Hyperdimensional Computing Exploiting Carbon Nanotube FETs, Resistive RAM, and Their Monolithic 3D Integration

Tony F. Wu, Haitong Li, Ping-Chen Huang, Abbas Rahimi, Gage Hills, Bryce Hodson, William Hwang, Jan M. Rabaey, *Fellow, IEEE*, H.-S. Philip Wong, *Fellow, IEEE*, Max M. Shulaker, and Subhasish Mitra, *Fellow, IEEE*

***Abstract*— The field of machine learning is witnessing rapid advances along several fronts: new machine learning models, new machine learning algorithms utilizing these models, new hardware architectures for these algorithms, and new technologies for creating energy-efficient implementations of such hardware architectures. Hyperdimensional (HD) computing represents one such model. Emerging nanotechnologies, such as carbon nanotube field-effect transistors (CNFETs), resistive random-access memory (RRAM), and their monolithic 3D integration, enable energy- and area-efficient hardware implementations of HD computing architectures. Such efficient implementations are achieved by exploiting several characteristics of the component nanotechnologies (e.g., energy-efficient logic circuits, dense memory, and incrementers naturally enabled by gradual reset of RRAM cells) and their monolithic 3D integration (enabling tight integration of logic and memory), as well as various characteristics of the HD computing model (e.g., embracing randomness that allows us to utilize rather than avoid inherent variations in RRAM and CNFETs, resilience to errors in the underlying hardware). We experimentally demonstrate and characterize an end-to-end HD computing nanosystem built using monolithic 3D integration of CNFETs and RRAM. Using our nanosystem, we experimentally demonstrate the pairwise classification of 21 languages with measured mean accuracy of up to 98% on >20 000 sentences (6.4 million characters), training using one text sample (~100 000 characters) per language, and resilient operation (98% accuracy) despite 78% of bits in HD representation being stuck at 0 or 1 in hardware. We also show that the monolithic 3D implementations of HD computing can have 35× improved energy-execution time product for training and inference of language classification data sets (while using 3× less area) compared to silicon CMOS implementations.**

***Index Terms*— Carbon nanotube (CNT), CNT field-effect transistor (CNFET), hyperdimensional (HD) computing, monolithic 3D integration, nanosystems, resistive random-access memory (RRAM).**

Manuscript received April 15, 2018; revised June 25, 2018 and August 20, 2018; accepted August 30, 2018. Date of publication October 3, 2018; date of current version October 22, 2018. This work was supported in part by the Defense Advanced Research Projects Agency, in part by National Science Foundation/Nanoelectronics Research Initiative/Global Research Collaboration Energy-Efficient Computing: from Devices to Architectures, in part by STARnet SONIC, and in part by the Stanford SystemX Alliance. This paper was approved by Guest Editor Patrick Mercier. *(Corresponding author: Tony F. Wu.)*

T. F. Wu, H. Li, B. Hodson, W. Hwang, and H.-S. P. Wong are with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305 USA (e-mail: tonyfwu@stanford.edu; haitongl@stanford.edu; bkhodson@stanford.edu; hwangw@stanford.edu; hspwong@stanford.edu).

P.-C. Huang is with Qualcomm, San Jose, CA 95110 USA (e-mail: pingchen.huang@berkeley.edu).

A. Rahimi is with the Department of Information Technology and Electrical Engineering, ETH Zürich, 8092 Zürich, Switzerland, and also with the Department of Electrical Engineering and Computing Sciences, University of California at Berkeley, Berkeley, CA 94720 USA (e-mail: abbas@ee.ethz.ch).

G. Hills and M. M. Shulaker are with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: ghills@mit.edu; shulaker@mit.edu).

J. M. Rabaey is with the Department of Electrical Engineering and Computing Sciences, University of California at Berkeley, Berkeley, CA 94720 USA (e-mail: jan_rabaey@berkeley.edu).

S. Mitra is with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305 USA, and also with the Department of Computer Science, Stanford University, Stanford, CA 94305 USA (e-mail: subh@stanford.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSSC.2018.2870560

I. Introduction

Hyperdimensional (HD) computing is a brain-inspired computing model that is built upon a set of well-defined operations on high-dimensional binary vectors [1]. As a result, it does not require brute-force hyperparameter tuning unlike deep neural networks [2]. HD computing promises fast, energy-efficient training and inference [3]. This paper is about efficient hardware realizations of HD computing by utilizing emerging nanotechnologies. Hardware realizations of HD computing are expected to (as with other machine learning computing models as well): 1) tightly integrate computing and storage to reduce energy consumption and latency associated with data transfer [3], [4]; 2) employ efficient circuit implementations (e.g., through approximation) that save energy while achieving "good enough" accuracy levels as required by the application [5]; and 3) utilize variability in underlying technologies (rather than minimize it) for computing models, such as HD that embrace randomness [6].

To achieve tight integration of energy-efficient memory and compute elements, we employ monolithic 3D integration, whereby multiple layers of transistors and memory are vertically integrated (sequentially) and connected using short, high-density interlayer vias (i.e., standard vias used to connect metal layers in the interconnects of integrated circuits today) [7], [8]. Compared to traditional chip stacking [9], monolithic 3D integration can provide orders of magnitude greater memory bandwidth [10]. As has been experimentally demonstrated [11], monolithic 3D integration is naturally enabled by carbon nanotube field-effect transistors (CNFETs) and resistive random-access memory (RRAM) due to low-temperature fabrication (<250 °C). CNFETs promise an order-of-magnitude improvement in energy-delay-product versus silicon CMOS [12], enabling energy-efficient computation. RRAM is an emerging memory technology that promises high-capacity and non-volatile (as well as multiple bits per cell) data storage with improved speed, energy efficiency, and density compared to dynamic RAM (DRAM) [13]. Computing system architectures using monolithic 3D integration of CNFETs and RRAM promise to significantly improve the energy efficiency (expressed as the product of energy and execution time) of a wide variety of emerging applications (e.g., deep learning and graph analytics), often in the range of 1000× [10], [14].

In this paper, we design, fabricate, and demonstrate an HD computing nanosystem for cognitive tasks, such as language recognition. We use monolithic 3D integration of CNFETs and RRAM. We exploit RRAM and CNFETs to create area- and energy-efficient monolithic 3D circuit blocks that combine CNFETs with fine-grained access to RRAM memories:

1) circuits that embrace inherent variations in RRAM and CNFETs with estimated 3× lower dynamic energy (versus silicon CMOS implementations at the same technology node) stemming from both the circuit topology and the use of energy-efficient CNFETs;
2) approximate incrementer circuits using gradual RRAM reset operation, which can use 30× fewer transistors versus full-digital incrementer implementations;
3) ternary content-addressable memory (TCAM) cells built using pairs of CNFETs and RRAM, which use 19× lower energy (simulated versus SRAM-based TCAM cells) due to reduced leakage of non-volatile RRAM.

Using our HD nanosystem hardware prototype, we experimentally demonstrate the followings:

1) pairwise classification of 21 European languages with measured accuracy of up to 98% on >20 000 sentences (6.4 million characters) per language pair;
2) on-chip learning using one text sample (∼100 000 characters) per language;
3) resilient operation (98% accuracy) despite 78% of bits in HD representation being stuck at 0 or 1 in hardware.

When implemented at a scaled technology node (e.g., 28 nm), HD nanosystems like ours are projected to offer 35× energy-efficiency (product of energy and execution time) improvement with a simultaneous 3× reduction in area footprint compared to conventional silicon CMOS implementations. For larger HD nanosystems, the benefits can be potentially greater.

Our previous paper [15] introduced our monolithic 3D nanosystem for HD computing. This paper extends [15] in the following ways.

1) We provide a detailed overview of system functionality as well as more measurement data to characterize our nanosystem.
2) We analyze the projected energy, delay, and area of HD computing nanosystems implemented using monolithic 3D integration of CNFETs and RRAM at scaled technology nodes (e.g., 28 nm) to demonstrate the benefits of monolithic 3D nanosystems for brain-inspired computing models.

The rest of this paper is organized as follows. Section II describes the HD computing model and the component nanotechnologies, Section III explains our HD nanosystem implementation and highlights the properties of nanotechnologies we rely on, and Section IV explains the projected benefits of implementing our system in scaled technology nodes. We conclude this paper in Section V.

II. HD Computing Model and Component Nanotechnologies

In this section, we introduce HD computing and the component nanotechnologies used in our nanosystem.

A. Hyperdimensional Computing

HD computing performs computations on high-dimensional vectors using a set of well-defined operations. One of its attractive features is learning using a few examples for various cognitive tasks (due to the non-iterative nature of its training mechanisms) [1]. In HD computing, inputs are represented using high-dimensional vectors (e.g., 10 000 bits), also referred to as HD vectors (HVs). HVs are processed using operations on binary vectors (discussed later in this section).

Training on input samples is equivalent to a process of combining HVs representing the inputs into a new HV of same dimensionality (which can be easily distinguished from other HVs during inference). The high-dimensional nature of HD framework [16], [17] allows for resilience to errors in the HVs. In general, increasing the dimension of HVs increases overall classification accuracy. However, it also manifests as large area footprint when implemented in hardware. HD computing has been shown to be effective (i.e., similar or higher accuracy compared to state-of-the-art methods, such as the *k*-nearest neighbors (KNNs) algorithm, support vector machines, or neural networks) for cognitive tasks, such as language classification, electromyography (EMG)-based hand gesture recognition, electroencephalograph (EEG) brain-machine interfaces, and human activity recognition [3], [18]–[21], while requiring fewer training samples [19]. Moreover, tasks beyond object recognition (such as analogy making and logical inference) have been demonstrated using HD computing with cellular automata [22].

In this paper, we focus on a case study for HD computing: classification of European languages given an input sentence. For language classification, an input sentence (i.e., a sequence of characters) is first transformed into a sequence of binary HVs. Each unique character is mapped to a unique HV through a process called *random projection* (see Fig. 1). Next, a series of three HD operations is performed on the sequence of HVs: 1) HD permutation; 2) HD multiplication; and 3) HD addition. HD permutation is a 1-bit rotating shift of an HV, HD multiplication is a bitwise XOR of two HVs, and HD addition is bitwise accumulation of multiple HVs followed by a threshold operation (to transform back to a binary HV). The same threshold is used for all elements in the HV. These three operations are performed on the sequence of binary HVs

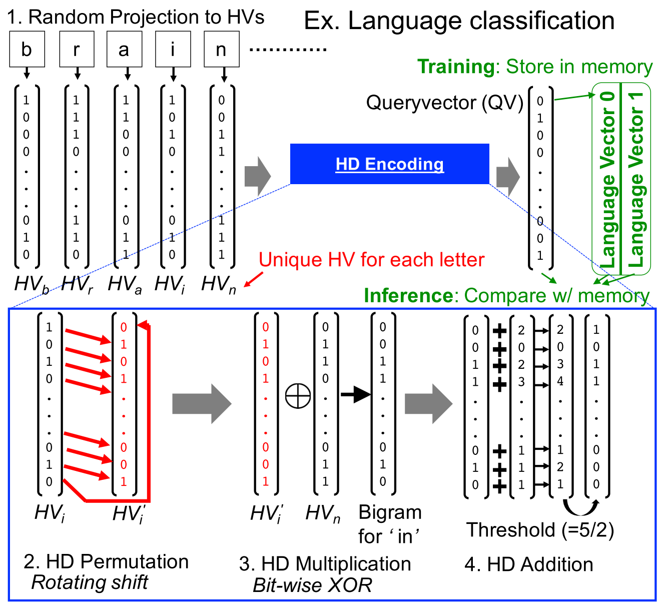


Fig. 1. HD computing, showing main operations. 1—random projection. 2—HD permutation. 3—HD multiplication. 4—HD addition in the context of language classification. QVs are stored as language vectors in memory during training. QVs are compared to stored language vectors during inference.

by the HD encoder to produce a single HV that represents the entire sentence called a *queryvector* (QV) (explained in more detail in Section III-B).

