

# An Accuracy Tunable Non-Boolean Co-Processor Using Coupled Nano-Oscillators

NEEL GALA, Indian Institute of Technology Madras

SARADA KRITHIVASAN, National Institute of Technology Trichy

WEI-YU TSAI, XUEQING LI, and VIJAYKRISHNAN NARAYANAN,

The Pennsylvania State University

V. KAMAKOTI, Indian Institute of Technology Madras

As we enter an era witnessing the closer end of Dennard scaling, where further reduction in power supply-voltage to reduce power consumption becomes more challenging in conventional systems, a goal of developing a system capable of performing large computations with minimal area and power overheads needs more optimization aspects. A rigorous exploration of alternate computing techniques, which can mitigate the limitations of Complementary Metal-Oxide Semiconductor (CMOS) technology scaling and conventional Boolean systems, is imperative. Reflecting on these lines of thought, in this article we explore the potential of non-Boolean computing employing nano-oscillators for performing varied functions. We use a two coupled nano-oscillator as our basic computational model and propose an architecture for a non-Boolean coupled oscillator based co-processor capable of executing certain functions that are commonly used across a variety of approximate application domains. The proposed architecture includes an accuracy tunable knob, which can be tuned by the programmer at runtime. The functionality of the proposed co-processor is verified using a soft coupled oscillator model based on Kuramoto oscillators. The article also demonstrates how real-world applications such as Vector Quantization, Digit Recognition, Structural Health Monitoring, and the like, can be deployed on the proposed model. The proposed co-processor architecture is generic in nature and can be implemented using any of the existing modern day nano-oscillator technologies such as Resonant Body Transistors (RBTs), Spin-Torque Nano-Oscillators (STNOs), and Metal-Insulator Transition (MITs). In this article, we perform a validation of the proposed architecture using the HyperField Effect Transistor (FET) technology-based coupled oscillators, which provide improvements of up to 3.5× increase in clock speed and up to 10.75× and 14.12× reduction in area and power consumption, respectively, as compared to a conventional Boolean CMOS accelerator executing the same functions.

CCS Concepts: • **Hardware** → **Emerging architectures**; *Emerging technologies*;

Additional Key Words and Phrases: Non-boolean computing, coupled oscillators, co-processor, Kuramoto, vector quantization, digit recognition, structural health monitoring, micro-architecture

This work is supported by the National Science Foundation under grant 1317560 and 1640081.

Authors' addresses: N. Gala and V. Kamakoti, Dept. of Computer Science and Engineering, IIT Madras; email: neelgala@gmail.com; S. Krithivasan, Dept. of Electrical and Electronics Engineering, NIT Trichy; email: ksharadak@gmail.com; W.-Y. Tsai, X. Li, and V. Narayanan, Dept. of Computer Science and Engineering, The Pennsylvania State University; emails: wzt114@psu.edu, lixueq@gmail.com, vxn9@cse.psu.edu.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.

© 2017 ACM 1550-4832/2017/09-ART1 \$15.00

<https://doi.org/10.1145/3094263>

**ACM Reference format:**

Neel Gala, Sarada Krithivasan, Wei-Yu Tsai, Xueqing Li, Vijaykrishnan Narayanan, and V. Kamakoti. An Accuracy Tunable Non-Boolean Co-Processor Using Coupled Nano-Oscillators. *J. Emerg. Technol. Comput. Syst.* 14, 1, Article 1 (September 2017), 28 pages.  
<https://doi.org/10.1145/3094263>

**1 INTRODUCTION**

Power consumption is directly proportional to both the performance of a chip and its functionality. Scaling of supply voltage and frequency are some of the common techniques employed to manage power in high-performance complex System-on-Chips (SoCs). However, the increase in static leakage power in Complementary Metal-Oxide Semiconductor (CMOS) devices has resulted in dealing with even greater challenges in the context of the classical or Dennard scaling (Dennard et al. 1974; Bohr 2007), thus making further reduction of supply voltage to reduce power consumption in conventional Boolean systems difficult. As a consequence, every progressive generation of CMOS technology has observed a consistent reduction in the net clock frequency increase over its previous generation. These limitations have led to high cooling costs in high-performance computing nodes and also driven a move to the multi-core era. Nonetheless, the inability to fit more cores within constrained power budgets has been projected to threaten even this new scaling paradigm (Esmailzadeh et al. 2011).

With the burst in mobile and hand-held computing devices such as smartphones, tablets, laptops, and so on, the nature of workloads (media processing, Digital Signal Processing (DSP) applications, image recognition, etc.) has observed a drastic deviation from conventional workloads. These workloads are known to be more context aware, have more natural interfaces, derive inputs directly from the analog world, and also share a common attribute that a precise answer at the output is not a necessity. To improve performance for such workloads, the industry has seen a renewal in *domain-specific processing*. This has led to an increased deployment of specialized accelerators/co-processors as part of general purpose computing systems (Kim et al. 2015; Price et al. 2014; Pham et al. 2012; Chen et al. 2014, 2015). Despite their benefits, these accelerators are limited to specific applications. Interestingly, the source of inputs for many of these applications is analog. These analog inputs are converted to digital and then processed. The question posed here is *whether one can avoid this conversion and work directly on the analog inputs*. If so, what would be the benefits in terms of *power consumption* and *performance*, the two big obstacles faced by CMOS technology. The rise of these challenges has led researchers to explore rigorously non-conventional devices beyond CMOS technology and also pursue alternate computing paradigms (Hoppensteadt and Izhikevich 1999; Nikonov et al. 2013; Narayanan et al. 2014). Along identical lines of thought, this article proposes *a non-Boolean computing co-processor using coupled nano-oscillators*.

Over the past years, nano-oscillator-based computing techniques (Roska et al. 2012; Shibata et al. 2012; Levitan et al. 2012; Csaba et al. 2012; Sharad et al. 2013) have earned significant recognition as an alternate computing paradigm. Prominent among them are spin torque oscillators and electrically coupled oscillator systems. These architectures believe in the notion of “*let physics do the computing*,” and hence, map applications to physical phenomenon. This has led to significant performance, area, and power benefits (Shukla et al. 2014). One such class of architectures is the set of weakly-coupled oscillators (Shibata et al. 2012). In Shibata et al. (2012), it is empirically shown that an array of such oscillators, when coupled together with distinctly close initial states, tend to synchronize. These synchronized oscillatory systems pose associative computational capabilities (Datta et al. 2014; Roska and Rodríguez-Vázquez 2002) that find use in a variety of application domains such as brain-inspired computing, neural networks, image processing, and the like. The

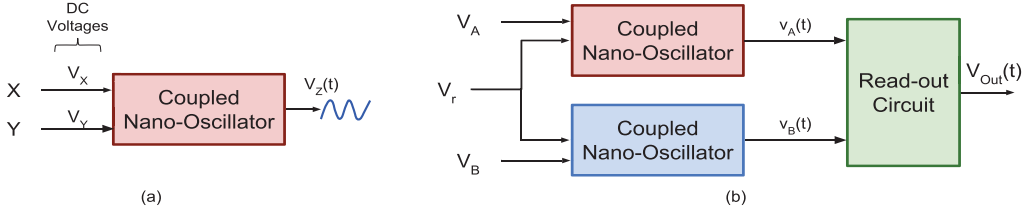


Fig. 1. (a) A generic coupled oscillator. (b) A pair of coupled oscillators with a read-out circuit to identify distance between input signals.  $V_A(t)$  and  $V_B(t)$  are tuned voltages corresponding to the input analog samples A and B.

image pixels, speech signals, and so on, from the *analog world* manifest themselves as voltages or currents that are supplied as inputs to the coupled oscillator circuits. Since these oscillator frameworks process the analog signals directly, the conversion from analog to digital domain is avoided. Additionally, the coupled oscillators can be implemented using only a few transistors, and therefore, the resulting circuit is not only compact in terms of area but also capable of working at a higher frequency while incurring lower power than its conventional Boolean counterparts.

Since the oscillators require the inputs to be within a certain range, the analog samples undergo necessary DC offsets, amplitude shrinking, and fine-tuning to map the sampled value to a unique voltage/current which can be directly applied to the coupled oscillator. Once the inputs with varying phases and frequencies are applied, the output of the coupled oscillator tries to settle down to a common phase or frequency. This process is known as *synchronization* and when a common phase or frequency is reached, the oscillator is said to be *locked-in* or *synchronized*. This locked-in mode is detected using peak detectors which assert a pulse when no more changes in phase or frequency are noticed at the output. A characteristic of coupled oscillators is the fact that *the time taken for a coupled oscillator to reach locked-in mode is directly proportional to the difference in the phase or frequency of the inputs*. This forms the basic computational model, which has been exploited in this article and explained with an example later in the text.

**The Basic Computational Model:** Figure 1(a) presents a generic coupled oscillator. Within each oscillator, there is a relaxation-oscillating signal, oscillating at the frequency positively correlated to the respective DC input voltage. In Figure 1(a),  $X$  and  $Y$  (numerical inputs) are represented by the DC voltages  $V_X$  and  $V_Y$ . The coupled oscillator block is composed of two voltage-controlled oscillators, fed with oscillating signals  $v_X(t)$  and  $v_Y(t)$ , each differing either in phase or frequency depending on the values of  $V_X$  and  $V_Y$ . The common output —  $v_Z(t)$  represents the coupled oscillating signal. This signal, over time, settles to a common frequency/phase. As noted earlier, the time required for the signal  $v_Z(t)$  to synchronize is directly proportional to the difference in the DC voltages  $V_X$  and  $V_Y$ . This property is exploited to perform different computations. As an example, consider two numerical samples A and B. Using appropriate tuning mechanisms (applying DC offset, shrinking amplitude, etc.), these samples are translated to DC signals  $V_A$  and  $V_B$ . Consider a base reference signal  $V_r$  that represents a reference numerical sample R. Now consider the setup in Figure 1(b) where two coupled oscillators are used. The top oscillator has the inputs  $V_A$  and  $V_r$  and the bottom oscillator has the inputs  $V_B$  and  $V_r$ . The top and bottom oscillators produce synchronized oscillating signals  $v_A(t)$  and  $v_B(t)$ , respectively. The read-out circuit reads these signals and, using appropriate peak-detectors and thresholding logic, identifies when each signal reaches synchronization (i.e., no more change in phase). The read-out circuit outputs the index of the oscillator that reaches synchronization first (0 for top and 1 for bottom) as well as the time taken by it to reach synchronization. If the time taken by the top oscillator to synchronize to  $v_A(t)$  is higher than the time taken by the bottom oscillator to reach  $v_B(t)$ , then the read-out circuit outputs a

1 indicating that  $A > B$ . Otherwise, the read-out circuit will output a 0 indicating  $B > A$ . Thus, by simply monitoring the synchronization times of the coupled oscillators, we can compare the samples  $A$  and  $B$ .

Many applications will need the actual value of the maximum analog signal in addition to the result of the comparison operation. This can be obtained by mapping the time required for synchronization by the oscillator to the frequency/phase of the input analog signal. The property that the time taken for synchronization is proportional to the phase/frequency difference of the input analog samples can be used to arrive at the required mapping function. Similar to how parameters of the CMOS standard cells are characterized by the foundry, the oscillator cells can also be *characterized* by the foundry yielding the mapping function. Since the inputs to the oscillators are bounded in value, the time taken for synchronization for various inputs will also be bounded. In this article, the feasibility of such a mapping is proved for the Kuramoto oscillator model (Chopra and Spong 2005). It is interesting to note that the aforementioned procedure obviates the need of an analog-to-digital conversion and has been highlighted in later sections.

Though a majority of the oscillator models proposed in literature have been optimized and proven to be stable and scalable in performing specific computations, most works have failed in providing a more generic architecture, which can employ such oscillatory systems in a co-processor-like environment. This article makes an attempt at providing a generalized nano-oscillator-based co-processor architecture that can execute certain functions that are commonly used across varied application domains. Following is a summary of the contributions of this article:

- (1) The article proposes a mechanism for realizing commonly employed computational functions using oscillator arrays in the context of approximate applications. It further uses a soft model based on Kuramoto oscillators to verify the correctness of the realization. These form the basic building blocks of the proposed co-processor. The different functions realized by these basic building blocks form the Instruction set of the co-processor. This work is novel and to the best of our knowledge, no work in the past has proposed such a generic oscillator-based co-processor.
- (2) The proposed architecture also provides a dynamic accuracy tunable knob that allows the programmer to change the accuracy of the outputs and gain benefits in runtime.
- (3) A set of real-world applications such as Vector quantization, Handwritten Digit Recognition, and Structural Health Monitoring are mapped onto the proposed co-processor architecture and the analysis of the tradeoff between accuracy and runtime is presented.
- (4) The article also addresses the steps and issues involved in validating the proposed architecture using the HyperField Effect Transistor (FET) oscillator technology and provides the benefits gained in terms of area, power, and speed as compared to a conventional Boolean logic-based accelerator.

