A 5 µw Standard Cell Memory-Based Configurable Hyperdimensional Computing Accelerator for Always-on Smart Sensing

A 5 µW Standard Cell Memory-based Configurable Hyperdimensional Computing Accelerator for Always-on Smart Sensing

Manuel Eggimann, *Graduate Student Member, IEEE,* Abbas Rahimi, Luca Benini, *Fellow, IEEE*

***Abstract*—Hyperdimensional computing (HDC) is a brain-inspired computing paradigm-based on high-dimensional holistic representations of vectors. It recently gained attention for embedded smart sensing due to its inherent error-resiliency and suitability to highly parallel hardware implementations. In this work, we propose a programmable all-digital CMOS implementation of a fully autonomous HDC accelerator for always-on classification in energy-constrained sensor nodes. By using energy-efficient standard cell memory (SCM), the design is easily cross-technology mappable. It achieves extremely low power, 5 µW in typical applications, and an energy efficiency improvement over the state-of-the-art (SoA) digital architectures of up to 3× in post-layout simulations for always-on wearable tasks such as Electromyography (EMG) gesture recognition. As part of the accelerator's architecture, we introduce novel hardware-friendly embodiments of common HDC-algorithmic primitives, which results in 3.3× technology scaled area reduction over the SoA, achieving the same accuracy levels in all examined targets. The proposed architecture also has a fully configurable datapath using microcode optimized for HDC stored on an integrated SCM-based configuration memory, making the design "general-purpose" in terms of HDC algorithm flexibility. This flexibility allows usage of the accelerator across novel HDC tasks, for instance, a newly designed HDC-algorithm for the task of ball bearing fault detection.**

***Index Terms*—Hyperdimensional Computing, Always-on, Edge Computing, Machine Learning, Hardware Accelerator, VLSI, Standard Cell Memory**

I. Introduction

ENERGY boundedness is the key design metric and constraint in the development of internet-of-things (IoT) devices [1, 2, 3]. With more and more sensor modalities integrated into IoT end nodes, the amount of data to process, and the complexity of the processing pipeline increases. Aiming for uninterrupted operation for years or even indefinitely within the tight power envelope of small batteries or environmental energy harvesting urges to drastically reduce the average power consumption of the sensor nodes themselves.

Observing that the majority of power consumption in today's wireless sensor devices is spent in data transmission [4] promotes moving data processing closer to the sensor. Instead of raw data transmission and centralized processing in the cloud, the data is processed continuously on these so-called *smart sensor* devices [5]. Only the analyzed portion of the information is transmitted (e.g., transmission of a single `imminent machine failure` message instead of the raw vibration and temperature data). This may not be easily achieved by application-specific integrated circuit (ASIC) designs because *general-purpose* always-on smart sensing systems operate in the µW range and also demand a programmable fabric. Therefore, the next evolution step towards fully self-sustainable always-on smart sensors requires the exploration of new avenues of hardware-software co-design and outside the realm of traditional von Neumann-based computing [6, 7].

An energy proportional sensor data processing scheme, where a wake-up circuit (WuC) detects patterns of interest and aggressively duty cycles other circuitry, is a viable solution to drastically reduce average power consumption [8, 9]. While there are numerous WuCs, e.g., for biosignal anomaly detection, sound/keyword spotting, incoming radio transmissions in the µW range, all of these solutions are highly application-specific. Considering the cost of custom silicon development and the rapidly widening range of application targets, there is a need for configurable and application-agnostic WuCs with more flexible pattern extraction capabilities than the simple threshold-based solutions, which can suffer from high false-positive rate and thus energy losses of unnecessary wake-ups.

Hyperdimensional computing (HDC) is a brain-inspired computing paradigm that excels in the learning curve, computational complexity of the training, and simplicity of operations for hardware. This makes it a perfect fit for energy-constrained inference applications, and, more specifically, for general-purpose always-on sensing [10, 11, 12]. It can also be combined with other low-power computing paradigms such as spiking neural networks operating at µW [13, 14, 15, 16]; for instance, HDC can efficiently compress and process spikes generated from event-based dynamic vision sensors [17].

In this work, we present the following contributions:

- We propose a novel flexible and highly energy-efficient all-digital HDC architecture for always-on smart sensing applications achieving up to 3× higher energy efficiency (191 $\mathrm{nJ/inference}$) over the SoA.
- As part of the architecture, we introduce novel hardware-friendly embodiments of common HDC operators resulting in 3.3× technology scaled area reduction.
- We provide an evaluation of latch-based associative memories at sub nominal supply voltage conditions in post-layout simulation, indicating the potential of at least 3.5× energy efficiency improvement compared to an SRAM based digital solution.
- We provide practical application case studies of our approach, including the first investigation (to the best of

our knowledge) on the feasibility of HDC for the task of ball bearing fault detection.
- Finally, using an all-digital approach enables us to publicly release our architecture under the permissive Solderpad open-source license[1].

The remainder of this paper is structured as follows. In Section II we elaborate on previous work in the domain of HDC accelerators and always-on classification circuitry and highlight the distinctive novel characteristics of the proposed approach. Section III analyzes in detail the modules of the proposed architecture. We continue with post-layout analysis on power and area of the design in different target technologies and different design parameter combinations in Section IV before we conduct an energy efficiency and accuracy analysis for several always-on cognitive sensing scenarios in Section V. Finally, we conclude in Section VI.

II. Related Work

Tackling the power-consumption challenge of always-on sensing in a hierarchical manner using WuCs to apply aggressive duty cycling on more involved data processing modules is not a new idea. In the recent past, there have been several publications on low-power always-on wake-up circuitry in various domains. Table I gives an exemplary overview of current general-purpose wake-up circuitry research using selected publications in the recent past.

Keyword spotting and voice activity detection (VAD) is a very actively researched target for always-on sensing; Giraldo et al. present a low power WuC for speech detection, speaker identification, and keyword spotting with integrated preprocessing blocks for MFCC generation and LSTM accelerator for classification [20]. Shan et al. proposed another implementation in the same application domain with state-of-the-art energy efficiency on the task of two-word keyword spotting using binarized depth-wise separable CNN's operating at near-threshold [21]. At the lower end of the power consumption spectrum Cho et al. present a 142 nW VAD circuitry with integrated analog-frontend that combines a configurable always-on time-interleaved mixer architecture with a heavily duty-cycled neural-network processor [18]. Although their analog input stage is highly specific to VAD only, the integrated 14 µW digital neural network processor could potentially be repurposed for other applications and we thus assume it to be general-purpose.

Monitoring life signals is another very active field; in the context of cardiac arrhythmia detection, Zhao et al. combine a level-crossing ADC with asynchronous QRS-complex detection circuitry with an artificial neural network accelerator to benefit from the energy advantage of non-Nyquist sampling [22]. Although these NN-based solutions achieve high accuracy at outstanding energy efficiency in their particular application domain, they are often hardwired for the respective task and do not support online training.

To the authors' knowledge, the only low power WuC with slightly more sophisticated pattern matching capabilities was introduced by Rovere et al.. Instead of analyzing the delta-encoded signal from the LC-ADC with hardwired detectors, they continuously match the input signal against a sequence of upper and lower amplitude thresholds with up to 16 threshold segments. This scheme equates to matching the input signal's approximate amplitude slope against a configurable pre-trained prototypical signal slope of interest [19]. Their approach proved successful for pathological ECG classification and binary hand gesture recognition (finger-snap or hand clapping). Still, detecting more complex patterns in the spatial or time dimension remains outside their proposed architecture's scope.

Hyperdimensional computing (HDC) is an energy-efficient and flexible computing paradigm for near-sensor classification that gracefully degrades in the presence of bit errors, and noise [23, 24, 25]. Various works showcased HDC's few-shot learning properties and energy efficiency in multiple domains like biosignal processing [26], language recognition [27], DNA sequencing [28], or vehicle type classification [29].

In emerging hardware implementations, the HDC's inherent error-resiliency is leveraged for novel non-volatile memory (NVM) based in-memory computing architectures [7, 30, 31]. Targeting FPGAs, efficient mappings of binary and bipolar HDC operations are proposed [32, 33, 34]. However, the only complete digital CMOS-based HDC accelerator was recently introduced by Datta et al.. They propose a data processing unit (DPU) based encoder design that interconnects with a ROM-based item memory, and a fully parallel associative memory [35]. While their implementation indeed excels in throughput, its' configurability as well as area- and energy efficiency are limited; their encoder architecture is restricted to what they call *generic* multi-stage HDC algorithms with a hardwired encoder depth in feedforward configuration imposing hard limits on the supported encoding schemes. From an energy efficiency and area standpoint, their design suffers a lot from using a large read-only-memory (ROM) for item memory (IM) and pipeline registers in the very wide datapath of every encoding layer.

Our proposed architecture targets the sub 25µW power envelope (resulting in a lifetime of about four years from a small lithium-thionyl chloride coin cell battery). The always-on smart sensing circuitry leverages the flexibility of HDC to perform energy-efficient end-to-end classification on a diverse

[1]Available under https://github.com/pulp-platform/hypnos

TABLE I
Comparison of state-of-the-art general-purpose WuCs with our proposed HDC-based WuC. Area is reported in 65nm and 22nm technology while power is reported in 22nm for a compute-intensive language classification algorithm and a typical always-on classification algorithm for EMG data. For Cho et al. only the neural network processor without application specific VAD circuitry is considered.

