HYBRID ACCELERATOR TO OVERCOME IMPERFECTIONS OF MIXED-SIGNAL DNN ACCELERATORS USING ALGORITHM-HARDWARE CO-DESIGN

A Dissertation
Presented to
The Academic Faculty

By

Payman Behnam

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Engineering
Department of Electrical & Computer Engineering

Georgia Institute of Technology

May 2023

HYBRID ACCELERATOR TO OVERCOME IMPERFECTIONS OF MIXED-SIGNAL DNN ACCELERATORS USING ALGORITHM-HARDWARE CO-DESIGN

Thesis committee:

Dr. Saibal Mukhopadhyay
Department of Electrical &
Computer Engineering
Georgia Institute of Technology

Dr. Tushar Krishna
Department of Electrical &
Computer Engineering
Georgia Institute of Technology

Dr. Alexey Tumanov
College of Computing
Georgia Institute of Technology

Date approved: April 21, 2023

Dedicated to my beloved parents

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The objective of the proposed research is to increase the robustness of mixed-signal accelerators with an energy-efficient mechanism. In recent years, PIM-based mixed-signal accelerators have been proposed as energy- and area-efficient solutions with ultra-high throughput to accelerate DNN computations. However, PIM designs are sensitive to imperfections such as noise, weight/conductance variations, and cell programming errors that substantially degrade the DNN accuracy. To address this issue, we propose a novel algorithm-hardware co-design framework called ***Harmonica*** that simultaneously avoids accuracy degradation due to imperfections, improves area utilization and execution time, and reduces energy consumption. ***Harmonica*** proposes to select imperfection-sensitive weights using an input channel-wise method and transfer them to a novel and robust digital accelerator while the main computations are performed in the analog PIM cores. ***Harmonica*** is adapted to leverage the preceding weight selection method by reducing ADC precision, employing smaller peripheral circuitry, and a hybrid quantization to optimize the design. Our comprehensive experiments show that even in the presence of imperfections as high as 50%, ***Harmonica*** reduces the accuracy degradation from 60% - 90% in designs without a protection solution (e.g., in ISAAC or SRE baselines) to 1% - 2% for different DNNs across diverse datasets. In addition, compared to the ISAAC (SRE), ***Harmonica*** improves the execution time, energy, area, power, area-efficiency, and power-efficiency by 26% (14%), 52% (40%), 28% (28%), 57% (45%), 43% (7.5×), and 91% (7.3×), respectively. By employing architecture-based differential cells, where two separated categories of crossbars are used for positive and negative weights, ***Harmonica*** outperforms ISAAC (SRE) by 75% (9.2×) and 2.65× (10.2×) in terms of area- and power-efficiency.

CHAPTER 1
INTRODUCTION

We are approaching the end of Moore's and Denard's scaling laws, where the increase in transistor counts, core frequency, and performance per watt have become more difficult than ever. We need to adopt new solutions that efficiently utilize bounded transistor count in nanometer integrated circuits that lead to more energy-efficient systems [1]. The primary source of energy consumption in today's computer systems is data movement. As an example, the total energy required for fetching two operands from the main memory to a GPU is 500X higher than the energy required for performing addition in the GPU core [2, 3]. On the other hand, machine learning has an ever-increasing number of applications in our daily lives, for which deep learning provides state-of-the-art results. Their widespread applications include search engines, language translation, computer vision, speech recognition, navigation systems, augmented and virtual reality, and autonomous vehicles. [4, 5, 6]. The inherent parallelism of deep learning and computer vision algorithms cannot be exploited only by software mechanisms. In fact, the recent advancements in deep learning algorithms have been made possible thanks to the massive improvement in their associated hardware counterparts [7, 8, 9, 10, 3]. In order to ameliorate the performance and energy efficiency of DNN applications, a breadth of work has been done on developing specialized hardware accelerators. Among them, assorted digital CMOS-based accelerators have been recommended to provide high-performance computing units. However, these conventional von-Neumann architectures need a lot of back-and-forth data movement between computation and perhaps off-chip storage units [11, 8, 12]. In addition, the number of operations and likewise the amount of memory requirement for storing weights and feature maps in modern DNNs have increased dramatically [13]. In order to ameliorate the performance and energy efficiency of DNN applications, we need ultra high-performance and energy-efficient

