A 0.44 μJ/dec, 39.9 μs/dec, Recurrent Attention In-memory Processor for Keyword Spotting

Hassan Dbouk, *Student Member, IEEE,* Sujan K. Gonugondla, *Student Member, IEEE,* Charbel Sakr, *Student Member, IEEE,* and Naresh R. Shanbhag, *Fellow, IEEE*

***Abstract*—This paper presents a deep learning-based classifier IC for keyword spotting (KWS) in 65 nm CMOS designed using an algorithm-hardware co-design approach. First, a Recurrent Attention Model (RAM) algorithm for the KWS task (the KeyRAM algorithm) is proposed. The KeyRAM algorithm enables accuracy vs. energy scalability via a confidence-based computation (CC) scheme, leads to a $2.5\times$ reduction in computational complexity compared to state-of-the-art (SOTA) neural networks, and is well-suited for in-memory computing (IMC) since the bulk (89%) of its computations are 4b matrix-vector multiplies. The KeyRAM IC comprises a multi-bit multi-bank in-memory computing (IMC) architecture with a digital co-processor. A sparsity-aware summation scheme is proposed to alleviate the challenge faced by IMCs when summing sparse activations. The digital co-processor employs diagonal major weight storage to compute without any stalls. This combination of the IMC and digital processors enables a balanced trade-off between energy efficiency and high accuracy computation. The resultant KWS IC achieves SOTA decision latency of $39.9\,\mu$s with a decision energy $< 0.5\,\mu$J/dec which translates to more than $24\times$ savings in the energy-delay product (EDP) of decisions over existing KWS ICs.**

***Index Terms*—In-memory computing (IMC), keyword spotting (KWS), machine learning, recurrent attention networks**

I. Introduction

SPEECH has emerged as a natural mode for humans to interact with intelligent Edge devices including smart phones and personal digital assistants [1]. A full fledged speech recognition system is complex and power hungry making it prohibitive for resource-constrained applications at the Edge. Preceding a speech recognizer with an 'always-on' keyword spotting (KWS) system enables such devices to continually sense, detect, and classify speech segments under stringent energy, computational and storage constraints.

A typical keyword spotting (KWS) pipeline (Fig. 1) processes raw audio in two stages: feature extraction and classification. The classification stage, which is often implemented via neural networks, dominates the complexity of the KWS system and presents a challenge for efficient KWS realization. Recently, various deep learning-based KWS classifier algorithms such as deep/convolutional neural networks (DNNs/CNNs) and recurrent NNs (RNNs) have been shown to achieve high

Hassan Dbouk, Charbel Sakr, and Naresh R. Shanbhag are with the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA (e-mail: hdbouk2@illinois.edu; sakr2@illinois.edu; shanbhag@illinois.edu).

Sujan K. Gonugondla was with the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA. He is now with Amazon, Seattle, WA, USA (e-mail:gsujan@amazon.com).

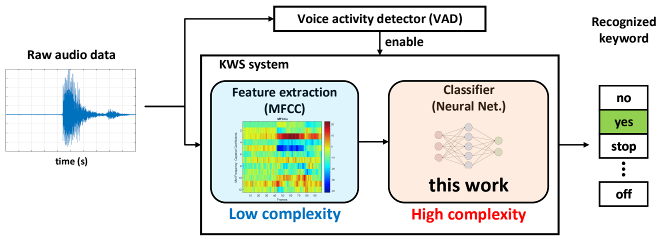


Fig. 1. A typical keyword spotting (KWS) pipeline. The classifier, which is implemented via neural networks, dominates the complexity.

($> 90\%$) accuracies [2] but at the expense of very high computational costs making them unsuitable for deployment on Edge platforms.

To that end, existing KWS ICs [3]–[7] aim at implementing off-the-shelf NN-based classifiers for KWS while focusing on minimizing the power consumption in order to maintain continuous operation. These ICs have mostly been: 1) implemented in digital utilizing low-power techniques such as voltage overscaling [4], [5], [7]; and 2) benchmarked with simple datasets, e.g., TIMIT. One exception is [6] which implements a binarized RNN-based KWS on an in-memory computing (IMC) architecture for the more complex Google Speech dataset [8]. Recently, [7] reported the lowest power consumption of $510\,$nW for a KWS IC by implementing depth-wise separable convolutional neural networks (DSCNNs) [9]. However, the implementation of [7] is limited to two keywords and requires $64\,$ms to classify one speech segment. For a ten keyword classification, the power penalty increases to $16\,\mu$W [6].

Realistically, speech related events occur infrequently which makes the 'always-on' detection of wake words using a KWS IC to be energy sub-optimal. Especially since a voice activity detector (VAD) IC can achieve $P_{\text{VAD}} < 200\,$nW [10], which is much less than the KWS system. Therefore, by gating the KWS system with a VAD (Fig. 1), the total system power P_{tot} can be significantly reduced and the power constraint on the KWS system is relaxed, as long as the audio events occur infrequently. In fact, it can easily be shown that the total system power can be expressed as:

$$P_{\text{tot}} = P_{\text{VAD}} + \alpha E_d \tag{1}$$

where α is the voice activity rate (activity/s) and E_d is the decision energy of the KWS IC. Expression (1) implies that the traditional viewpoint of minimizing the power of KWS ICs needs to be reconsidered. Instead, designers need to reduce the energy per decision E_d for a specified decision latency. This can be done by minimizing the decision-level energy-delay product (EDP) of the KWS IC, which is the focus of this work.

With that design goal in mind, this work employs an algorithm-hardware co-design approach to realize KWS for Edge devices with $< 1\,\mu$J/decision and a latency of $< 50\,\mu$s/decision. To the best our knowledge, this is the first work to propose using a Recurrent Attention Model (RAM) [11], previously proposed for image classification, for KWS (KeyRAM algorithm) *and* the first IC implementation of RAM for KWS (KeyRAM IC). The use of RAM for the KWS task reduces the computational complexity of inference compared to state-of-the-art neural network-based algorithms (Fig. 6) at iso-accuracy. The KeyRAM algorithm is mapped onto an IC consisting of two multi-bit sparsity-aware IMCs and a digital co-processor with a 96×512 standard 6T SRAM. The IC stores all model weights on-chip to further increase the energy efficiency. As a result, up to $7.6\times$ savings in energy/decision and $> 24\times$ savings in EDP of decisions over state-of-the art IMC ICs for KWS is achieved, while realizing the lowest reported decision latency of $39.9\,\mu$s. Preliminary measurement results were reported in [12].

This paper is organized as follows: Section II provides the necessary background on IMC and RNNs. Section III explains the KeyRAM algorithm and demonstrates its effectiveness. The circuit and architectural implementation details of the KeyRAM IC are described in Section IV. Measurement results demonstrating the classifier's confidence, accuracy, latency, and energy are presented in Section V. Section VI concludes this paper.

II. Background

A. In-Memory Computing

In-memory computing (IMC) architectures [13]–[22] have drawn much interest due to their ability to reduce the energy and latency cost of memory accesses for machine learning applications. The IMC concept was first introduced in [23] and was later followed by numerous prototype ICs [13]–[15] demonstrating its effectiveness. Since then numerous IMC architectures [17]–[22] spanning different bitcell layouts, e.g., 6T, 8T, 10T, and bit-precision support, e.g., binary, ternary, multi-bit, have been proposed. In this section, we focus on the IMC design in [13], [14], which is a mixed-signal multi-bit dot-product processor based on embedding the processing into the standard 6T SRAM bitcell array (BCA). This architecture realizes dot products efficiently, while preserving the density of the BCA. The IMC stores B_w-bit weights in the BCA in a column major format and computes dot-products between these weights and inputs stored in buffers in three steps: 1) bitcells (BCs) storing digital weights are concurrently accessed via pulse-width modulated word-lines (WLs), such that the voltage discharge ΔV_{BL} on each bit-line (BL) is proportional to the multi-bit weight in the memory; 2) the BL discharges are multiplied with the corresponding input data from buffers via column pitch-matched charge redistribution-based multipliers;