The rest of the article is organized as follows. Section 2 highlights the related and past work. Section 3 presents an analysis of real-world applications leading to the identification of basic building blocks and thus the Instruction Set of the proposed co-processor architecture. Section 4 elaborates how each of the building blocks can be realized using oscillatory systems. It further proves the correctness of the realization by simulating the functionalities of the basic building blocks on a framework that uses a Kuramoto oscillator-based soft model. Section 5 elaborates how various applications can be mapped on to the proposed co-processor architecture. The methodology of implementing the proposed architecture using modern oscillator technologies such as HyperFET and the resulting benefits in area, speed, and power over corresponding Boolean CMOS-based implementations are presented in Sections 6 and 7. Section 8 concludes the article.

## 2 RELATED WORK

This section presents a summary of oscillator-based solutions for real-world applications reported in literature followed by an account of practical realization of nano-oscillators.

### 2.1 Oscillator-Based Solutions

Several works in the past have exploited oscillator-based systems to perform a variety of computations. Specifically, efforts in employing coupled oscillator systems to perform certain special computations have gained significant attention.

Many physical phenomena found in basic sciences such as chemistry (Chopra and Spong 2005), physics (Bozis and Hadjidemetriou 1999), and biology (Winfree 1967) can be modeled as synchronization of coupled oscillators. Several software models involving complex mathematical analysis have been proposed to understand and mimic such phenomena. Some of these models have also been successfully applied to image analysis applications (Horn and Opher 1999; Liu and Wang 1999; Johnson 1994). A software model of neural oscillators performing image segmentation based on correlation has been explored in Wang and Terman [1997] and Chen and Wang [2002]. The authors of Kyamakya et al. (2010) have demonstrated the use of coupled oscillators as well as cellular neural networks to enhance the contrast of an image. Other image processing related tasks employing coupled oscillators can be found in Roska et al. (2012). Scientists have also found oscillatory behavior in the human nervous system and have attempted to build a similar system using conventional CMOS components such as ring oscillators, Phase Locked Loop (PLL)s, and the like. However, overcoming issues such as scalability in size, power, and area have always been a major task for such devices and applications.

### 2.2 Nano-Oscillators

The aforementioned issues in scalability have opened up avenues for researchers to explore new materials such as Nano-Electro-Mechanical Switching (NEMS) devices, Spin-Torque Nano-Oscillators (STNOs), and Metal-Insulator Transition (MITs) materials toward practical realization of oscillators. Motivated by the use of such oscillators for solving real-world applications, the possibility of coupling multiple oscillator systems in a controlled and programmable manner is also explored (Kaka et al. 2006).

Resonant Body Transistors (RBTs) are a type of NEMS device that is designed to use electrically transduced motion to modulate the current through the transistor (Weinstein and Bhawe 2010). Coupling between RBTs is accomplished through feedback mechanisms leading to oscillating frequencies of 100MHz (Bartsch et al. 2012).

The operation of STNOs is based on the persistent interaction between electrical current in the presence of magnetic fields. This interaction leads to self-sustaining oscillatory changes in the resistance of the device (Csaba et al. 2012). For a given device, the frequency of oscillation is controlled by the DC current driving the device. Coupling in STNOs can be achieved in multiple ways - electrical coupling, magnetic coupling, interconnects, fluctuations in resistivity, and so on.

Oscillations achieved in several MIT-based devices (Yang et al. 2011) are very slow, making detection of synchronization extremely difficult.

Vanadium Dioxide ( $\text{VO}_2$ )-based architectures are used for implementation of the oscillator pairs. These oscillators use an RC time constant model, whose simulations are based on real devices fabricated using  $\text{VO}_2$  (Shukla et al. 2014). These devices are referred to as HyperFETs.

Prior works, especially in image processing, have focused on developing dedicated oscillator-based systems targeted for only specific tasks of the algorithms (Narayanan et al. 2014). In contrast to such previous efforts, this work proposes a general oscillator-based co-processor architecture



Table 1. List of Applications and Algorithms That Are Often Accelerated Through Hardware

Applications	Algorithms	Dominant Computations
Eye Detection	Vector Quantization	Distance and Degree of Match (DoM)
Image Segmentation	$k$ -means, K-medians	Distance
Digit Recognition	$k$ -nearest Neighbors	Distance
Pitch Estimation	Peak Detection	Max & Sort
Speech Recognition	Vector Quantization	Distance, Min and Max
Structural Health Monitoring	Peak Detection	Max and Sort

that realizes commonly found functions across a wide variety of applications. The main processor offloads these computations to the co-processor for efficient execution.

### 3 PROPOSED CO-PROCESSOR INSTRUCTION SET ARCHITECTURE

Recent work in Gubbi et al. (2013) has indicated the rising trend for Internet of Things (IoT) in the contemporary embedded systems market, specifically in hand-held and wearable gadgets. Devices employed at the data collection points for such applications often derive their inputs from the analog world (e.g., image, speech, text). Converting the analog signals to digital form and transmitting these massive amounts of data to the cloud server for further processing is often considered to be energy inefficient. This limitation has been overcome by including in-situ processing units in the form of specific co-processors and accelerators to perform preliminary computations at the data collection point and then transmit only the processed data to the server, thereby significantly reducing the amount of data transferred over the network. These co-processors/accelerators are aimed at being power and energy efficient. While several works in the past have met this constraint (i.e., being power/energy efficient), very few works have made a focused attempt in achieving this efficiency by working directly on the analog inputs, thereby avoiding the overheads of analog-to-digital conversion (Narayanan et al. 2014).

Column 1 of Table 1 lists some of the current common applications found in today's designs that employ dedicated hardware (accelerators). The primary source of inputs for the majority of these applications is analog in nature, which gets translated to digital domain using A/D converters. Column 2 of the same table shows the algorithms employed by the respective applications. Column 3 of Table 1 presents the dominant computations involved in executing the respective algorithms. It should be noted here that Table 1 is in no way exhaustive and contains only a few representatives of the category of applications/algorithms being targeted in this article.

We shall briefly discuss on the algorithms mentioned in Table 1. Vector Quantization (VQ) is a lossy data compression technique employed by many real-world applications such as iris/eye detection (Kekre et al. 2010), speech recognition (Kekre et al. 2011), density estimation, and so on. The computation, at the core of VQ, involves finding the degree of match between a given input cluster and already available clusters. The  $k$ -nearest neighbor algorithm, which employs the distance computation kernel, is actively used in applications such as handwritten digit recognition (Babu et al. 2014) and image segmentation (Liu et al. 2008). Some applications related to speech and structural health monitoring of civil structures often deal with identifying the peak input from a given set of analog samples (Bakht and Mufti 2015). An important property of each of the applications listed in Table 1 is that they are inherently error-resilient in nature, i.e., these applications can tolerate some amount of error at their outputs or inputs without significant loss in quality of service. In fact, for such applications, computing a perfect solution is almost impossible and a

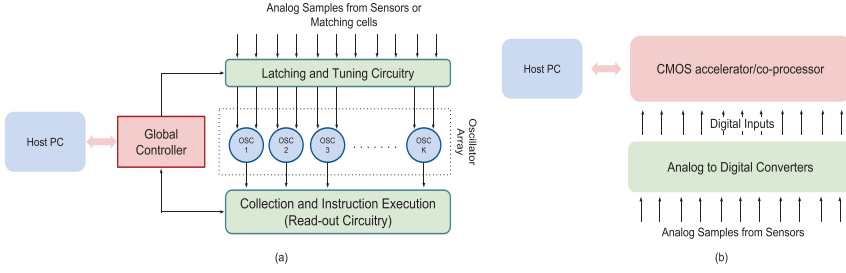


Fig. 2. (a) Proposed architecture of the oscillator-based co-processor. (b) A conventional CMOS accelerator.

near-perfect solution is acceptable. This notion of in-built error resiliency provides us leverage to design systems that can compute approximately leading to tolerable errors at the output.

Using the previously stated analysis and information, the following basic Instruction Set Architecture (ISA) of the co-processor is identified that can support the execution of the aforementioned algorithms/applications -

- **$N^{th}$ -Distinct Min**: To find the  $N^{th}$  distinct minimum among “K” input analog samples.
- **$N^{th}$ -Distinct Max**: To find the  $N^{th}$  distinct maximum among “K” input analog samples.
- **Degree of Match ( $V_1, V_2$ )**: For a given pair of vectors containing up to “K” analog samples ( $V_1, V_2$ ), to find the number of pairs of samples (elements) that are close to each other.
- **Sort( $N, Ord$ )**: Depending on the *Ord* parameter, which can either be *Increasing* or *Decreasing*, this instruction outputs the first  $N$  smallest or largest samples, respectively, from a given input set of “K” analog samples.

The following sections describe how each of these instructions can be computed through coupled oscillators and how they can be employed in the field to execute an application. Since these oscillators directly work on analog samples, the analog-to-digital converters are no longer required, thus making the entire system more compact and energy efficient.

#### 4 PROPOSED CO-PROCESSOR ARCHITECTURE

Figure 2(a) shows the block diagram of the proposed oscillator-based co-processor. The architecture primarily comprises an array of “K” coupled oscillators. These oscillators can either be phase coupled or frequency coupled. Figure 2(b) shows the conventional design where Analog to Digital Converter (ADC)s are used to convert the input Analog samples into the Digital domain, which is processed by a conventional CMOS accelerator/co-processor.

The latching circuitry will latch the analog signals either from the external analog sensors or from analog matching cells (Bui and Shibata 2008) to the respective oscillator inputs. At most, “K” pairs of analog signals (one pair per oscillator) can be latched. Based on the instruction to be executed, the global controller decides the assignments of the analog signal pairs to the oscillators.

The read-out circuitry, on the other hand, is responsible for collecting the outputs of the “K” oscillators and to execute the instruction using them. The global controller indicates to the read-out circuitry the instruction to be executed. A detailed diagram of the read-out circuitry is in Figure 3(a). It comprises — a  $\log(K)$ -counter, a K-input Winner Take All (WTA) circuit, a  $\log(K)$  value-register, a timer, a  $\log(K) \times K$ -bit memory array, and a  $\log(K)$  comparator. As seen in Figure 3(a), there are six inputs to the read-out circuitry, namely, Reset-counter, Write Value, Write Time-limit, AddrOut, Mux Input, and the K-oscillator outputs. These lines are driven by the global controller depending on the instruction being executed. It should also be noted that

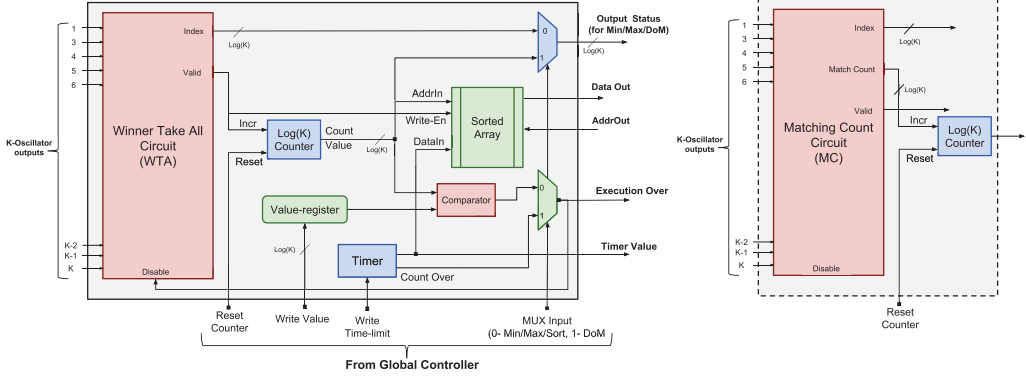


Fig. 3. (a) Overall block diagram of the read-out circuitry of the co-processor mentioned in Figure 2. (b) Modifications required to implement  $N^{th}$  Min/Max in the presence of duplicates.

the  $j^{th}$  oscillator output from the array drives the  $j^{th}$  input of the WTA circuit. There are four outputs from the read-out circuitry, namely, Output Status, Data Out, Execution Over, and Timer Value, which are read by the global controller. Table 2 provides the description of the functionality of each component of the read-out circuit.