For language classification, HVs are combined into bigrams (i.e., HVs that represent groups of two adjacent letters) (or trigrams for three, tetragrams for four, and so on) using the HD permutation and HD multiplication operations. To create a bigram, the HV that corresponds to a preceding character is first rotated, and then, bitwise XOR is performed with the HV that corresponds to the current character, which encodes a bigram. Then, all bigrams in the sentence are accumulated bitwise, producing the QV (as shown in Fig. 1).

During training, this QV is chosen to represent a particular language (in its entirety, i.e., all sentences corresponding to a language are encoded into a single QV) and stored in memory, which is called a *language vector* (see Fig. 1). During inference, a QV is compared to all of the stored language vectors using the Hamming distance (i.e., the number of elements that do not match between the vectors). The language represented by the language vector that corresponds to the least Hamming distance from QV is chosen as the output language.

B. Component Nanotechnologies

In this paper, we use CNFETs for logic circuits, RRAM for memories, and their monolithic 3D integration (through low-temperature fabrication) to realize the hardware implementations of HD computing architectures.

1) Carbon Nanotube Field-Effect Transistors: CNFETs represent an emerging transistor technology, which promises an order of magnitude improvement in energy-delay-product (a metric of energy efficiency) for digital circuits [12]. CNFETs use multiple CNTs (cylindrical structures of carbon atoms 1–2 nm in diameter) that act as channels (see Fig. 2). CNTs simultaneously enable high carrier mobility and excel-

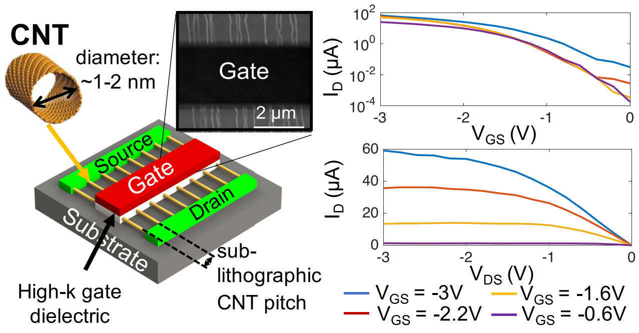


Fig. 2. CNFET. The scanning electron microscopy image shows multiple parallel CNTs in the channel region. Example measured I–V curves are shown.

lent electrostatic control in CNFETs [23], enabling highly energy-efficient digital logic circuits. CNFETs can be used in high-performance complementary logic (with 10^6 I_{ON}/I_{OFF} ratio, the drive current/off-state leakage current) [24], [25] and can be built at scaled gate lengths (5 nm) [26]. Hysteresis-free CNFETs have been demonstrated in [27]. CNFETs have been fabricated as negative capacitance FETs with sub-55-mV/decade subthreshold swing at room temperature [28].

Imperfections inherent in CNTs, such as mis-positioned CNTs (that can lead to stray conducting paths resulting in incorrect functionality) and metallic CNTs (i.e., CNTs with little or no bandgap), were a major limitation in the past—the *imperfection-immune paradigm* overcomes these obstacles through a combination of fabrication and design techniques [29]–[31]. The imperfection-immune paradigm enables wafer-scale fabrication and VLSI-compatible design of CNFET circuits. This paradigm has since enabled experimental demonstrations, such as the first CNT computer and the first 3D nanosystem consisting of over 2 million CNFETs on a single die [11], [32]. Since CNFETs can be fabricated at low temperature (<250 °C), they naturally enable monolithic 3D integration (discussed in Section II-B3).

In addition to process variations that exist in silicon transistors (e.g., variations in threshold voltage, channel length, and oxide thickness), CNFETs are subject to CNT-specific variations, such as CNT count variations (i.e., variations in the number of CNTs in a CNFET). These variations cause drive current variations, which can manifest as delay variations in digital circuits. These variations can be suppressed using optimized process and circuit design [30]. However, in this paper, we utilize these inherent variations (instead of suppressing them) to perform the random projection operation for HD computing. In particular, we capitalize on variations in CNT count and threshold voltage (discussed in detail in Section III-A).

2) Resistive RAM: RRAM is an emerging memory technology, which promises high-capacity, non-volatile data storage (10-year retention) and can be fabricated at low temperature (<250 °C) [13]. RRAM is fabricated as a metal oxide switching layer (insulator) sandwiched between two metallic electrodes. While RRAM can be realized using various metal–insulator–metal material combinations [13], we use Pt/HfO_X/TiN RRAM, where the bottom electrode is Pt, the top electrode is TiN, and the switching layer is HfO_X (see Fig. 3).

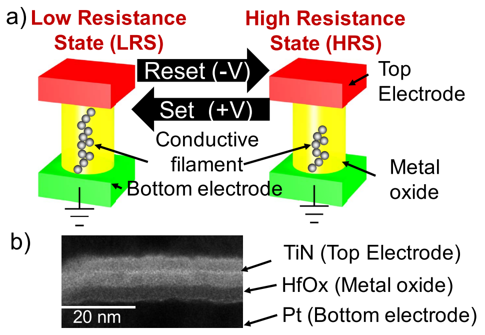


Fig. 3. RRAM. (a) Set operation increases the conductive filament length, while the reset operation decreases the conductive filament length. (b) Transmission electron micrograph of RRAM cell.

Unlike [33] and [34] that suggest bitwise HD operations inside RRAM cells, we utilize the following properties of RRAM in our demonstration: digital storage (i.e., store a 0 or 1), gradual reset (i.e., ability to increment the RRAM resistance in a fine-grained manner), and its stochasticity (i.e., variations in RRAM resistance) for HD computing (see Section III).

Three main operations are typically performed on an RRAM cell: set, reset, and read. The *set* operation transforms the cell from high-resistance state (HRS) to low-resistance state (LRS) by applying a positive voltage (i.e., *set voltage*, V_{set}) across the top and bottom electrodes [13]. In our chip, the set voltage is 2.6 V. A transistor is used to limit the current for the set operation (called the *compliance current*). This creates or lengthens a filament of oxygen vacancies from the bottom electrode to the top electrode. As the length of the conductive filament increases, the resistance of the RRAM decreases [13]. In most cases, a higher set voltage (called *forming voltage*, 4 V in our chip) is applied to form the filament (once) after fabrication. However, forming-less RRAM cells have also been demonstrated [13], [35]. A *reset operation* transforms the cell from LRS to HRS by applying a negative voltage (i.e., *reset voltage*, V_{reset}) across the top and bottom electrodes, rupturing the filaments between the electrodes. In our chip, the reset voltage is −2.6 V. RRAM cells with <2-V set/reset voltage (∼10-ns pulse duration) and 10–100 HRS/LRS resistance ratio have been demonstrated [13], [36]. RRAM is also subject to variations in its resistance, stemming from the stochastic size and shape of the conductive filament after a set or reset operation [13]. A *read* operation detects the state of the cell (e.g., HRS or LRS) by sensing the current after applying a small voltage (e.g., 0.5 V in our chip) across the two electrodes. This voltage is small enough to not change the resistance of the cell.

Although RRAM has limited write (i.e., set/reset) endurance (10^{12} cycles at the cell level [36] and 10^5–10^7 cycles at the array level [37], [38]), HD computing is robust to endurance-related errors [18]. Moreover, our nanosystem only performs onewrite cycle (i.e., set and reset) for each text sample inferred or trained and can endure one year of continuous operation (i.e., repeatedly training the entire training set and inferencing all of the sentences in the data set) on the data set given an RRAM write endurance of 10^7 cycles.

Although many cell structures (e.g., 1 transistor–1 RRAM cell, 1 transistor–*n* RRAM cell, 1 selector–1 RRAM cell [13]) may be used, in our chip, we use the 1 transistor–1 RRAM cell (1T-1R) structure (see Fig. 5) as it prevents current overshoot during the set operation [13]. Array-level implementations using the 1T-1R RRAM structure have been demonstrated up to 16 Gb of capacity [39].

A single RRAM cell can store a single bit or multiple bits (see [40]). To demonstrate multi-bit storage in RRAM cells, one or a combination of set or reset parameters are adjusted to change the resistance of the cell to an intermediate value (between LRS and HRS): compliance current in the set operation, reset voltage, and set or reset pulse duration. These parameters can also be adjusted to gradually increase the RRAM cell resistance (i.e., increasing the resistance incrementally). This gradual increase in RRAM cell resistance has been demonstrated for a variety of switching layers (i.e., the material in which the filament forms, such as HfO_X [41]) [13] by using short pulse durations (50-μs pulses in this paper) during the reset operation. In this paper, we call this behavior as *gradual reset*.

3) Monolithic 3D Integration: Monolithic 3D integration is a process, whereby tiers of circuits (i.e., a layer of logic, memory, or sensors) are fabricated on top of each other on the same substrate. Monolithic 3D integration uses inter-layer vias (ILVs), standard vias used to connect adjacent metal layers in the interconnect stack of today's silicon CMOS technologies, to connect between tiers of circuits. This is in contrast to chip stacking using through-silicon vias (TSVs) with typical pitches of around 10 μm [9]. ILVs can have the same pitch as metal interconnects (100 nm at 28-nm technology node [42]), enabling significantly denser vertical connectivity compared to TSVs [43], a key to tight integration between logic and memory.

Monolithic 3D integration requires low-temperature fabrication for upper tiers of circuits (<400 °C) as higher temperatures can damage existing circuits (transistors and interconnects) on the bottom tiers. While this is difficult for traditional silicon CMOS technologies (e.g., high-temperature requirements for dopant activation >1000 °C), it is naturally enabled by CNFETs and RRAM due to their low-temperature fabrication [13], [44]. For our chip, we fabricate CNFETs and RRAM in which all fabrication steps on the wafer have a maximum temperature of 200 °C. Monolithic 3D integration of CNFETs, RRAM, and silicon transistors has been demonstrated [45], demonstrating compatibility with silicon CMOS. Recently, a four-layer monolithic 3D nanosystem consisting of >2 million CNFETs with 1-Mb RRAM on top of >1 million silicon transistors has been demonstrated [11].

