that spin wave propagation is slower than charge by 10x [15][16]. While the initial designs considered here were ripple-carry adders, more benefits may be obtained through architectural optimization. Additional aspects need to be considered for large-scale designs such as communication and clocking. Large distance communication may be addressed by using charge based interconnects to reduce propagation delays, with a trade-off in power consumption.

VII. SUMMARY

We have presented a new multi-valued computation framework using waves for post-CMOS integrated circuits, called Wave Interference Functions (WIFs). Spin waves were used to illustrate WIF paradigm, however it is generic and applicable to other wave phenomena as well. This is a completely new direction compared to previous approaches that map wave interactions to conventional Boolean and Majority logic. We leverage the intrinsic *Interference Functions* to realize multi-valued logic resulting in compact circuit implementation, and compressed multi-valued data encoding and communication reduces interconnect requirements. WIF showed tremendous benefits in terms of power, performance and area compared to equivalent 45nm CMOS adder designs even at higher bit-width. This new paradigm completely changes conventional assumptions on circuit and processor designs. It could potentially lead to new processor micro-architecture directions that exploit compact functional implementation with WIF to extract much higher parallelism than what can be achieved today. It also provides a pathway to efficient post-CMOS IC implementations for applications like image processing, big data analytics, many-valued decision diagrams, artificial neural networks etc. that are inherently suited for multi-valued computation.

REFERENCES

[1] 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.

[2] 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.

[3] 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.

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

[5] C. M. Allen, and D. D. Givone, "A minimization technique for multiple-valued logic systems," IEEE Transactions on Computers, vol. C-17, no. 2, pp. 182 – 184, Feb. 1968.

[6] S. P. Onneweer, and H. G. Kerkhoff, "High-radix current-mode CMOS circuits based on the truncated-difference operator," In Proc. 17th ISMVL, pp. 188–195, 1987.

[7] T. Temel, and A. Morgul, "Implementation of multi-valued logic, simultaneous literal operations with full CMOS current-mode threshold circuits," Electronics Letters 38, 4, pp. 160 –161, Feb. 2002.

[8] T. Temel, and A. Morgul , "Implementation of multi-valued logic gates using full current-mode CMOS circuits," Analog Integrated Circuits and Signal Processing 39, 191–204, 2004.

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

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

[11] P. P. Tirumalai, and J. T. Butler, "Prime and non-prime implicants in the minimization of multiple-valued logic functions," in proceedings of Nineteenth International Symposium on Multiple-Valued Logic (1989), pp.272,279, 29-31 May 1989.

[12] J. Yunjian, and R. K. Brayton, "Don't cares and multi-valued logic network minimization," in proceedings of IEEE/ACM International Conference on Computer Aided Design, (ICCAD-2000), pp.520,525, 5-9 Nov. 2000.

[13] Z. G. Vranesic, E. S. Lee, and K. C. Smith, "A many-valued slgebra for switching systems," IEEE Transactions on Computers, vol.C-19, no.10, pp.964-971, Oct. 1970.

[14] 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 2010 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH), pp.11,16, 17-18 June 2010.

[15] S. Rakheja, A. Naeemi, and J. D. Meindl, "Physical limitations on delay and energy dissipation of interconnects for post-CMOS devices," in proceedings of 2010 International Interconnect Technology Conference (IITC), pp. 1-3, 6-9 June 2010.

[16] P. Shabadi and C. A. Moritz, "Post-CMOS hybrid spin-charge nanofabrics," in proceedings of 2011 11th IEEE Conference on Nanotechnology (IEEE-NANO), pp.1399-1402, 15-18 Aug. 2011.