#### 4.1 Mapping Instructions to Architecture

In this section, we map each instruction on to the architecture described in the previous sections. Without loss of generality, we assume that the number of oscillators present in the array is  $K$ . This value can be defined at the time of design based on the application requirements. We also consider the fact that physical nano-oscillators work over only a certain input voltage range and thus assume that the input analog samples are bounded in value as well, i.e., each *input sample*  $\in (a_{MIN}, a_{MAX})$ . This assumption also holds for many practical applications — e.g., pixel values are usually evaluated in the range (0,256).

**4.1.1  $N^{th}$  - Distinct Max.** Algorithm 1 explains the steps involved in performing the  $N^{th}$ -Distinct Max instruction on the co-processor using the read-out circuitry shown in Figure 3. In situations where multiple analog samples correspond to the same MAX value, the smaller index is returned by the read-out circuitry. Thus, Algorithm 1 returns the index of the  $N^{th}$ -Distinct Max.

Algorithm 1 can be used for other computations as well by setting different values to  $N$ . To find the maximum of all  $K$  samples, the value of  $N$  is set to 1. The median of a set of  $K$  samples can be found by setting  $N$  to  $K/2$ . In scenarios where the input set of samples is smaller than the available number of oscillators, the two inputs of the left-over oscillators are fed with extreme values -  $a_{MIN}$  and  $a_{MAX}$ . This ensures that the left-over oscillators will be the last ones to synchronize. In the opposite situation (i.e., *input samples*  $> K$ ), the input set is divided into chunks of  $K-N$ . The  $N$  distinct maximal values of the first chunk are used again in the second chunk to find the new maximum. The  $N^{th}$  distinct maximum of the last chunk will provide the final result.

In scenarios where duplicates are present at the inputs, it may be more relevant to an application to find just the  $N^{th}$ -Max instead of the  $N^{th}$ -Distinct Max. In such a case, it is necessary to record all the oscillators, which shall synchronize simultaneously, and increment the counter by the value. This can be achieved by using a Match-Counting (MC) circuit instead of a WTA circuit as shown in Figure 3(b). This circuit counts the total number of oscillators that have synchronized at a particular instant and outputs this number on the signal *Match Count*, which is used to increment the counter.



Table 2. Description of Each Component in the Read-out Circuitry

Component	Inputs	Outputs	Description
Winner Take All (WTA)	<ul style="list-style-type: none"> <li>– K inputs such that <math>j^{th}</math> oscillator output is connected to the <math>j^{th}</math> input where <math>1 \leq j \leq K</math></li> <li>– <i>Disable</i> (1-bit)</li> </ul>	<ul style="list-style-type: none"> <li>– <i>Index</i> (Log(K)-bits)</li> <li>– <i>Valid</i> (1-bit)</li> </ul>	<ul style="list-style-type: none"> <li>• <i>Index</i>: The WTA circuit outputs the index of the first oscillator to synchronize. Since the maximum value to be output is K, the Index signal is Log(K) bits wide.</li> <li>• <i>Valid</i>: Each time a synchronization occurs, a pulse is sent out on the Valid output.</li> <li>• <i>Disable</i>: Deactivates the WTA circuit when a pulse is driven on it.</li> </ul>
Log(K)-Counter (Binary counter of Log(K) bits)	<ul style="list-style-type: none"> <li>– <i>Incr</i> (1-bit)</li> <li>– <i>Reset</i> (1-bit)</li> </ul>	– <i>CountValue</i> (Log(K)-bits)	<ul style="list-style-type: none"> <li>• The <i>Valid</i> signal from the WTA drives the <i>Incr</i> input that, in turn, increments the counter value by 1.</li> <li>• If the <i>Reset</i> input is asserted by the global controller, then the counter value is set back to 0. The <i>Reset</i> input is usually asserted at the beginning of each instruction execution.</li> </ul>
Value-Register	– Log(K) bits from global controller	Log(K)-bits Value	This register holds the value provided by the global controller. It is typically used by some instructions to compare the number of oscillators that have synchronized so far.
Comparator	<ul style="list-style-type: none"> <li>– <i>CountValue</i></li> <li>– <i>Value-Register</i></li> </ul>	1-bit Output	This is a Log(K)-bits comparator that compares the value stored in the Log(K)-counter and the <i>Value-Register</i> . It sends a pulse (low-high) on the 1-bit output when these values are equal.
Timer	– <i>Timer-Limit</i> (16-bits)	<ul style="list-style-type: none"> <li>– <i>CountOver</i> (1-bit)</li> <li>– <i>Timer Value</i> (16-bits)</li> </ul>	<ul style="list-style-type: none"> <li>• <i>Timer-Limit</i>: This signal is controlled by the global controller, which sets the maximum value to which the counter should count. This signal also triggers the counter to start counting from 0.</li> <li>• <i>CountOver</i>: When the timer reaches <i>Timer-Limit</i>, this one-bit signal is asserted.</li> <li>• <i>Timer Value</i>: This gives the current count value of the Timer.</li> </ul>
Sorted Array	<ul style="list-style-type: none"> <li>– <i>Write-En</i> (1-bit)</li> <li>– <i>AddrIn</i> (Log(K)-bits)</li> <li>– <i>DataIn</i> (Log(K)-bits)</li> <li>– <i>AddrOut</i> (Log(K)-bits)</li> </ul>	– <i>DataOut</i> (Log(K)-bits)	<ul style="list-style-type: none"> <li>• This array stores up to “K” 16-bit values and is used as a queue.</li> <li>• The <i>AddrIn</i> input is driven by the Log(K)-Counter, the <i>DataIn</i> by the <i>Timer Value</i> signal, and the <i>Write-En</i> by the <i>Valid</i> output bit of the WTA.</li> <li>• Whenever the <i>Valid</i> signal is triggered, the <i>Timer Value</i> is stored in the location pointed Log(K)-Counter and the latter is incremented, thus implementing the queue functionality.</li> <li>• The <i>AddrOut</i> is used by the host-PC to access the sorted array for the final results.</li> </ul>

**ALGORITHM 1:**  $N^{th}$ -Distinct Max

---

**Input:** K Input analog samples  $\langle A_1, A_2, A_3, \dots, A_k \rangle$  and the value  $N$   
**Output:**  $N^{th}$  distinct maximum of the input samples  
 /\* Actions performed by the global controller \*/

---

- 1 Reset Log(K)-Counter by asserting the Reset Counter input;
- 2 Write  $N$  in the *Value-Register* using *Write Value* input;
- 3 Set MUX Input to 0;
- 4 Latch  $\langle A_i, a_{MAX} \rangle$  to the  $i^{th}$  oscillator where  $1 \leq i \leq K$ ;
- 5 Set the *Timer-Limit* to  $2^{16}$  which will trigger the timer to count from 0;

---

- 6 Once the mentioned previously are performed by the global controller, the read-out circuit works as described later in the text and as inferred from Table 2;
- 7 Whenever an oscillator synchronizes, the WTA increments the Log(K) counter;
- 8 Since the oscillators are fed with the input sample on one of their inputs and  $a_{MAX}$  as their other input, the first one to synchronize will be the sample closest to  $a_{MAX}$  and, hence, the maximum among all the samples;
- 9 The Log(K) counter reaches  $N$  when the oscillator with the  $N^{th}$  distinct maximum sample synchronizes;
- 10 Since the *Value-Register* is set to  $N$ , the comparator comparing the *Value-Register* and the Log(K) counter sends a pulse on its output;
- 11 The pulse from the comparator triggers the output signal *Execution Over* and disables the WTA circuit;
- 12 The *Output Status* line, driven by the WTA's index output, holds the index of the oscillator synchronized at this instance;
- 13 The *Timer Value* output signal holds the time taken for the oscillator holding the  $N^{th}$  maximum sample to synchronize. As the WTA is disabled, this value will be held till the next instruction is to be executed;
- 14 The *Timer Value* can be read by the host-pc, which uses a mapping function that maps the *Timer Value* to the original value of the  $N^{th}$ -Distinct Maximum;

---

**4.1.2  $N^{th}$  - Distinct Min:** This instruction operates similarly to the  $N^{th}$  Max instruction with the exception of latching the smallest range value (i.e.,  $a_{MIN}$ ) at the second input of all the oscillators. Setting  $N \leftarrow 1$ , outputs the minimum of all K numbers and setting  $N \leftarrow K$ , outputs the largest number. The read-out circuitry for this instruction functions the same way as that for the  $N^{th}$ -Distinct Max instruction. Figure 3(b) can be used to find the  $N^{th}$ -Min in the presence of duplicates at the inputs.

**4.1.3 Degree of Match - DoM( $V_1, V_2$ ).** The DoM instruction finds the degree of closeness of two given equal length input vectors ( $V_1, V_2$ ) of samples. For example, let  $V_1 = \langle A_1, A_2, A_3, \dots, A_k \rangle$  and  $V_2 = \langle B_1, B_2, B_3, \dots, B_k \rangle$ . Thus,

$$DoM(V_1, V_2) = \sum_{i=1}^k f_i$$

$$\text{such that, } f_i = \begin{cases} 1 & \text{if } |a_i - b_i| \leq T_h \\ 0 & \text{otherwise} \end{cases}$$

Here,  $T_h$  is the threshold value set by the user/application. In other words, DoM outputs the number of pairs among ( $V_1, V_2$ ) whose absolute difference is less than the threshold  $T_h$ . Algorithm 2 shows the steps involved in performing DoM.

**ALGORITHM 2:** Degree of Match - DoM( $V_1, V_2$ )

---

**Input:** Two Vectors ( $V_1, V_2$ ) each with  $K$  Input samples and Time-limit

Let  $V_1 = \langle A_1, A_2, A_3 \dots A_k \rangle$

Let  $V_2 = \langle B_1, B_2, B_3 \dots B_k \rangle$

**Output:** Number of matching samples

/\* Actions performed by the global controller

\*/

- 1 Reset Log(K)-Counter by asserting the Reset Counter input;
  - 2 Set MUX Input to 1;
  - 3 Latch  $\langle A_i, B_i \rangle$  to the  $i^{th}$  oscillator where  $1 \leq i \leq K$ ;
  - 4 Set the *Timer-Limit* to an optimal value which will trigger the timer to count from 0;
- 
- 5 Once the previously mentioned are performed by the global controller the read-out circuit works as inferred in Table 2;
  - 6 Whenever an oscillator synchronizes the WTA increments the Log(K) counter;
  - 7 As the Timer reaches the Timer-Limit, a pulse is sent over the Count Over signal which in turn triggers the Execution Over output signal and also disables the WTA circuit;
  - 8 The *Output Status* output line holds the Log(K) counter value, which indicates the number of oscillators that have synchronized within this set Timer-Limit and represents the Degree of Match between  $V_1$  and  $V_2$ ;
- 

In order to map this function on to the oscillator-based co-processor, we assume that the number of samples present in each vector are equal to the number of available oscillators (i.e.,  $K$ ). To mimic the functionality of  $T_h$  in the oscillator domain, we use an upper bound on the time (i.e., *Timer-Limit*) for synchronization of the oscillators. This ensures that only those oscillators that synchronize within the *Timer-Limit* will contribute to the output count, while oscillators with widely separated input pairs will take a longer time to synchronize and thus not contribute to the closeness. This concept only holds true for slow oscillator technologies where time to synchronization can be captured easily by the timer module. In case of HyperFET-like oscillator models, an Exclusive or (XOR)-gate-based circuitry such as the one proposed in Datta et al. (2014) can be employed to capture the number of matching elements between the two input vectors.

The *Timer-Limit* is programmable and needs to be provided by the global controller while executing the instruction. Selecting this *Timer-Limit* value is crucial for proper functioning of this instruction and is dependent on the range of inputs that is being supported by the oscillators. The *Timer-Limit* will definitely be lower than the time taken for an oscillator with the input pair  $\langle a_{MIN}, a_{MAX} \rangle$  to synchronize since this pair is the maximum difference possible for any pair within the given range. Our observations based on the functional simulation suggest that the *Timer-Limit* value depends on the application. This forms the motivation to keep *Timer-Limit* as a programmable parameter in the co-processor. For example, consider a face recognition application where the inputs to the oscillators are pixel values. An ideal match happens when both the inputs of the oscillators have the same pixel values. This would require minimal time for synchronization for all the oscillators. However, usually the test images are different than the template images and, thus, some amount of difference in pixel values is acceptable by the applications to find the best match. The typical value of this difference in pixels can help identify the value of *Timer-Limit* to be used for this application. The suggested method to obtain an optimal value of *Timer-Limit* is to carry out simulations on existing benchmarks and fine-tune this parameter until acceptable results are obtained.