	Cho et al. [18]*	Rovere et al. [19]	**This Work**
Applications	General-Purpose	General-Purpose	General-Purpose
Technology	180nm	130nm	65nm / 22nm
Cross Tech.	High	Low	High
Power Envelope	~14 µW	~2.2 µW	max. ~25 µW, typ. ~5 µW
Classification Scheme	NN	Threshold Sequence	HDC
Configurability	High	Medium	High
Area	15.6 mm^2	0.054 mm^2	1.43 mm^2, 0.29 mm^2

set of input signals. We achieve higher configurability, a reduction of 3.1× in area and up to 3.3× improvement in energy efficiency than the current SoA in HDC acceleration and present a first-in-class flexible and technology agnostic digital CMOS architecture for near sensor smart sensing wake-up circuitry.

III. Programmable HDC-Accelerator Architecture

A. Hyperdimensional Computing

Hyperdimensional Computing (HDC) or vector symbolic architectures (VSAs) in general, is a brain-inspired compute paradigm that recently is gaining attention [23]. Its core idea is to map low-dimensional input data, i.e., raw sensor data or features thereof, to vectors of very high dimensionality (cardinality in the order of thousands). The procedure of input to HD space mapping is commonly called *hyperdimensional encoding*. HDC defines simple operations on vectors to aggregate their information into a single vector. *Binding* a vector V_a to another vector V_b creates a vector that is dissimilar to both inputs and thus may be used to represents the mapping $V_a : V_b$. *Bundling* several input vectors yields a vector most similar to all of its inputs, therefore representing the set of its input vector. The unary *Permutation* operation maps a single vector deterministically to an entirely unrelated subspace. Combining these three operations on multiple channels or a time-sequence of mapped input vectors (using a so-called *item memory*) captures high-level signal characteristics of the underlying data in an error-resilient and flexible manner. This process is commonly called HDC encoding [36].

The inverse mapping of HD Vectors to the low dimensional output space, i.e., the index of a classification result, is enabled by the *Associative Lookup* operation. This operation finds the most similar vector to the input within a set of stored HD vectors.

For both, training and classification, the same encoder is used to map input data to a high-dimensional vector. Then the associative memory is used to store these vectors during training, or find the closest one during classification. This will further enable online learning that has been initially explored in [37].

There are various embodiment options for VSAs, differing in the concrete representation of the individual dimensions and actual implementations of *Binding*, *Bundling* and the similarity metric. In this work, we concentrate on the so-called binary spatter code (BSC), a digital CMOS-friendly VSA that uses a single bit per dimension. BSC uses XOR for the binding and majority vote for the bundling operation with Hamming distance as the implied similarity metric for associative lookups.

B. Overview

Figure 1 illustrates the three major components of the accelerator, which we describe in detail in the following subsections; the **associative memory** (AM) stores the prototype vectors and performs the associative lookup operations,

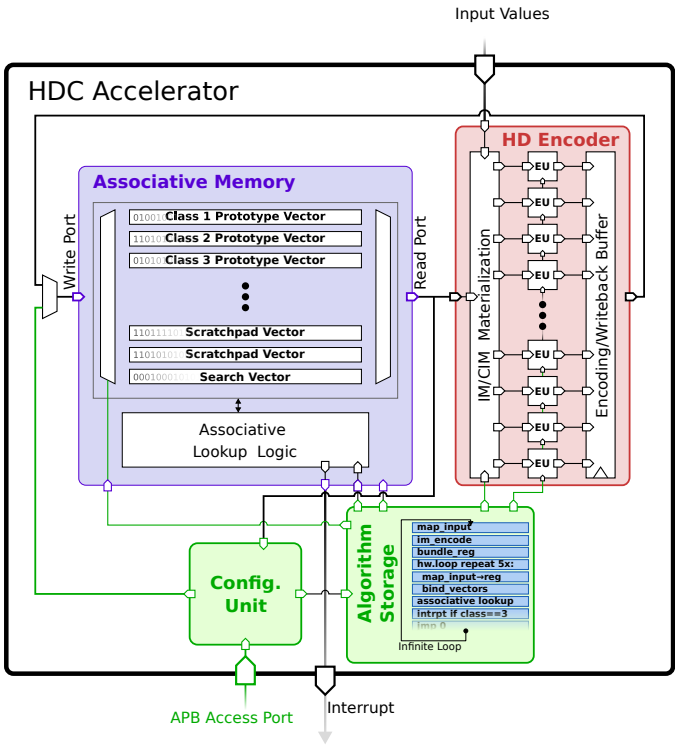


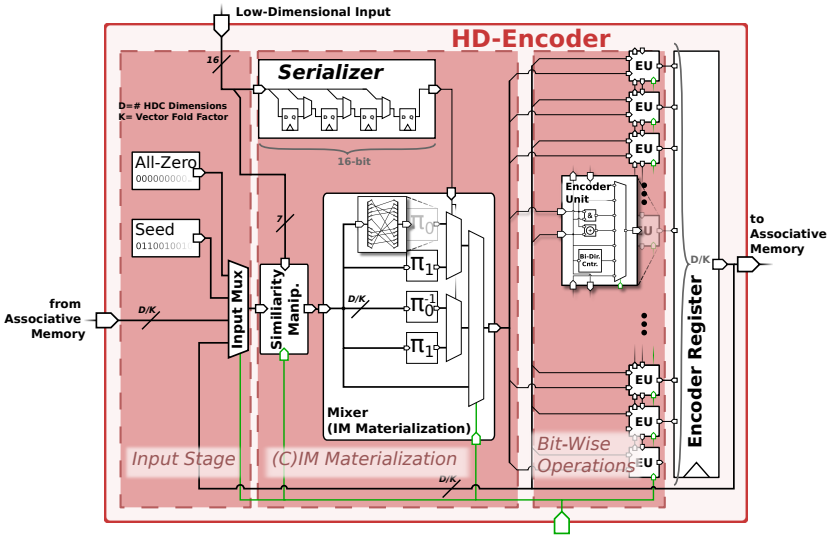
Fig. 1. High-level structure of the proposed HDC accelerator. The associative memory (blue) is responsible for storage and associative lookup of prototype vectors and serves as a scratchpad memory for the HD Encoder (red). Encoder and associative memory are orchestrated by user-programmable algorithm storage (green).

the final step of most HDC algorithms. The *hyperdimensional encoder* (HD-Encoder) is responsible for mapping low-dimensional input values to HD-vectors. It operates on HD vectors from the AM or its own output in an iterative manner. The AM and HD-encoder are managed by a small controller circuit that sequentially consumes a stream of compact microcode instructions and accordingly reconfigures the datapath. A tiny user-programmable configuration memory supplies this microcode stream.

C. HD-Encoder

The first step of every HDC classification algorithm is mapping a dense input space to a high-dimensional holistic representation. Most algorithms encode the input data into a single high-dimensional search vector. The search vector is then compared with prototype vectors stored in the AM that represent the different classes. The differences between the various HDC algorithms mainly lay in the particular encoding algorithms. They are crafted to capture relevant characteristics from the raw data, e.g., amplitude distribution, spatial or temporal features, and are highly application dependent.

Figure 2a illustrates our proposed encoder architecture. It consists of three main components connected in a combinational pipeline. The input stage of the encoder multiplexes between 4 different input sources; the all-zeros vectors, a hardwired random seed vector, a vector addressed from AM,



(a)

AM	Associative Memory
EU	Encoder Unit
HDC	Hyper-Dimensional Computing
BSC	Binary Spatter Code
(C)IM	(Continuous) Item Memory Mapping
SCM	Standard Cell Memory

(b)

Fig. 2. (a) Architecture of the HDC Encoder responsible for (Continouus) Item Memory materialization and search vector encoding. The width of the datapath is a function of the HDC dimensionality (D) and the design parameter K (discussed in Section III-C3). (b) Table of accelerator-related acronyms.

or the HD-encoder's own output. The IM materialization stage maps input data to item vectors using either quasi-orthogonal vectors (IM) or continuous item mapping (CIM). The encoder's bitwise encoder units perform binary or unary operations on the individual bits of the vectors.

There are no pipeline registers in the very wide datapath between the encoder stages. Although this design choice reduces throughput, it increases the energy efficiency of our architecture.

1) Encoder Units: The Encoder Unit processes one dimension of the input vector. Besides the combinational logic for the binary and unary bitwise operations, each unit contains an output Flip-Flop that stores the result after each encoding cycle.

Additionally, there is one saturating bidirectional 5-bit counter per unit to perform the bundling operations. Analyses in [32] showed that for dimensions up to 10000, a 5-bit saturating counter implementation still achieves the same bundling capacity as a full precision model.

2) Mixing Stage: The Mixer submodule visualized in figure 2a generates quasi-orthogonal pseudo-random HD vectors. The rematerialization, i.e., on-the-fly regeneration, of such vectors is an area-efficient alternative to explicit storage of large numbers of item vectors required for input to HD space mapping.

The mixer stage feeds the input vector selected by the encoder input stage through one of two hardwired random permutations π_0 and π_1. The encoder maps a given low-dimensional binary input datum w from the input domain $\mathbb{D}$ to the pseudo-random HD-vector V_w by iteratively applying one of the two permutations to a hardwired seed vector S:

$$V_w = \prod_{k=0}^{n} \pi_i S, \text{ for } i = \begin{cases} 0 & \text{, if } w_k = 0 \\ 1 & \text{, if } w_k = 1 \end{cases} \quad (1)$$

where w_k denotes the k$^{\text{th}}$ bit position in the input word w's binary representation and $n = \log_2 |\mathbb{D}|$. The resulting HD-vectors V_w are all quasi-orthogonal, given that π_0 and π_1 do not commute.