III. Implementing HD Computing in Hardware

Fig. 4 shows the overall architecture of our nanosystem. It contains three main blocks: random projection unit, HD encoder, and HD classifier. Inputs (text characters) are streamed into the random projection unit and mapped to HVs (explained in detail in Section III-A); the HD encoder applies HD operations (HD permutation, HD multiplication, and HD

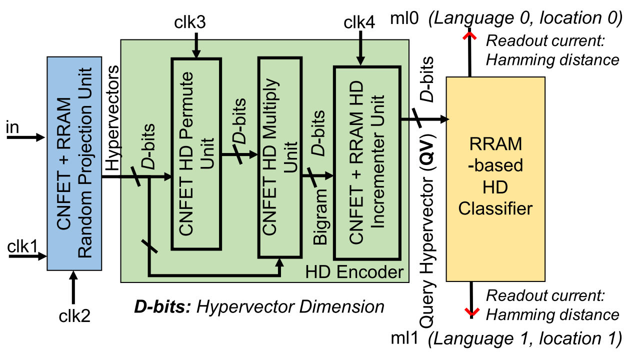
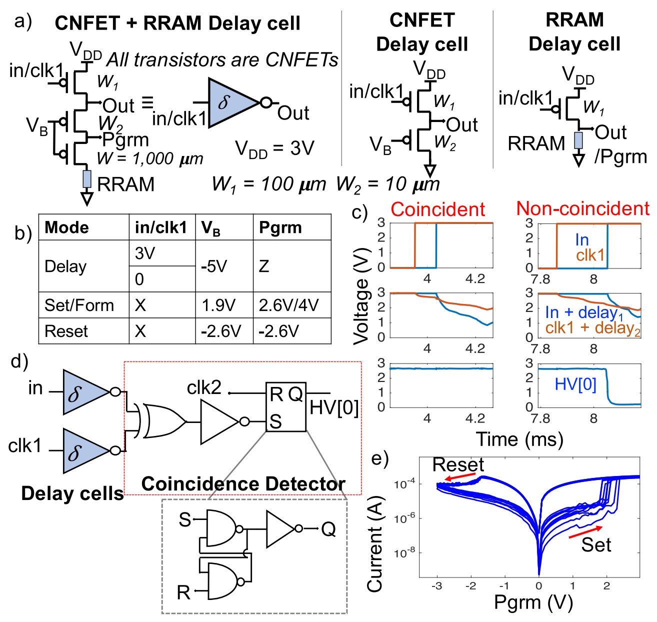

in
clk1
clk2
CNFET + RRAM Random Projection Unit
Hypervectors
clk3
clk4
D-bits
CNFET HD Permute Unit
CNFET HD Multiply Unit
Bigram
CNFET + RRAM HD Incrementer Unit
HD Encoder
Query Hypervector (QV)
RRAM -based HD Classifier
ml0 (Language 0, location 0)
Readout current: Hamming distance
ml1 (Language 1, location 1)
D-bits: Hypervector Dimension

Fig. 4. Block diagram of our HD nanosystem.



Mode	in/clk1	V_B	Pgrm
Delay	3V	-5V	Z
	0		
Set/Form	X	1.9V	2.6V/4V
Reset	X	-2.6V	-2.6V

Fig. 5. (a) Schematics of CNFET + RRAM delay cell, CNFET delay cell, and RRAM delay cell (with only an RRAM cell in the pull-down network), showing (b) operating voltages for inputs and (c) measured waveforms using CNFET + RRAM delay cells and coincidence detectors. (d) Schematic of coincidence detector. (e) Programming (Set and Reset) curve of the CNFET + RRAM delay cell using the *Pgrm* pin.

addition, explained in detail in Section III-B) to produce a QV to represent the entire text; the HD classifier stores the QV (in RRAM) during training or compares the QV to stored (i.e., *learned*) HVs using the Hamming distance metric during inference (explained in detail in Section III-C). Our nanosystem uses four-phase operations, governed by four clocks (clk1, clk2, clk3, and clk4, explained in Section III-B). While this architecture is tailored specifically for language classification, additional random projection units may be required for other applications, such as biosignal classification using EMG [19] and EEG [20].

A. Random Projection

A key requirement for HD computing is that the unique inputs should be mapped to unique HVs that are at a maximal (Hamming) distance (i.e., Hamming distance of 50% of the HV dimension, D) from all other HVs [1]. Moreover, the same inputs must map to the same HVs. One approach approximates this mapping through random projection (i.e., randomly choose the mapping) to obtain the large Hamming distances [3], [18], [30]. In this paper, we achieve the Hamming distance of 16 between HVs for $D = 32$.

To achieve random projection, several methods can be used. Random projection may be hard-coded or reconfigurable (i.e., the mapping of inputs to HVs can be changed) during either test or operation in the field. For hard-coded random projection, the mapping is determined during design time (e.g., hard-coded into the RTL as a truth table) [18]. While this method may be comparable in terms of energy and delay versus some reconfigurable methods, it generally has a larger area footprint (details in Section IV). Reconfigurable random projection may be realized in several different ways: using pseudo-random number generators [e.g., linear feedback shift registers (LFSRs)] or utilizing natural randomness (e.g., stochastic behavior of RRAM [33]) to generate the mapping. This mapping can be stored in an on-chip memory for lookup during random projection to recall the same HV for each unique input. We compare these methods in detail in Section IV.

In this paper, we implement random projection by embracing inherent variations in our component nanotechnologies: variations in RRAM resistance values (discussed in Section II-B2) and in CNFET drive currents (discussed in Section II-B1). Our approach can achieve up to 22× less energy compared to other random projection methods outlined earlier (details in Section IV).

To utilize inherent variations in RRAM and CNFETs, we create *delay cells* [PMOS-only CNFET inverters with an additional RRAM in the pull-down network (see Fig. 5)] to translate device-level variations into delay variations of the delay cell circuits. In addition, we also fabricated delay cells with CNFETs only and delay cells with RRAM only to characterize the individual effects of RRAM and CNFET variations (also shown in Fig. 5). The pull-up CNFETs are all sized the same (i.e., same width) with the pull-down transistor in CNFET delay cells sized at 1/10 of the pull-up transistor for increased gain and swing of the PMOS-only logic. In CNFET + RRAM delay cells, an additional wide transistor [see Fig. 5(a)] is inserted in the pull-down network and used to set and reset the RRAM by applying a voltage on the drain of the CNFET (*Pgrm*) [see Fig. 5(e)]. During delay mode [see Fig. 5(b)], Pgrm is set to high impedance to allow the delay cell to operate as an inverter.

To quantify delay variations of delay cells, we use the σ/μ metric (σ: standard deviation of delays and μ: mean delay). HD computing generally requires HVs corresponding to inputs to be nearly orthogonal to each other (the Hamming distance close to 50% of HV dimension D) [1]. To generate HVs using delay cells, each possible input (26 letters of the alphabet and the space character) is first time-encoded and mapped to an evenly spaced delay [maximum delay $T = 2$ μ (see Fig. 6)]. For example, "A" = 0, "B" = $T/27$, "C" = $2T/27$, and so on. To calculate each bit of the HV, random delays are added to clk1 and input (in) using CNFET + RRAM delay cells. If the resulting signals are coincident (the falling edges are close enough to set the SR latch), the output is "1"

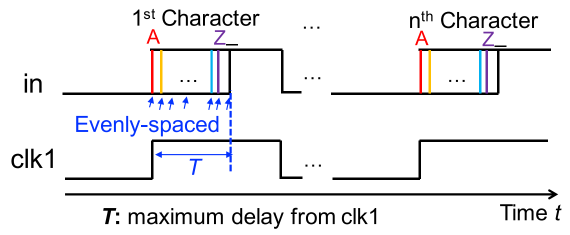


Fig. 6. Time encoding of a sentence of length n; 27 possible characters are encoded as a time delay from a reference clock, *clk1*. The encodings of the 27 characters are evenly spaced (e.g., “A” = 0, “B” = T/27, “C” = 2T/27, and so on) with $T = 2\mu$ ($2\times$ the mean delay cell delay).

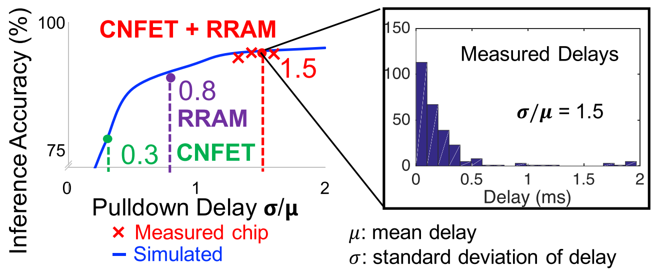
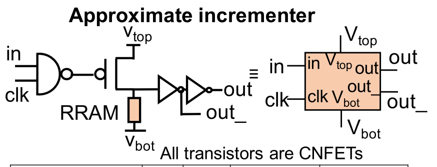


Fig. 7. Delay variation and its effect on inference accuracy. Higher inference accuracy can be achieved by using delay cells with both CNFETs and RRAM since the generated HVs have higher Hamming distance values from each other (i.e., the individual characters of the alphabet are mapped to HVs that are further apart from each other). Measured inference accuracies (each from a different chip) are similar to simulated accuracies (for the same delay variations). In general, higher Hamming distance values between HVs result in higher classification accuracy [1].



Mode	in	clk	V_{top}	V_{bot}
Increment	3V	3V	0V	2.6V
	0			
Read	X	0	2.5V	2.0V
Set/Form	X	0	3.7V/4V	1.1V/0V

Fig. 8. Approximate incrementer using RRAM.

(Fig. 5(c) and (d), a coincidence detector). Each element in the HV is an output of a single coincidence detector consisting of an XNOR gate and an SR latch. Our 3D nanosystem contains 64 delay cells and 32 coincidence detectors, forming the random projection unit.

In our design, larger variations (thus, larger σ/μ) correspond to the greater Hamming distance between HVs. Through simulations of our nanosystem (see Fig. 7), we show that higher values of σ/μ correspond to higher classification accuracy. Delay cells and coincidence detectors were simulated in SPICE using a CNFET model [46] and a resistor for the RRAM. CNT count variations and RRAM resistance variations were swept until the desired delay variations were achieved. Then, HVs that correspond to each character in the alphabet were extracted from the SPICE simulations. This mapping of the 27 characters was then used to simulate pairwise language classification on a software implementation of the HD computing algorithm [18].

In general, the smaller the width of the pull-down transistor (corresponding to width W_2), the higher the σ/μ. To increase variations in CNFET drive current, W_2 is set to the minimum width corresponding to the technology node. While delay cells can be created using CNFETs only (i.e., CNFET *delay cells*), our measured CNFET delay cells only achieve $\sigma/\mu = 0.3$, which results in low pairwise classification accuracy (76%). Alternatively, by utilizing inherent RRAM variations only (i.e., RRAM *delay cells*, with only an RRAM cell in the pull-down network), we achieve $\sigma/\mu = 0.8$, which corresponds to higher pairwise classification accuracy compared to CNFET delay cells. However, by combining the RRAM resistance variations and CNFET drive current variations, our *CNFET + RRAM* delay cells achieve $\sigma/\mu = 1.5$ (measured), corresponding to the mean Hamming distance of 16 for 32-bit HVs, resulting in higher accuracy (see Fig. 7, where actual chip measurements are superimposed on top of simulation results to obtain classification accuracy).

To initialize delay cells, the RRAM resistance is first reset to HRS and then set to LRS. To reset the RRAM to HRS, V_B is set to −2.6 V, whereas Pgrm is set to −2.6 V. To set the RRAM to LRS, V_B is set to 1.9 V and Pgrm is set to 2.6 V (see Fig. 5). This process is performed before training. During random projection operation, the resistance of the RRAM in each delay cell does not change (i.e., the voltage across the RRAM (0–1 V) is not enough to change its resistance). Thus, the same input will generate the same output HV.