**ALGORITHM 3:** Sorting - Sort( $N, \text{Ord}$ )

---

**Input:**  $K$  Input analog samples  $\langle A_1, A_2, A_3, \dots, A_K \rangle$  and Value  $N$   
**Output:** The smallest/largest  $N$  samples in increasing/decreasing order.  
 /\* Actions performed by the global controller \*/

---

- 1 Reset Log( $K$ )-Counter by asserting the Reset Counter input;
- 2 Write  $N$  in the Value-Register using Write Value input;
- 3 Set MUX Input to 0 Latch  $\langle A_i, a_{MAX} \rangle$  for decreasing order or  $\langle A_i, a_{MIN} \rangle$  for increasing order to the  $i_{th}$  oscillator,  $1 \leq i \leq K$ ;
- 4 Set the *Timer-Limit* to  $2^{16}$ , which will trigger the timer to count from 0;

---

- 5 Once the aforementioned actions are performed by the global controller the read-out circuit works as inferred in Table 2;
- 6 Whenever an oscillator synchronizes, the WTA increments the Log( $K$ ) counter and triggers the *Write-En* signal to allow writing data into the Sorted-Array;
- 7 The Log( $K$ ) counter output acts as an address and drives the *AddrIn* signal of the Sorted-Array;
- 8 Each time the Log( $K$ ) counter increments, the *Timer Value* of the Timer is stored in the Sorted-Array at the address pointed by the Log( $K$ ) counter. The Log( $K$ ) counter reaches  $N$  when the oscillator with the  $N^{th}$  maximum sample synchronizes;
- 9 Since the Value-Register is set to  $N$ , the comparator comparing the Value-Register and the Log( $K$ ) counter sends a pulse on its output;
- 10 The pulse from the comparator triggers the output signal *Execution Over* and disables the WTA circuit;
- 11 The host-pc can now read the  $N$  *Timer Values* of the top  $N$  analog samples in increasing or decreasing order;

---

In scenarios, where the number of samples ( $N$ ) in the vector is more than the available oscillators, the samples can be broken down into chunks of  $K$  and fed to the oscillator array in iterations. The timer will be initialized to a time limit for every chunk, while the counter value gets accumulated over all  $N/K$  iterations by the host-pc.

**4.1.4 Sorting - Sort( $N, \text{Ord}$ ).** The working of this instruction is shown in Algorithm 3. This instruction stores the time taken by the smallest or the largest  $N$  analog samples to synchronize from the given input  $K$  analog samples in the Sorted-Array memory in increasing or decreasing order, respectively. These *Timer Values* are read by the host-pc, which can then estimate the values of the analog samples using the mapping function. Alternatively, the index of the oscillators in sorted order can also be stored in the array and the host-pc can use this information to directly access the analog samples.

In presence of  $p$  duplicates at the inputs, we can employ the partial circuitry of Figure 3(b) to ensure that there are  $p - 1$  gaps between the addresses of distinct inputs. Extra circuitry can further be added to ensure multiple writes into the Sorted array when the Match-Count signal is greater than one.

## 4.2 Experimental Setup

The co-processor architecture mentioned previously was functionally verified. For the purpose of this functional verification, a MATLAB<sup>TM</sup> model of the Kuramoto oscillator (Kuramoto 2012; Chopra and Spong 2005, 2009) was employed. In the Kuramoto model, the oscillators achieve a phase-lock over time. The amount of time required for the inputs to synchronize/lock indicates the closeness of the inputs. Thus, oscillators with almost the same inputs will synchronize faster

Table 3. Parameter Values for the Kuramoto Oscillator

Parameter	Value
No. of synchronizing inputs	2
Strength of synchronization	349
Time step size	0.000042
Input scaling factor	0.009075
Input integer range ( $a_{MIN}, a_{MAX}$ )	(1,32)

than oscillators with widely separated inputs. The values of different parameters of the Kuramoto oscillator used during our simulations is shown in Table 3. In order to mimic generic physical oscillator models, we have configured the Kuramoto model to use only two inputs. Using the settings stated in Table 3, the oscillator model was able to generate a unique time of synchronization for each pair of input values ( $a, b$ ) that has a unique difference. Maintaining a unique time of synchronization for every pair ( $a, b$ ) with a unique difference is crucial for the accurate execution of any instruction. The input values ( $a, b$ ) were picked from the range ( $a_{MIN}, a_{MAX}$ ) with a resolution of  $1/32$ . In other words, the values of  $a$  and  $b$  satisfy the following equations:

$$a = a_{MIN} + k_1 * h$$

$$b = a_{MIN} + k_2 * h,$$

where, the resolution  $h = \frac{(a_{MAX} - a_{MIN})}{32}$  and  $k_1, k_2 \geq 0$  are integers.

The choice of this resolution is to ensure that the Kuramoto model reflects the physical oscillator models as closely as possible in terms of functionality. Most voltage controlled nano-oscillators operate for a defined input voltage range and also have a defined granularity of unique voltages, which can be supplied to the oscillators. Section 6 discusses on these issues in more detail in the context of HyperFET technology.

The reader should note here that since Kuramoto is a mathematical model, the inputs to the oscillator model are actually integers, which can be treated as representatives of the analog signal parameters such as amplitudes, phase, or frequency. It is known that analog signals are continuous, and a true representation should have infinitely fine granularity. However, we use the integer representations instead of the full-precision (e.g., floating point) due to the difficulty in quantifying the precision of the oscillator model. In physical designs, analog structures often suffer from the variations, thereby affecting the signals. This approximated computing gives us the liberty of making fuzzier decisions by treating the inputs and outputs as low-precision signals. Therefore, rather than building a model using high-precision signals, we use the finite-precision for representing the signals, which can actually be finer in the analog implementations.

Each of the instruction/functions of the proposed ISA were verified on this Kuramoto model using a random set of input pairs generated using MATLAB<sup>TM</sup>. The outputs of the Kuramoto model were compared against the outputs of a C++ program performing the same function on the same set of input integers. For applications where the input range is other than (1,32) or where a resolution other than  $h$  is desired, the scaling factor, coupling strength, and the step-size of Table 3 need to be fine-tuned so as to maintain the property of unique time of synchronization for each unique difference within the given input range.

**4.2.1 Interpreting the Output.** For the  $N^{th}$ -Distinct MAX/MIN instructions, the proposed architecture outputs the index of the maximum/minimum number and the time taken for its

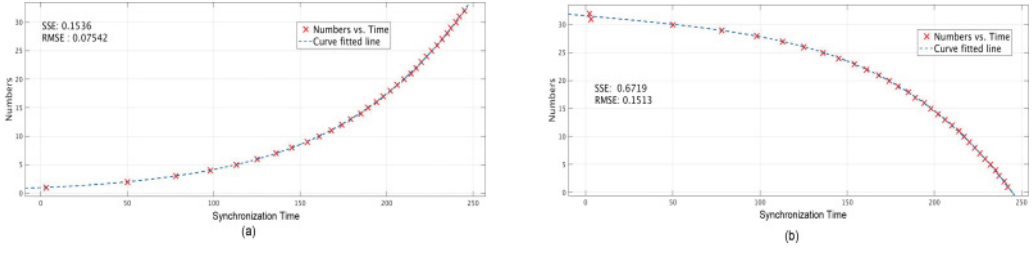


Fig. 4. Curve-fit for (a)  $N^{th}$ -Distinct MIN and (b)  $N^{th}$ -Distinct MAX.

synchronization as well. Using the index to find the maximum/minimum showed 100% compliance with the C++ outputs for unique vectors. In cases where duplicates existed in the inputs, the lower index containing the maximum/minimum element was output by the read-out circuit. While the index of the maximum/minimum element is useful, often the value of the maximum/minimum is required for further processing by the host-pc. It would thus be required to store the input analog samples using analog matching cells and then perform an analog-to-digital conversion only for this sample to produce the necessary digital value for the host-pc. Another technique to produce a digital estimate of the original analog sample is to utilize the synchronization time. Since the times for synchronization are unique for each analog sample when compared with a reference sample (value of zero when performing  $N^{th}$ -Distinct MIN), one can easily create a mapping of this time to numbers within the input range. Figure 4(a) shows how the time of synchronization varies for up to 32 integer numbers within the range [1,32] when compared with 0. We now use the curve-fitting tool of MATLAB<sup>TM</sup> to generate a polynomial of the best-fit curve. The host-pc can now use this polynomial on the *Timer Value* output by the read-out circuit to estimate the value of the original input while performing  $N^{th}$ -Distinct MIN function. Figure 4(a) shows that the best-fit curve results in a Root Mean Square Error (RMSE) error of 0.07542 and a Sum of Squares due to Error (SSE) of 0.1536 for the  $N^{th}$ -Distinct Min. Figure 4(b) shows the same data for the MAX operation. The function Sort(N, Ord) stores the individual synchronization times in the array, which the host-pc can convert to digital values of the analog samples using the polynomial for  $N^{th}$ -Distinct MAX or  $N^{th}$ -Distinct MIN in case of increasing or decreasing order, respectively.

We reiterate the fact that the applications mentioned in Table 1, which are supported by the proposed co-processor, are approximate applications (Chippa et al. 2013). In other words, these applications bear a certain amount of inherent error resilience such that if the computations involved in these applications are executed approximately, they have minimal or insignificant impact on the final output quality. For such applications, a *good* result (with tolerable errors) is sufficient as compared to a *best* (error-free) result. Works in Gala et al. (2013), Gala et al. (2014), and Gala et al. (2015) have shown that hardware catering to such applications can be optimized and designed for significantly lower power consumption without crossing the tolerable error limits set by the applications. Along the same lines of argument, we state that the error produced by the mapping function proposed previously can be used to estimate the original values of the samples without significant loss of output quality for the target applications being supported by the proposed co-processor. Though we have demonstrated the curve-fitting technique only for the Kuramoto model in this article, the same methodology can be applied to other models as well. If the input range is very small, the host-pc can maintain a look-up table to map the synchronization times to the digital values as well in lieu of the curve-fitted function.



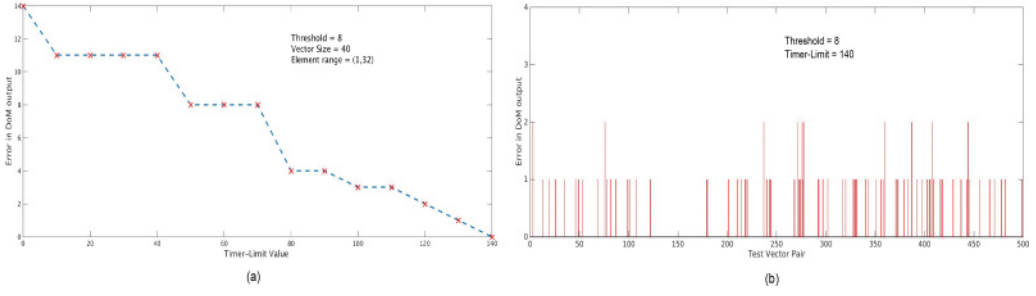


Fig. 5. (a) Graph shows how the DoM grows more accurate with increasing *Timer-Limit* value. (b) This graphs shows that *Timer – Limit* = 140 obtained from (a) cannot provide an accurate answer for all inputs.

As described in previous sections, for the proper operation of the DoM instruction, it is crucial to find the optimal value of the *Timer-Limit* parameter. This parameter is fine-tuned using the threshold value set by the application. Figure 5 provides a correlation between the *Timer-Limit* parameter and a given threshold value  $T_h$ . Figure 5(a) was obtained by varying the *Timer-Limit* value of the co-processor and comparing the output value of DoM with the output of a conventional C++ program performing DoM using  $T_h = 8$  as the threshold for a single pair of vectors of size 40, such that each element of the vector is within the range (1, 32). It can be seen from Figure 5(a) that for a single pair of vectors and a given value of threshold  $T_h$ , one can easily find the optimal value of *Timer-Limit*, which provides an accurate answer. However, when the same values of  $T_h$  and *Timer-Limit* are used across random pairs of vectors of the same size, minor differences in output are observed for a few pairs as seen in Figure 5(b). Thus, by manipulating the value of *Timer-Limit*, the programmer can obtain different levels of accuracy at the output while gaining benefits in runtime (and thereby energy).