For algorithms that require random access to the item memory, the above scheme rematerializes the item vector with time complexity $\mathcal{O}(\log_2 |\mathbb{D}|)$.

Our proposed IM-mapping approach is more area-efficient than storing random vectors in a large ROM and scales well to large input domains whose cardinality is unknown in advance. From a hardware perspective, the mixer stage translates to N 4-input and N 2-input multiplexers, where N denotes the datapath width and some moderate wiring overhead caused by the random permutations.

3) Vector Folding: With default parameters, the proposed HD-Accelerator contains a datapath width equal to the size of a whole HD-Vector. However, as will be analyzed in more detail in Chapter IV, going for a more parallel architecture does not always yield the most energy-efficient design for a given target technology. Thus, in addition to other design parameters, the RTL exposes the *Vector Fold* parameter; it allows to tune the design for the optimal amount of parallelism to improve energy efficiency. Increasing the value of *the vector fold* splits a single D-dimensional vector into K smaller subparts of equal size. The datapath of the accelerator shrinks accordingly and only processes one subpart at a time. While the throughput of the accelerator at constant frequency decreases by K, the area of the HD-Encoder, dominated by the saturating counters, reduces similarly by a factor of K.

4) Similarity Manipulator: The Similarity Manipulator stage transforms the mixing stage's output vector by flipping a configurable number of its bits. This operation is a fundamental building block of various high-level operations like *binarized B2B bundling* [32], CIM mapping [24] and exponential forgetting. Figure 3 shows its internal structure; the 7-bit input word w is first mapped to a 128-bit unary representation w_{unary}. This unary representation is spread to the target HD-vector dimensionality D/K by repeating each

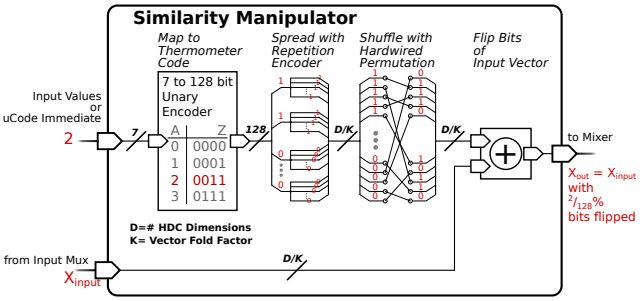


Fig. 3. Structure of the Similarity Manipulator stage

bit of w_{unary} $\frac{D}{K \times 128}$ times. The resulting vector passes through a hard-wired random permutation to distribute the 'ones' over all the vector dimensions. The result is XOR-ed with the input vector. A limitation of the proposed solution is that a uniform distribution of the input words does not yield equal distribution of the probabilities for a bit to be set across the HD-Vector's input dimensions. A multi-cycle approach can be used for operations where equal bit-flipping probability is a hard requirement; first, a bitmask with the desired bit-density is generated by passing the all-zero vector through the manipulator stage with the input word w. This mask is subsequently mixed in the Mixing stage using the same input word w to randomize the position of the 'ones' in the bitmask. The resulting bitmask is ultimately XOR-ed with the input HD-vector within the encoder units.

D. Fully-Synthesizable Associative Memory

For a given search vector, the AM looks up the most similar vector currently stored within the memory. However, the obvious approach to combine traditional SRAMs to store the HD-vectors with digital logic yields suboptimal results. Although SRAMs are the go-to solution for fast and area-efficient volatile on-chip memory, conventional SRAM macro generators are not optimized for the extremely wide memory aspect ratios needed for parallel access to HD-vectors. Also they are less energy-efficient under low V_{DD} conditions for low bandwidth applications [38, 39]. The nature of hyperdimensional computing with lots of simple, component-wise operations demands a non-von Neumann scheme of computation with computational logic intermixed with memory cells.

1) Using Latch Cells as Memory Primitives: Figure 4 shows the structure of the AM in our design; latch cells are used as primitive memory elements instead of flip-flops due to their lower area (-10%) and energy (-20%) footprint [39]. Each row of the memory consists of D/K latch cells and a single glitch-free clock gate. These row clock gates are activated by the one-hot encoded write address. A two-port design allows fetching a new HD-Vector from AM into the HD-encoder while simultaneously writing back the previous result without any stalls or energy costly pipeline registers in the wide datapath.

In most HDC-based classification schemes, the AM only keeps hold of the prototype vectors representing the individual classes. The proposed architecture differs in that regard by

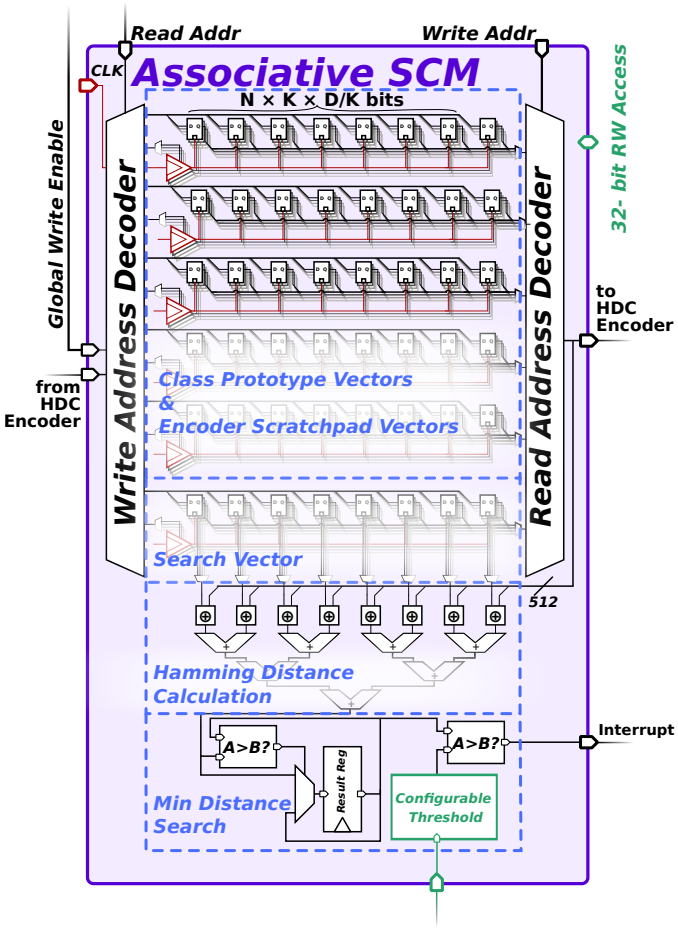


Fig. 4. Architecture of the latch-cell based all-digital AM. Vectors can be read and written simultaneously in subrows of length D/K. The last vector within the memory acts as the search vector for the associative lookup logic. The D/K-bit adder tree for the popcount operation is shared by all memory rows. The distance of the most similar entry is compared with configurable threshold and conditionally raises an interrupt line to an external peripheral (e.g. power management unit in an SoC)

using rows of the AM to store the iterative encoding process's intermediate results. The AM thus serves the double purpose of a register file for entire HD-vectors (or vector subparts in case *vector fold* $K > 1$).

Although latch cells drastically reduce the impact on area footprint compared to flip-flops, their usage can complicate static timing analysis (STA). Due to their transparent nature during write access, one must take care not to introduce combinational loops. While Teman et al. suggest decoupling the memory by using flip-flops at the IO boundary of the memory [38], we repurposed the output register in the encoder stage to break combinational loops. This approach, coupled with multi-cycle path constraints for STA [38], allows treating the AM like a regular flip-flop-based synchronous design during synthesis.

2) Associative Lookup Logic: As can be seen in figure 4, the HD-vector slot acts as the search vector in the proposed architecture. While we could directly use the write input into the memory as the search word, this would prevent the vector folding feature's usage since our write port would not have

a full vector width anymore. The lookup logic iterates over each memory row, calculating the Hamming distance between one subpart of the search vector and a subpart of one of the stored HD-vectors at a time. The control logic accumulates the Hamming distance between the subparts and iteratively determines the most similar entry's index and distance.

E. An ISA for HD-Computing

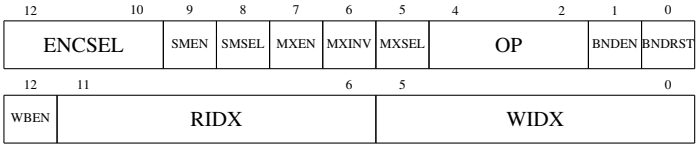
Previously proposed HDC accelerator designs hardwired large portions of their datapath to execute HD-algorithms of a particular structure [24]. On the other hand, the architecture we are proposing is not bound to execute only one specific class of algorithms. A control unit continuously reconfigures the datapath according to a stream of microcode instructions fetched from a tiny embedded configuration memory. This allows the accelerator to be reconfigured at runtime to execute algorithms of a much larger variety by altering the microcode stored in the configuration memory. After configuration, the algorithm is executed autonomously without any further interaction of a host processor. We propose a 26-bit instruction set architecture (ISA) with the encoding space split into 25-bit low-level datapath configuration instructions and 25-bit complex high-level instructions.

Although this structure bears some resemblence with a conventional processor, there are some key differences: The amount of area and energy overhead induced by the control path is less than 1% (16kBit CAM, 65nm) and thus much lower compared to a conventional core. Also, with only 18 available instructions in total, the microcode does not aim to be Turing complete. We introduce just enough configurability to support execution of HDC operations in arbitrary order.