hardware accelerators, which might not be feasible through conventional von-Neumann architectures. In recent years, processing in memory (PIM) and near memory (PNM) have been considered as fast and energy-efficient solutions to build Deep Neural Network (DNN) accelerators. Among them, ReRAM-based mixed-signal DNN accelerators have absorbed a great deal of interest due to their high density, throughput, and area-/power -efficiency [14, 15, 16, 17, 18, 19, 10, 20, 21, 22, 23]. Yet, they are still in the early stages of development since there are several fundamental challenges that need to be addressed properly to make them commercialized.

Firstly, these accelerators are very sensitive to imperfections such as stochastic process variations and programming errors due to their operations in the continuous analog regimes that degrade their accuracy significantly [24, 20, 25, 26, 21, 22, 27]. Secondly, since ReRAM cells are non-ideal devices, even if the cells are programmed to store the exact weight values, the current that flows on bitline can deviate from the actual current, that is supposed to be, especially in the architectures like ISAAC that exploit bias and offset subtraction for mapping the weights [9]. Furthermore, to have higher throughput, more wordlines are needed to be activated concurrently. By activating more wordlines, the amount of deviation in the accumulated current of the bitlines, which is the input of analog to digital converter (ADC), becomes larger. This leads to an overlap of neighboring states in ADC's output and hence erroneous digital results [20, 28]. In practice, without any protection mechanism, we cannot activate many wordlines (i.e, greater than 16) at the same time since it results in significant throughput degradation [29, 20, 28]. Thirdly, the employed ADCs in the mixed-signal accelerators consume significant area and power consumption (e.g., between 30% to 50% of the area, and between 50% to 70% of power consumption) [9, 10, 30, 31]. Several designs leverage smaller ADCs to perform the computations in lower-bit precision to save area and energy consumption at the cost of the undesired accuracy drop [22]. These properties make these mixed-signal DNN accelerators an unsuitable choice for practical applications.

Recently, several solutions have been proposed to make these accelerators more robust [32, 33, 34, 27, 35, 36, 37, 38, 39, 40, 24]. For instance, Long et al. suggest retraining a DNN model mixed with a noise model [34]. However, retraining is a cost-prohibitive procedure and cannot capture all the non-idealities of real applications precisely. Feinberg et al. propose an error correction scheme using AN-code to restore the accuracy loss [27]. AN codes rectify errors in linear functions and only consider the random telegraph noise. Moreover, it adds hardware overheads, which makes it less scalable [38]. The closest prior work to our proposal is Individual Weight Selection (IWS) [32, 33]. IWS categorizes weights into two groups of sensitive and non-sensitive to imperfections using a Hessian matrix of the weights. The sensitive weights are processed by a centralized error-resilient digital accelerator while non-sensitive ones are processed by the analog accelerator. According to IWS, since non-sensitive weights run on analog cores, the errors that occur in the analog units will not impact the model accuracy significantly. Unfortunately, IWS does not consider