This error is due to the fact that the *Timer-Limit* parameter can only provide an estimate of the  $T_h$  in the oscillator domain, and for more accurate results, the oscillator parameters in Table 3 will require further changes and fine-tuning. While further fine-tuning is possible in Kuramoto (a mathematical model), real-world physical oscillators are restricted by technology and physical parameters, and obtaining direct mapping between *Timer-Limit* and  $T_h$  would be extremely difficult, if not impossible.

As will be shown later in Section 5.1, in the context of approximate applications, DoM-using oscillators turn out to be a cost-efficient approximate substitute for Euclidean functions to find the closeness between vectors. Since DoM merely matches the elements of the vectors, more granular differences in the distance, which are captured in the conventional Euclidean calculation may not get captured by DoM, thus leading to minor differences in output. For example, consider a three-element reference vector  $V_{ref} = (3, 4, 5)$ . The task at hand is to find which of the vectors -  $V_a = (7, 8, 9)$  and  $V_b = (6, 7, 8)$  is closer to  $V_{ref}$ . A Euclidean estimation will conclude  $V_b$  to be the winner. However, due to the approximate nature of DoM discussed previously, the proposed oscillator may declare both,  $V_a$  and  $V_b$  to be equidistant from  $V_{ref}$ , which amounts to slight errors in application outputs. A similar situation may arise when multiple elements of both the input vectors is the same, wherein the counter in the proposed architecture will only be incremented by one rather than the number of repetitions present. However, as will be shown later in Section 5, such corner-case scenarios where both the vectors are exactly the same are very rare in real-world applications and the error presented due to DoM for other cases is small and tolerable by the applications. Applications that require more precise calculations will require extra hardware (such as the XOR network presented in Shukla et al. (2014)) in the read-out circuitry.

**ALGORITHM 4:** Algorithm for Vector Quantization

---

```

1 Choose the number of clusters - M;
2 Choose M random vectors (from the input set of vectors) to be the initial set of centroids for each
  cluster;
3 while All vectors not assigned do
4   Choose a new input vector  $x$ ;
5   Find the cluster closest to  $x$  using a suitable distance metric;
6   Update the centroid of the chosen cluster;
7 end

```

---

**5 APPLICATIONS/ALGORITHMS**

This section demonstrates how the proposed co-processor can be employed to solve a set of common real-world applications. Three case studies are presented in this section.

**5.1 VQ**

VQ has traditionally been defined as a lossy data compression technique. It works by dividing a large set of vectors into groups having approximately the same number of vectors closest to each other. Each group is represented by its centroid point. Over the years, VQ has been widely applied in pattern recognition (speech and image), density estimation, and clustering-based applications (Furui 1991; Nasrabadi and King 1988). In these applications, the elements of the vectors are often referred to as *attributes* of the vectors, i.e., a vector with  $p$  elements is said to have  $p$  attributes. In such cases, the centroid of a cluster is a vector  $C$  with  $p$  elements, where each individual element of  $C$  is said to be the centroid of a particular attribute across all vectors associated with that cluster. Centroids are usually calculated using various metrics such as mean, median, mean-squared, and so on. In this article, we use K-medians as a distance metric to calculate the centroids of the clusters. This particular metric allows for better outlier detection, reduces the absolute deviation present within a cluster, and is easily computed as compared to  $k$ -means (Poonam and Dutta 2012). The conventional method followed by VQ is shown in Algorithm 4.

The *While Loop* in Algorithm 4 is often referred to as the clustering/training step and most implementations of VQ use  $k$ -means,  $k$ -nearest neighbors, and the like, to perform this step. In applications such as speaker identification, which employ VQ, the training phase is often followed by a prediction phase where an unknown input vector is provided and the system identifies the *closest* cluster it may belong to. The *closeness* is measured with respect to a threshold. If, for the given input vectors, generated by the speech signal input of the speaker, none of the clusters are close to it (within the threshold), then no valid cluster is found for the same. We now present steps on how the training and the prediction phase of VQ can be carried out on the proposed co-processor.

In this article, we assume that the number of available oscillators ( $K$ ) in the co-processor is the maximum of the number of attributes in any vector and the number of vectors associated with a cluster. Algorithm 5 shows the steps involved in performing the training phase of VQ using the co-processor.

It should be noted here that the proposed co-processor is responsible only for computations and will not be involved with handling memory/data. This particular task of handling memory structures is off-loaded to the host-pc/processor. For example, the task of choosing  $N$  initial centroid is to be performed by the host-pc. When a new vector is chosen for clustering, the host-pc initializes the variables *MAX* to 0 and *ChosenCluster* to  $N + 1$ . The host-pc then commands the co-processor

**ALGORITHM 5:** Training Phase of VQ to Classify  $V$  Vectors into  $N$  Clusters

---

**Input:** Number of Clusters -  $N$ , Input set of vectors-  $V$   
**Output:**  $N$  clusters with updated centroids

```

1 Choose  $N$  random vectors from  $V$  to be initial centroids;
2 while Not all vectors classified do
3    $MAX \leftarrow 0$ ;
4    $ChosenCluster \leftarrow N + 1$ ;
5   Choose an unclassified vector  $V_i$ ;
6   for All clusters  $C_j$  do
7      $M \leftarrow DoM(V_i, C_j)$ ;
8     if  $M > MAX$  then
9        $MAX \leftarrow M$ ;
10       $ChosenCluster \leftarrow j$ ;
11    end
12  end
13  for All attributes  $A_i$  do
14    for All vectors  $V_k$  within  $ChosenCluster$  do
15       $NewCentroid \leftarrow N\text{-Distinct Max}(N/2)$ ;
16    end
17  end
18 end

```

---

to perform *DoM* of the chosen vector with every cluster (lines 6–12). Here, the attributes of the input vector are matched with the corresponding attributes of the cluster. The number of oscillators to be synchronized within a time period provides the degree of match between the vector and the cluster. During this execution, the host-pc is responsible for maintaining the maximum matching cluster so far. If at the end of line 12 the value of *ChosenCluster* is  $N + 1$ , then the particular vector is said to be an outlier and the host-pc can decide to ignore the vector or randomly assign a cluster to it. Once the closest cluster to the vector is chosen, the host-pc updates its centroids by executing the  $N - \text{Max}(N/2)$  instruction on the co-processor for each attribute of the vectors associated with that cluster.

In the prediction phase, the computation involves performing only the first two steps of Algorithm 6. The max *DoM* obtained is compared against a certain threshold. If the *DoM* is lower than the threshold value, it gets classified as an outlier, else it belongs to the cluster with the max *DoM*. The vector is also classified as an outlier if no cluster is found to be close to the vector and the *ChosenCluster* variable holds the value  $N + 1$ .

**5.1.1 Experimental Setup.** The previously mentioned VQ algorithm was executed on the co-processor using the Kuramoto model described in the previous section. We used a set of 50 random vectors to be classified into three clusters. Each vector had eight attributes.

To quantify the quality of clustering, we use sum total of the absolute deviation present within a cluster across all three clusters as a metric. We refer to this metric simply as *clustering-deviation*. We compared the *clustering-deviation* from the co-processor against that obtained by a traditional C++ algorithm for the same set of input vectors. Typically, a clustering solution where the final *clustering-deviation* is minimal is considered to be a better solution than others. Tests were run across 1,000 random sets of 50 input vectors each. Each attribute of the input vectors was a number within  $[1, 32]$ . The results of this experiment are discussed in the next section.

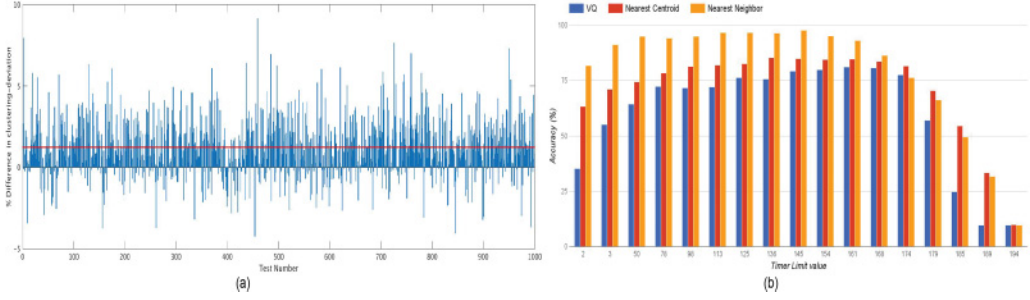


Fig. 6. (a) The percentage variation of clustering deviation produced by the proposed co-processor compared against the clustering deviation produced by a C++ program. (b) The variation of recognition accuracy with changing values of *Timer-Limit* for Nearest Neighbor, Nearest Centroid Classifier, and VQ-based algorithms.

---

**ALGORITHM 6:** Prediction Phase of VQ
 

---

**Input:** Centroids of  $N$  Clusters, Input vectors  $V$   
**Output:** Closest Cluster or Outlier Detection

```

1 while Not all vectors classified do
2    $MAX \leftarrow 0$ ;
3    $ChosenCluster \leftarrow N + 1$ ;
4   for All clusters  $C_j$  do
5      $M \leftarrow DoM(V, C_j)$ ;
6     if  $M > MAX$  then
7        $MAX \leftarrow M$ ;
8        $ChosenCluster \leftarrow j$ ;
9     end
10  end
11 end
12 return  $ChosenCluster$ 
  
```

---

**5.1.2 Results and Insights.** The percentage offset of the *clustering-deviation* of the co-processor with respect to the *clustering-deviation* of the C++ program is plotted in Figure 6(a). The plot contains data for each of the 1,000 random runs. The horizontal line indicates the average percentage offset of 1.22% over the 1,000 random runs for the co-processor. All the vertical lines above the zero axis indicate that the clustering by the co-processor was poorer than the traditional C++ algorithm, however, the offset is minimal or almost negligible. The reason for difference in the *clustering-deviation* is owed to the slight error present in the execution of the DoM function elaborated in Section 4.2.1. While VQ itself is a lossy compression technique, applications employing VQ are approximate as well and can tolerate this small offset in the output.

It can also be seen that in nearly 30% of the input cases, the vertical lines fall below the zero axis indicating that the co-processor has performed better clustering than the C++ implementation.

## 5.2 Handwritten Digit Recognition using VQ

Another real-world application that has been mapped to the proposed architecture is the Handwritten Digit Recognition. The optical digit recognition dataset, made available at University of California Irvine (UCI)'s Machine Learning (ML) repository (Lichman 2013), is used to evaluate the accuracy. Pre-processing programs from National Institute of Standards and Technology (NIST)

are used to reduce the  $32 \times 32$  bitmaps to an  $8 \times 8$  input matrix, where each element is within the range  $[1, 16]$ . For our experiments, we train the circuit using the training set of 3,823 samples and use the 1,791 test samples from the validation set (different from the training set) to evaluate the recognition accuracy.

We implemented three algorithms to perform digit recognition. The first attempt was using the *Nearest Neighbor* approach, where a particular test image was compared against all the training images and the label of the training image was assigned to the test image. Algorithm 6 can be used to perform this operation where the clusters are each of the training images and the input vectors were the test images. As mentioned earlier, the value of *Timer-Limit* defines the accuracy at the output of the proposed architecture. Figure 6(b) shows the variation of digit recognition accuracy with changing values of *Timer-Limit*.

The second approach was based on the *Nearest Centroid Classifier*. In this approach, 10 clusters were defined—one for each digit. The centroids of these clusters were pre-calculated such that each cluster included only those training images which had the same label. Now the prediction of each test image was performed using Algorithm 6 where 10 clusters were used using the approach and the same set of test images were applied as input vectors. Though this approach turns out to be more runtime efficient than the *Nearest Neighbor* approach, this benefit comes at a cost of reduced accuracy in recognition as shown in Figure 6(b).

The VQ algorithm from the previous section was also employed to perform digit recognition. In order to perform the training phase of VQ for this application, we set  $N$  to 10 (i.e., 10 clusters—one for each digit) such that the initial centroids were those obtained from the *Nearest Centroid Classifier* approach. Once the final set of centroids for each cluster is obtained, they were fed into the prediction Algorithm 6. The accuracy for various *Timer-Limit* values using the VQ approach is shown in Figure 6(b). The reduced accuracy of VQ is because of the fact that it is a lossy technique, better suited for compression-based applications rather than recognition-based applications.

One can observe from Figure 6(b) that for each of the aforementioned algorithms, initially, as the *Timer-Limit* value is increased, the accuracy increases. However, for higher *Timer-Limit* values, the accuracy drops since almost all the oscillators start synchronizing within the limit. Thus, one can easily conclude that an accuracy-vs-speed tradeoff is involved, which can be programmed by the user. For higher accuracies, a higher *Timer-Limit* would be required, thereby increasing the runtime and energy consumption of the overall application. However, if inaccuracies are tolerable, then the runtime as well as the energy consumption can be reduced.