1) Low-level Instructions: The Low-level instructions directly encode the select signals of the multiplexers within the HD-encoder and the address lines of the HD-memory. They thus control the transformation and data flow within the HD-encoder unit during input data encoding. Figure 5 summarizes the function of the bitfields with a single 25-bit low-level instruction. They provide fine-grained control over the datapath with the RIDX and WIDX fields acting like source and destination register operands in a conventional ISA. However, since the Encoder unit contains an output Flip-Flop, many vector transformation operations can be performed without AM access using feedback.

2) High-level Instructions: The high-level instructions encode multi-cycle HDC operations and instructions for code size reduction and host interaction.

a) High-level HDC Operations: For several HDC transformations, there are dedicated high-level multi-cycle instructions; The *AM_SEARCH* instruction starts the associative lookup procedure within the AM. The vector currently stored at the highest index is used as the search vector. As its only operand, the instruction takes an immediate that limits the search space to a maximum index. Only vectors stored at an index smaller than the given maximum are considered during the lookup operation. The immediate value thus allows partitioning the AM dynamically into scratchpad and prototype memory.



ENCSEL	Select between the all-zero vector, a vector from AM and the current HD-encoder output as input for the encoder stage
SMEN	Enable/Bypass Similarity Manipulator Stage
SMSEL	Select between external input data and internal register as input for similarity manipulator stage
MXEN	Enable/Bypass Mixing Stage
MXINV	Select Inverse Permutation set in Mixing Stage
MXSEL	Select between permutation π_0 and π_1 or if MXINV is set between π_0^{-1} and π_1^{-1}.
OP	Select operations to be performed in Encoder Units.
BNDEN	Enable bundle counter thus bundling the current encoder output.
BNDRST	Reset the bundle counter to its initial value
WBEN	Enable write back of the encoder output to AM at index WIDX. If disabled HD-encoder is only stored in output buffer.
RIDX	Read index in case vector from AM is used as encoder input.
WIDX	Write index if the result of current iteration is written back to AM (WBEN = 1).

Fig. 5. Low-level Instruction format

The *MIX* instruction applies multiple mixing cycles to the current content of the encoder register and hence is the basis of IM-mapping. The mixing value is either an immediate or an externally supplied input data, e.g. from an external sensor.

b) Host interaction and Code Size Reduction: An autonomous WuC requires to conditionally signal a target system about the result of the classification algorithm. The proposed design uses a dedicated interrupt instruction to conditionally (or unconditionally) assert an interrupt signal line. The instruction has two operands:

- *Similarity Threshold* - The interrupt is not raised if the last associative lookup operation yielded a result with a Hamming distance higher than the given value.
- *Index Threshold* - The interrupt signal is not raised if the index of the most similar vector found in the last associative lookup operation is higher than the given threshold.

One use case of these thresholds is to wake up the target system only if the HDC classification algorithm detects one particular class with a certainty above a specific threshold.

For the architecture to be autonomous and energy-efficient, the amount of memory required to map a given HD algorithm to the proposed ISA must be kept small.

Thus the algorithm storage in our design supports up to 3 nested hardware loops. Each loop is initiated with a single instruction containing a 10-bit immediate for the number of iterations and a 10-bit immediate for the instruction address that marks the end of the loop body.

The combination of dedicated instructions for commonly used HDC algorithmic primitives and code size reducing features like hardware loops results in a high expressiveness of the ISA. All examined HDC algorithms (see Section V) can be mapped with less than 64 instructions.

3) An Example Configuration for Language Recognition: Language Recognition is a commonly used example applica-

tion in the field of HDC [40, 25, 27, 31, 30, 7, 10]. The task is to determine the language given a sentence in the form of a character string. For a text corpus with 21 European languages, HDC achieves accuracies of up to 96.7% [40]. The algorithm consists of four main steps; in the preprocessing step, the test sentence is split into so-called n-grams, substrings of the test sentence, obtained when applying a sliding window of size n over the character string. In the next step, the individual n-grams of the sentence are each mapped to an HD-vector according to

$V_{\mathtt{n\text{-}gram}} = \pi^{n-1}(V_{\mathtt{n\text{-}1}}) \oplus \pi^{n-2}(V_{\mathtt{n\text{-}2}}) \oplus \ldots \oplus V_0$, with $V_{\mathtt{k}}$ denoting the HD-vector corresponding to the character at index k within the n-gram. This vector is obtained through IM mapping using 27 random HD-vectors (26 characters in the Latin alphabet plus one for whitespaces). π^k denotes the repeated application of a bit permutation (most commonly a binary shift operation), and $\oplus$ is the bind operator (XOR for BSC). The n-gram vectors $V_{\mathtt{n\text{-}gram}}$ for the test sentence are then bundled together to a single search vector $V_{\mathtt{sentence}}$ and in the final step compared with prototype vectors for each language in the AM. The model of the described algorithm, thus the prototype vectors are obtained by bundling together all sentence vectors $V_{\mathtt{sentence}}$ of the training dataset of a language.

In practice, an n-gram size of 4 proved to yield the best performance in terms of accuracy [40].

Listing 1 shows the above algorithm for n=4 in Pseudocode:

```
1:  i ← 0
2:  char_vec_{i-[0,1,2,3]} ← 2048'b0
3:  ngram_{i-[0,1]} ← 2048'b0
4:  for char in sentence do
5:      char_vec_i ← im_map(char)
6:      ngram_i ← π(ngram_{i-1}) ⊕ char_vec_i ⊕ π^4(char_vec_{i-4})
7:      i ← i + 1
8:  end for
9:
10: search_vec ← bundle(ngram_0, ngram_1, ...)
11: idx ← 0
12: min_distance ← ∞
13: class_idx ← 0
14: for p in prototype vectors do
15:     distance ← popcount(search_vec ⊕ p)
16:     if distance < min_distance then
17:         min_distance ← distance
18:         class_idx ← idx
19:     end if
20: end for
```

Listing 1: Pseudocode of an HDC algorithm for language recognition.

Instead of recalculating the same character vectors repeatedly when sliding over the sentence, we recursively compute the n-gram using a FIFO structure [27]. Mapping the above algorithm to the proposed ISA with an AM size of 16 vectors and vector fold of one results in the code in listing 2.

We omitted the initialization steps that would correspond to lines 1-3 in the pseudo-code listing for simplicity. The body of the algorithm (lines 4-8, listing 1) maps to the 12 instructions (lines 1-22) in listing 2. After the hardware loop (line 2 - 22) the search vector is extracted from the MSBs of the bundling counter (line 24) and the search vector is compared with the prototype vectors (line 25) which corresponds to lines 10-20 in listing 1. The instruction on line 26 triggers an interrupt if the processed sentence belongs to the classes represented by

```
1  start:
2    hw.loop0 nr_characters_in_sentence, end_loop
3    # ENCSEL='output register', MXEN=1
4    enc_reg → mix → enc_reg
5    # ENCSEL='memory', RIDX=12, OP=bind, WBEN=1, WIDX=11
6    mem[12] → mix → bind → mem[11]
7    # ENCSEL='memory', RIDX=13, MXEN=1, WBEN=1, WIDX=12
8    mem[13] → mix → mem[12]
9    # ENCSEL='memory', RIDX=14, MXEN=1, WBEN=1, WIDX=13
10   mem[14] → mix → mem[13]
11   # ENCSEL='memory', RIDX=15, MXEN=1, WBEN=1, WIDX=14
12   mem[15] → mix → mem[14]
13   # Generate seed for char vector
14   # ENCSEL='zero', SMEN=1, SMSEL='50%', OP='passthrough'
15   zero_vec → man 50% → enc_reg
16   #IM-Map char represented with 5-bits
17   MIX_EXT 5 #5+2 cycles
18   # ENCSEL='output register', WBEN=1, WIDX=15
19   enc_reg → mem[15]
20   # ENCSEL='memory', RIDX=11, OP='bind', BNDEN=1
21   mem[11] → bind → bundle
22 end_loop:
23   # OP='threshold bundling counters', WBEN=1, WIDX=15
24   threshold_bndl_cntrs → mem[15]
25   am_search nr_classes #nr_classes+2 cycles
26   intr 400, 2
27   jmp start
```

Listing 2: Microcode mapping of the language classification algorithm in pseudo code. Arrows indicate that operations happen in a combinational pipeline configured according to a low-level instruction (see figure 5). The relevant config bitfields for these instructions are indicated in the comments starting with '#' before the relevant line.

prototype 1 or 2 with a Hamming distance of less or equal to 400 bits. The final unconditional jump causes the algorithm to start over again, either immediately if the interrupt conditions are not met or after the host processor clears the pending interrupt.

IV. Implementation and Results

In this section, we evaluate the proposed architecture in terms of area and power consumption. In Section IV-B, we present an overhead analysis of the proposed associative memory. Finally, in subsection IV-C we compare the area and power consumption of the whole accelerator for two different technologies nodes and examine the influence of the vector fold parameter on the efficiency for a given target technology.