B. HD Encoder

The HD encoder transforms a sequence of HVs representing a sequence of characters into a single vector, QV, representing the entire text. It follows the method described in [18]. Groups of two adjacent HVs in the sequence (bigrams) are first combined into a single HV (representing the bigram) using the HD multiplication and HD permutation operations. The current HV (HV_2) is multiplied with a permuted (rotating shift) version of the previous HV $\rho(HV_1)$)to form the HV representing the bigram ($\rho(HV_1) \star HV_2$). The permutation of the previous HV is used to preserve the order of the letters and distinguish between the pairs of characters in different order (e.g., “ab” versus “ba”). In hardware, we use a 1-bit shift with D-latches and a bitwise XOR to implement the HD permutation and HD multiplication operations, respectively.

Then, all HVs (each representing a bigram) are added together using the HD addition operation. The HD addition operation corresponds to counting the number of ones in each bit location of the HVs. For example, HD addition of binary vectors 0100, 0101, and 1011 produces (1, 2, 1, 2). Thresholding is then performed to transform the vector back into a binary vector [e.g., (1, 2, 1, 2) turns into 0101 for threshold of 1.5]. The threshold value is set to half of the number of total HVs added (e.g., threshold 50 for 100-character sentence). For language classification, the average input sentence contains fewer than 128 characters (requiring 7 bits of precision to perform HD addition on 128 HVs) [47]. In hardware, we use incrementers to implement the HD addition operation.

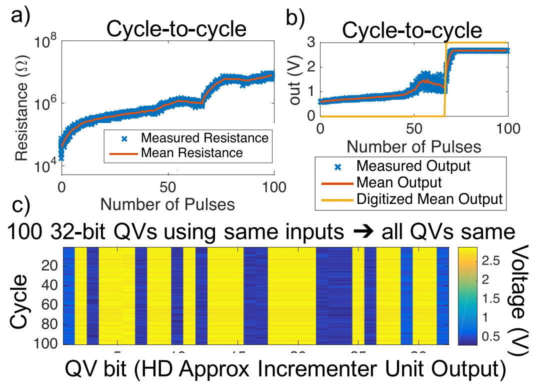


Fig. 9. Characterization of approximate incrementer using CNFETs and RRAM. (a) Gradual resistance increase in the RRAM when more reset pulses are applied (single cell shown). (b) Measured output of the approximate incrementer. (c) 100 repeated cycles of the same HV sequence at the inputs to the HD incrementer, showing consistent outputs (same QVs are generated).

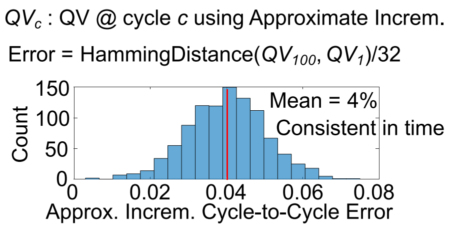


Fig. 10. Characterization of approximate incrementer error, showing mean cycle-to-cycle error (i.e., fraction of QV bits different in cycle 100 and cycle 1) of just 4%.

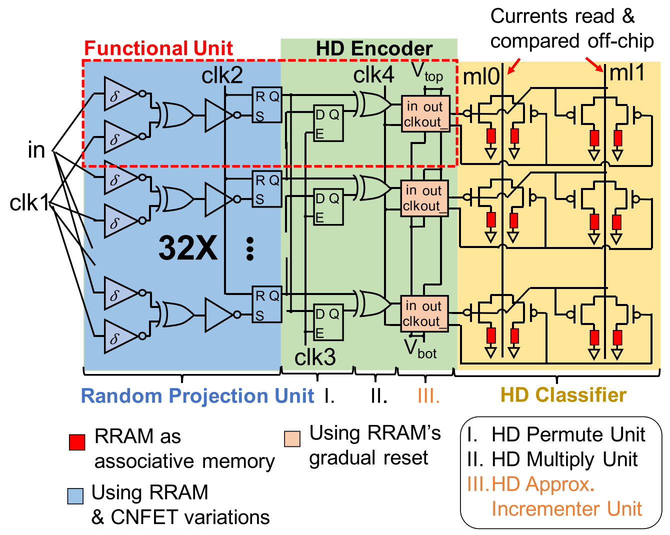


Fig. 11. Schematic of our HD nanosystem. It consists of 32 functional units connected in parallel. Each block (Random Projection unit, HD encoder, and HD classifier) uses a different unique property RRAM and/or CNFETs. The Random Projection Unit uses RRAM and CNFET variations in delay cells to randomly map inputs (in) to HVs. clk2 is used to reset this unit. The HD encoder uses CNFET digital logic (to permute and multiply HVs) and HD approximate incrementers using the gradual reset property of RRAM to accumulate HVs. Then, the CNFET and RRAM TCAM cells compare the HVs using current summing in the HD classifier.

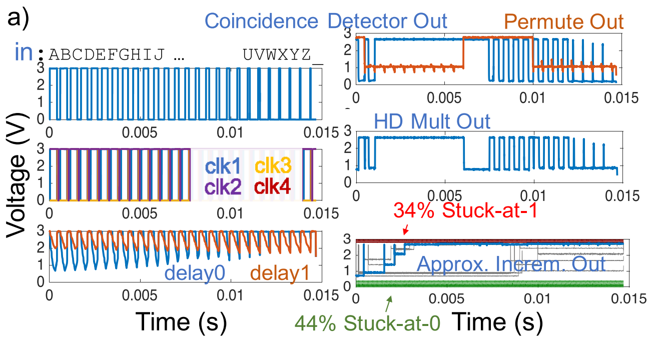
To implement an approximate incrementer, we utilize the gradual reset property of RRAM to realize area-efficient HD addition [41] (see Fig. 8). When in its LRS, the RRAM can be incrementally reset by applying short reset pulses, gradually increasing the resistance to HRS [see Fig. 9(a)]. This provides the ability to store non-binary values inside the RRAM. We leverage the multiple values of RRAM resistance that can be programmed by performing a gradual reset (V_{top}− V_{bot}: −2.6 V, 50-μs pulsewidth, and 1-ms period) to count the number of ones in each bit location of the HV (see Fig. 8). When the input (in, corresponding to a bit location of an HV) is "1," a reset pulse is applied to the RRAM cell when the clock is high. When the value is read ($V_{top} - V_{bot}$: +0.5 V), a pair of inverters threshold (transform) the sum to a binary value. The threshold value, typically representing half of the total number of bits incremented, can be adjusted using the voltages V_{top} and V_{bot} (determining the voltage range of the input of the first inverter). Multiple (one per HV bit) approximate incrementers form the HD approximate incrementer unit.

Since the incremented value (for each location of an HV) is thresholded to a binary value, the impact of errors due to the approximate nature of the HD approximate incrementer unit is somewhat mitigated (with measured mean cycle-to-cycle error 4%, showing consistency across time) (see Fig. 10). Here, we define the cycle-to-cycle error as the normalized Hamming distance (i.e., the Hamming distance divided by the HV dimension) of a QV at cycle 1 and a QV at cycle 100. The same sequence of input HVs (with dimension 32) was used to generate the QVs [see Fig. 9(c)]; thus, identical QVs should be generated each cycle. To arrive at a mean error of 4%, 1000 trials were performed.

To clear the sum, a set operation is performed on the RRAM (V_{top}–V_{bot}: +2.6 V). In HD computing, one approximate incrementer is used for each bit in the HV (i.e., D incrementers, where $D =$ HV dimension). Each such approximate incrementer uses eight transistors and a single RRAM cell. In contrast, a digital 7-bit incrementer (used for language classification of sentences of 128 characters) may use 240 transistors. Thus, for large values of D (e.g., 10 000), the savings' transistor count can be significant (30×).

We create multiple *functional units* (shown in Fig. 11): each functional unit contains a delay cell pair with a coincidence detector, a D-latch for HD permutation, an XOR gate for HD multiplication, and an HD approximate incrementer for HD addition. The functional units, when connected in parallel, form the random projection unit and the HD encoder. Fig. 12 shows measured waveforms of functional unit. The functional units use a four-phase operation (each controlled by a separate clock: clk1, clk2, clk3, and clk4): project, multiply and increment, permute, and reset projection. First, during the random projection phase, a new input (character) is transformed into an HV (described in Section III-A). Then, the HV is multiplied by a permuted HV (of the previous input character) and incremented. The HV is then permuted via the HD permutation operator. Each D-latch takes its data input from the output of the coincidence detector belonging to a neighboring functional unit. Finally, the coincidence detectors are all reset (each SR-latch is reset, described in Section III- A). This four-phase process is repeated for every single character in the text to produce the final output of the HD encoder (QV).

a)



b)

	Project	Multiply + Increm.	Permute	Reset Proj.
in	+3V	0	0	0
clk1	+3V	0	0	0
clk2	0	0	0	+3V
clk3	0	0	+ 3V	0
clk4	0	+3V	0	0
V_{top}	0	0	0	0
V_{bot}	0	+2.6V	0	0

Fig. 12. (a) Measured waveforms of a functional unit (see Fig. 11), showing individual output waveforms for each of the HD operations. A sample output of a coincidence detector, HD permute, and HD multiplication is shown; 32 approximate incrementer waveforms (from 32 functional units) are superimposed, showing 34% of the approximate incrementer outputs stuck-at-1 and 44% of the outputs stuck-at-0. (b) Each cycle of operation of the functional unit includes four steps: 1) an input character is projected into a HV; 2) multiply the increment; 3) permute the HV for the next cycle; and 4) reset the random projection unit for the next cycle.

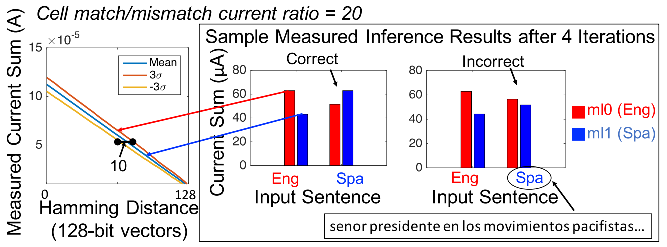


Fig. 13. Measured output currents of the HD classifier distinguishing between English and Spanish languages. When an English sentence is classified correctly, *ml*0 shows higher current. When a Spanish sentence is classified correctly, *ml*1 shows higher current.

C. HD Classifier

For each QV generated by the HD encoder, training or inference is performed by the HD classifier. During training, the QV is stored in the HD classifier in a location (specified during operation) representing the trained language. It is then referred to as a language vector (described in Section II-A). During inference, the QV is compared (using the Hamming distance metric) to all the stored language vectors in parallel. The language that represents the language vector corresponding to the shortest Hamming distance is reported as the classified language.

The HD classifier is implemented using 2-CNFET transistors and 2-RRAM cells (2T2R) TCAM cells to form an associative memory [48] (see Fig. 11). The TCAM cells store the language vectors and calculate the Hamming distance between the language vectors and the QV. TCAM cells are used to include the flexibility of using language vectors and QVs with smaller dimension (i.e., <32) by setting the unused cells to a "do not care" state (LRS–LRS or HRS–HRS).

Initially, all RRAM cells are in HRS. During training, a matchline [e.g., *ml*0 or *ml*1 (see Fig. 11)] corresponding to the language (e.g., *ml*0 for English and *ml*1 for Spanish) is set to 3 V, writing the QV into the RRAM cells (see Fig. 11). Since the gates of the transistors in the TCAM cell are connected to complementary signals, when writing the QV, each RRAM cell in the TCAM cell will store complementary values (i.e., LRS–HRS or HRS–LRS) in order to compute a bitwise XOR during inference.