### 5.3 Structural Health Monitoring

The process of implementing damage detection and characterization of large civil structures (e.g., bridges) and tall buildings is referred to as Structural Health Monitoring. This process involves observing a civil structure periodically and making measurements of structural properties like deformation, strain, and the like, using an array of sensors, extracting damage related features from these measurements and finally deducing the current health state of the structure. In this article, we focus on the health monitoring systems employed for large-scale bridges.

Developing long-term monitoring systems for large-scale bridges is a critical task, since a failure in such a structure can lead to fatal accidents and huge economic losses. However, studies (Mita and Takhira 2003) have shown that high-precision sensors and sophisticated networks are not necessarily an apt solution for the task. In some cases, a simple measurement of the peak strain suffered by the structure is sufficient to detect damages. Authors of Bakht and Mufti (2015) have shown how peak strain detection across regular intervals can be used to calculate the axle load of moving trucks and also their velocity. The work presented in ASTM (1990) discusses how primary and secondary strain peaks can be used to estimate the fatigue life of the bridge.

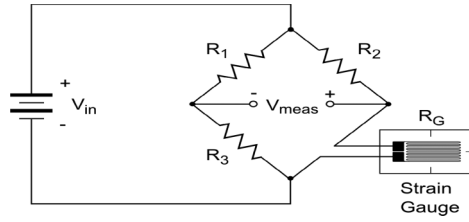


Fig. 7. A Wheatstone bridge with an active strain gauge.

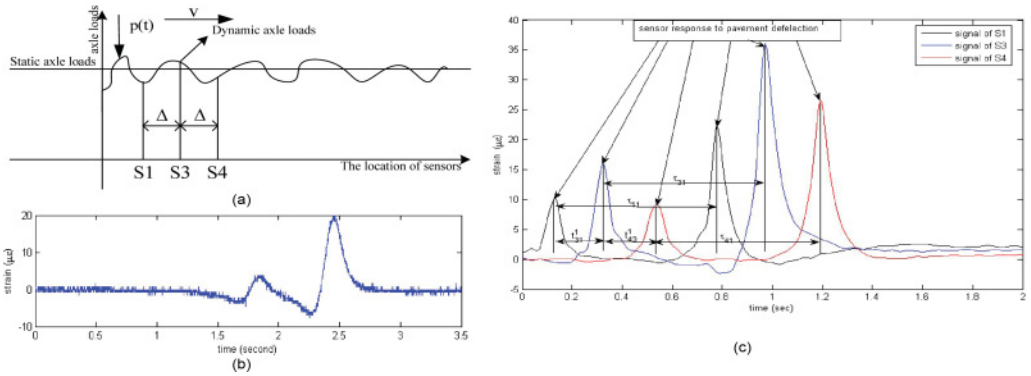


Fig. 8. (a) Location of the sensors traversed by a wheel of force  $p(t)$  and speed  $v$ . (b) Strain time histories of sensor S1 for a moving two-axle vehicle. (c) Example of sensor responses on the bridge. This data has been borrowed from Zhang et al. (2008) for demonstration purpose only.

Several different types of sensors have been proposed in the past to measure strains in bridges and civil structures (Westermo and Thompson 1994; Muto et al. 1992; Kakizawa and Ohno 1996). The most simple method to measure strain is through a strain gauge, a device whose variation in electrical resistance is directly proportional to the amount of strain incurred on the device. The observed changes in strain are usually very small and thus require measurement of very small changes in resistance. This can typically be achieved using a bridge configuration along with a voltage excitation source. Figure 7 shows a Wheatstone bridge, where one of its arms is an active strain gauge that will capture any changes in the strain as changes in output voltage  $V_{meas}$ . Typically, over a period of time, sufficient samples (50-75 samples) of voltage fluctuations are captured, converted to digital form, and then worked upon for primary and peak detection.

The work in Zhang et al. (2008) shows how an array of well-placed strain gauge sensors can be utilized to find the axle load of vehicles crossing a bridge. The work in Zhang et al. (2008) shows that the axle weight of a vehicle is a statistical relation of the strain induced on the sensors and the speed of the vehicle. Figure 8(a) shows the placement of three strain gauge sensors on the bridge. As a vehicle crosses a given sensor, the strain induced by the vehicles front and rear wheels will lead to large voltage changes which are captured by the Wheatstone bridge. Figure 8(b) shows the unfiltered strain-time histories of a 2-axle vehicle crossing sensor S1. The first peak is due to front wheel load while the second peak is due to rear wheel. Once these peaks have been determined, the speed can be calculated by dividing the distance between the strain sensor S1 and S3 by the difference in time between the axle first crossing S1 and then crossing S3 (refer to Figure 8(c)). The number of peaks in the strain-time histories can also be analyzed to find the number of heavy vehicles crossing the bridge and provide an estimate of the traffic pattern over the bridge.



**ALGORITHM 7:** Peak Detection**Input:**  $N$  Strain samples**Output:** Primary and secondary peaks

- 1  $M \leftarrow 2$ ;
- 2 Sort-( $M$ ,Dec);
- 3 Read the first and second element from the sorted array.

In current day conventional systems, the analog signal  $V_{meas}$  obtained from the Wheatstone bridge is converted to digital form using ADCs and then processed for further analysis. The advantage of using nano-oscillators for computation is the fact that they can directly work with analog signals, thus, giving away to the use of ADCs. Thus, to compute the primary and secondary peak among  $M$  samples, we simply use the  $N^{th}$ -Distinct Max instruction with  $N \leftarrow 1$  and  $N \leftarrow 2$ , respectively. In order to perform both the operations simultaneously, one can use the Sort- $N$  instruction with  $N \leftarrow 2$ . A sorted order of the peaks also helps in performing outliers analysis and further damage detection. It is also easier to filter out smaller vehicle peaks by ignoring the array after a threshold peak value has been crossed. Algorithm 7 shows the steps involved in obtaining the primary and secondary peaks from the co-processor. Once the sorting is performed, the host-pc can read the required sample values from the sorted array using AddrOut and DataOut ports of the read-out circuitry (Figure 3).

**5.3.1 Results.** Similar to the VQ analysis, we generated random sets of 50 samples in the range (0,300) and performed sort and  $N_{th}$ -Distinct Max instructions on them using the Kuramoto model. The output was verified against a conventional C++ implementation. In scenarios where there are multiple elements of the same value, the  $N_{th}$ -Distinct Max operation returns the lowest oscillator number containing the Max value. While the index output of the co-processor for Sorting and  $N^{th}$ -Distinct Max gave perfect results, using the timer values and a mapping function for estimating the peaks suffered from slight error (RMSE of .209), as mentioned in Figure 4.

## 6 PHYSICAL VALIDATION USING HYPERFET

While the Kuramoto model used in this article provides a generic soft model for coupled nano-oscillators, it still does not effectively capture all the challenges and issues involved in physical nano-oscillator implementations. In this section, we validate the proposed model using the HyperFET technology-based coupled nano-oscillators as proposed in Tsai et al. (2016) and address some of these challenges.

### 6.1 The HyperFET Model

Recent advances in the manufacturability of systems of complex oxides capable of performing spontaneous metal-insulator transitions under the application of a critical electric field (or temperature) have created an interest in realizing systems of coupled nano-oscillators. A material of particular interest is Vanadium Dioxide ( $\text{VO}_2$ ).  $\text{VO}_2$  is a transition oxide capable of achieving insulator-metal transitions just above room temperature. This phase transition is marked by an abrupt change in conductivity of up to five orders of magnitude as illustrated in Figure 9(a). The work of Datta et al. (2014), Parihar et al. (2014), and Jerry et al. (2016) has demonstrated how a  $\text{VO}_2$ -based compact, scalable, inductor-less, and a functionally dense relaxation oscillator capable of multi-GHz operation can be designed and fabricated in detail. The work in Tsai et al. (2016) shows how two  $\text{VO}_2$ -based Voltage Controlled Relaxation Oscillators (VCROs) can be coupled together through a capacitor in a traditional topology to build a two or more coupled nano-oscillator.

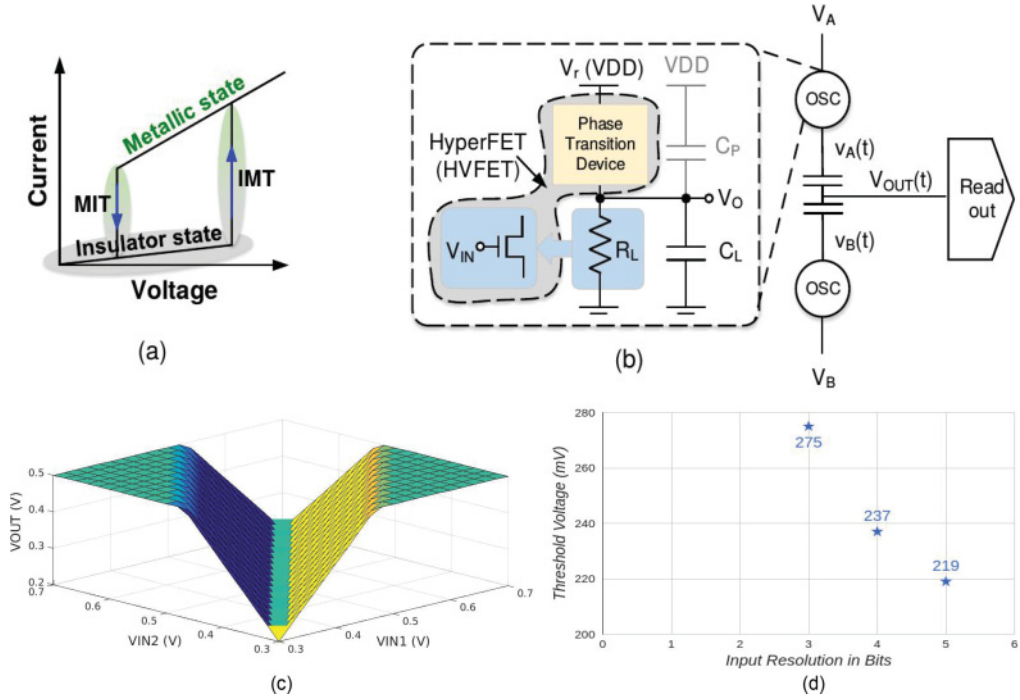


Fig. 9. (a) The insulator-metal transition for the  $\text{VO}_2$  in HyperFET. (b) Schematic of the traditional  $\text{VO}_2$ -based VCRO pair coupled through a capacitor. (c) The non-linear behavior of  $V_{OUT}$  with  $V_A(t)$  and  $V_B(t)$ . (d) The impact of threshold on the input resolution.

An example of this coupled oscillator circuit is replicated in Figure 9(b) for reference. When  $V_A(t) = V_B(t)$ , both VCROs synchronize at the same frequency and the phase difference between the outputs of the two oscillators stabilizes at  $\pi$ . As the difference between the  $V_A(t)$  and  $V_B(t)$  increases, the frequency difference also increases and, thus, out-of-phase oscillations can be observed at  $V_{OUT}$  after a few cycles. In Tsai et al. (2016), a few different topologies are introduced for supporting a different number of coupled oscillators and different read-out interfaces.

It is observed that the output signal  $V_{OUT}$  of Figure 9(b) reflects the relationship of input voltages  $V_A(t)$  and  $V_B(t)$ . The pulses of  $V_{OUT}$  will be minimal and of the same amplitude when  $V_A(t) = V_B(t)$ . In the case of  $V_A(t) \neq V_B(t)$ , the amplitude of  $V_{OUT}$  increases.  $V_{OUT}$  thus shares a non-linear relation with the inputs  $V_A(t)$  and  $V_B(t)$ , which has been captured in Figure 9(c). We exploit this characteristic of the device to differentiate between synchronizing and non-synchronizing inputs by employing a simple 1-bit comparator (1-bit ADC) at  $V_{OUT}$ . This comparator logic generates a pulse when the amplitude of  $V_{OUT}$  is below a particular threshold indicating the occurrence of synchronization. For  $V_{OUT}$  values above this threshold, the output of the comparator remains idle.