A. Methodology

We followed the subsequent methodology for the area and power analysis; the purely digital design written in SystemVerilog RTL was first synthesized with Synopsys Design Compiler 2018.6 using default settings for mapping effort. We evaluate the design's performance in two different target technologies: The first one is a 65 nm Low-Leakage Low-K process node using a high Vth (HVT) standard cell library to minimize cell leakage at low operating frequencies required by the HDC accelerator. If not denoted otherwise, all numbers were obtained with the typical case library characterization at 1.0 V, 25 °C. The second technology we targeted is a 22nm FDSOI node using an UHVT and SLVT library. The library characterization at 0.8 V, 25 °C without body biasing at the typical-typical corner was used. Using Cadence Innovus 2018, we performed place and route with an eight-layer metal stack

for the 65 nm node targeting a core area utilization of 80%. For the 22 nm node, a ten-layer metal stack with a target core area utilization of 70% was used. Post-layout power numbers were obtained with Cadence Voltus using switching activity for all internal nodes extracted from a timing back-annotated post-layout simulation of the HDC algorithms in Mentor Questasim 2019.

B. Energy and Area overhead Analysis of SCM-based AMs

Table II provides an evaluation of the area overhead and energy efficiency for a fully combinational and the row-sequential AM architecture described in Section III-D. To get an accurate estimate of the delay and power consumption at sub nominal voltages, the complete standard cell library was recharacterized with spice simulations using Cadence Liberate for a V_{DD} corner of 0.6 V. At this voltage, all standard cells within the library are still operational in spice simulation.

6T-bitcell-based SRAMs that are readily available in all commercial technology nodes are no longer operational at such low voltages[39, 41]. Although there are specialized low-voltage SRAMs for sub-threshold operation [42], they are custom-tailored for a particular technology and not readily available for all technology nodes. Furthermore, experiments by Andersson et al. indicate that customized SCMs can still have an energy advantage over sub-threshold SRAMs for small memory sizes [43].

At the 0.6V operating corner, we see a 4× improvement in energy efficiency for the sequential architecture and almost 5× for the fully parallel version compared to operation at nominal voltage. The full-parallel implementation is 2.6x more energy efficient than the sequential one. However, for most HDC algorithms, the vast majority of the proposed HDC accelerator's compute time is spent on vector encoding, during which the AM lookup logic stays idle. For this reason, we focus on the row-sequential SCM AM architecture, which has a better trade-off between energy efficiency during lookup operation and static leakage power in the subsequent analysis.

C. Tuning for Maximum Energy Efficiency

As will be further elaborated in Section V, the high amount of parallelism in the datapath and the efficiency of the proposed ISA in executing common HD-algorithms allows the architecture to be clocked at fairly low frequencies while still achieving realtime processing capabilities for many target applications. Figure 6a shows the power breakdown of the proposed architecture synthesized with an AM size of 16kBit (16x 1024 bits) while processing an EMG gesture recognition classification algorithm for different degrees of vector folding. Since higher vector fold values result in less datapath parallelism, we adjusted the frequency for each different vector fold configuration to achieve identical throughput for all configurations. In other words, although the different configurations run at different frequencies, they perform the same amount of useful work per time interval with different degrees of sequentiality.

We see entirely orthogonal tendencies for the two different technology nodes in energy efficiency versus Vector Fold. For 65nm, the overall energy efficiency increases with lower vector folds, thus a higher degree of parallelism, while we see the opposite effect in GF22.

The reason behind this effect becomes evident when we have a closer look at the area breakdown in figure 6b. For a Vector Fold value of one, almost 60% of the accelerator area is occupied by the HD-Encoder. In a technology node like GF22 with SLVT cells, the design is dominated by leakage power. Increasing the vector fold that directly affects the encoder's datapath width has a large effect on the overall area and thus static current draw of the accelerator.

Although the fully synthesizable architecture's technology independence would make it easy to switch to a different technology node with lower leakage, this is not always a possibility, especially when the device is integrated into a larger system. For these situations, the vector fold feature, in addition to its function as a control knob to trade-off area for maximum throughput, provides the means to tune the design for maximum energy efficiency depending on the target technologies' leakage behavior.

V. Applications and Use Cases

As thoroughly discussed in Section III, the proposed HDC accelerator uses hardware-friendly embodiments of commonly used HDC primitives and combines them with a programmable control path. In this section, we take a closer look at the achieved accuracy of the proposed architecture when configured to execute different classification problems using state-of-the-art HDC algorithms. Both, to validate the soundness of the algorithmic transformations and to compare the energy efficiency with other fully digital HDC accelerators.

Training and accuracy evaluation was performed using a bit-true model of the primitive accelerator operations written in python that was verified against the hardware in RTL simulation. Energy numbers were obtained from simulation of the algorithm on the post-layout netlist using a representative subset of cycles.

A. Accuracy Analysis on Text classification and EMG Gesture Recognition

As mentioned earlier, the language classification of textual data is a prime example for classification with HDC. While this application does not fit the context of always-on smart sensing, it serves the purpose of validating the accuracy implications of the permutation-based item memory materialization described in Section III-C2. We tackle the same classification task to classify the text samples into 21 Indo-European languages [40]. We use the same HDC algorithm described in Section III-E with an n-gram size of five, which is identical to the algorithm used by Rahimi et al.. Figure 7 indicates the achieved accuracy using a vector fold factor of 1 for different dimensionalities; for 8192-bit HD vectors, the modified HDC operators achieve an accuracy of 94.52%. This accuracy is almost identical to the results reported by Datta et al. on their accelerator (95.2%) [35]. The algorithm maps to only 14 HDC ISA instructions and has a memory requirement of five vector slots in the AM, in addition to the 21 language

TABLE II
AREA AND ENERGY EFFICIENCY COMPARISON OF SCM-BASED 128 BY 128 BIT AM- AND SRAM-BASED AM-ARCHITECTURE IN 65NM TECHNOLOGY USING ALL THREE AVAILABLE VT FLAVORS. THE MOST ENERGY EFFICIENT SRAM CONFIGURATION GENERATED BY THE AVAILABLE SRAM MACRO GENERATOR COLLECTION FOR THE TARGET TECHNOLOGY WAS CHOSEN.

	Area [kGE]	Throughput [MOPS/s]		Energy Efficiency [pJ/lookup]		Leakage Power [µW]	
		@1.2 V	@ 0.6 V	@ 1.2 V	@ 0.6 V	@1.2 V	@ 0.6 V
SRAM + Digital AM	17	2.56	—	3280	—	1.5	—
Sequential SCM AM	101	1.29	0.23	2353	556	7.5	1.7
Full parallel SCM AM	265	13.80	1.54	921	188	81.0	15.1

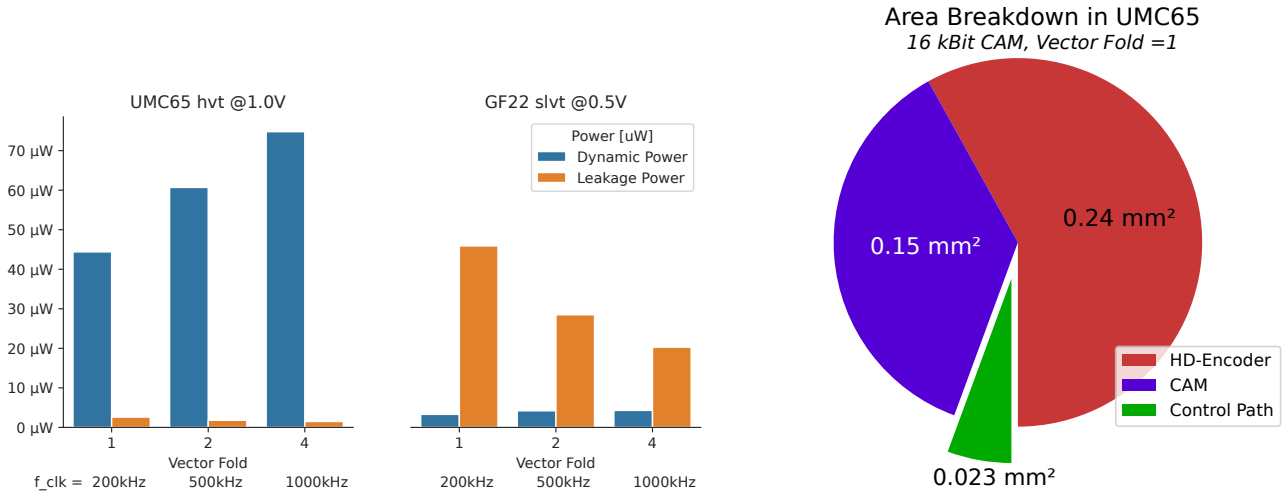


Fig. 6. (a) Post-layout-simulated power consumption of the HDC accelerator (16 vectors, 1024 bits each) when executing a realtime HDC algorithm for different vector folds in 65nm and 22nm technology. (b) Area breakdown of the HDC algorithm for a vector fold of 1, placed and routed in UMC 65nm.

prototype vectors, for intermediate results during the encoding process. For a vector fold of 1, the algorithm executes at 14 cycles per processed input character, which results in 1400 cycles to classify a single sentence.

The second application we evaluate is hand gesture recognition on electromyography (EMG) data recorded on the subject's forearm. We used the dataset and preprocessing pipeline from [26]; the data consists of recordings from the subject performing five different hand gestures captured by a 64-channel EMG sensor array with a sampling rate of 1kSPS. The actual HDC classification algorithm works as follows; for each time sample, the 64 channel values are continuously mapped to HD-vectors using the similarity manipulator module described in Section III-C4 and bound to a per-channel label vector, generated in the mixer stage. Bundling the resulting 64 channel vectors together yields a single HD-vector that represents the state of all channels for a given instance in time. Five of these vectors are combined to a 5-gram analog to the language classification algorithm to form the search vector for associative lookup against the prototype vector. Training of the prototype vectors works like classification, but many search vectors corresponding to the same gesture are bundled together to form the prototype vector.