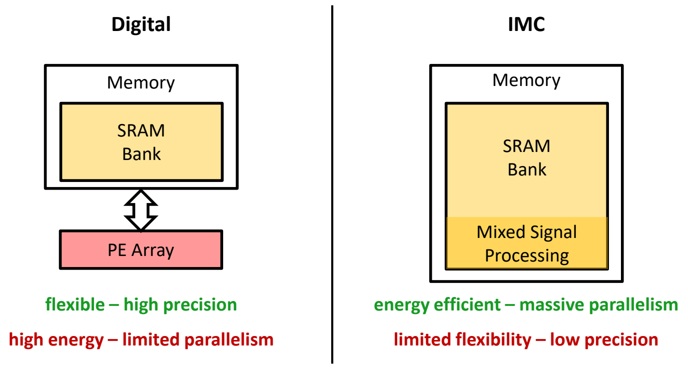

Digital
Memory
SRAM Bank
PE Array
flexible – high precision
high energy – limited parallelism
IMC
Memory
SRAM Bank
Mixed Signal Processing
energy efficient – massive parallelism
limited flexibility – low precision

Fig. 2. A high-level comparison between a von Neumann digital architecture and an IMC architecture.

and 3) the multiplier outputs are summed across the columns via charge sharing across BLs resulting in the final dot product which is converted to digital via ADCs. The accuracy of the IMC computations can be traded off with energy by controlling the word-line voltage V_{WL}. While IMC architectures (Fig. 2) have shown remarkable gains ($100\times$) in the energy-delay product (EDP) over a von Neumann equivalent with minimal loss in inference accuracy [14], they typically are limited to low-precision computations, and lack the flexibility of the von Neumann architecture due to their mixed-signal nature.

B. Recurrent Neural Networks

Recurrent neural networks (RNNs) [24] are a special type of fully connected (FC) networks characterized by a hidden state vector. RNNs maintain the hidden state vector via a feedback structure, which enables them to store temporal features when processing data sequentially. This makes RNNs suitable for speech related tasks which exhibit strong temporal correlations.

Given a length T sequence of N-dimensional input vectors $\mathbf{x}_1, ..., \mathbf{x}_T$, at each timestep t, an RNN updates its L-dimensional hidden state vector $\mathbf{h}_t$ as follows:

$$\mathbf{h}_t = f_h(\mathbf{h}_{t-1} \parallel \mathbf{x}_t) = \sigma\Big(\mathbf{W}_h \cdot (\mathbf{h}_{t-1} \parallel \mathbf{x}_t) + \mathbf{b}_h\Big) \quad (2)$$

where $\sigma(.)$ is an element-wise non-linearity function (typically a ReLU or a sigmoid), $(. \parallel .)$ denotes the vector concatenation operation, and $\mathbf{W}_h \in \mathbb{R}^{L\times(L+N)}$ and $\mathbf{b}_h \in \mathbb{R}^{L}$ are the learned weights and bias terms associated with the RNN. The initial hidden state vector $\mathbf{h}_0$ is the all zeros vector. In an M-way classification setup, an RNN is also equipped with a classifier that maps the hidden state vector into class probabilities. Specifically, we have

$$\mathbf{y} = f_y(\mathbf{h}_T) = \mathrm{softmax}\Big(\mathbf{W}_y \cdot \mathbf{h}_T + \mathbf{b}_y\Big) \quad (3)$$

where $\mathbf{y} \in [0,1]^M$ is a vector of class scores, $\mathbf{W}_y \in \mathbb{R}^{M\times(L+N)}$ and $\mathbf{b}_y \in \mathbb{R}^M$ are the learned weights and bias terms associated with the classifier. The classifier needs to operate on the final hidden state vector in the sequence $\mathbf{h}_T$, which is a function of all the previous input vectors. During testing,

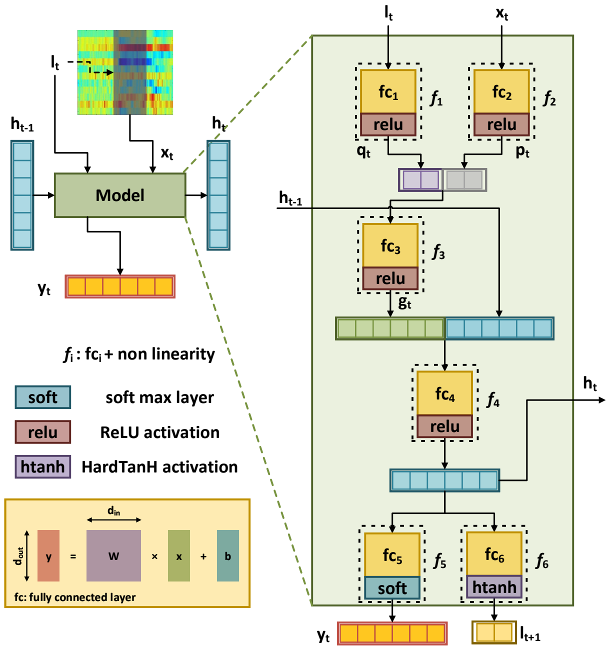


Fig. 3. The Recurrent Attention Model (RAM) for KWS.

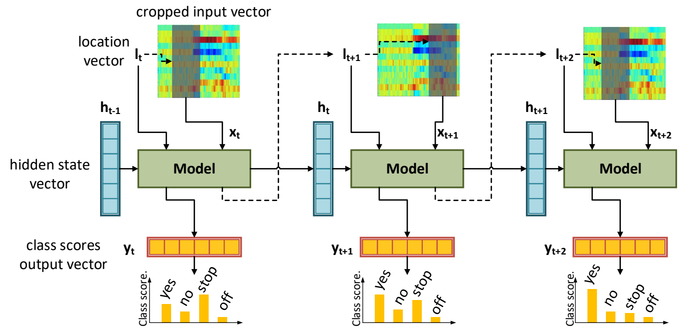


Fig. 4. Improvement in confidence of keyword 'yes' as more glimpses are processed.

the pre-softmax output vector $\tilde{\mathbf{y}}$ is sufficient for predicting the associated class $c \in \{1, ..., M\}$, since we have:

$$c = \arg\max_{i\in[M]}\{y_i\} = \arg\max_{i\in[M]}\{\tilde{y}_i\} \tag{4}$$

Similar to other network architectures, RNNs can be trained in a supervised learning setup using labeled data and the stochastic gradient descent (SGD) algorithm. The required gradients can be computed using back-propagation through time (BPTT) [25], which is essentially the traditional back-propagation algorithm with the recursive computation of the RNN unrolled.

III. The KeyRAM Algorithm

This sections describes the KeyRAM algorithm and compares its performance against state-of-the art neural network-based algorithms for KWS. KeyRAM employs the RAM algorithm [11], originally proposed for image classification, for the KWS task. It employs an RNN in a feedback loop to selectively process input subsets (glimpses) in the feature space, as opposed to processing the entire input feature, thereby reducing the computational complexity.

A. *KeyRAM Algorithm*

The KeyRAM algorithm (Fig. 3) employs 6 fully connected layers (f_1 to f_6) to track the informative features across multiple glimpses t via a hidden state vector $\mathbf{h}_t$. At each glimpse t, KeyRAM combines the previous hidden state vector $\mathbf{h}_{t-1}$ with the glimpse vector $\mathbf{g}_t$ (which is obtained from the input patch vector $\mathbf{x}_t$ at location $\mathbf{l}_t$) to compute $\mathbf{h}_t$:

$$\mathbf{h}_t = f_4(\mathbf{h}_{t-1} \,\|\, \mathbf{g}_t) = \text{ReLU}\Big(\mathbf{W}_4 \cdot (\mathbf{h}_{t-1} \,\|\, \mathbf{g}_t) + \mathbf{b}_4\Big) \tag{5}$$

The glimpse vector $\mathbf{g}_t$ combines information from the input patch vector and the location vector via two stages of processing involving three fully connected layers:

$$\mathbf{g}_t = f_3(\mathbf{q}_t \,\|\, \mathbf{p}_t) = \text{ReLU}\Big(\mathbf{W}_3 \cdot (\mathbf{q}_t \,\|\, \mathbf{p}_t) + \mathbf{b}_3\Big) \tag{6}$$

where $\mathbf{p}_t$ and $\mathbf{q}_t$ are the processed input patch vector $\mathbf{x}_t$ and location vector $\mathbf{l}_t$ respectively. They are computed as follows:

$$\mathbf{p}_t = f_2(\mathbf{x}_t) = \text{ReLU}\Big(\mathbf{W}_2 \cdot \mathbf{x}_t + \mathbf{b}_2\Big) \tag{7}$$

$$\mathbf{q}_t = f_1(\mathbf{l}_t) = \text{ReLU}\Big(\mathbf{W}_1 \cdot \mathbf{l}_t + \mathbf{b}_1\Big) \tag{8}$$