Works such as Shukla et al. (2015), Frougier et al. (2016), and Srinivasa et al. (2016) have been successful in experimentally fabricating HyperFETs, in which a MOSFET is built together with  $\text{VO}_2$  within the same die. Furthermore, the  $\text{VO}_2$  material can also be fabricated on top of the source or drain contact, thereby allowing the oscillators and the traditional MOSFET logic-based read-out circuit to exist within the same die without significant overheads. However, due to the emerging nature of these devices, the fabrication is limited to only a few devices and the entire design fabrication is not viable. Consequently, we use models calibrated with the physical realizations in this

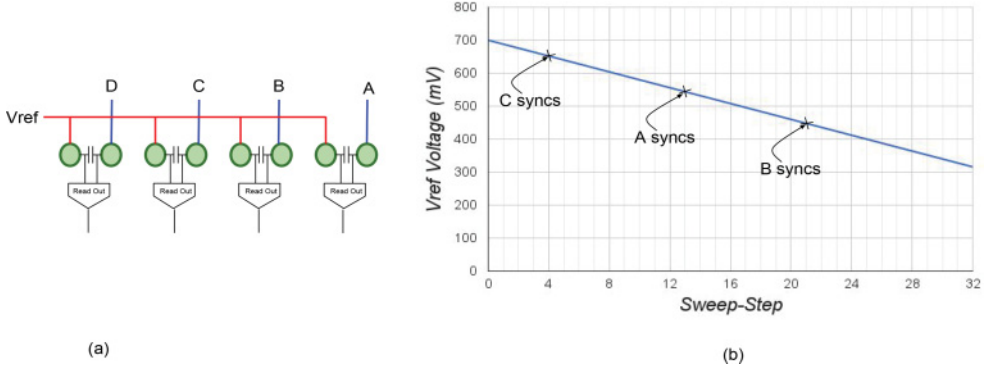


Fig. 10. (a) An array of four two-coupled oscillators. (b) Behavior for (a) when  $V_{ref}$  is swept from 700mV to 300mV in steps of 12mV.

work. Further, we have tuned the interface between the analog sensors and the coupled oscillators to fit the input voltage range supported by the device. In this article, we used the parameters from Tsai et al. (2016) to build a MATLAB model of the device, which supports an input voltage range of 0.3V-0.7V with an output voltage ( $V_{OUT}$ ) range of 0.2V-0.5V. Through MATLAB simulations, the devices are used for a 5-bit resolution for a threshold voltage of 219mV. In other words, the input voltages to the oscillator take up to 32 values between 0.3V-0.7V and the consecutive voltages are at least 12mV apart. Choosing a different threshold value leads to different input resolutions as shown in Figure 9(d).

## 6.2 Validating the ISA on HyperFET

The previously described coupled oscillator model achieves synchronization within very few cycles, which cannot be captured by the timer module or be directly used by the algorithms proposed in Section 4. We now describe the modifications required in the context of HyperFET-based oscillators to perform the necessary operations proposed in this article.

Let's consider the task of finding the 3<sup>rd</sup> maximum for a given set of four input samples - A, B, C, and D. Using the exact model and the granularity characteristics derived previously, we can assume the samples to take the following values: A=544mV, B=500mV, C=652mV, and D=400mV and the threshold voltage to be 219mV. Thus, the task of finding the 3<sup>rd</sup> maximum should return the sample B as the answer.

We use an array of four two-coupled oscillators to perform the required task as shown in Figure 10(a). As a first step, the input samples A, B, C, and D are latched onto one input of the respective oscillators. The other input of each oscillators is tied to common voltage source  $V_{ref}$ . As a next step, we sweep the voltage of  $V_{ref}$  from 700mV to 300mV in steps defined by the granularity selected (i.e., 12mV). As the  $V_{ref}$  voltage hits 652mV, the output of oscillator-C drops to 200mV causing the output  $V_{OUTC}$  of the 1-bit comparator to go high. This signal is fed to the WTA circuit of Figure 3, which increments the counter to one and the rest of the circuit progresses, as highlighted in Algorithm 1. At this point, the outputs of oscillators A, B, and D are 373mV, 468mV, and 500mV, respectively. Since each of these values are above the threshold limit of 219mV, the respective comparators do not produce any pulse. Similarly, when  $V_{ref} = 544mV$ , the output  $V_{OUTA}$  goes high and the counter is incremented to two. Finally, when  $V_{ref}$  hits 500mV, the output of oscillator-B drops to 200mV causing the counter to increment to three. The co-processor latches this as the output of the function. The timer module can be used to capture the number of sweeps

Table 4. Improvement in Speed, Power, and Area of the Proposed HyperFET Co-processor Against a Conventional CMOS Accelerator

	Proposed Read-Out Circuit	Conventional Accelerator	Gain
Speed	1.42GHz	$\approx 400\text{MHz}$	3.5×
Power	3.98mW	42.79mW	10.75×
Area	8,414	118,864	14.12×

required to find the  $3^{rd}$  maximum and can use it for further processing by the host-pc as described earlier in this article. This particular principle of voltage sweeping can easily be extended to a generic N-oscillator array and to other functions such as  $N^{th}$ -Distinct Minimum and Sorting as well. Figure 10(b) captures the behavior of the circuit over the entire instruction execution for the previously mentioned experiment. To calculate the DoM between two vectors, say A ( $a_1, a_2, \dots, a_n$ ) and B ( $b_1, b_2, \dots, b_n$ ), the corresponding elements of each vector are latched as inputs to the same oscillator, i.e.,  $a_i$  and  $b_i$  will be inputs to oscillator  $osc_i$ . The number of oscillators to synchronize will represent the exact DoM between the vectors. However, if the element ordering is not necessary and only the number of similar elements needs to be calculated, then the voltage source  $V_{ref}$  can be swept with discrete values, each corresponding to an element of B, while the other input to the oscillators remain to be elements of vector A.

The instruction-set of the proposed co-processor in this article majorly exploits the comparator property of the oscillators while leveraging the advantages of low-power to obtain the required speed and resolution. With this regard, the previously stated voltage sweeping scheme can also be carried out using simple analog comparators. However, nano-oscillators offer the same functionality with reduced power consumptions and higher sensitivity and fast settling behavior as compared to analog comparators. Additionally, the HyperFET model is capable of comparing multiple inputs (as shown in Tsai et al. (2016)) by coupling more than two oscillators together as opposed to employing an exponential number of comparators in the conventional approach. The work in Tsai et al. (2016) also presents some of the complex functions that can be performed using oscillators, which turn out to be very costly using simple comparators. The oscillators, by virtue of tuning their threshold values, provide a notion of *approximate matching*, which can be utilized by applications where grouping similar data is required. This property of configurable tolerance for approximation cannot be implemented using traditional voltage comparators. While the proposed co-processor majorly uses the comparator property of the coupled oscillators, the architecture is not limited to just this functionality and can be easily extended to compute more complex instructions as well.

## 7 AREA, POWER, AND SPEED BENEFITS

In this section, we present a comparison of the efficiency of the proposed HyperFET-based co-processor against a conventional CMOS accelerator performing the same instructions. For the oscillator-based co-processor, the global controller and the readout circuitry were designed in RTL using Bluespec System Verilog (BSV). The conventional CMOS circuitry performing the same functions was designed in BSV as well. Each of the circuits were designed to handle up to 32 simultaneous inputs, i.e., Sorting or DoM between 32 numbers or samples in parallel. Both designs were synthesized using the Synopsys Design Compiler using a 32nm technology. The speed, area, and power benefits of the proposed co-processor (including the oscillators and readout circuit) over the conventional CMOS accelerator are shown in Table 4. The implementation of the Match-Counting circuit of Figure 3(b) adds a further area and power overheads of 5.5% and 1.5%, respectively. The

improvement in speed is obtained due to the reduction in critical path of the design. The readout circuitry is also significantly smaller as compared to the conventional CMOS accelerator, which employs a large sorting circuitry and 32 comparators for performing the DoM function. In addition to the gains listed in Table 4, the HyperFET-based solution also has additional benefits at the analog digital interfaces. The HyperFET-based co-processor employs only a single bit comparator (1-bit ADC) for each oscillator at its output, which is much more power efficient than a 5-bit ADC ( $\approx 6\text{mV}$ ) employed for each input of the conventional CMOS accelerator.

While those results in Table 4 have already shown promising results with the model in Tsai et al. (2016), it could be even more encouraging when applied with further scaled devices when more orders of resistance difference in difference phases becomes possible, or when smaller operation voltage is feasible. In this article, we have employed HyperFETs to perform linear functions. However, the non-linear characteristics of the oscillator (shown in Figure 9(c)) can be further exploited to execute functions such as norm-distance calculation commonly used in image processing applications. The HyperFETs have shown to be much more power and area efficient in such scenarios than their conventional Boolean counterparts (Shukla et al. 2014; Tsai et al. 2016).

## 8 CONCLUSION

An ideal computing system would be one which is compact and consumes minimal power. With the end of classical Dennard Scaling and the eminent fall of Moore's law, it is not possible to further shrink size or reduce power consumption of conventional computing chips. It has thus become necessary to explore and adapt to alternate computing paradigms which provide a more elegant solution to the power crisis. This article explores the non-Boolean computing paradigm, which employs nano-oscillators for performing computations. A nano-oscillator-based co-processor is proposed, which is capable of performing basic functions commonly found in today's applications. The article explains in detail micro-architecture of the required co-processor. The processor is aimed at catering to approximate applications, which derive direct analog inputs and can tolerate a certain amount of error at the output. The article also describes how the accuracy of the output of the processor can be tuned by the programmer providing a runtime-vs-accuracy tradeoff. The article also verifies the proposed architecture with the help of a soft model of the Kuramoto oscillators. Several real-world applications have also been mapped on the architecture to highlight benefits. The validation of the architecture was carried out using the HyperFET technology-based coupled nano-oscillators, which provide a  $3.5\times$  speed-up in clock frequency while simultaneously reducing the area and power consumption by  $10.75\times$  and  $14.12\times$ , respectively, as compared to a conventional accelerator. The article also highlights some of the non-idealities and challenges involved in realizing physical oscillator implementations in the context of HyperFET transistors.

## REFERENCES

- ASTM. 1990. Standard practices for cycle counting in fatigue analysis. In *American Society for Testing and Materials, West Conshohocken, PA*. <https://www.astm.org/Standards/E1049.htm>.
- U. R. Babu, Y. Venkateswarlu, and A. K. Chintha. 2014. Handwritten digit recognition using k-nearest neighbour classifier. In *Proceedings of the 2014 World Congress on Computing and Communication Technologies (WCCCT)*, 60–65.
- Baidar Bakht and Aftab Mufti. 2015. *Bridges: Analysis, Design, Structural Health Monitoring, and Rehabilitation*. Springer International Publishing, Cham, Chapter Structural Health Monitoring, 307–354.
- Sebastian Thimotee Bartsch, Alexandru Rusu, and A. M. Ionescu. 2012. Phase-locked loop based on nanoelectromechanical resonant-body field effect transistor. *Applied Physics Letters* 101, 15 (2012), 153116.
- M. Bohr. 2007. A 30 year retrospective on Dennard's MOSFET scaling paper. *IEEE Solid-State Circuits Society Newsletter* 12, 1 (2007), 11–13.
- G. Bozis and John D. Hadjidemetriou. 1999. On the continuation of periodic orbits from the restricted to the general three-body problem. *Celestial Mechanics* 13, 2 (1999), 127–136.