The whole algorithm maps very well to HDC ISA, requiring only 12 instructions and two memory slots for intermediate results. The inner loop over the 64 channels in the algorithm is executed in only two cycles for a folding factor of 1, which results in a total of **678 cycles** to classify a single 500ms window of data. Consequently, realtime classification of 64 EMG channels implies an accelerator clock frequency of only **1356 Hz**.

While the data preprocessing flow we used in our experiments was identical to [26], the HDC algorithm, although identical in general structure, differs in a few crucial aspects from the baseline implementation. Moin et al. perform CIM mapping of the individual samples to HDC vectors using scalar multiplication of the sample value with a per-channel bipolar label vector, effectively leaving the binary domain [26]. Moreover, the bundling operation to form a time sample vector is implemented as a scalar addition of the integer-valued vectors before thresholding the result back to a bipolar representation with positive values mapped to +1 and negative values to -1. Even though the proposed algorithm modification stays strictly in the binary domain, there is only a small drop in accuracy; with 8192 dimensions, the proposed architecture achieves 96.31% accuracy while Moin et al. report an accuracy of 99.44% accuracy using 10'000-bit vectors and arbitrary precision bundling [26].

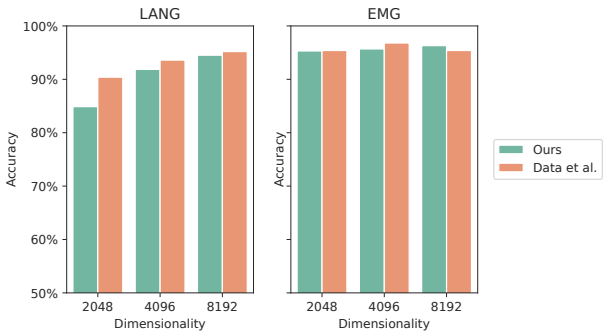


Fig. 7. Achieved accuracies for the target applications using different HD-Vector sizes.

B. Ball Bearing anomaly detection

Predictive maintenance, also known as condition-based maintenance, is a term for the process of estimating the current condition of in-service equipment to anticipate component failure. The goal is to switch to a maintenance scheme where components are replaced once they approach their end-of-life instead of fixed maintenance intervals based on preventive replacement according to the statistically expected lifetime [44]. As part of our algorithmic investigations, we investigate the feasibility of HDC for the task of ball bearing fault prediction using vibration data from low power accelerometer sensors.

For our analysis, we use the IMS Bearing Dataset provided by the University of Cincinnati [45]. They recorded vibration data at a sampling rate of 20kHz from 4 different ball bearings on a loaded shaft rotating at a constant 2000rpm. We concentrated on the first of the three recording sets, which contains 1 second data records obtained with an interval of 10 minutes in a run-to-failure experiment that lasted 35 days with an accumulated operating time of about 15 days.

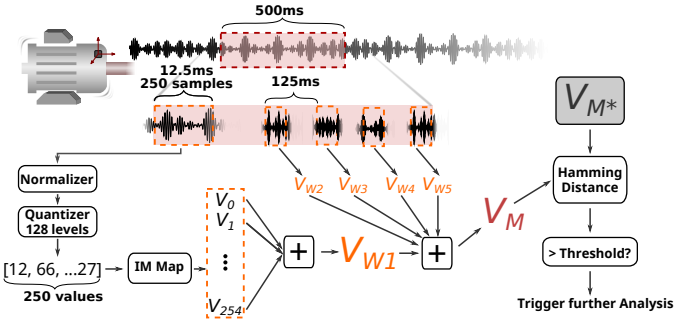


Fig. 8. Illustration of the proposed HDC-based ball bearing anomaly detection algorithm. V_{M*} denotes the online trained calibration vector from the first 24 operating hours of the ball bearing.

Figure 8 illustrates the basic classification procedure. The algorithm requires an initial calibration phase where a prototype vector representing the ball bearing's normal operating condition is generated. With the inherent feature of HDC that classification and training are of almost equivalent computational complexity, online training with HDC imposes negligible additional energy costs. The current control path of the proposed HDC accelerator allows for online training algorithms to be encoded in the algorithm storage but requires an external control entity, e.g., a general-purpose core that provides the labels during algorithm execution.

The algorithm's basis is the encoding of small time windows from the raw vibration data to *measurement vectors* V_M. Each time window consists of 250 samples (12.5ms). The sample values are first normalized using a pre-trained normalization factor and quantized to 7 bits. Each sample value is then mapped to an HD-vector using IM mapping, and the whole window of 250 samples is bundled together to a *window vector* V_W. Five of these window vectors with an interval of 125ms are again bundled together to form a single *measurement vector* V_M. The resulting vector thus approximates the amplitude distribution over a 0.5-second time frame.

The general idea behind the proposed analysis scheme is to generate a prototype vector V_{M*} using the first couple of *measurement vectors* after commissioning. We then track the evolution of Hamming distance over time for subsequent measurement vectors. We calibrated the prototype vector using 100 random measurement vectors from the first 24 operating hours of the respective ball bearing in our experiments. Similarly, the normalization factor is generated using the 99% quantile of the amplitude within the same 24 hours after commissioning. The proposed algorithm can be mapped to **9 HDC ISA instructions** and requires two vector slots, one for the calibration vector and one for intermediate results.

Figure 9 shows the evolution of Hamming distance over time with an exponential moving average filter with a halflife of five hours. This feature can be computed very efficiently without the need for a large ring buffer. The line color indicates the labels proposed by experts on the manual analysis of the dataset [46].

By the end of the IMS ball bearing experiment, bearings 3 and 4 failed, while bearings 1 and 2 were severely worn down but did not fail yet. We see a sharp increase in Hamming distance for all four ball bearings several hours before the actual failure, in the case of ball bearing 3, even several days before the actual inner race failure.

While the proposed algorithm certainly does not replace more involved analysis on time and frequency domain features, the results suggest that it can act as a first filtering stage for aggressive duty cycling of more power-intensive analysis schemes when combined with simple thresholding. However, more experiments on larger datasets and possibly with more complex HDC encoding schemes will be required to quantify the benefits of an HDC-based ball bearing fault predictor.

C. Energy Efficiency Analysis and Comparison

Table III summarizes the performance of the three introduced HDC algorithms, language classification (LANG), EMG gesture recognition (EMG), and ball bearing anomaly detection (BEARING). While EMG and BEARING represent typical workloads for streaming wake-up applications on life sensor data, we picked LANG for its common use in HDC literature as a benchmark application.[40, 25, 27, 31, 30, 7, 10]

TABLE III
MEMORY REQUIREMENTS AND POST-LAYOUT ENERGY NUMBERS OF SELECTED HDC ALGORITHM ON THE PROPOSED ARCHITECTURE WITH AN AM SIZE OF 32 X 2048 BIT, VECTOR FOLD 1

Algorithm	# of HDC instr.	Vector Memory	Cycles/classif.	Realtime Freq. [kHz]	Power [µW]		Min. Energy/classif. [nJ] @100 kHz	
					65nm	22nm	65nm	22nm
LANG	14	5 + 21	1400	100	86.5	23.7	1205	**332**
EMG	12	2 + 5	678	1.4	10.7	2.9	703	**191**
BEARING	9	1 + 1	12513	25	29.1	7.9	10913	**2966**

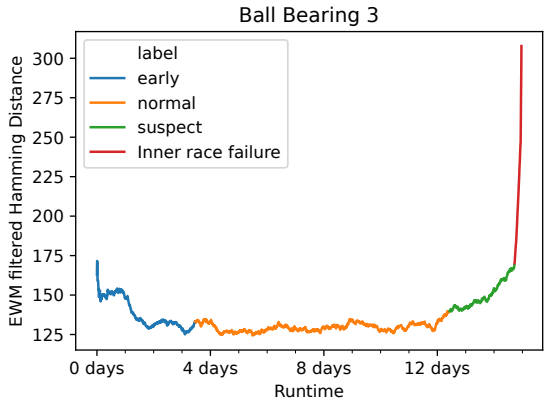


Fig. 9. Hamming distance evolution over time for ball bearings 3 in the IMS dataset. The Hamming distance was post-processed with an exponential moving average filter with a halflife of 5 hours. The other ball bearings in the dataset show a similar behavior.

Columns 2 & 3 report the number of HDC instructions and the total number of required HD vector memory to map the algorithm to the architecture. Column 4 shows the required minimum frequency for realtime execution of the algorithm (not applicable for LANG since there is no realtime constraint for this application). The last two columns indicate the power when operating at the aforementioned minimum frequency and the corresponding energy efficiency per classification. For LANG, we consider a single classification to be the processing of a 100-character string, the average sentence length in the Wortschatz corpora. For EMG and BEARING, a single classification is defined as the analysis of a 500ms window as described in the algorithm sections V-A and V-B.

In Table IV we compare the energy efficiency of our solution to the current SoA HDC accelerator architecture from Datta et al. [35] which differs in several aspects from our work; While we use the rematerialization approach introduced in section III-C2 for IM and CIM mapping, their design uses a large ROM to explicitly store the input to HD vector mapping. Also, they combine a heavily pipelined encoder with a fully combinational, flip-flop-based associative memory whereas our architecture opts for a pipeline-free sequential encoder with a vector sequential, latch-based AM design. Finally, the architecture proposed by Datta et al. is only *generic* (a subset of *general-purpose* architectures) according to their taxonomy on HDC algorithm classes established in [35]. This imposes a set of additional constraints on the structure of HDC algorithms that can be executed. In contrast, the microcode-based approach that our architecture follows allows for arbitrary HDC algorithm computation (including the *generic* ones) within the limits of the available AM and instruction memory resources.