Given the updated hidden state vector $\mathbf{h}_t$, the KeyRAM algorithm predicts both the up-to-date class probability scores $\mathbf{y}_t$ and the next location vector to process $\mathbf{l}_{t+1}$ as follows:

$$\mathbf{y}_t = f_5(\mathbf{h}_t) = \text{softmax}\Big(\mathbf{W}_5 \cdot \mathbf{h}_t + \mathbf{b}_5\Big) \tag{9}$$

$$\mathbf{l}_{t+1} = f_6(\mathbf{h}_t) = \text{hardtanh}\Big(\mathbf{W}_6 \cdot \mathbf{h}_t + \mathbf{b}_6\Big) \tag{10}$$

After training, the class prediction of the KeyRAM algorithm improves in confidence (class score) as more glimpses are processed (Fig. 4). To do so, the location vectors produced over time must correspond to locations in the input feature that are highly informative. This dynamic nature of KeyRAM enables a natural trade-off between complexity and accuracy, which is translates to an energy/latency vs. accuracy trade-off, where the number of glimpses per decision n_G acts as a tuning knob. In order to train the KeyRAM algorithm, we use the same hybrid supervised loss function from [11], which formulates RAM as a reinforcement learning problem, with the location network acting as an agent whose loss function is based on a reward, whereas the rest of the layers are learned using the standard cross-entropy loss function.

B. *Enabling Accuracy-Energy Scalability*

The decision accuracy of the KeyRAM algorithm is proportional to the class confidence level or the class score, which increases with the number of glimpses n_G and hence the energy cost. We introduce a confidence-based computation

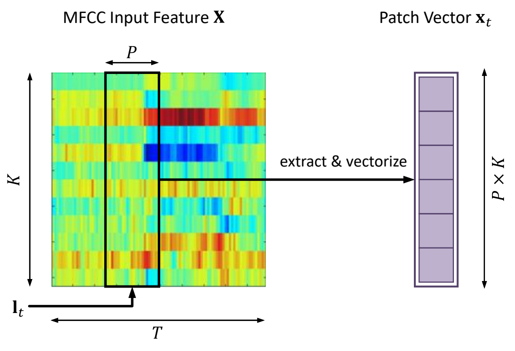


Fig. 5. The patch extractor function ϕ used in KeyRAM.

(CC) strategy to provide a seamless trade-off between accuracy and energy. At each glimpse t, let $c_t \in [M]$ be the predicted class label obtained from $\mathbf{y}_t$ in (9) as follows:

$$c_t = \arg\max_{i\in[M]}\{y_{i,t}\} \tag{11}$$

and $y_{c_t,t} \in [0,1]$ be its associated class score. For a specific threshold $\tau \in [0,1]$, the KeyRAM algorithm with the CC strategy (KeyRAM-CC) terminates at glimpse $t = n_G$ if one of the following conditions holds: $y_{c_{n_G},n_G} > \tau$ or $n_G = N_G$, where N_G is the maximum number of glimpses allowed per decision, i.e., $n_G \leq N_G$. The choice of the confidence threshold τ can be used to trade-off the decision accuracy w.r.t. complexity. A small value of τ implies that the classifier isn't confident about its decision resulting in lower accuracy but also lower energy consumption since decisions are obtained with smaller values of n_G. On the other hand, a large value of τ implies a higher value of n_G, i.e., more glimpses per decision, and hence better accuracy at the expense of more energy consumption.

C. *Enabling Audio Classification*

To enable audio classification, we use Mel-frequency Cepstral Coefficient (MFCC) [26] features as KeyRAM inputs. Since audio features exhibit only temporal invariance, we propose a location vector $\mathbf{l}_t$ that points to the time index of the MFCC feature.

Figure 5 describes the input patch vector generation for a given MFCC feature matrix $\mathbf{X} \in \mathbb{R}^{K\times T}$ where K denotes the number of Mel-features used and T is the number of time samples. The location vector $\mathbf{l}_t$ points to a time index, and a rectangular patch of size $K \times P$ is extracted accordingly. The input patch vector $\mathbf{x}_t \in \mathbb{R}^{PK}$ is therefore obtained by vectorizing the rectangular patch. This process can be described using the patch extractor function ϕ:

$$\mathbf{x}_t = \phi(\mathbf{X}, P, \mathbf{l}_t) \tag{12}$$

The wider the patch (higher P), the more information is contained in $\mathbf{x}_t$, and more processing required per glimpse. Note that for 2D images, the location vector $\mathbf{l}_t$ would be two dimensional, where each entry corresponds to a separate image axis. However, for KWS, we are only interested in one axis, and therefore $\mathbf{l}_t$ becomes a scalar l_t.

TABLE I
LAYER DIMENSIONS AND ACTIVATION FUNCTIONS FOR THE KEYRAM ALGORITHM. VALUES IN **BOLD** ARE USED IN THE KEYRAM IC.

Layer	d_{in}	d_{out}	σ
f_1	**2**	d_l (**63**)	ReLU
f_2	$K \times P$ (**8 × 8**)	d_x (**64**)	ReLU
f_3	$d_x + d_l$ (**64 + 63**)	d_g (**127**)	ReLU
f_4	$d_g + d_h$ (**127 + 127**)	d_h (**127**)	ReLU
f_5	d_h (**127**)	M (**2-to-10**)	softmax
f_6	d_h (**127**)	**2**	hardtanh

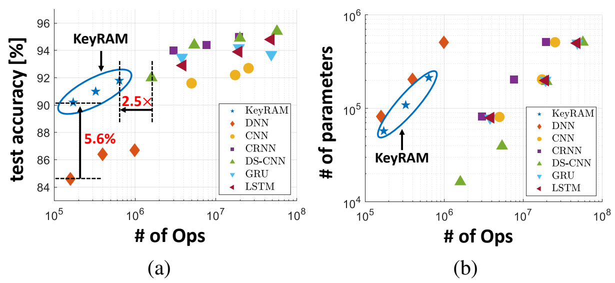
Table I details both the activation functions as well as the shapes of the 6 fully connected layers as a function of: d_l the dimension of $\mathbf{q}_t$; d_x the dimension of $\mathbf{p}_t$; d_g the dimension of $\mathbf{g}_t$; and d_h the dimension of $\mathbf{h}_t$, for using KeyRAM with M keywords, K Mel-features, and a patch of width P.

D. *Performance of KeyRAM*

To benchmark the effectiveness of the KeyRAM algorithm, we use the recent Google Speech [8] dataset. We compare KeyRAM against several classical neural network architectures for KWS reported in [2]. Similar to [2], we use 12 keywords (including 'silence' and 'unknown' labels) from the Google Speech dataset. Figure 6(a) plots the performance, in terms of classification accuracy, of different network architectures and KeyRAM vs. number of floating-point operations required for a single inference. The complexity of the KeyRAM is varied by changing the number of glimpses and the model size. KeyRAM reduces the number of operations required for inference by $\sim 2.5\times$ compared to state-of-the-art neural network-based algorithms from [2] at iso-accuracy. For the same number of operations, KeyRAM is able to achieve a $\sim 5.6\%$ improvement in the classification accuracy as well. Figure 6(b) shows that KeyRAM remains competitive in terms of storage, which is reflected in the number of model parameters required.

IV. IMPLEMENTATION

This section describes the architecture of the KeyRAM prototype IC in 65 nm CMOS.



(a) (b)

Fig. 6. The accuracy-complexity-storage requirements of KeyRAM compared to other neural network classification algorithms for the 12-class Google Speech dataset [8]: (a) test accuracy, and (b) number of parameters vs. number of operations.

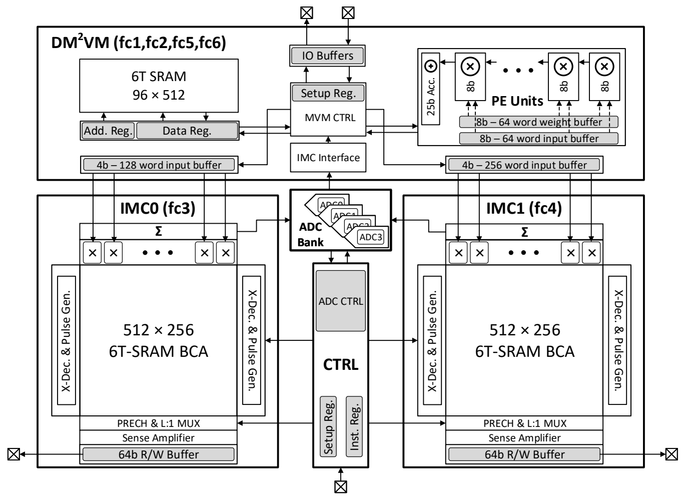


Fig. 7. The KeyRAM IC architecture.