- Trong Tu Bui and Tadashi Shibata. 2008. Compact bell-shaped analog matching-cell module for digital-memory-based associative processors. *Japanese Journal of Applied Physics* 47, 4S (2008), 2788.
- Ke Chen and DeLiang Wang. 2002. A dynamically coupled neural oscillator network for image segmentation. *Neural Networks* 15, 3 (2002), 423–439.
- Tianshi Chen, Zidong Du, Ninghui Sun, Jia Wang, Chengyong Wu, Yunji Chen, and O. Temam. 2015. A high-throughput neural network accelerator. *IEEE Micro* 35, 3 (May2015), 24–32.
- Yunji Chen, Tao Luo, Shaoli Liu, Shijin Zhang, Liqiang He, Jia Wang, Ling Li, Tianshi Chen, Zhiwei Xu, Ninghui Sun, and Olivier Temam. 2014. DaDianNao: A machine-learning supercomputer. In *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'14)*. IEEE Computer Society, Washington, D.C., 609–622.
- Vinay K. Chippa, Srmat T. Chakradhar, Kaushik Roy, and Anand Raghunathan. 2013. Analysis and characterization of inherent application resilience for approximate computing. In *Proc. DAC*. 113:1–113:9.
- N. Chopra and M. W. Spong. 2005. On synchronization of Kuramoto oscillators. In *Proceedings of the 44th IEEE Conference on Decision and Control, 2005 and 2005 European Control Conference (CDC-ECC'05)*. 3916–3922.
- N. Chopra and M. W. Spong. 2009. On exponential synchronization of Kuramoto oscillators. *IEEE Trans. Automat. Control* 54, 2 (Feb.2009), 353–357.
- G. Csaba, M. Pufall, D. Nikonov, G. Bourianoff, A. Horvath, T. Roska, and W. Porod. 2012. Spin torque oscillator models for applications in associative memories. In *Proceedings of the 2012 13th International Workshop on Cellular Nanoscale Networks and Their Applications (CNNA)*. 1–2.
- S. Datta, N. Shukla, M. Cotter, A. Parihar, and A. Raychowdhury. 2014. Neuro inspired computing with coupled relaxation oscillators. In *Proceedings of the 2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*. 1–6.
- Robert H. Dennard, Fritz H. Gaensslen, Hwa nien Yu, V. Leo Rideout, Ernest Bassous, Andre, and R. Leblanc. 1974. Design of ion-implanted MOSFETs with very small physical dimensions. *IEEE J. Solid-State Circuits* (1974), 256.
- Hadi Esmaeilzadeh, Emily Blem, Renee St. Amant, Karthikeyan Sankaralingam, and Doug Burger. 2011. Dark silicon and the end of multicore scaling. In *Proceedings of the 38th Annual International Symposium on Computer Architecture (ISCA'11)*. ACM, New York, NY, 365–376.
- J. Frougier, N. Shukla, D. Deng, M. Jerry, A. Aziz, L. Liu, G. Lavallee, T. S. Mayer, S. Gupta, and S. Datta. 2016. Phase-transition-FET exhibiting steep switching slope of 8mV/decade and 36 2016 IEEE Symposium on VLSI Technology. 1–2.
- S. Furui. 1991. Vector-quantization-based speech recognition and speaker recognition techniques. In *Proceedings of the 1991 Conference Record of the Twenty-Fifth Asilomar Conference on Signals, Systems and Computers*. 954–958.
- Neel Gala, V. R. Devanathan, K. Srinivasan, V. Visvanathan, and V. Kamakoti. 2014. ProCA: Progressive configuration aware design methodology for low power stochastic ASICs. In *Proceedings of the 2014 27th International Conference on VLSI Design and 2014 13th International Conference on Embedded Systems*. 342–347.
- Neel Gala, V. R. Devanathan, V. Visvanathan, and V. Kamakoti. 2015. Best is the enemy of good: Design techniques for low power tunable approximate application specific integrated chips targeting media-based applications. *Journal of Low Power Electronics* 11, 2 (2015), 133–148.
- Neel Gala, V. R. Devanathan, V. Visvanathan, V. Gandhi, and V. Kamakoti. 2013. Tunable stochastic computing using layered synthesis and temperature adaptive voltage scaling. In *Proceedings of the 2013 5th Asia Symposium on Quality Electronic Design (ASQED)*. 103–112.
- Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. 2013. Internet of things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems* 29, 7 (2013), 1645–1660.
- Frank C. Hoppensteadt and Eugene M. Izhikevich. 1999. Oscillatory neurocomputers with dynamic connectivity. *Phys. Rev. Lett.* 82, 14 (Apr.1999), 2983–2986.
- David Horn and Irit Opher. 1999. Collective excitation phenomena and their applications. In *Pulsed Neural Networks*. MIT Press, 297–320.
- Matthew Jerry, Wei yu Tsai, Baihua Xie, Xueqing Li, Vijay Narayanan, Arijit Raychowdhury, and Suman Datta. 2016. Phase transition oxide neuron for spiking neural networks. In *Proceedings of the 74th Device Research Conference (DRC)*.
- John L. Johnson. 1994. Pulse-coupled neural nets: Translation, rotation, scale, distortion, and intensity signal invariance for images. *Applied Optics* 33, 26 (1994), 6239–6253.
- S. Kaka, M. R. Pufall, W. H. Rippard, T. J. Silva, S. E. Russek, and J. A. Katine. 2006. Mutual phase-locking of microwave spin torque nano-oscillators. In *Proceedings of the IEEE International INTERMAG 2006 Magnetics Conference*. 2–2.
- T Kakizawa and S Ohno. 1996. Utilization of shape memory alloy as a sensing material for smart structures. In *Proc. Advanced Composite Materials in Bridges and Structures*. 67–74.
- H. B. Kekre, A. A. Athawale, and G. J. Sharma. 2011. Speech recognition using vector quantization. In *Proceedings of the International Conference Workshop on Emerging Trends in Technology (ICWET'11)*. ACM, New York, NY, 400–403.
- H. B. Kekre, T. K. Sarode, V. A. Bhargadi, A. A. Agrawal, R. J. Arora, and M. C. Nair. 2010. Iris recognition using vector quantization. In *Proceedings of the International Conference on Signal Acquisition and Processing, 2010 (ICSAP'10)*. 58–62.



- Y. Kim, I. Hong, and H. J. Yoo. 2015. 18.3 A 0.5V 54 W ultra-low-power recognition processor with 93.5% accuracy geometric vocabulary tree and 47.5% database compression. In *Proceedings of the 2015 IEEE International Solid-State Circuits Conference (ISSCC)*. 1–3.
- Yoshiki Kuramoto. 2012. *Chemical Oscillations, Waves, and Turbulence*. Vol. 19. Springer Science & Business Media.
- K. Kyamakya, J. C. Chedjou, M. A. Latif, and U. A. Khan. 2010. A novel image processing approach combining a coupled nonlinear oscillators'-based paradigm with cellular neural networks for dynamic robust contrast enhancement. In *Proceedings of the 2010 12th International Workshop on Cellular Nanoscale Networks and Their Applications (CNNA)*. 1–7.
- S. Levitan, Y. Fang, D. Dash, T. Shibata, D. Nikonov, and G. Bourianoff. 2012. Non-boolean associative architectures based on nano-oscillators. In *Proceedings of the 2012 13th International Workshop on Cellular Nanoscale Networks and Their Applications (CNNA)*. 1–6.
- M. Lichman. 2013. UCI Machine Learning Repository. Retrieved from <http://archive.ics.uci.edu/ml>.
- Hongzhi Liu, Qiyong Guo, Mantao Xu, and I-Fan Shen. 2008. Fast image segmentation using region merging with a k-nearest neighbor graph. In *Proceedings of the 2008 IEEE Conference on Cybernetics and Intelligent Systems*. 179–184.
- Xiuwen Liu and DeLiang L. Wang. 1999. Range image segmentation using a relaxation oscillator network. *IEEE Transactions on Neural Networks* 10, 3 (1999), 564–573.
- Akira Mita and Shinpei Takhira. 2003. A smart sensor using a mechanical memory for structural health monitoring of a damage-controlled building. *Smart Materials and Structures* 12, 2 (2003), 204.
- Norio Muto, Hiroaki Yanagida, Teruyuki Nakatsuji, Minoru Sugita, Yasushi Ohtsuka, and Yasuhiro Arai. 1992. Design of intelligent materials with self-diagnosing function for preventing fatal fracture. *Smart Materials and Structures* 1, 4 (1992), 324.
- V. Narayanan, S. Datta, G. Cauwenberghs, D. Chiarulli, S. Levitan, and P. Wong. 2014. Video analytics using beyond CMOS devices. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*. 1–5.
- Nasser M. Nasrabadi and Robert A. King. 1988. Image coding using vector quantization: A review. *IEEE Transactions on Communications* 36, 8 (1988), 957–971.
- Dmitri E. Nikonov, Gyorgy Csaba, Wolfgang Porod, Tadashi Shibata, Danny Voils, Dan Hammerstrom, Ian A. Young, and George I. Bourianoff. 2013. Coupled-oscillator associative memory array operation for pattern recognition. In *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits* 1, 85–93.
- A. Parihar, N. Shukla, S. Datta, and A. Raychowdhury. 2014. Exploiting synchronization properties of correlated electron devices in a non-Boolean computing fabric for template matching. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 4, 4 (Dec2014), 450–459.
- P. H. Pham, D. Jelaca, C. Farabet, B. Martini, Y. LeCun, and E. Culurciello. 2012. NeuFlow: Dataflow vision processing system-on-a-chip. In *Proceedings of the 2012 IEEE 55th International Midwest Symposium on Circuits and Systems (MWSCAS)*. 1044–1047.
- Poonam and M. Dutta. 2012. Performance analysis of clustering methods for outlier detection. In *Proceedings of the 2012 2nd International Conference on Advanced Computing Communication Technologies*. 89–95.
- M. Price, J. Glass, and A. P. Chandrakasan. 2014. 27.2 A 6mW 5k-word real-time speech recognizer using WFST models. In *Proceedings of the 2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. 454–455.
- T. Roska, A. Horvath, A. Stubendek, F. Corinto, G. Csaba, W. Porod, T. Shibata, and G. Bourianoff. 2012. An associative memory with oscillatory CNN arrays using spin torque oscillator cells and spin-wave interactions architecture and end-to-end simulator. In *Proceedings of the 2012 13th International Workshop on Cellular Nanoscale Networks and Their Applications (CNNA)*. 1–3.
- Tamás Roska and Ángel Rodríguez-Vázquez. 2002. Toward visual microprocessors. *Proc. IEEE* 90, 7 (2002), 1244–1257.
- Mrigank Sharad, Karthik Yogendra, and Kaushik Roy. 2013. Energy efficient computing using coupled dual-pillar spin torque nano oscillators. *IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*. 28–29.
- T. Shibata, R. Zhang, S. Levitan, D. Nikonov, and G. Bourianoff. 2012. CMOS supporting circuitries for nano-oscillator-based associative memories. In *Proceedings of the 2012 13th International Workshop on Cellular Nanoscale Networks and Their Applications (CNNA)*. 1–5.
- N. Shukla, A. Parihar, M. Cotter, M. Barth, X. Li, N. Chandramoorthy, H. Paik, D. G. Schlom, V. Narayanan, A. Raychowdhury, and S. Datta. 2014. Pairwise coupled hybrid vanadium dioxide-MOSFET (HVFET) oscillators for non-Boolean associative computing. In *Proceedings of the 2014 IEEE International Electron Devices Meeting (IEDM)*. 28.7.1–28.7.4.
- Nikhil Shukla, Arun V. Thathachary, Ashish Agrawal, Hanjong Paik, Ahmedullah Aziz, Darrell G. Schlom, Sumeet Kumar Gupta, Roman Engel-Herbert, and Suman Datta. 2015. A steep-slope transistor based on abrupt electronic phase transition. *Nature Communications* 7, 6 (2015), 7812.
- S. Srinivasa, A. Aziz, N. Shukla, X. Li, J. Sampson, S. Datta, J. P. Kulkarni, V. Narayanan, and S. K. Gupta. 2016. Correlated material enhanced SRAMs with robust low power operation. *IEEE Transactions on Electron Devices* 63, 12 (2016), 4744–4752.

- Wei-Yu Tsai, Xueqing Li, Matt Jerry, Baihua Xie, Nikhil Shukla, Huichu Liu, Nandhini Chandramoorthy, Matthew Cotter, Arijit Raychowdhury, Donald Chiarulli, S. P. Levitan, S. Datta, J. Sampson, N. Ranganathan, and V. Narayanan. 2016. Enabling new computation paradigms with HyperFET-an emerging device. In *IEEE Transactions on Multi-Scale Computing Systems* 2, 1 (2016), 30–48.
- David W. L. Wang and David Terman. 1997. Image segmentation based on oscillatory correlation. *Neural Computation* 9, 4 (1997), 805–836.
- Dana Weinstein and Sunil A Bhawe. 2010. The resonant body transistor. *Nano Letters* 10, 4 (2010), 1234–1237.
- B. D. Westermo and L. D. Thompson. 1994. Smart structural monitoring: A new technology. *Sensors* 11, 11 (1994), 15–18.
- Arthur T. Winfree. 1967. Biological rhythms and the behavior of populations of coupled oscillators. *Journal of Theoretical Biology* 16, 1 (1967), 15–42. <http://www.sciencedirect.com/science/article/pii/0022519367900513>.
- Zheng Yang, Changhyun Ko, and Shriram Ramanathan. 2011. Oxide electronics utilizing ultrafast metal-insulator transitions. *Annual Review of Materials Research* 41 (2011), 337–367.
- Wenbin Zhang, Chunguang Suo, and Qi Wang. 2008. A novel sensor system for measuring wheel loads of vehicles on highways. *Sensors* 8, 12 (2008), 7671.

Received June 2016; revised March 2017; accepted April 2017