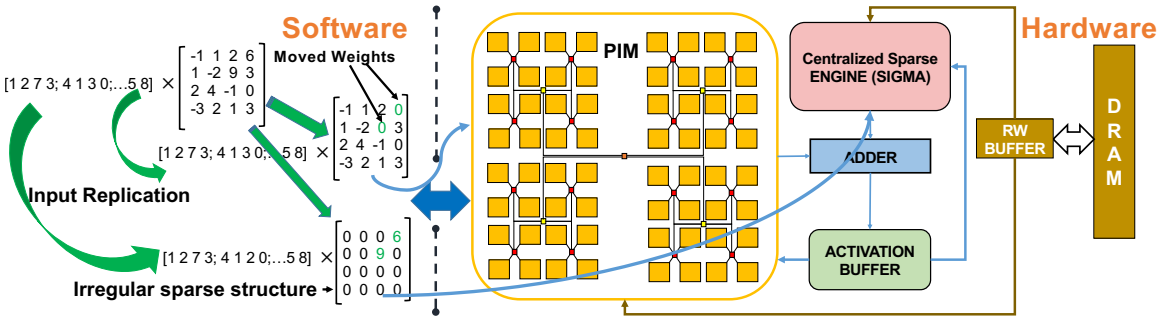

Figure 1.1: Illustration of the individual weight selection method (IWS) and its corresponding architecture [32, 33].

More specifically, IWS typically moves an irregular sparse set of weights to the digital accelerator. Therefore, the digital cores require the entire input channels for correct computation. Broadcasting the same input data to both digital and analog accelerators imposes a huge data movement between analog and digital accelerator units (see Fig. 1.1). This overhead is more intensified in the design considered by IWS, where a centralized sparse/irregular GEMM digital accelerator (i.e., SIGMA [41]) is employed to perform sparse computation. For instance, SIGMA exploits a configurable systolic array-based architecture with remarkable interconnection overheads (i.e., Table 5.6). Besides, having a

centralized digital accelerator means that for each layer, even if the analog unit is finished earlier, it needs to wait for a long time until the digital cores are finished or vice versa. IWS also recommends writing the weights of the next layers at the same ReRAMs crossbar when processing the current layer is ended. Writing the weights of all layers into the same crossbar is on the critical latency path and time-consuming. It also has significant energy consummation and endurance issues [42, 43, 44].

Hence, designing not only a reliable and variation-resilient but also an energy-efficient, high-performance mixed-signal accelerator is essential to make them practical for real-world applications. In this work, we propose ***Harmonica***, an algorithm-hardware co-design framework that not only addresses the aforementioned problems, but also opens more doors to optimize the area, power, and energy consumption of the PIM-based mixed-signal accelerators. To this end, ***Harmonica*** narrows down the selected sensitive weights to the filter associated with input channels. The selected input channels are the only part of the input needed by the digital accelerator. As a result, ***Harmonica*** **prevents input/activation replication of digital accelerator cores into analog ones**.

On the analog accelerator side, the channels are mapped to the rows of crossbars. Since sensitive weight channels are entirely selected to be placed in digital units, their corresponding rows in analog crossbar units are zeroed uniformly. This feature evenly reduces the value of aggregated partial sums over the bit lines. Hence, we can **employ low precision ADCs** that significantly reduce area and power consumption with a negligible accuracy loss [9, 31]. Moreover, channel-wise sensitive weight selection enables us to quantize the weights in the analog part with a lower precision compared to the digital cores **without any need to retraining/post-training** while maintaining minimal accuracy loss.

In summary, ***Harmonica*** brings the following contributions:

- In contrast with the previous works [34, 38], the proposed method **doesn't need any retraining**. This is important as training is a costly procedure and it may not capture all kinds of imperfections that may occur during run-time operation.

- Unlike IWS [32, 33], the proposed channel-wise weight selection method **avoids replicating the input activation** into both digital and analog cores and significantly reduces data movement and energy consumption.

- Unlike IWS [32, 33], the proposed solution **avoids forming irregular sparse weight sets**, as the whole weight channel along with its corresponding input activation are transferred to the digital units, which leads to having a simplified mechanism to process them.

- We observed that weight distribution in ***Harmonica*** allows us hybrid quantization (reducing the quantization precision on the analog side more compared to the digital side) with a negligible accuracy drop. This quantization is based on input channels' weight distillation per layer depending on the placement of input channels without a need for retaining or fine-tuning.

- Due to transferring sensitive weight channels to digital cores uniformly, ***Harmonica*** enables us to utilize **smaller ADC and peripheral circuitry**.

- Since ***Harmonica*** provides a robust design to the conductance variation, we are able to **activate more wordlines of the analog crossbar at the same time** with less bitline current deviation and accuracy degradation.