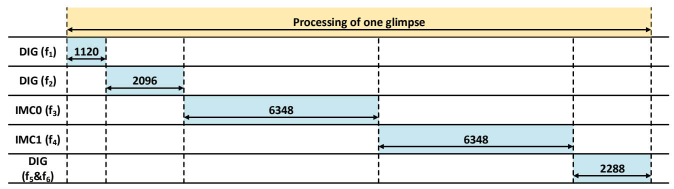


Fig. 8. Timing diagram of the KeyRAM IC.

A. Architectural Mapping

The main challenges in implementing the KeyRAM algorithm using an IMC-based architecture are threefold: 1) the irregular layer shapes in KeyRAM makes it difficult to map onto IMC architectures, where the support for arbitrary length dot products (DPs) is limited; 2) the sparsity of the input activations present a challenge for any IMC implementation employing a charge sharing summation scheme, such as that of [13]; and 3) the precision requirements for different layers vary. In fact it is quite common in DNN quantization [27]–[30] to allocate higher bit-widths for the first and last layers. This presents an issue for IMC architectures, as they are typically meant for low-precision operations, due to the inherent non-idealities associated with mixed-signal computing.

However, we find via precision analysis, that KeyRAM's early (f_1 and f_2) and final (f_5 and f_6) layers require 8b precision, whereas the intermediate layers f_3 and f_4 require 4b precision (both weights and activations) in order to maintain accuracy. Furthermore, Table II indicates that our design choice of 8b for f_1/f_2/f_5/f_6 and 4b for f_3/f_4 achieves an accuracy within 0.2% of that of an 8b baseline whereas a fully 4b KeyRAM algorithm would suffer from an accuracy loss of $\sim 3\%$ compared to the 8b baseline thereby indicating that extreme quantization, e.g., binarization, is not an option.

The parameters values of the KeyRAM architecture chosen for IC implementation are detailed in Table I. The number of operations of the resultant network architecture is dominated

TABLE II
ACCURACY OF THE KEYRAM ALGORITHM FOR THE 7-CLASS GOOGLE SPEECH DATASET VS. DIFFERENT PRECISION PROFILES USING 4 GLIMPSES. TRAINING SETUP AND DETAILS ARE PROVIDED IN SECTION V-A.

Layer Precision Assignment						
f_1	f_2	f_3	f_4	f_5	f_6	Accuracy [%]
8b	8b	8b	8b	8b	8b	91.42
4b	4b	4b	4b	4b	4b	88.16
8b	**8b**	**4b**	**4b**	**8b**	**8b**	**91.25**

(89.2%) by the internal layers f_3 and f_4. Both layers require 127 dot-products to be computed, of length 128 and 255 respectively (bias addition is realized by augmenting the input vector by 1). Fortunately, these require 4b operations. Thus, an IMC is well-suited for implementing f_3 and f_4. The remaining layers f_1, f_2, f_5, and f_6 (10.8% of the total complexity) are computed in 8b precision digital using a novel and optimized dataflow. Layers f_5 and f_6 are fused together and implemented using a single matrix-vector multiply (MVM) routine, since they share the same input vector $\mathbf{h}_t$.

B. Architecture and Timing

The KeyRAM architecture (Fig. 7) processes a single glimpse using 5 operating modes (Fig. 8) without the need for off-chip data transfer since all weights are stored on-chip. Internal vector buffers are utilized to transfer data between operating modes. These buffers allow us to swap the output of one stage into the input of the next stage without stalling.

The architecture comprises of: 1) two IMC blocks (IMC0 and IMC1) to implement multi-bit MVMs via temporal folding into a sequence of dot products. The IMC block is based on the single-bank IMC architecture in [13] which implements a single multi-bit dot-product per read cycle. Each IMC block in KeyRAM consists of a standard 6T SRAM 512×256 bit-cell array (BCA) with per-column multipliers and a cross-column adder. The two IMC blocks share four 6-b ADCs and implement f_3 and f_4 respectively; 2) a diagonal major MVM kernel (DM^{2}VM) to efficiently implement f_1, f_2, f_5 and f_6 in digital; and 3) a digital control block (CTRL) for timing synchronization. The CTRL block is synchronized with a 1 GHz main clock generated off-chip. The 1 GHz frequency was chosen primarily for design simplicity, i.e., to be able to employ a synchronous IMC architecture and be able to generate 1 ns pulses required for execution of its various stages of computation. Note that, as established in Section I, the infrequent nature of speech related events implies that the 1 GHz clock can be gated during the long and frequent inactivity periods of the KWS IC. All internal clocks are generated from the main clock. The IC can be configured to perform 2–to–10 way classification.

C. IMC Block

The proposed multi-bit IMC architecture (Fig. 9) has two modes of operations: 1) a regular SRAM R/W mode and 2) a DP mode in which it can implement sparse dot-products between a buffered input vector and the weights stored in the BCA as described in Section II. The BCA employs a

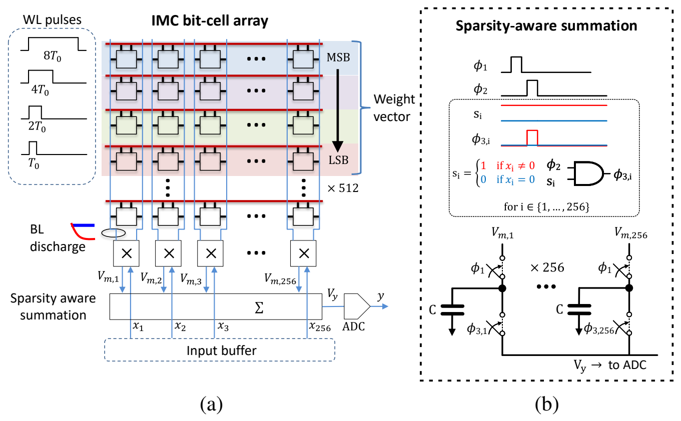


Fig. 9. The multi-bit IMC bank: (a) architecture, and the (b) sparsity-aware summation scheme.

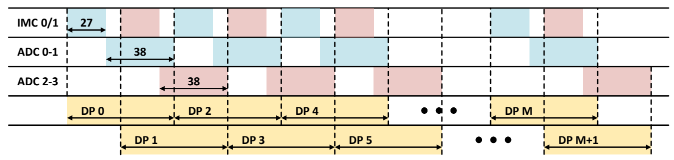


Fig. 10. Timing diagram for realizing an MVM on the IMC.

standard 6T SRAM bitcell designed with logic rules occupying $1.93\,\mu\text{m}^2$. In the DP mode, the IMC can access four ($B_w = 4$) consecutive rows simultaneously via the WL drivers. The resultant voltage drop on each bitline (ΔV_{BL}) is proportional to the 4b digital weight stored in that column. By tuning both the wordline voltage V_{WL} and the minimum pulse width T_0, we can control the maximum bitline discharge voltage $\Delta V_{\text{BL-max}}$. In this work, we fix $T_0 = 250\,\text{ps}$, and vary V_{WL} between $0.5\,\text{V}$ and $0.8\,\text{V}$, which translates to a maximum bitline discharge voltage between $0.32\,\text{V}$ and $0.54\,\text{V}$. The energy consumption of the IMC is a strong function of $\Delta V_{\text{BL-max}}$ and thus V_{WL}. A small $\Delta V_{\text{BL-max}}$ implies more energy efficiency, at the expense of compute accuracy, as demonstrated in [14]. The presence of PVT variations prohibits the use of extremely small $\Delta V_{\text{BL-max}}$ values, and thus the value of V_{WL} must to be chosen with care. In this work we find that a $V_{\text{WL}} = 0.7\,\text{V}$ is sufficient for maintaining proper operation.