During inference, *ml*0 and *ml*1 are set to a low voltage (e.g., 0.5 V), and the current on each matchline is read as an output. When a QV bit is equal to the value stored in a TCAM cell (match), the current is high (~1.3 μA); otherwise (mismatch), the current is low (~0.06 μA). Each individual TCAM cell has current ratio of ~20 between the match and mismatch states (see Fig. 12). Cell currents are summed on *ml*0 and *ml*1 with each current inversely proportional to the Hamming distance (shown in Fig. 13). The line with the most current corresponds to the output class (read and compared off-chip).

We perform *pairwise* language classification on 21 European languages (e.g., English versus Spanish, English versus German, and Spanish versus French). Sample texts from the Worschatz Corpora [49] for training texts and sample sentences from the European Parallel Corpus [47] are used for inference. By testing the functional units individually, we determine that 25 out of the 32 functional units have outputs permanently stuck at 0 or 1 (34% at 1 and 44% at 0). Thus, some QV and language vector bits are either stuck at 0 or 1.

The cause of these stuck-at bits at the outputs of the functional units (bits corresponding to the QV) is primarily be attributed to performing all fabrication steps in an academic facility. To analyze the cause of these stuck-at bits, we examine each functional unit's electrical behavior (e.g., switching behavior of internal nodes and leakage current). We determined that the cause of 80% of the stuck-at bits can either be attributed to weaknesses in the gate dielectric of a CNFET in the functional unit (i.e., the breakdown voltage of the gate dielectric being much lower than the expected 16 V for 20 nm of aluminum oxide, characterized experimentally), causing significant gate leakage current and dielectric breakdown in the CNFETs [see Fig. 14(a)], or due to the high current drive variability in the CNFETs. The following behaviors were observed: incorrect combinational logic behavior (e.g., XOR gate behaving like an AND gate), SR latches or D-latches not being able to hold state (e.g., latch cannot hold a "1" or "0" state), or incorrect behaviors of access transistors to the RRAM (in the delay cells or the approximate incrementer). One or a combination of these behaviors can result in the output of the functional units (bits corresponding to the QV) being stuck. While CNFETs with the high current drive variability are needed for delay cells (in Section III-A), they can reduce the yield of digital logic (i.e., cause incorrect switching behavior) [31]. We found no correlation in the location of the observed behaviors [i.e., different functional

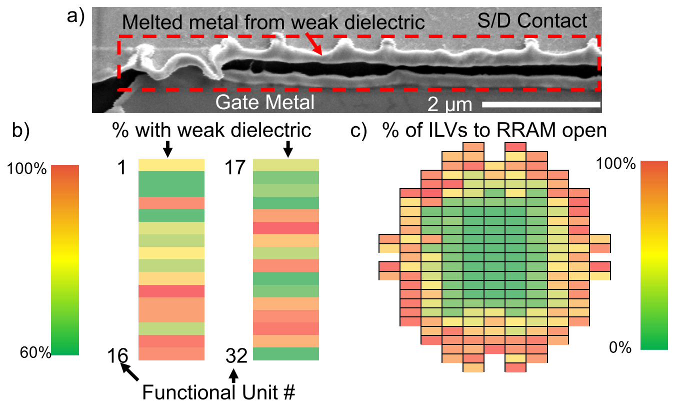


Fig. 14. (a) Scanning electron micrograph (SEM) of gate dielectric failure. (b) Distribution of functional unit failure due to gate dielectric weakness. (c) Distribution of open ILVs across the wafer.

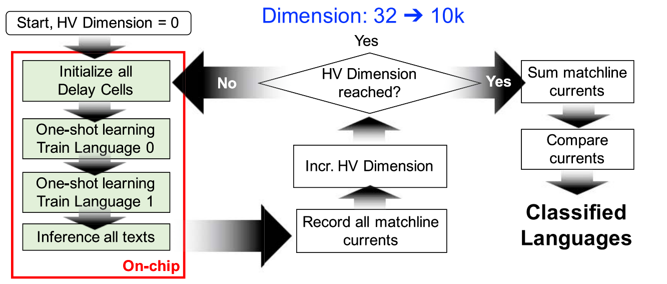


Fig. 15. Iterative method to emulate HD computing systems with larger vector dimension.

units as well as different transistors in the functional units were affected across the wafer, see Fig. 14(b)].

The dielectric weakness can be mitigated using more robust gate dielectric deposition procedures, such as those used in foundry processes, which are not available in our academic facility (e.g., using a machine that only deposits one type of material to avoid contamination). The other 20% of the stuck-at bits are attributed to open vias to the RRAM (confirmed by measuring test structures for the vias), causing open circuits; thus, the RRAM cells could not be set and reset. We found that dies on the edges of the wafer tended to exhibit more of these open vias [see Fig. 14(c)] since the etch rate using our equipment is spatially nonuniform for the vias. Despite these stuck-at bits, our nanosystem still achieves 98% pairwise classification accuracy using our iterative method, described in the following.

D. Emulating Larger HD Computing Systems

We emulate larger HD computing systems (more bits per HV, i.e., higher values of D) by running our fabricated nanosystem iteratively, as illustrated in Fig. 15. In each iteration, language pairs are trained (each language is trained on one text sample of 100 000 characters), generating new language vectors stored in the HD classifier, and all inferences are performed. The currents from all inferences are recorded off-chip. After each iteration, the RRAM in the delay cells is cycled (i.e., reset to HRS, then set to LRS). This provides a new set of mappings from input characters to HVs in the random projection unit. This creates a different segment of a higher dimension HV, generating another segment of the QV for each sentence encoded. Then, more iterations are performed. After all iterations, the recorded currents from *ml*0 and *ml*1 of the corresponding sentences are summed and compared off-chip.

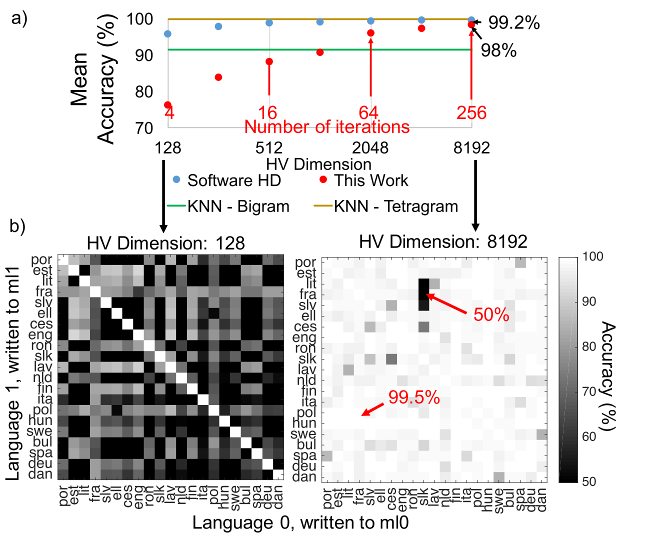


Fig. 16. (a) Measured inference accuracy for various vector dimensions. As the vector dimension increases, our nanosystem accuracy approaches the software benchmark. (b) Accuracy values for all language pairs using HV dimension of 128 and 8192. Accuracy for most language pairs increases significantly with larger HV dimension.

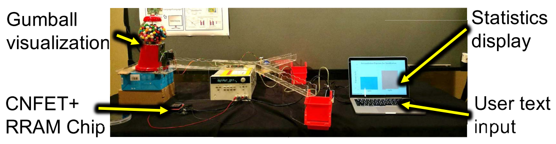


Fig. 17. Image of the live demonstration of the brain-inspired 3D nanosystem at the ISSCC 2018.

For a single iteration, the accuracy is 59%. Using 256 iterations with 22% non-stuck QV bits ($D = 8192$ with 1792 non-stuck QV bits), our nanosystem can categorize between two European languages with a mean accuracy (i.e., arithmetic mean of inference accuracies between all language pairs) of 98% (see Fig. 16). The classification energy per iteration is measured at 540 μJ (3-V supply, average power 5.4 mW, 1-kHz clock frequency, and 1-μm gate length). The reported accuracy is the percentage of 84 000 sentences that were classified correctly (data set: 420 language pairs, 200 sentences per language pair) [47]. We compare our nanosystem against a software implementation of the HD algorithm as well as a software implementation of a KNNs algorithm (state-of-the-art for language classification [18]). For bigrams, using KNN shows less accuracy than both the software HD algorithm and our nanosystem using >64 iterations. However, by using tetragrams, the KNN algorithm shows comparable accuracy as the software HD algorithm, albeit using 20× more memory (1.7 MB versus 86 KB), as shown in [18]. Using 256 iterations (and an additional 2 KB of off-chip memory for recorded currents), our nanosystem approaches the accuracy of the software HD implementation (using 8192-bit HVs with HV sparsity 0.5, achieving 99.2% accuracy) [21].

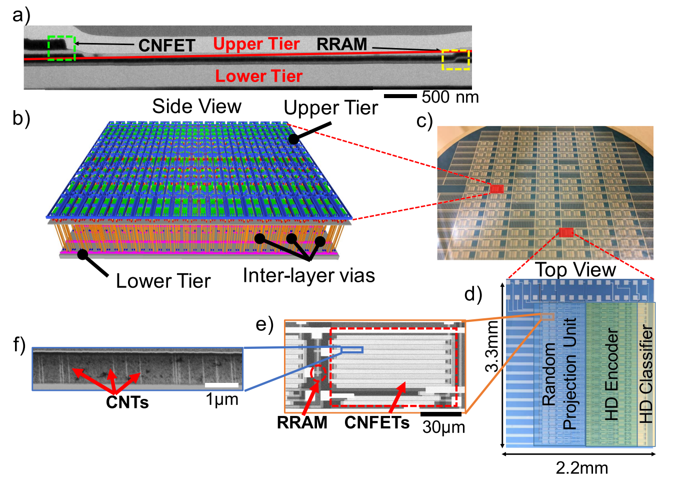


Fig. 18. (a) Transmission electron micrograph of our chip, showing CNFETs on the upper tier and RRAM on the lower tier. (b) Computer rendering of our nanosystem, showing CNFETs on the upper tier and RRAM on the lower tier connected with ILVs. (c) Photograph of wafer, showing wafer-scale fabrication. (d) Die photograph, showing the three main blocks. (e) SEM of delay cell. (f) SEM of a CNFET channel, showing multiple parallel CNTs.

Fig. 17 shows an image of a live demonstration by Wu *et al.* [15] at the IEEE International Solid-State Circuits Conference (ISSCC) on February 13, 2018. Users were able to train the 3D nanosystem (Fig. 18) in any of the 21 European languages and classify sentences based on their text input. As a visualization, for each inference, a gumball was redirected to a container representing the language identified.

IV. Projected Benefits

In order to quantify the benefits of our nanosystem, we compare our implementation (exploiting monolithic 3D of CNFETs and RRAM) with a comparable silicon CMOS-based digital design at a 28-nm node in simulation (modified from an existing design in [18]). The silicon CMOS-based digital design, implemented as a conventional 2-D chip, includes the following [see Fig. 19(a)].