- Considering algorithm and hardware innovations, ***Harmonica*** can **further reduce the area, power, and energy consumption of mixed-signal accelerators**, where a fraction of the saved area can be utilized to add a **proposed digital accelerator** that is more robust against variation and protects the sensitive weights. Moreover, Harmonica proposes a novel data flow for the proposed architecture.

- ***Harmonica*** also accelerates computing sensitive weights compared to IWS by employing fewer Eigen pairs. ***Harmonica*** also utilizes binary search that reduces the search complexity from *O(n)* to *O(logn)*.

CHAPTER 2

HARMONICA MECHANISM

2.0.1 System Overview

Fig. 2.1 shows the Harmonica overview, which is designed to address the problems of existing solutions discussed in chapter 1. Compared with IWS [32, 33] (i.e., Fig. 1.1), ***Harmonica*** has several algorithm and hardware innovations. The algorithm novelties includes innovations in Hessian and sensitivity weight computation acceleration, channel-wise weight selection, and input channel-wise weight quantization. Hardware innovations include novelties in ADC resolution reduction, designing the digital accelerator, and data flow. We describe the innovations in algorithms and hardware in the following.

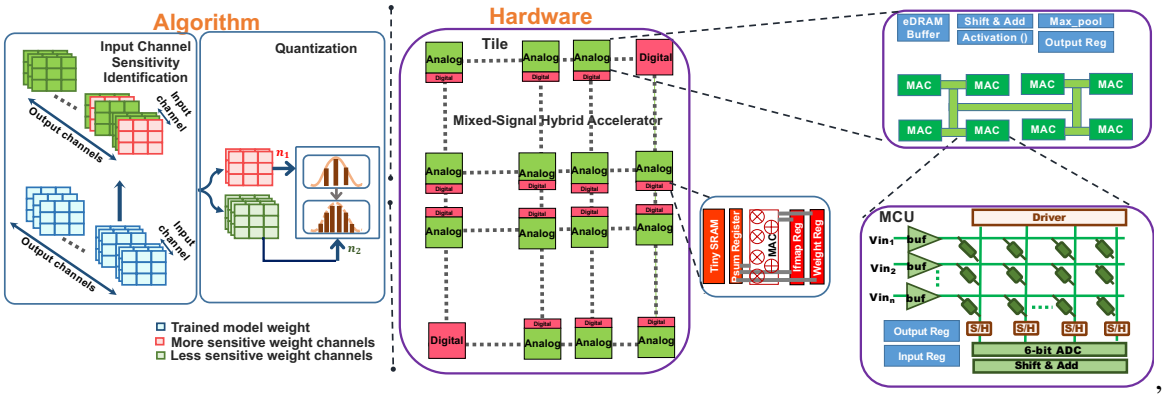

Figure 2.1: Illustration of the proposed input channel-wise weight selection.

2.0.2 Accelerating Weight Sensitivity Computation

In order to find important weights, we need to take into account the sensitivity of each parameter to the imperfections. This sensitivity can be calculated using either the gradient [45, 46] or the Hessian [47, 48, 33, 32] of the parameters. Current research shows Hessian-based solutions reach superior performance in terms of estimating the robustness of the parameters. The Hessian matrix describes the change rate of the elements of the gradient along the weight matrices. It indicates the amount of curvature of the loss manifold at a particular point. A

Hessian matrix's eigenvalues indicate how much curvature exists along the Eigenvector direction where higher curvature means more sensitivity to the imperfections [48, 32].

Similar to IWS, we avoid computing the full Hessian matrix as we need to compute only the m Hessian Eigen pairs. However, we argue that for the channel-wise weight selection, we need to compute fewer Eigen pairs as more sensitive weights are finally transferred to the digital accelerator cores. The intuition here is that as more weights are selected in channel-wise weight selection, the lack of precision due to less number of iterations is compensate. This increases computation speed.