The IMC can realize dot-products between signed weights and unsigned inputs. In each column, the sign of the weight is computed by comparing the BL and BLB voltages, and the absolute value of the weight ($\Delta V_{\text{BL-abs}} = \min(\Delta V_{\text{BL}}, \Delta V_{\text{BLB}})$) is multiplied with the corresponding digital input via a column pitch-matched charge re-distribution-based multiplier (similar to that of [14]). The IMC requires two separate voltage sharing rails that compute the partial dot-products using the positive weights and the negative weights separately. Therefore, at the end of each DP operation, two output voltages V_p and V_n need to be sampled and converted to digital. This is done via two separate ADCs, one for each rail voltage. The IMC design can be easily modified to support signed inputs by computing the partial product sign instead of the weight sign via a simple XOR gate between the weight sign and the input sign signals in each column. The partial product sign can be used to decide which voltage sharing rail to use, and thereby realize a fully signed DP operation efficiently.

In order for the ADC delay not to bottleneck the DP operation, we ping-pong between two pairs of ADCs in every read cycle. The CTRL block sequences the DP operations in order to realize the full MVM. Figure 10 details the timing information in terms of number of cycles required by each stage in the DP operation, and the ping-pong ADC schedule. The ADC outputs are scaled and shifted to calibrate for offsets followed by a ReLU non-linearity, and are subsequently routed to the input buffers.

The support for input-sparse dot-products is crucial for proper operation while realizing neural networks. This is due to the use of the ReLU activation function, which increases the input activations sparsity to $\sim 50\% - 70\%$. Sparse input vectors present a challenge for the charge sharing summation scheme used in [13] as the multiplier output voltage spread shrinks. A *sparsity-aware summing* (SAS) method (Fig. 9(b)) is proposed in which the per-column multiplier output voltages ($V_{m,i}$) are selectively charge shared based on whether the corresponding input element is zero or not. Th SAS scheme operates in two stages and is synchronized by the CTRL block via the control signals ϕ_1 and ϕ_2. When ϕ_1 switches, all $V_{m,i}$'s are dumped onto per-column unit capacitors. Next, ϕ_2 switches to trigger the summation process. In every column, signal $\phi_{3,i}$ is derived from ϕ_2 to select whether or not $V_{m,i}$ should be dumped onto the charge sharing rail. In the process, the output swing is preserved thereby improving the ADC accuracy. In order for this to work, the number of non-zero elements N_z in the N-dimensional input vector must be known to properly scale the post-ADC output. We compute N_z as the inputs are streamed in thereby incurring minimal latency overhead.

The effectiveness of the proposed SAS scheme is validated via numerical simulations. Ideally, we would like to compute the DP:

$$y_0 = \sum_{i=1}^{N} x_i w_i \tag{13}$$

where $x_i \in \{0, ..., 1 - 2^{-4}\}$ and $w_i \in \{-(1 - 2^{-3}), ..., 1 - 2^{-3}\}$ are the quantized inputs (unsigned) and weights (signed), respectively. The charge sharing summation is equivalent to an averaging operation, which results in the following computation of the DP:

$$y_1 = N \cdot Q\Big(\frac{1}{N} \sum_{i=1}^{N} x_i w_i\Big) \tag{14}$$

where $Q(.)$ is the B_y-bit ADC quantization function. In the presence of the SAS scheme, the averaging occurs over the non-zero partial products, which can be expressed as:

$$y_2 = N_z \cdot Q\Big(\frac{1}{N_z} \sum_{i=1}^{N} x_i w_i\Big) \tag{15}$$

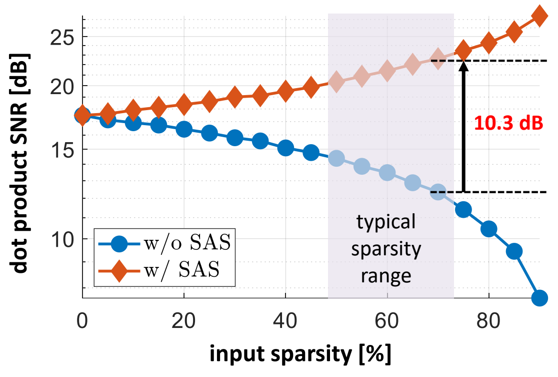


Fig. 11. The DP SNRs with and without the proposed SAS scheme vs. the input sparsity, using: $N = 128$, $B_x = B_w = 4$ and $B_y = 6$.

The signal-to-noise ratio (SNR) between the ideal output y_0 and the actual output y_j ($j \in \{1, 2\}$) is defined as follows:

$$\mathrm{SNR}_j = 10 \log_{10} \left(\frac{\sigma^2_{y_0}}{\sigma^2_{(y_0 - y_j)}} \right) \tag{16}$$

Figure 11 plots SNR_1 (without SAS) and SNR_2 (with SAS) against the input sparsity for randomly sampled weights and inputs. As expected, when the inputs are not very sparse ($<$ 20%), the impact of SAS is negligible. However, for the typical input sparsity range observed in KeyRAM, the SAS scheme improves the DP SNR by 5 dB-to-10 dB.

D. DM^{2}VM Block

The DM^{2}VM block is a synthesized digital co-processor operating using a 500 MHz clock consisting of: 1) a 96×512 standard 6T SRAM for parameter storage; 2) an array of 64 processing elements (PEs), each implementing an 8b$\times$8b multiply-accumulate (MAC) operation; 3) a local controller to manage the SRAM R/W functionality, synchronize operations with the main clock and manage data buffers. The block digitally computes all MVM operations in f_1, f_2, f_5 and f_6 with 8b inputs and 8b weights using three operating modes (f_5 and f_6 are fused together). The MVM dot-products are computed via the PE array, with weights fetched from the SRAM and inputs streamed in from a dedicated buffer. The 25-b accumulated dot-product outputs are truncated to 8b per algorithmic requirements. The DM^{2}VM processor is designed to minimize idle cycles when inputs/outputs are streamed in/out since the diagonal major architecture is able to complete an $N \times M$ MVM in a fixed number $N + M$ of cycles irrespective of whether $N > M$ or vice-versa. In contrast, an input (output) stationary architecture requires between $2N + M$ ($N + 2M$) and $N + M$ cycles, respectively, based on whether the input/output/both are streamed or not.

Figure 12 illustrates the principle of the DM^{2}VM kernel for a 4×4 FC layer. During the setup phase, the weight matrix is stored in the SRAM in diagonal major format. The DM^{2}VM begins computation as soon as the first input is streamed in and stops exactly when the final output is streamed out without any stalls. Each PE cycle consists of 8 clock cycles, involving fetching one weight diagonal from the SRAM and computing the partial sums using the shifted inputs. Once all the inputs are shifted in, the shift register flips direction and starts streaming out the completed output sums. The bias vector is pre-fetched in a dedicated buffer, and each element is added to the final output term once its streamed out. This makes the DM^{2}VM well-matched to the diverse set of MVM dimensions utilized by the KeyRAM algorithm.

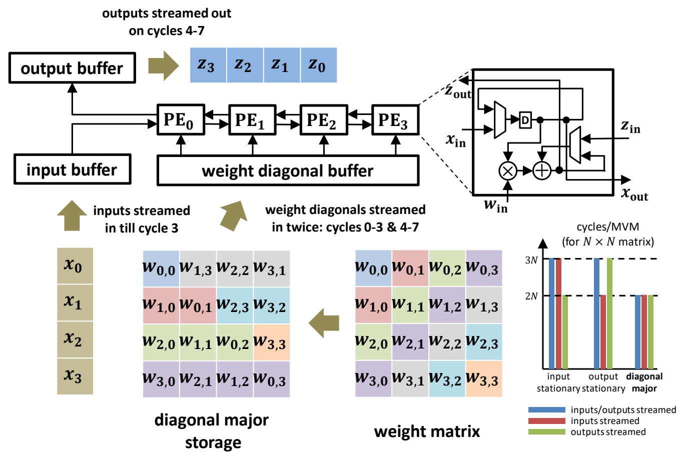
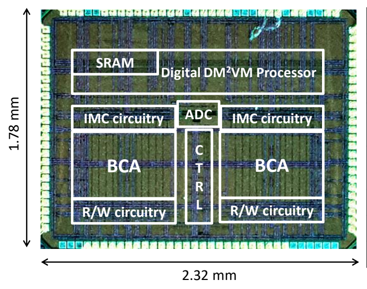


Fig. 12. The principle of the DM^{2}VM block for a 4×4 FC layer.



Technology	65nm
Die Size	1.78mm × 2.32 mm
Memory Capacity	38kB
Nominal Supply	1.0V
CTRL Frequency	1GHz
Latency	39.9μs
Energy/dec	0.44μJ
Algorithm	RAM

Fig. 13. Die micrograph and chip summary.