Among other algorithms, Datta et al. report the energy numbers for EMG and LANG executed on a 32 by 2048-bit accelerator in TSMC28. We achieve a technology scaled area reduction by **3.3**×. This can be explained by massive area reductions in all major components of the accelerator. The most considerable effect has the on-the-fly pseudo-random materialization of the item vectors used in our design, which removes the necessity to incorporate a large ROM to store all possible item vectors. In fact, 62% of the overall area in Datta et al. is occupied by a large 1024 by 2048 bit ROM. Besides the area and energy implications, the ROM-based solution has the added drawback of having a hardwired partitioning of the memory; one for the item memory, containing quasi-orthogonal vectors, and one for continuous item memory vectors, where the pair-wise Hamming distance between the vectors correlates to the difference of the corresponding input values. Another large reduction in area is achieved in the AM, where our solution uses latch cells and sequentially calculates the Hamming distance in contrast to the baseline, which uses a flip-flop-based fully parallel implementation.

In fairness, one has to notice that [35], with a maximum clock frequency of 434MHz, has a much higher peak throughput than our solution due to its parallel and heavily pipelined architecture. However, the results in Table III suggest that algorithms used for always-on sensing do not benefit from such a high throughput, and energy efficiency is the key metric by which we should judge the performance of the different approaches.

As we can see in Table IV, the energy efficiency differences between the two architectures depend a lot on the algorithm at hand. For LANG, the achieved energy efficiency is slightly worse (+31%) than the baseline, which is still impressive considering the 3.3× reduction in area. For EMG, on the other hand, we achieve a 3.1× improvement in energy efficiency. This is in contrast to Datta et al.'s architecture, where LANG exhibits a higher energy efficiency than EMG. This can be explained by the difference in the computational complexity of orthogonal and continuous item mapping in our architecture. In LANG, input values are mapped to quasi-orthogonal vectors using the mixing stage (III-C2), which requires $log_2(N)$ cycles, where N denotes the cardinality of the input set. The overhead of this iterative approach considerably lowers the energy advantage of not using a large ROM for item memory

TABLE IV
AREA AND ENERGY EFFICIENCY COMPARISON WITH THE CURRENT STATE-OF-THE-ART HDC ACCELERATOR ARCHITECTURE. THE TERMS *generic* AND *general-purpose* WERE INTRODUCED BY DATTA ET AL. IN [35].

	Technology	Area [kGE]	Architecture Type	IM / CIM resolution [bit]	Energy eff.[nJ/inference]	
					LANG	EMG
Datta et al.	TSMC28	3618	generic	10 or 10	250	610
Our Work	**GF22**	**1094**	**general-purpose**	**arbitrary and 7**	**332**	**191**

generation. For EMG, on the other hand, the input values are mapped continuously using the similarity manipulator, which can be performed in a single cycle and can even be combined with a bundling or bin operation in the subsequent encoder units. Hence, for this algorithm, the effect of not requiring a ROM comes to display and causes the EMG task to execute more efficiently than the LANG task.

In general, we can say that for very high input value resolutions, the overhead of iterative item vector generation starts to dominate the overall energy consumption of our architecture. Thus, for an application-specific accelerator with a small fixed input resolution using IM-based encoding, a ROM-based IM might be more energy efficient than our approach. Still, the fact that the computational complexity of the rematerialization approach grows with the logarithm of the input space instead of linear ROM area scaling suggests an advantage of our architecture for larger input space cardinality. In any case, the proposed architecture excels in its energy proportionality to the desired HDC algorithm. The ROM-based approach in [35] has an almost fixed cost for item memory mapping with an upper limit on the supported resolution. For example, in LANG, only 13% (27 out of 1024 item vectors) of all ROM entries are required to map the input values.

VI. CONCLUSION

In this work, we presented a novel all-digital cross-technology mappable HDC accelerator architecture with a highly configurable datapath using a newly proposed microcode ISA optimized for HDC. Place and routed in GF 22nm technology, the architecture improves on the current state-of-the-art both in area and energy efficiency by a factor of up to 3.1× and 3.3× respectively. The architecture achieves an energy efficiency of 192 $\mathrm{nJ/inference}$ for the task of EMG gesture classification with an always-on compatible typical power consumption of 5 µW. Our post-layout simulation experiments on different digital associative memory architectures in Section IV-B indicate a significant potential for latch-based associative memories to push the limits of energy efficiency when operating at sub-nominal voltage and can already outperform the energy efficiency of commercial-off-the-shelf SRAM macros at nominal voltage. In Section V we demonstrated that our newly introduced rematerialization scheme for IM and CIM mapping have a negligible impact on classification accuracy with a drop of less than 0.5% compared to a ROM-based approach used by the current SoA HDC accelerator. As part of the analysis, we proposed a novel HDC-based end-to-end classification algorithm for ball bearing anomaly detection that maps to only 9 HDC microcode instructions. While our experiments in Section V-C indicated that the energy efficiency of a rematerializing IM is inferior to a ROM-based solution for low input resolutions, the proposed CIM mapping scheme outperforms the current SoA in energy efficiency, area usage, and flexibility. Finally, we provided the first open-source release of a complete HDC Accelerator platform which is possible due to the all-digital nature of the proposed architecture.

VII. ACKNOWLEDGMENTS

This work was supported in part by the Croatian-Swiss Research Programme, project #180625, *"Heterogeneous Computing Systems with Customized Accelerators"*, by the Swiss National Science Foundation and by the European Unions Horizon 2020 Research and Innovation Program through the project MNEMOSENE (Grant 780215).

REFERENCES

[1] B. Chatterjee *et al.*, "Context-Aware Intelligence in Resource-Constrained IoT Nodes: Opportunities and Challenges," *IEEE Design & Test*, vol. 36, no. 2, pp. 7–40, Apr. 2019.

[2] S. Bagchi *et al.*, "New Frontiers in IoT: Networking, Systems, Reliability, and Security Challenges," *IEEE Internet of Things Journal*, pp. 1–1, Jul. 2020.

[3] D. Newell *et al.*, "Review of Power Conversion and Energy Management for Low-Power, Low-Voltage Energy Harvesting Powered Wireless Sensors," *IEEE Transactions on Power Electronics*, vol. 34, no. 10, pp. 9794–9805, Oct. 2019.

[4] V. Shnayder *et al.*, "Simulating the Power Consumption of Large-scale Sensor Network Applications," in *Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems*, ser. SenSys '04. New York, NY, USA: ACM, Nov. 2004, pp. 188–200.

[5] B. Spencer *et al.*, "Smart Sensing Technology: Opportunities and Challenges," *Structural Control and Health Monitoring*, vol. 11, pp. 349–368, Oct. 2004.

[6] A. Sebastian *et al.*, "Memory devices and applications for in-memory computing," *Nature Nanotechnology*, vol. 15, no. 7, pp. 529–544, Jul. 2020.

[7] G. Karunaratne *et al.*, "In-memory hyperdimensional computing," *Nature Electronics*, vol. 3, no. 6, pp. 327–337, Jun. 2020.

[8] I. Miro-Panades *et al.*, "SamurAI: A 1.7MOPS-36GOPS Adaptive Versatile IoT Node with 15,000× Peak-to-Idle Power Reduction, 207ns Wake-Up Time and 1.3TOPS/W

ML Efficiency," in *2020 IEEE Symposium on VLSI Circuits*, Jun. 2020, pp. 1–2.

[9] D. Ma *et al.*, "Sensing, Computing, and Communications for Energy Harvesting IoTs: A Survey," *IEEE Communications Surveys Tutorials*, vol. 22, no. 2, pp. 1222–1250, Dec. 2020.

[10] L. Ge *et al.*, "Classification using Hyperdimensional Computing: A Review," *IEEE Circuits and Systems Magazine*, vol. 20, no. 2, pp. 30–47, Apr. 2020.

[11] A. Rahimi *et al.*, "Hyperdimensional biosignal processing: A case study for EMG-based hand gesture recognition," in *2016 IEEE International Conference on Rebooting Computing (ICRC)*. IEEE, Oct. 2016, pp. 1–8.

[12] F. Montagna *et al.*, "PULP-HD: Accelerating Brain-Inspired High-Dimensional Computing on a Parallel Ultra-Low Power Platform," in *Proceedings of the 55th Annual Design Automation Conference on - DAC '18*. ACM Press, Jun. 2018, pp. 1–6.

[13] S.-G. Cho *et al.*, "A 2048-Neuron Spiking Neural Network Accelerator With Neuro-Inspired Pruning And Asynchronous Network On Chip In 40nm CMOS," in *2019 IEEE Custom Integrated Circuits Conference (CICC)*, Apr. 2019, pp. 1–4.

[14] S. Moradi *et al.*, "A Scalable Multicore Architecture With Heterogeneous Memory Structures for Dynamic Neuromorphic Asynchronous Processors (DYNAPs)," *IEEE transactions on biomedical circuits and systems*, vol. 12, no. 1, pp. 106–122, Feb. 2018.

[15] M. Davies *et al.*, "Loihi: A Neuromorphic Manycore Processor with On-Chip Learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, Jan. 2018.