1) Silicon CMOS SRAM lookup for reconfigurable random projection. Each of the 27 unique HVs is pre-determined off-chip and stored in on-chip SRAM. They can be determined by using pseudo-random number generators, such as LFSRs. The same 27 HVs are recalled repeatedly (in arbitrary order); thus, we use a RAM. While many methods can be used to generate unique HVs, the majority of the time and energy is spent reading the on-chip SRAMs (i.e., random projection) since the HVs are only generated on system initialization; thus, we do not include its energy and time in our calculations. The address of the SRAM corresponds to the ASCII representation of the input character, such that the corresponding HV is the output of the SRAM during a read operation.
2) Silicon CMOS digital 7-bit incrementers for each element of the HD addition operator in the HD encoder. A threshold function is used to transform the incremented value into a binary value a digital comparator.

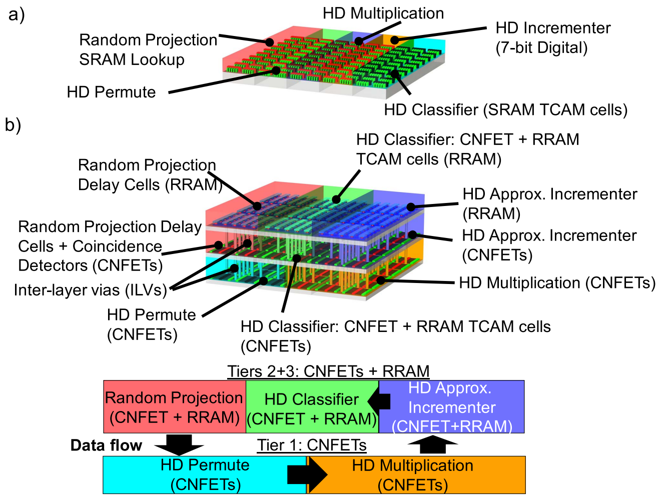

a)
HD Multiplication
Random Projection SRAM Lookup
HD Incrementer (7-bit Digital)
HD Permute
HD Classifier (SRAM TCAM cells)
b)
HD Classifier: CNFET + RRAM TCAM cells (RRAM)
Random Projection Delay Cells (RRAM)
HD Approx. Incrementer (RRAM)
Random Projection Delay Cells + Coincidence Detectors (CNFETs)
HD Approx. Incrementer (CNFETs)
Inter-layer vias (ILVs)
HD Multiplication (CNFETs)
HD Permute (CNFETs)
HD Classifier: CNFET + RRAM TCAM cells (CNFETs)
Tiers 2+3: CNFETs + RRAM
Random Projection (CNFET + RRAM)
HD Classifier (CNFET + RRAM)
HD Approx. Incrementer (CNFET+RRAM)
Data flow
Tier 1: CNFETs
HD Permute (CNFETs)
HD Multiplication (CNFETs)

Fig. 19. (a) Silicon-only HD system. (b) Monolithic 3D HD nanosystem (using CNFETs and RRAM), showing the 3D structure as well as the flow of data (HVs) across the tiers.

3) Silicon CMOS SRAM TCAM cells [50] are used in the HD classifier in the place of CNFET and RRAM TCAM cells.

The monolithic 3D design [see Fig. 19(b)] uses the same architecture as our experimental demonstration using CNFETs and RRAM for all of the individual modules (see Fig. 4) with the following design modifications.

1) HV dimension (D) 8192 to achieve 98% accuracy without using the iterative method described in Section III-D.
2) An additional tier to facilitate the place-and-route (P&R) of a system with larger HV dimension ($D = 8192$). The CNFET-only blocks (i.e., HD permute and HD multiplication) are on the first tier, while the next tiers have the CNFET + RRAM blocks (i.e., Random projection, HD classifier, and HD approx. incrementer), as shown in Fig. 19(b).

In our fabricated chip, we used two tiers since the HV dimension was small ($D = 32$); thus, the interconnects between the circuit blocks were short. However, a larger HV dimension (e.g., $D = 8192$) with a similar floor plan would yield longer interconnect lengths due to limited horizontal routing resources (we used four metal layers per routing direction); thus, an additional tier is utilized to reduce the interconnect distance. Fabricating an additional tier of CNFETs is feasible and has been demonstrated in [11].

A. Methodology

To compare the area, energy, and execution time of the two designs, we perform a full physical design. Fig. 20 explains our physical design flow.

First, we use a variation-aware nanosystem design kit (NDK) [51] to generate a CNFET standard cell library. The NDK uses an experimentally calibrated, SPICE-compatible, virtual-source CNFET model [46] and accounts for CNT-specific variations, such as those in CNT density and CNT diameter [31]. The standard cell library is generated by first extracting the interconnect resistances and parasitic capacitances (RC) from a standard cell library of an existing

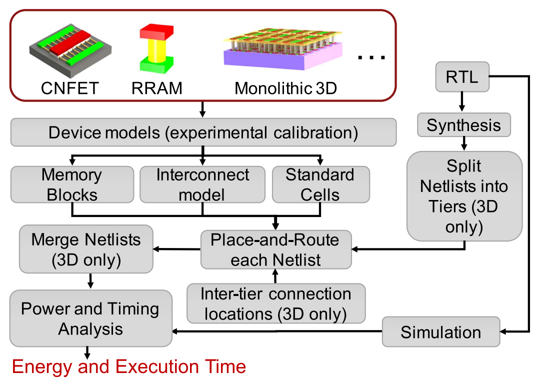

CNFET
RRAM
Monolithic 3D
...
Device models (experimental calibration)
Memory Blocks
Interconnect model
Standard Cells
RTL
Synthesis
Split Netlists into Tiers (3D only)
Merge Netlists (3D only)
Place-and-Route each Netlist
Inter-tier connection locations (3D only)
Power and Timing Analysis
Simulation
Energy and Execution Time

Fig. 20. Simulation methodology for monolithic 3D ICs as well as 2-D ICs using emerging technologies. For the silicon-CMOS implementation, we use device models from and industry silicon foundry rather than those for emerging technologies.

silicon CMOS process design kit (PDK) using Synopsys StarRC; in our case, an industry 28-nm PDK is used. Then, the transistor models in the extracted layouts are replaced by the CNFET model; thus, the CNFETs will have the same width and contacted gate pitch as the original silicon standard cells. Parameters, such as power supply voltage and gate length, are optimized for the lowest energy-delay product (using the methodology in [52]). In our designs, we use $V_{\text{DD}} = 0.5$ V for CNFETs (1 V for silicon CMOS) and a gate length of 27 nm. The standard cells are then re-characterized for leakage power, input pin capacitance, propagation delay tables, output slew rate tables, and internal switch energy and timing constraint tables using the NDK (as in standard Electronic Design Automation (EDA) tools for timing/power characterization, such as the Cadence Liberate).

For the delay cells, approximate incrementers, and HD classifier (all of which use RRAM and CNFETs and span two tiers), layout was performed using the Cadence Virtuoso to determine their area footprints. After extracting the interconnect RCs, energies and delays were determined using SPICE. RRAM characteristics (i.e., set/reset voltage and resistance) found in the literature ($V_{\text{set}} = 0.8$ V, $V_{\text{reset}} = -0.8$ V, HRS 1.7 MΩ, and LRS 30 kΩ) [13], [41], [53] were used. Physical (e.g., pin location), power, and timing information for delay cells and approximate incrementers were added to the standard cell library to be included in the netlists. The HD classifier is represented as a black box in the layout (as well as netlist) to account for its presence in floor planning and P&R and its power and timing were used in P&R.

We perform synthesis and P&R using the Synopsys Design Compiler (DC) and IC Compiler, respectively. To facilitate P&R for a monolithic 3D design (since no commercial EDA tool exists), the gate-level netlist is partitioned into two separate netlists, one representing tier 1 and another representing tiers 2 and 3 [see Fig. 20(b)]. The netlist representing tier 1 consists of CNFET-only circuits, such as the HD multiplication and HD permutation units, while the other netlist, representing tiers 2 and 3, consists of CNFET and RRAM circuits, such as the random projection unit, HD approximate incrementer, and the HD classifier. Tiers 2 and 3 are combined into a single netlist since each custom standard cell or black box already contains both CNFETs and RRAM (described earlier). The inter-tier connections are treated as I/O ports in each netlist, and their location is determined prior to P&R according to a manually determined floor plan. Each netlist is P&R'd separately.

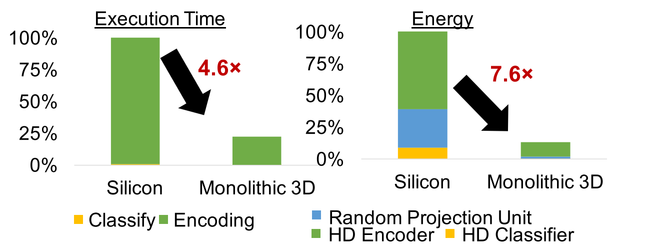


Fig. 21. Simulated benefits for 28-nm implementations of HD computing configured for pairwise language classification, showing 35× system-level energy efficiency (execution time × energy) benefits.

TABLE I
COMPARISON OF PRIOR CNFET AND/OR RRAM DEMONSTRATIONS

	This work	**[25]**	**[11]**	**[56]**	**[25]**	**[33]**
Year	2018	2017	2017	2017	2017	2016
Supply Voltage	3V	0.4V	3V	1.9V	2V	-
CNFET Logic	PMOS	CMOS	PMOS	CMOS	CMOS	-
Operating Frequency	1 KHz	-	2 KHz	282 MHz	100 Hz	-
(S)ystem vs (T)ech Demo	S	T	S	T	S	T
Gate length	1 µm	5 nm	1 µm	100nm	1 µm	-
# of CNFETs	1952	2	$>2^{21}$	12	132	-
# of RRAM	224	-	2^{20}	-	-	4
# of Cascaded CNFET Logic Stages	16	-	>20	6	9	-
Write Condition (RRAM) V_{set}, V_{reset}	2.6V, -2.6V	-	2V, -2V	-	-	1.1V, -1.5V
RRAM properties utilized	I, II, III	-	I	-	-	I, III

I. Digital memory, II. Approximate inference, III. Stochasticity

After P&R, interconnect RCs are extracted for each of the two layouts (containing standard cells and black boxes) along with their final netlists. A wrapper (Verilog) module that instantiates and connects each final netlist is then created (using Synopsys DC), representing the entire 3D nanosystem, and used for power and timing analysis.

B. Simulation Results

In this section, we compare the *system-level energy efficiency*, quantified as energy × execution time (for training and inference of the entire data set), and area footprint of the two systems (see Fig. 19). For this comparison, we maintain the same pairwise classification accuracy (98% between 21 European languages) for the two systems.

When we compare system-level energy efficiency and area footprint, the monolithic 3D design (using CNFETs and RRAM) provides 35× and 3× benefits versus the 2-D silicon CMOS-based digital design (see Fig. 21), respectively. Our design achieves 4.6× faster execution time. A combination of using CNFETs (with higher drive strength than silicon FETs at half the supply voltage [14]) and area reduction (shorter wires) from monolithic 3D integration and smaller circuits (HD approximate incrementer) allows for faster clock rate and, thus, a 4.6× faster execution time. Simultaneously, our design achieves 7.6× lower energy. By utilizing the inherent variations of CNFETs and RRAM, the energy consumption of random projection is reduced by 22× compared to the silicon CMOS SRAM lookup approach. Using CNFET and RRAM TCAM cells instead of SRAM TCAM cells for the HD classifier reduces the classifier energy by 19×. This is primarily due to a reduction of leakage energy (since the HD classifier is idle for most of the execution time) by using non-volatile memory (RRAM) instead of SRAM. The HD encoder energy is reduced by 5.4×, owing to a combination of using CNFETs (instead of silicon FETs) and RRAM approximate incrementers. While the monolithic 3D design only uses two tiers of CNFET logic, a 3× area reduction was achieved, thanks to the area savings of using RRAM versus SRAM and RRAM approximate incrementers versus digital incrementers.