V. Experimental Results

This section describes the measured results from the prototype IC, and compares its performance with existing KWS ICs. The 65 nm CMOS IC is packaged in a 100-pin QFN. Figure 13 shows the die micrograph of the 65nm CMOS IC along with its summary.

A. Setup

Measurements on the IC are performed using the Google Speech dataset [8]. The KeyRAM architecture from Section IV was trained for 7-way classification using 11 k data samples each corresponding to a 1 s keyword sampled at 16 kHz. The inputs to the classifier on the chip are 8-channel MFCCs extracted within 40 ms windows with 20 ms overlap resulting in a 8×49-dimensional feature vector ($K = 8$, $T = 49$). The input patch dimension at each glimpse is 8×8 ($P = 8$) and the locator is a one-dimensional scalar. Figure 14 shows the

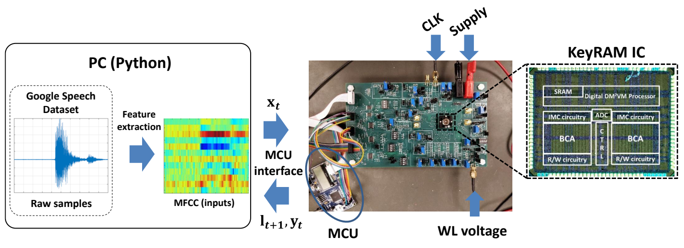


Fig. 14. Measurement setup showing data transfer between the PC to the KeyRAM IC via a microcontroller (MCU).

TABLE III
MEASURED CLASSIFIER ACCURACY ACROSS GLIMPSES.

Glimpses (n_G)	1	2	3	4
Accuracy [%]	60.58	72.12	83.65	90.38

measurement setup used for testing the KWS IC. MFCC feature extraction of raw audio samples from the Google Speech dataset is performed on the PC. The microcontroller (MCU) transfers the MFCC glimpses from the PC to the IC, and the location and class scores from the IC to the PC. All KeyRAM parameters are stored on-chip. The total I/O requirements is 3.4 Mbps when operating at maximum throughput.

B. Decision Accuracy, Energy, and Latency

Figure 15 shows how the confidence of the classifier IC improves with the number of glimpses processed when the input sample corresponds to the label 'off'. After two glimpses, the IC is unable to recognize the correct label, but with low confidence. By processing an extra glimpse, the IC makes the correct decision by assigning the label 'off' to the sample, and it does so with high confidence. Table III indicates that the measured classification accuracy of KeyRAM improves with the number of glimpses n_G used per decision. Processing more glimpses makes the classifier more confident in its decisions, and hence improves its accuracy. When averaged over the test set, the measured classification accuracy is 90.38% at $V_{\mathrm{WL}} = 0.7\,\mathrm{V}$ with $n_G = 4$ glimpses.

The decision energy and latency linearly increase with n_G (Fig. 16). Combined with a tunable WL pulse amplitude V_{WL},

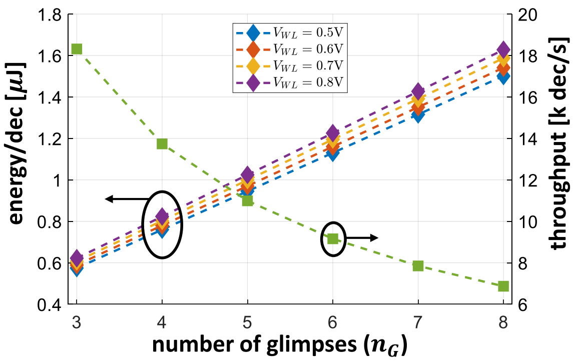


Fig. 16. Measured decision throughput and energy for varying V_{WL} and the number of glimpses n_G showing KeyRAM's dynamic energy-throughput trade-off.

the KeyRAM IC incorporates dynamic energy-accuracy trade-offs, e.g., the energy/glimpse varies from $0.19\,\mu$J-to-$0.21\,\mu$J as V_{WL} varies from 0.5 V-to-0.8 V. The measured energy breakdown (Fig. 17) shows that layers f_3 and f_4, which account for 89% of computations, consume 68% of the total energy consumption. These savings are attributed to the use of IMC. The CTRL energy ($0.08\,\mu$J/glimpse) gets amortized as the problem increases.

C. KeyRAM with CC Strategy

In Fig. 15, we notice that the classifier need not process the 4th glimpse as the confidence after the 3rd glimpse was very high. This provides some evidence that the CC strategy presented in Section III can be useful to minimize the energy. In this section, we investigate the efficacy of using this strategy.

Figure 18 plots the measured classification error and average energy per decision of KeyRAM with CC (KeyRAM-CC) while varying the threshold parameter τ. The accuracy-energy trade-off is evident, as the average energy per-decision increases with τ, for more glimpses per input sample n_G are required on average, while the classification error generally decreases. For $\tau \geq 0.5$, the test error plateaus around 9.62%, which is the test error associated with a constant glimpse strategy ($n_G = N_G$)

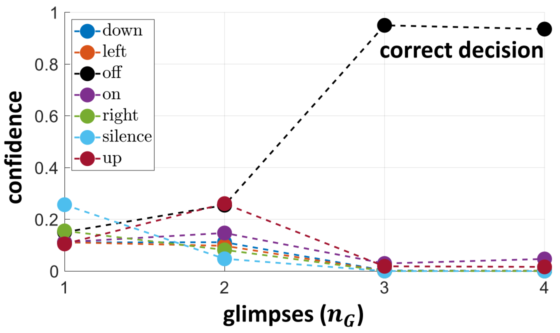


Fig. 15. Measured classifier confidence across glimpses for one input sample when the true class label is 'off', using $V_{\mathrm{WL}} = 0.7\,\mathrm{V}$.

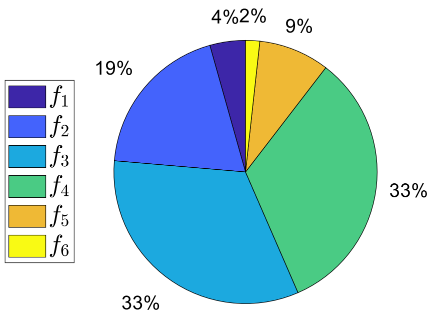


Fig. 17. Measured energy breakdown of the KeyRAM IC at $V_{\mathrm{WL}} = 0.7\,\mathrm{V}$ and $n_G = 4$.

TABLE IV
COMPARISON TABLE.

	[3]	[4]	[5]	[6]	[7]	This Work
Technology	65 nm	65 nm	65 nm	65 nm	28 nm	65 nm
Algorithm	DNN	LSTM	LSTM	Binarized-RNN	DSCNN	RAM
Dataset	TIDIGITS	TIMIT	TIMIT	Google Speech	Google Speech	Google Speech
# of Classes	11	39	4[a]	10	2	7
Test Accuracy [%]	98.35	80.4	–	90.2	94.6	90.38
On-chip Storage [kB]	747.52	82	32	18	2	38
Area [mm^2]	9.61	1.57	1.035	6.2	0.23	4.13
Energy/Decision [μJ]	6.4[c]	9.54[c]	0.06	3.36	0.03[c]	0.44
Decision Latency [μs]	37000[c]	770[c]	12000[c]	130	64000[b]	39.9
# of MACs/Decision	–	–	5.8 k – 27.2 k	–	151.4 k	200 k
# of Parameters	–	–	–	–	2.18 k	91 k
Energy-Delay Product [pJ.s]	239 k[c]	7.3 k[c]	720	430	1.92 k[c]	17.56
Supply Voltage [V]	0.6 – 1.2	0.75 – 1.24	0.575	0.9 – 1.1	0.41	1
Energy Efficiency [TOPS/W]	–	3.08	–	11.7	–	0.91

[a]4 binary classifiers [b]includes MFCC calculation [c]estimated from reported data

with $n_G = 4$. Therefore, KeyRAM-CC achieves an accuracy equal to that achieved by the constant glimpse strategy while consuming $\sim 2\times$ less energy per decision on average.

Comparison with a digital architecture implementing a standard RNN with the same model size, Figure 19 shows a $4.1\times$ savings in decision energy of which approximately $1.48\times$ is attributed to the use of the KeyRAM algorithm using constant glimpse strategy with $n_G = 4$ and $2.74\times$ is due to IMC. An additional $1.81\times$ savings is achieved on average by employing the CC strategy ($\tau = 0.5$) for a total of $7.3\times$ savings in decision energy.