[16] C. Frenkel *et al.*, "A 0.086-mm 2 12.7-pJ/SOP 64k-Synapse 256-Neuron Online-Learning Digital Spiking Neuromorphic Processor in 28-nm CMOS," *IEEE transactions on biomedical circuits and systems*, vol. 13, no. 1, pp. 145–158, Feb. 2019.

[17] M. Hersche *et al.*, "Integrating event-based dynamic vision sensors with sparse hyperdimensional computing: A low-power accelerator with online learning capability," in *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*, ser. ISLPED '20. New York, NY, USA: Association for Computing Machinery, Aug. 2020, pp. 169–174.

[18] M. Cho *et al.*, "17.2 A 142nW Voice and Acoustic Activity Detection Chip for mm-Scale Sensor Nodes Using Time-Interleaved Mixer-Based Frequency Scanning," in *2019 IEEE International Solid- State Circuits Conference - (ISSCC)*, Feb. 2019, pp. 278–280.

[19] G. Rovere *et al.*, "A 2.2-μ W Cognitive Always-On Wake-Up Circuit for Event-Driven Duty-Cycling of IoT Sensor Nodes," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 8, no. 3, pp. 543–554, Sep. 2018.

[20] J. S. P. Giraldo *et al.*, "Vocell: A 65-nm Speech-Triggered Wake-Up SoC for 10- μ W Keyword Spotting and Speaker Verification," *IEEE Journal of Solid-State Circuits*, vol. 55, no. 4, pp. 868–878, Apr. 2020.

[21] W. Shan *et al.*, "14.1 A 510nW 0.41V Low-Memory Low-Computation Keyword-Spotting Chip Using Serial FFT-Based MFCC and Binarized Depthwise Separable Convolutional Neural Network in 28nm CMOS," in *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*, Feb. 2020, pp. 230–232.

[22] Y. Zhao *et al.*, "A 13.34 μW Event-Driven Patient-Specific ANN Cardiac Arrhythmia Classifier for Wearable ECG Sensors," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 14, no. 2, pp. 186–197, Apr. 2020.

[23] P. Kanerva, "Hyperdimensional Computing: An Introduction to Computing in Distributed Representation with High-Dimensional Random Vectors," *Cognitive Computation*, vol. 1, no. 2, pp. 139–159, Jun. 2009.

[24] A. Rahimi *et al.*, "Efficient Biosignal Processing Using Hyperdimensional Computing: Network Templates for Combined Learning and Classification of ExG Signals," *Proceedings of the IEEE*, vol. 107, no. 1, pp. 123–143, Jan. 2019.

[25] A. Rahimi *et al.*, "High-Dimensional Computing as a Nanoscalable Paradigm," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, no. 9, pp. 2508–2521, Sep. 2017.

[26] A. Moin *et al.*, "An EMG Gesture Recognition System with Flexible High-Density Sensors and Brain-Inspired High-Dimensional Classifier," *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, Feb. 2018.

[27] A. Joshi *et al.*, "Language Geometry Using Random Indexing," in *Quantum Interaction*, ser. Lecture Notes in Computer Science, J. A. de Barros *et al.*, Eds. Cham: Springer International Publishing, Jan. 2017, pp. 265–274.

[28] M. Imani *et al.*, "HDNA: Energy-efficient DNA sequencing using hyperdimensional computing," in *2018 IEEE EMBS International Conference on Biomedical Health Informatics (BHI)*, Mar. 2018, pp. 271–274.

[29] D. Kleyko *et al.*, "Brain-like classifier of temporal patterns," in *2014 International Conference on Computer and Information Sciences (ICCOINS)*, Jun. 2014, pp. 1–6.

[30] T. F. Wu *et al.*, "Hyperdimensional Computing Exploiting Carbon Nanotube FETs, Resistive RAM, and Their Monolithic 3D Integration," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 11, pp. 3183–3196, Nov. 2018.

[31] H. Li *et al.*, "Hyperdimensional computing with 3D VRRAM in-memory kernels: Device-architecture co-design for energy-efficient, error-resilient language recognition," in *2016 IEEE International Electron Devices Meeting (IEDM)*, Dec. 2016, pp. 16.1.1–16.1.4.

[32] M. Schmuck *et al.*, "Hardware Optimizations of Dense Binary Hyperdimensional Computing: Rematerialization of Hypervectors, Binarized Bundling, and Combinational Associative Memory," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 15, no. 4, pp. 32:1–32:25, Oct. 2019.

[33] S. Salamat *et al.*, "F5-HD: Fast Flexible FPGA-based Framework for Refreshing Hyperdimensional Comput-

ing," in *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '19. New York, NY, USA: Association for Computing Machinery, Feb. 2019, pp. 53–62.

[34] S. Salamat *et al.*, "Accelerating Hyperdimensional Computing on FPGAs by Exploiting Computational Reuse," *IEEE Transactions on Computers*, vol. 69, no. 8, pp. 1159–1171, Aug. 2020.

[35] S. Datta *et al.*, "A Programmable Hyper-Dimensional Processor Architecture for Human-Centric IoT," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 3, pp. 439–452, Sep. 2019.

[36] M. Imani *et al.*, "Exploring Hyperdimensional Associative Memory," in *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. Austin, TX: IEEE, Feb. 2017, pp. 445–456.

[37] A. Moin *et al.*, "A wearable biosensing system with in-sensor adaptive machine learning for hand gesture recognition," *Nature Electronics*, vol. 4, no. 1, pp. 54–63, Jan. 2021.

[38] A. Teman *et al.*, "Power, Area, and Performance Optimization of Standard Cell Memory Arrays Through Controlled Placement," *ACM Transactions on Design Automation of Electronic Systems*, vol. 21, no. 4, pp. 1–25, May 2016.

[39] P. Meinerzhagen *et al.*, "Benchmarking of Standard-Cell Based Memories in the Sub-V_T Domain in 65-nm CMOS Technology," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 1, no. 2, pp. 173–182, Jun. 2011.

[40] A. Rahimi *et al.*, "A Robust and Energy-Efficient Classifier Using Brain-Inspired Hyperdimensional Computing," in *Proceedings of the 2016 International Symposium on Low Power Electronics and Design*, ser. ISLPED '16. San Francisco Airport, CA, USA: Association for Computing Machinery, Aug. 2016, pp. 64–69.

[41] M. E. Sinangil *et al.*, "A reconfigurable 65nm SRAM achieving voltage scalability from 0.25–1.2V and performance scalability from 20kHz–200MHz," in *ESSCIRC 2008 - 34th European Solid-State Circuits Conference*, Sep. 2008, pp. 282–285.

[42] B. Mohammadi *et al.*, "A 128 kb 7T SRAM Using a Single-Cycle Boosting Mechanism in 28-nm FD–SOI," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 4, pp. 1257–1268, Apr. 2018.

[43] O. Andersson *et al.*, "Ultra Low Voltage Synthesizable Memories: A Trade-Off Discussion in 65 nm CMOS," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 6, pp. 806–817, Jun. 2016.

[44] S. Selcuk, "Predictive maintenance, its implementation and latest trends," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, vol. 231, no. 9, pp. 1670–1679, Jul. 2017.

[45] H. Qiu *et al.*, "Wavelet filter-based weak signature detection method and its application on rolling element bearing prognostics," *Journal of Sound and Vibration*, vol. 289, no. 4, pp. 1066–1090, Feb. 2006.

[46] J. Ben Ali *et al.*, "Linear feature selection and classification using PNN and SFAM neural networks for a nearly online diagnosis of bearing naturally progressing degradations," *Engineering Applications of Artificial Intelligence*, vol. 42, pp. 67–81, Jun. 2015.

Manuel Eggimann (GS'18) Received his B.Sc. and consecutive M.Sc. degree in electrical engineering and information technology from the ETH Zurich, Switzerland in 2018. He is currently pursuing a Ph.D. degree at the ETH Zurich Integrated Systems Laboratory. His research interests include low-power hardware design, edge-computing and energy efficient SoC interconnect architectures. He is the recipient of the best paper award at the 2019 IEEE 8th International Workshop on Advances in Sensors and Interfaces.

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, CA, USA, in 2015. Since then, he held post-doctoral research positions in the Department of Electrical Engineering and Computer Sciences at the University of California, Berkeley, CA, USA, and in the Department of Information Technology and Electrical Engineering at ETH Zürich, Zürich, Switzerland. In 2020, he has joined the IBM Research-Zürich laboratory in Rüschlikon, Switzerland, as a Research Staff Member. 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 2015 Outstanding Dissertation Award in the area of "New Directions in Embedded System Design and Embedded Software" from the European Design and Automation Association, and the ETH Zürich Postdoctoral Fellowship in 2017. He was a co-recipient of the Best Paper Nominations at DAC (2013) and DATE (2019), and the Best Paper Awards at BICT (2017) and BioCAS (2018).

Luca Benini (F'07) received a Ph.D. degree in electrical engineering from Stanford University, USA, in 1997. He has served as the Chief Architect of the Platform2012/STHORM Project with STMicroelectronics, Grenoble, France, from 2009 to 2013. Currently, he holds the chair of Digital Circuits and Systems at ETH Zurich, Switzerland, and is a full professor at the University of Bologna, Italy. He has published more than 1000 peer-reviewed articles and five books. His current research interest includes energy-efficient computing systems design from embedded to high performance. Dr. Benini is a fellow of the ACM and a member of the Academia Europaea. He was a recipient of the 2016 IEEE CAS Mac Van Valkenburg Award and the 2019 IEEE TCAD Donald O. Pederson Best Article Award.