V. Conclusion

This paper illustrates how various properties of heterogeneous nanotechnologies can be effectively exploited and combined to realize brain-inspired computing architectures that tightly integrate computation and storage, enable energy-efficient computation, employ approximation, embrace randomness, and exhibit resilience to errors. We have experimentally demonstrated the followings:

1) pairwise classification of 21 European languages with measured accuracy of up to 98% (comparable to software implementations) on >20 000 sentences (6.4 million characters) per language pair;
2) learning from a few examples (often referred to as one-shot learning) using one text sample (∼100 000 characters) per language;
3) resilient operation (98% accuracy) despite hardware errors (circuit outputs stuck at 0 or 1).

In simulation, we demonstrate significant system-level energy-efficiency and area benefits when brain-inspired computing models are implemented using monolithic 3D integration compared to silicon CMOS implementations.

Future research directions include: 1) demonstration of larger nanosystems at scaled technology nodes [significant progress is being been made along this direction (see Table I)]; 2) expanding the domain of HD nanosystem demonstrations beyond language recognition (e.g., computer vision and healthcare applications); and 3) applying the nanosystems' principles presented in this paper to a broader class of (brain-inspired) computing systems that leverage emerging nanotechnologies (see [34], [54], [55]).

Acknowledgment

The authors would like to thank Dr. E. Beigne of Laboratoire d'électronique des technologies de l'information (CEA-LETI) for fruitful discussions.

References

[1] P. Kanerva, "Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors," *Cognit. Comput.*, vol. 1, no. 2, pp. 139–159, Oct. 2009.
[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.
[3] A. Rahimi *et al.*, "High-dimensional computing as a nanoscalable paradigm," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 9, pp. 2508–2521, Sep. 2017.
[4] H.-S. P. Wong and S. Salahuddin, "Memory leads the way to better computing," *Nature Nanotechnol.*, vol. 10, no. 3, pp. 191–194, 2015.
[5] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017.
[6] A. Alaghi and J. P. Hayes, "Survey of stochastic computing," *ACM Trans. Embedded Comput. Syst.*, vol. 12, no. 2, 2013, Art. no. 92.
[7] S. Wong, A. El, P. Griffin, Y. Nishi, F. Pease, and J. Plummer, "Monolithic 3D integrated circuits," in *Proc. Int. Symp. VLSI Technol., Syst. Appl. (VLSI-TSA)*, Apr. 2007, pp. 1–4.
[8] P. Batude *et al.*, "3DVLSI with CoolCube process: An alternative path to scaling," in *Proc. Symp. VLSI Technol.*, Jun. 2015, pp. T48–T49.
[9] P. Leduc *et al.*, "Enabling technologies for 3D chip stacking," in *Proc. VLSI-TSA*, Apr. 2008, pp. 76–78.
[10] M. M. S. Aly *et al.*, "Energy-efficient abundant-data computing: The N3XT 1,000x," *IEEE Comput.*, vol. 48, no. 12, pp. 24–33, Dec. 2015.
[11] M. M. Shulaker *et al.*, "Three-dimensional integration of nanotechnologies for computing and data storage on a single chip," *Nature*, vol. 547, pp. 74–78, Jul. 2017.
[12] L. Chang, "Short course," in *IEDM Tech. Dig.*, Dec. 2012.
[13] H.-S. P. Wong *et al.*, "Metal–oxide RRAM," *Proc. IEEE*, vol. 100, no. 6, pp. 1951–1970, Jun. 2012.
[14] W. Hwang *et al.*, "3D nanosystems enabled embedded abundant-data computing," in *Proc. CODES+ISSS*, 2017.
[15] T. Wu *et al.*, "Brain-inspired computing exploiting carbon nanotube FETs and resistive RAM: Hyperdimensional computing case study," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2018, pp. 492–494.
[16] G. E. Hinton, "Mapping part-whole hierarchies into connectionist networks," *Artif. Intell.*, vol. 46, nos. 1–2, pp. 47–75, 1990.
[17] T. A. Plate, "Holographic reduced representations," *IEEE Trans. Neural Netw.*, vol. 6, no. 3, pp. 623–641, May 2015.
[18] A. Rahimi, P. Kanerva, and J. M. Rabaey, "A robust and energy-efficient classifier using brain-inspired hyperdimensional computing," in *Proc. ISPLED*, 2016, pp. 64–69.
[19] A. Rahimi, S. Benatti, P. Kanerva, L. Benini, and J. M. Rabaey, "Hyperdimensional biosignal processing: A case study for EMG-based hand gesture recognition," in *Proc. ICRC*, Oct. 2016, pp. 1–8.
[20] A. Rahimi, P. Kanerva, J. del R. Millán, and J. M. Rabaey, "Hyperdimensional computing for noninvasive brain-computer interfaces: Blind and one-shot classification of EEG error-related potentials," in *Proc. BICT*, 2017.
[21] A. Rahimi. *HDC Language Recognition*. [Online]. Available: https://github.com/abbas-rahimi/HDC-Language-Recognition
[22] O. Yilmaz, "Analogy making and logical inference on images using cellular automata based hyperdimensional computing," in *Proc. 2015th Int. Conf. Cogn. Comput., Integrating Neural Symbolic Approaches*, 2015, pp. 1–27.
[23] J. Appenzeller, "Carbon nanotubes for high-performance electronics—Progress and prospect," *Proc. IEEE*, vol. 96, no. 2, pp. 201–211, Feb. 2008.
[24] M. M. Shulaker, G. Pitner, G. Hills, M. Giachino, H.-S. P. Wong, and S. Mitra, "High-performance carbon nanotube field-effect transistors," in *IEDM Tech. Dig.*, Dec. 2014, pp. 33.6.1–33.6.4.
[25] Y. Yang, L. Ding, J. Han, Z. Zhang, and L.-M. Peng, "High-performance complementary transistors and medium-scale integrated circuits based on carbon nanotube thin films," *ACS Nano*, vol. 11, no. 4, pp. 4124–4132, 2017.
[26] C. Qiu, Z. Zhang, M. Xiao, Y. Yang, D. Zhong, and L.-M. Peng, "Scaling carbon nanotube complementary transistors to 5-nm gate lengths," *Science*, vol. 355, no. 6322, pp. 271–276, Jan. 2017.
[27] R. S. Park, G. Hills, J. Sohn, S. Mitra, M. M. Shulaker, and H.-S. P. Wong, "Hysteresis-free carbon nanotube field-effect transistors," *ACS Nano*, vol. 11, no. 5, pp. 4785–4791, 2017.
[28] T. Srimani *et al.*, "Negative capacitance carbon nanotube FETs," *IEEE Electron Device Lett.*, vol. 39, no. 2, pp. 304–307, Feb. 2018.
[29] J. Zhang *et al.*, "Carbon nanotube robust digital VLSI," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 31, no. 4, pp. 453–471, Apr. 2012.
[30] M. M. Shulaker, G. Hills, T. F. Wu, Z. Bao, H.-S. P. Wong, and S. Mitra, "Efficient metallic carbon nanotube removal for highly-scaled technologies," in *IEDM Tech. Dig.*, Dec. 2015, pp. 32.4.1–32.4.4.

[31] G. Hills *et al.*, "Rapid co-optimization of processing and circuit design to overcome carbon nanotube variations," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 34, no. 7, pp. 1082–1095, Jul. 2015.

[32] M. M. Shulaker *et al.*, "Carbon nanotube computer," *Nature*, vol. 501, no. 7468, pp. 526–530, 2013.

[33] H. Li *et al.*, "Four-layer 3D vertical RRAM integrated with FinFET as a versatile computing unit for brain-inspired cognitive information processing," in *Proc. IEEE Symp. VLSI Technol.*, Jun. 2016, pp. 1–2.

[34] H. Li *et al.*, "Hyperdimensional computing with 3D VRRAM in-memory kernels: Device-architecture co-design for energy-efficient, error-resilient language recognition," in *IEDM Tech. Dig.*, Dec. 2016, pp. 16.1.1–16.1.4.

[35] Z. Fang, H. Y. Yu, X. Li, N. Singh, G. Q. Lo, and D. L. Kwong, "$HfO_x/TiO_x/HfO_x/TiO_x$ multilayer-based forming-free RRAM devices with excellent uniformity," *IEEE Electron Device Lett.*, vol. 32, no. 4, pp. 566–568, Apr. 2011.

[36] Y.-B. Kim *et al.*, "Bi-layered RRAM with unlimited endurance and extremely uniform switching," in *Proc. Symp. VLSI Technol. Dig. Tech. Papers*, Jun. 2011, pp. 52–53.

[37] A. Grossi *et al.*, "Fundamental variability limits of filament-based RRAM," in *IEDM Tech. Dig.*, Dec. 2016, pp. 4.7.1–4.7.4.

[38] Z. Chen *et al.*, "Performance improvements by SL-current limiter and novel programming methods on 16 MB RRAM chip," in *Proc. IMW*, May 2017, pp. 1–4.

[39] R. Fackenthal *et al.*, "A 16 Gb ReRAM with 200 MB/s write and 1 GB/s read in 27 nm technology," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2014, pp. 338–339.

[40] S. S. Sheu *et al.*, "A 4 Mb embedded SLC resistive-RAM macro with 7.2 ns read-write random-access time and 160 ns MLC-access capability," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2011, pp. 200–202.

[41] T. Cabout *et al.*, "Role of Ti and Pt electrodes on resistance switching variability of HfO_2-based resistive random access memory," *Thin Solid Films*, vol. 533, pp. 19–23, Apr. 2013.

[42] C. W. Liang *et al.*, "A 28 nm poly/SiON CMOS technology for low-power SoC applications," in *Symp. VLSI Technol. Dig. Tech. Papers*, Jun. 2011, pp. 38–39.

[43] P. Batude, "Advances, challenges and opportunities in 3D CMOS sequential integration," in *IEDM Tech. Dig.*, Dec. 2011, pp. 7.3.1–7.3.4.

[44] N. Patil *et al.*, "Wafer-scale growth and transfer of aligned single-walled carbon nanotubes," *IEEE Trans. Nanotechnol.*, vol. 8, no. 4, pp. 498–504, Jul. 2009.

[45] M. Shulaker *et al.*, "Monolithic 3D integration of logic and memory: Carbon nanotube FETs, resistive RAM, and silicon FETs," in *IEDM Tech. Dig.*, Dec. 2014, pp. 27.4.1–27.4.4.

[46] C.-S. Lee and H.-S. P. Wong, "Stanford virtual-source carbon nanotube field-effect transistors model," nanoHUB, 2015, doi: 10.4231/D3BK16Q68.

[47] P. Koehn. (2005). *Europarl: A Parallel Corpus for Statistical Machine Translation*. [Online]. Available: http://www.statmt.org/europarl/

[48] L. Zheng, S. Shin, S. Lloyd, M. Gokhale, K. Kim, and S.-M. Kang, "RRAM-based TCAMs for pattern search," in *Proc. ISCAS*, May 2016, pp. 1382–1385.