D. Comparison with Existing KWS ICs

Table IV compares the KeyRAM IC with state-of-the-art digital [3]–[5], [7] and in-memory [6] KWS IC implementations. Table IV shows that the proposed KeyRAM IC requires more MACs per classification compared to current KWS ICs in [5], [7]. The reason being that the KeyRAM IC processes a higher number of keywords (7) on a more challenging Google speech dataset as compared to [5] which considers 4 keywords from the simpler TIMIT dataset, and [7] which processes only 2 keywords from the Google speech dataset. KeyRAM, using the CC strategy with $\tau = 0.5$, achieves $7.6\times$ reduction in energy/decision compared to the IMC [6]. In addition, KeyRAM achieves $> 24\times$ reduction in the decision energy-delay product (EDP) compared to other KWS implementations. It also achieves the lowest reported decision latency of 39.9 μs.

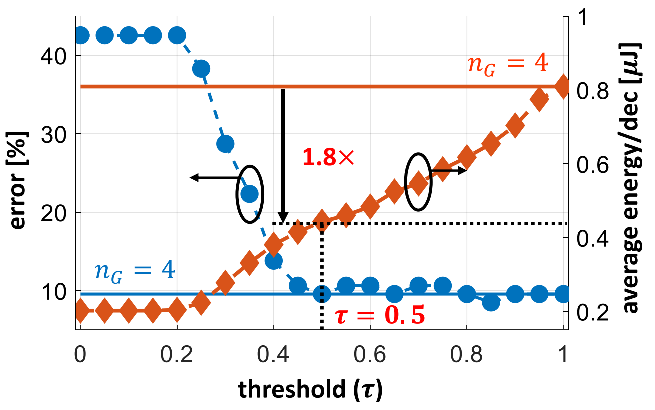


Fig. 18. Measured classification error and average energy per decision of KeyRAM-CC for different threshold values, using $V_{\text{WL}} = 0.7$ V.

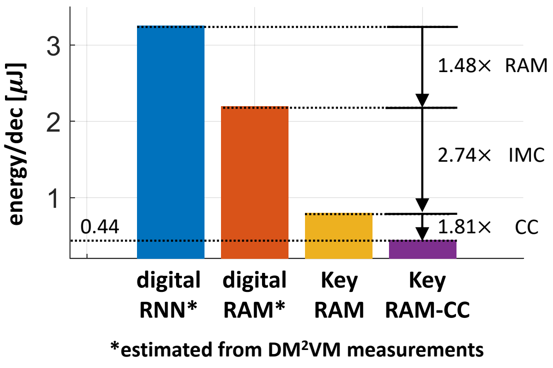


Fig. 19. Energy savings of KeyRAM compared to an all-digital implementation of RAM and RNN at iso-model size, with $n_G = 4$ and $V_{\text{WL}} = 0.7$ V.

VI. CONCLUSION

This paper illustrates the importance of algorithm-hardware co-design approach in minimizing the energy consumption and latency of decision-making systems in silicon. Specifically, adapting the RAM algorithm [11] from image classification to the KWS task provided $\sim 2\times$ reduction in energy-per-decision with no loss in accuracy and enabled on-chip storage of all weights. Combined with selective mapping of RAM layers to IMC [13] enabled this KWS IC to achieve a total of $> 24\times$ reduction in the decision-level EDP over SOTA. Such algorithm-hardware co-design approaches will be critical in other applications such as video inference as circuit and system designers seek to pervasively deploy AI-enabled semiconductors.

ACKNOWLEDGMENT

This work was sponsored by the AFRL and DARPA under agreement FA8650-18-2-7866 as part of the FRANC program and a grant from Sandia National Laboratories.

References

[1] I. McGraw, R. Prabhavalkar, R. Alvarez, M. G. Arenas, K. Rao, D. Rybach, O. Alsharif, H. Sak, A. Gruenstein, F. Beaufays *et al.*, "Personalized speech recognition on mobile devices," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 5955–5959. 1

[2] Y. Zhang, N. Suda, L. Lai, and V. Chandra, "Hello edge: Keyword spotting on microcontrollers," *arXiv preprint arXiv:1711.07128*, 2017. 1, 4

[3] M. Price, J. Glass, and A. P. Chandrakasan, "A scalable speech recognizer with deep-neural-network acoustic models and voice-activated power gating," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2017, pp. 244–245. 1, 9

[4] F. Conti, L. Cavigelli, G. Paulin, I. Susmelj, and L. Benini, "Chipmunk: A systolically scalable 0.9 mm 2, 3.08 GOP/s/mW@ 1.2 mW accelerator for near-sensor recurrent neural network inference," in *IEEE Custom Integrated Circuits Conference (CICC)*, 2018, pp. 1–4. 1, 9

[5] J. S. Giraldo and M. Verhelst, "Laika: A 5uW programmable LSTM accelerator for always-on keyword spotting in 65nm CMOS," in *IEEE 44th European Solid State Circuits Conference (ESSCIRC)*, 2018, pp. 166–169. 1, 9

[6] R. Guo, Y. Liu, S. Zheng, S.-Y. Wu, P. Ouyang, W.-S. Khwa, X. Chen, J.-J. Chen, X. Li, L. Liu *et al.*, "A 5.1 pJ/neuron 127.3 us/inference RNN-based speech recognition processor using 16 computing-in-memory SRAM macros in 65nm CMOS," in *Symposium on VLSI Circuits*. IEEE, 2019, pp. C120–C121. 1, 9

[7] W. Shan, M. Yang, J. Xu, Y. Lu, S. Zhang, T. Wang, J. Yang, L. Shi, and M. Seok, "14.1 a 510nw 0.41 v 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)*. IEEE, 2020, pp. 230–232. 1, 9

[8] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *arXiv preprint arXiv:1804.03209*, 2018. 1, 4, 7

[9] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017. 1

[10] S. Oh, M. Cho, Z. Shi, J. Lim, Y. Kim, S. Jeong, Y. Chen, R. Rothe, D. Blaauw, H.-S. Kim *et al.*, "An acoustic signal processing chip with 142-nw voice activity detection using mixer-based sequential frequency scanning and neural network classification," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 11, pp. 3005–3016, 2019. 1

[11] V. Mnih, N. Heess, A. Graves *et al.*, "Recurrent models of visual attention," in *Advances in Neural Information Processing Systems*, 2014, pp. 2204–2212. 2, 3, 9

[12] H. Dbouk, S. K. Gonugondla, C. Sakr, and N. R. Shanbhag, "KeyRAM: A 0.34 uJ/decision 18 k decisions/s recurrent attention in-memory processor for keyword spotting," in *2020 IEEE Custom Integrated Circuits Conference (CICC)*. IEEE, 2020, pp. 1–4. 2

[13] M. Kang, S. K. Gonugondla, A. Patil, and N. R. Shanbhag, "A multi-functional in-memory inference processor using a standard 6T SRAM array," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 2, pp. 642–655, 2018. 2, 5, 6, 9

[14] S. K. Gonugondla, M. Kang, and N. R. Shanbhag, "A variation-tolerant in-memory machine learning classifier via on-chip training," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 11, pp. 3163–3173, 2018. 2, 6

[15] J. Zhang, Z. Wang, and N. Verma, "In-memory computation of a machine-learning classifier in a standard 6T SRAM array," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 4, pp. 915–924, 2017. 2

[16] M. Kang, S. K. Gonugondla, S. Lim, and N. R. Shanbhag, "A 19.4-nj/decision, 364-k decisions/s, in-memory random forest multi-class inference accelerator," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 7, pp. 2126–2135, 2018. 2

[17] A. Biswas and A. P. Chandrakasan, "Conv-sram: An energy-efficient SRAM with in-memory dot-product computation for low-power convolutional neural networks," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 1, pp. 217–230, 2018. 2

[18] H. Valavi, P. J. Ramadge, E. Nestler, and N. Verma, "A 64-tile 2.4-Mb in-memory-computing cnn accelerator employing charge-domain compute," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 6, pp. 1789–1799, 2019. 2

[19] J.-W. Su, X. Si, Y.-C. Chou, T.-W. Chang, W.-H. Huang, Y.-N. Tu, R. Liu, T.-W. Lu, Pei-Jungand Liu, J.-H. Wang, Z. Zhang, H. Jiang, S. Huang, C.-C. Lo, R.-S. Liu, C.-C. Hsieh, K.-T. Tang, S.-S. Sheu, S.-H. Li, H.-Y. Lee, S.-C. Chang, S. Yu, and M.-F. Chang, "A 28nm 64Kb inference-training two-way transpose multibit 6T SRAM compute-in-memory macro for AI edge chips," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2020, pp. 240–241. 2

[20] J. Yue, Z. Yuan, X. Feng, Y. He, Z. Zhang, X. Si, R. Liu, M.-F. Chang, X. Li, H. Yang, and Y. Liu, "A 65nm computing-in-memory-based CNN processor with 2.9-to-35.8TOPS/W system energy efficiency using dynamic-sparsity performance-scaling architecture and energy-efficient inter/intra-macro data reuse," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2020, pp. 234–235. 2

[21] Q. Dong, M. E. Sinangil, B. Erbagci, D. Sun, W.-S. Khwa, H.-J. Liao, Y. Wang, and J. Chang, "A 351 TOPS/W and 372.4 GOPS compute-in-memory SRAM macro in 7nm FinFET CMOS for machine learning applications," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2020, pp. 242–243. 2

[22] X. Si, Y.-N. Tu, W.-H. Huang, J.-W. Su, P.-J. Lu, J.-H. Wang, T.-W. Liu, S.-Y. Wu, R. Liu, Y.-C. Chou, Z. Zhang, S.-H. Sie, W.-C. Wei, Y.-C. Lo, T.-H. Wen, T.-H. Hsu, Y.-K. Chen, W. Shih, C.-C. Lo, R.-S. Liu, C.-C. Hsieh, K.-T. Tang, N.-C. Lien, W.-C. Shih, Y. He, Q. Li, and M.-F. Chang, "A 28nm 64Kb 6T SRAM computing-in- memory macro with 8b MAC operation for AI edge chips," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2020, pp. 246–247. 2

[23] M. Kang, M.-S. Keel, N. R. Shanbhag, S. Eilert, and K. Curewitz, "An energy-efficient VLSI architecture for pattern recognition via deep embedding of computation in SRAM," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 8326–8330. 2

[24] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Eleventh annual conference of the international speech communication association*, 2010. 2

[25] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986. 3

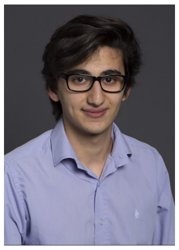
[26] P. Mermelstein, "Distance measures for speech recognition, psychological and instrumental," *Pattern recognition and artificial intelligence*, vol. 116, pp. 374–388, 1976. 4

[27] C. Sakr and N. Shanbhag, "An analytical method to determine minimum per-layer precision of deep neural networks," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 1090–1094. 5

[28] C. Sakr and N. R. Shanbhag, "Per-tensor fixed-point quantization of the back-propagation algorithm," in *7th International Conference on Learning Representations, ICLR 2019*, 2019. 5

[29] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European conference on computer vision*. Springer, 2016, pp. 525–542. 5

[30] D. Zhang, J. Yang, D. Ye, and G. Hua, "Lq-nets: Learned quantization for highly accurate and compact deep neural networks," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 365–382. 5



Hassan Dbouk received the B.E. degree with High Distinction in 2017 from the Department of Electrical and Computer Engineering at the American University of Beirut, Lebanon. He received his M.S. degree in 2019 from the Department of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign, where he is currently pursuing his Ph.D. degree. His research interests lie at the intersection of machine learning, circuits, and computer architecture. He is the recipient of the Rambus fellowship from the ECE department at the University of Illinois in 2020-2021.

Sujan K. Gonugondla received the Bachelor's and Master's in Technology degrees in Electrical Engineering from the Indian Institute of Technology Madras, Chennai, India, in 2014 and the Ph.D. degree in Electrical and Computer Engineering from the University of Illinois at Urbana-Champaign, Urbana, IL, USA in 2020. Since June 2020, he has been with Amazon, where he works as a Research Scientist. His research interests are in energy-efficient integrated circuits, and low complexity algorithms for machine learning systems, specifically algorithm hardware co-design for inference under resource-constraints.
Sujan K. Gonugondla is a recipient of the Dr. Ok Kyun Kim Fellowship 2018-19 and the M. E. Van Valkenburg Graduate Research Award 2019-20 from the ECE department at the University of Illinois at Urbana-Champaign, the ADI Outstanding Student Designer Award 2018 and the SSCS Predoctoral Achievement award in 2020. He has received Best Student Paper Awards in International Conference on Acoustics, Speech and Signal Processing (ICASSP) in 2016, and International conference in Circuits and Systems (ISCAS) in 2018.

Charbel Sakr is a PhD student at the University of Illinois working with Professor Naresh Shanbhag in the Coordinate Sciences Laboratory. He obtained his Engineering degree from the American University of Beirut in 2015 graduating with High Distinction. He then joined the University of Illinois and obtained his Masters degree in 2017 from the Electrical and Computer Engineering Department where he is now a PhD candidate. His research interests are in resource-constrained machine learning, with a focus on analysis and implementation of reduced precision algorithms and models.

Naresh R. Shanbhag received his B.Tech. (Indian Institute of Technology, New Delhi, 1988), M.S. (Wright State University, 1990), and his Ph.D. degree (University of Minnesota, 1993) all in Electrical Engineering. From 1993 to 1995, he worked at AT&T Bell Laboratories at Murray Hill where he was the lead chip architect for AT&T's 51.84 Mb/s transceiver chips over twisted-pair wiring for Asynchronous Transfer Mode (ATM)-LAN and very high-speed digital subscriber line (VDSL) chip-sets. Since August 1995, he is with the Department of Electrical and Computer Engineering, and the Coordinated Science Laboratory at the University of Illinois at Urbana-Champaign, where he is presently a Jack Kilby Professor of Electrical and Computer Engineering. He was a visiting faculty at the National Taiwan University (August 2007-December 2007) and Stanford University (August 2014-December 2014). His research interests are in the design of robust and energy-efficient integrated circuits and systems for communications including VLSI architectures for error-control coding, and equalization, noise-tolerant integrated circuit design, error-resilient architectures and systems, and system-assisted mixed-signal design. He has more than 200 publications in this area and holds twelve US patents. He is also a co-author of the research monograph Pipelined Adaptive Digital Filters published by Kluwer Academic Publishers in 1994.
Dr. Shanbhag received the 2010 Richard Newton GSRC Industrial Impact Award, became an IEEE Fellow in 2006, received the 2006 IEEE Journal of Solid-State Circuits Best Paper Award, the 2001 IEEE Transactions on VLSI Best Paper Award, the 1999 IEEE Leon K. Kirchmayer Best Paper Award, the 1999 Xerox Faculty Award, the Distinguished Lecturership from the IEEE Circuits and Systems Society in 1997, the National Science Foundation CAREER Award in 1996, and the 1994 Darlington Best Paper Award from the IEEE Circuits and Systems Society. Dr. Shanbhag is serving as an Associate Editor for the IEEE Journal on Exploratory Solid-State Computation Devices and Circuits (2014-16), served as an Associate Editor for the IEEE Transaction on Circuits and Systems: Part II (97-99) and the IEEE Transactions on VLSI (99-02 and 09-11), respectively. He has served as the General Chair of the IEEE Workshop on Signal Processing Systems (2013), and the IEEE International Symposium on Low-Power Design (ISLPED 2012), the Technical Program co-Chair of the 2010 ISLPED, and served on the technical program committee of a number of conferences including the International Solid-State Circuits Conference (ISSCC, 2007-11).
Dr. Shanbhag led the Alternative Computational Models in the Post-Si Era research theme, in the DOD and Semiconductor Research Corporation (SRC) sponsored Microelectronics Advanced Research Corporation (MARCO) center under their Focus Center Research Program (FCRP) from 2006-12. Since January 2013, he is the founding Director of the Systems On Nanoscale Information fabriCs (SONIC) Center, a 5-year multi-university center funded by DARPA and SRC under the STARnet phase of FCRP. In 2000, Dr. Shanbhag co-founded and served as the Chief Technology Officer of Intersymbol Communications, Inc., a venture-funded fabless semiconductor start-up that provides DSP-enhanced mixed-signal ICs for electronic dispersion compensation of OC-192 optical links. In 2007, Intersymbol Communications, Inc., was acquired by Finisar Corporation, Inc..