[49] U. Quasthoff, M. Richter, and C. Biemann, "Corpus portal for search in monolingual corpora," in *Proc. 5th Int. Conf. Lang. Resour. Eval.*, 2006, pp. 1799–1802.

[50] K. Nii *et al.*, "A 28 nm 400 MHz 4-parallel 1.6 Gsearch/s 80 Mb ternary CAM," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2014, pp. 240–241.

[51] G. Hills. *Variation-Aware Nanosystem Design Kit*. Accessed: Oct. 17, 2017. [Online]. Available: https://nanohub.org/resources/22582

[52] G. Hills, "Energy-efficient digital VLSI using carbon nanotube field-effect transistors," Ph.D. dissertation, Dept. Elect. Eng., Stanford Univ., Stanford, CA, USA, 2017.

[53] C. Y. Chen *et al.*, "Understanding the impact of programming pulses and electrode materials on the endurance properties of scaled Ta_2O_5 RRAM cells," in *IEDM Tech. Dig.*, Dec. 2014, pp. 14.2.1–14.2.4.

[54] W. Hwang, W. Wan, S. Mitra, and H.-S. P. Wong, "Coming up N3XT, after 2D scaling of Si CMOS," in *Proc. ISCAS*, May 2018, pp. 1–5.

[55] S. Yu, B. Gao, Z. Fang, H. Yu, J. Kang, and H.-S. P. Wong, "A neuromorphic visual system using RRAM synaptic devices with Sub-pJ energy and tolerance to variability: Experimental characterization and large-scale modeling," in *IEDM Tech. Dig.*, Dec. 2012, pp. 10.4.1–10.4.4.

[56] S.-J. Han *et al.*, "High-speed logic integrated circuits with solution-processed self-assembled carbon nanotubes," *Nature Nanotechnol.*, vol. 12, pp. 861–865, Jul. 2017.

Tony F. Wu received the B.S. degree in electrical engineering from the California Institute of Technology, Pasadena, CA, USA, in 2011, and the M.S. degree in electrical engineering from Stanford University, Stanford, CA, USA, in 2013, where he is currently pursuing the Ph.D. degree.

His current research interests include design and fabrication of monolithic 3D integrated systems using emerging technologies.

Mr. Wu was a recipient of the 2018 Electronics Materials Symposium Ross N. Tucker Award.

Haitong Li received the B.S. degree in microelectronics from Peking University, Beijing, China, in 2015, and the M.S. degree in electrical engineering from Stanford University, Stanford, CA, USA, in 2017, where he is currently pursuing the Ph.D. degree.

His current research interests include brain-inspired in-memory computing with emerging devices and architectures.

Mr. Li was a recipient of the 2016 IEEE EDS Masters Student Fellowship.

Ping-Chen Huang received the B.S. and M.S. degrees in electrical and communication engineering from National Taiwan University, Taipei, Taiwan, and the Ph.D. degree in electrical engineering from the University of California at Berkeley, Berkeley, CA, USA, in 2015.

She was with Academia Sinica Institute of Astronomy and Astrophysics (ASIAA), Taipei, where she was involved in millimeter-wave circuit designs for the telescope front end in Atacama Large Millimeter/submillimeter Array (ALMA) Project. She is currently with Qualcomm Atheros, Inc., San Jose, CA, USA. Her current research interests include low-power analog-/mixed-signal/RF circuits, sensor interfaces, and developing neuro-inspired sensor signal processing architectures for efficient implementations in current and future nanotechnologies.

Abbas Rahimi received the B.S. degree in computer engineering from the University of Tehran, Tehran, Iran, in 2010, and the M.S. and Ph.D. degrees in computer science and engineering from the University of California at San Diego, San Diego, CA, USA, in 2015.

He held a post-doctoral research position at the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, CA, USA, for two years. In 2017, he joined the Department of Information Technology and Electrical Engineering, ETH Zürich, Zürich, Switzerland. He is currently with the Berkeley Wireless Research Center, Berkeley. His current research interests include brain-inspired computing, approximate computing, massively parallel integrated architectures, and embedded systems and software with an emphasis on improving energy efficiency and robustness.

Dr. Rahimi received the ETH Zurich Post-Doctoral Fellowship. His doctoral dissertation has received the 2015 Outstanding Dissertation Award in the area of New Directions in Embedded System Design and Embedded Software from the European Design and Automation Association. He has also received the Best Paper Candidate at Design Automation Conference (DAC) 2013 and the Best Paper at Conference on Bio-inspired Information and Communications Technologies (BICT) 2017.

Gage Hills received the Ph.D. degree from Stanford University, Stanford, CA, USA, in 2018, advised by Prof. S. Mitra and co-advised by Prof. H.-S. P. Wong.

He is currently a Post-Doctoral Researcher with the Massachusetts Institute of Technology, Cambridge, MA, USA. His current research interests include development of very large-scale integrated circuits using nanotechnologies, such as carbon nanotube field-effect transistors.

Bryce Hodson received the B.Eng. degree in electrical and computer engineering from the University of Arizona, Tuscon, AZ, USA, in 2017. He is currently pursuing the M.S. degree with the Department of Electrical Engineering, Stanford University, Stanford, CA, USA.

In 2017, he was a Technical Intern at Intel Corporation. His current research interests include design and fabrication for reliability.

William Hwang received the B.S. degree in electrical engineering, the B.S. degree in Material Science Engineering (MSE), and the M.S. degree in MSE from the University of Washington, Seattle, WA, USA, in 2015 and 2016, and the M.S. degree in electrical engineering from Stanford University, Stanford, CA, USA, in 2018, where he is currently pursuing the Ph.D. degree.

His current research interests include energy-efficient computing systems, enabled by monolithic 3D integration of emerging technologies.

Jan M. Rabaey (F'95) holds the Donald O. Pederson Distinguished Professorship with the University of California at Berkeley, Berkeley, CA, USA, where he is currently the Electrical Engineering Division Chair. He is also the Founding Director of the Berkeley Wireless Research Center, Berkeley, and the Berkeley Ubiquitous SwarmLab, University of California at Berkeley. He has made high-impact contributions to a number of fields, including advanced wireless systems, low-power integrated circuits, sensor networks, and ubiquitous computing. His current interests include the conception of the next-generation integrated wireless systems over a broad range of applications and exploring the interaction between the cyber and the biological world.

He has been involved in a broad variety of start-up ventures.

Dr. Rabaey is a member of the Royal Flemish Academy of Sciences and Arts of Belgium. He was a recipient of major awards, including the IEEE Mac Van Valkenburg Award, the European Design Automation Association Lifetime Achievement Award, and the Semiconductor Industry Association University Researcher Award.

H.-S. Philip Wong (F'01) was with the IBM T.J. Watson Research Center, Yorktown Heights, NY, USA, from 1988 to 2004. In 2004, he joined Stanford University, Stanford, CA, USA, as a Professor of electrical engineering, where he is currently the Willard R. and Inez Kerr Bell Professor with the School of Engineering.

He held various positions from a Research Staff Member to the Senior Manager at IBM, while he was a Senior Manager, he had the responsibility of shaping and executing IBM's strategy on nanoscale science and technology as well as exploratory silicon devices and semiconductor technology. During his time at IBM, he managed pathfinding research on high-k/metal gate, strained silicon, alternative channel materials, such as Ge and III–V, multigate FinFET, ultrathin SOI—many of these have now become product technology at various companies. His research aims at translating discoveries in science into practical technologies. His works have contributed to advancements in nanoscale science and technology, semiconductor technology, solid-state devices, and electronic imaging. His current research interests include a broad range of topics, including carbon electronics, 2-D layered materials, wireless implantable biosensors, directed self-assembly, device modeling, brain-inspired computing, nonvolatile memory, and monolithic 3D integration.

Dr. Wong served as the Sub-Committee Chair of the IEEE International Solid-State Circuits Conference (ISSCC) from 2003 to 2004 and the General Chair of the International Electron Devices Meeting (IEDM) in 2007. He is currently the Chair of the IEEE Executive Committee of the Symposia of VLSI Technology and Circuits. He served as the Editor-in-Chief of the IEEE TRANSACTIONS ON NANOTECHNOLOGY from 2005 to 2006. He is the Faculty Director of the Stanford Non-Volatile Memory Technology Research Initiative (NMTRI) and the Founding Faculty Co-Director of the Stanford SystemX Alliance—an industrial affiliate program focused on building systems.

Max M. Shulaker received the B.S., M.S., and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, USA.

In 2016, he joined the Electrical Engineering and Computer Science (EECS) Department, Massachusetts Institute of Technology, Cambridge, MA, USA, as an Assistant Professor. During his Ph.D. degree, his research on carbon nanotube-based transistors and circuits resulted in the first digital systems built entirely using carbon nanotube FETs (including the first carbon nanotube microprocessor), the first monolithic 3D integrated circuits combining arbitrary vertical stacking of logic and memory, and the highest performance and highly scaled carbon nanotube transistors to-date.

Subhasish Mitra (F'13) was a Principal Engineer at Intel Corporation. He is currently a Professor of electrical engineering and computer science with Stanford University, Stanford, CA, USA, where he directs the Stanford Robust Systems Group and co-leads the computation focus area of the Stanford SystemX Alliance. He is also a Faculty Member of the Stanford Neurosciences Institute, Stanford. He holds the Carnot Chair of Excellence in Nanosystems at Laboratoire d'électronique des technologies de l'information (CEA-LETI), Grenoble, France. He and his students published several award-winning papers at major venues, including the ACM/IEEE Design Automation Conference (DAC), the IEEE International Solid-State Circuits Conference, the IEEE International Test Conference, the IEEE Transactions on CAD, and the IEEE VLSI Test Symposium, and the Symposium on VLSI Technology. His current research interests include robust computing, nanosystems, VLSI design, validation, test and Electronic Design Automation (EDA), and neurosciences. He, jointly with his students and collaborators, demonstrated the first carbon nanotube computer and the first 3D nanosystem with computation immersed in data storage. These demonstrations received wide-spread recognitions (cover of NATURE, Research Highlight to the United States Congress by the National Science Foundation (NSF), highlight as Important, Scientific Breakthrough by the BBC, Economist, EE Times, the IEEE Spectrum, MIT Technology Review, National Public Radio, New York Times, Scientific American, Time, Wall Street Journal, Washington Post, and numerous others worldwide). His earlier work on X-Compact test compression has been a key to the cost-effective manufacturing and high-quality testing of almost all electronic systems. X-Compact and its derivatives have been implemented in widely used commercial EDA tools.

Dr. Mitra served on the Defense Advanced Research Projects Agency's (DARPA) Information Science and Technology Board as an Invited Member. He is a fellow of the Association for Computing Machinery (ACM). He received the ACM Special Interest Group on Design Automation (SIGDA)/IEEE Council on EDA (CEDA) A. Richard Newton Technical Impact Award in EDA (a test of time honor), the Semiconductor Research Corporation's Technical Excellence Award, the Intel Achievement Award (Intel's highest corporate honor), and the Presidential Early Career Award for Scientists and Engineers from the White House (the highest United States honor for early career outstanding scientists and engineers). At Stanford, he has been honored several times by graduating seniors for being important to them during their time at Stanford.