In addition, to compute sensitive weights, which we describe in Sec. 2.0.3, unlike IWS, where in an iterative manner, after increasing every percentage of the protected weights, an inference is run to test the new accuracy, **Harmonica** utilizes binary search to apply inference and test the accuracy for selecting the most sensetive channels. This accelerates compilation time as it reduces the searching complexity from $O(n)$ to $O(logn)$.

2.0.3 Input Channel-wise Weight Selection

Generally, the input to a CNN model is a 4D tensor ($B \times H \times W \times C$), where B, H, W, and C represent the batch size, height, width, and the number of channels, respectively. Similarly, weight parameters consist of K number of 3D *kernel tensors* of dimension ($R \times R \times C$), where K is the number of output channels. R and C denote the height (and width), and depth of the kernel which is equal to the number of channels of the input activations. **Harmonica** proposes a channel-wise algorithm that extracts weights that are sensitive to imperfections. It then maps these weights to the digital accelerator cores, while the rest are in the analog accelerator ones.

Algorithm algorithm 1 and Fig. 2.1 show the proposed solution. **Harmonica** takes the already trained network parameters as input and generates a binary mask that assigns either 0 or 1 values to all the input channels of every kernel in the network, where 1 represents that channel needs to be protected. In order to compute this mask, we need to take into account the sensitivity of each parameter with respect to the imperfections. **Harmonica** calculates

the weight sensitivities using the equation Equation 2.1 as outlined in [33]:

$$s = \sum_{i=1}^{m} (|\lambda_i| q_i^2) \otimes w^2 \tag{2.1}$$

Where w is the weight parameter matrix and λ_i, q_i are the eigenvalue and the corresponding eigenvector of the Hessian of the parameter matrix, and m is the number of top-eigenpairs that are considered. After calculating the summation, we estimate s by taking the Hadamard product of the obtained *sum* and the weight matrix, w. By increasing the number of iterations (e.g., m), the precision of he sensitivity is increased. However, compared to IWS, m has a smaller value as we explained earlier.

Next, in order to reduce the redundant data-movement and avoid input data repetition, instead of considering the sensitivity values of each parameter independently as suggested in previous works [33, 32], **Harmonica** proposes input channel-wise aggregated form. If P and Q are number of input channels found to be more and less sensitive to variation, respectively, where Q = C – P , then we can say the number of parameters mapped to the digital and analog accelerators for this specific kernel will be $W_d \in R_{R\times R\times P\times K}$, and $W_a \in R_{R\times R\times Q\times K}$, respectively. The sensitivity of a particular input channel is calculated based on the aggregation of all the parameters sensitivity values along that dimension as equation Equation 2.2 shows:

$$s_i = \sum_{K} \left(\sum_{R} \left(\sum_{R} s \right) \right) \tag{2.2}$$

Where s_i is the aggregated sensitivity values for the W kernel. The proposed algorithm computes s_i value for all channels of all existing layers of the given DNN and sorts them according to their magnitude[1] The proposed algorithm utilizes binary search to find the channels that need to be processed with the digital accelerator. Initially, it starts its computation by assuming that the top 50% channels are mapped on the digital cores. Then, it applies variation to all of the weights and computes the new accuracy, $ACC_{calculated}$ until

[1]we tried different methods like maxing, averaging, mean square error, etc. Among them, channel aggregating has been found to provide better accuracy.

Algorithm 1 Input Channel-wise Weight Selection

Result: Analog and Digital mapping of the channels
Input: Trained weights, λ_i eigenvalues and v_i corresponding eigenvectors, $ACC_{desired}$
Calculate sensitivity of each parameter using: $s_w \equiv \left(\sum_{i=1}^{m} |\lambda_i| q_i^2\right) \odot w^2$ $SOL = [\,]$
for $i < num\_layers$ **do**
  for $j < num\_channels$ **do**
    SOL_{i_j} = channel wise aggregation of sensitivity values $SOL.append(SOL_{i_j})$
  end
end
SOL.**sort()** $Digital_{channel} =$ [top 50% of SOL] $Analog_{channel} =$ rest of SOL
Apply (50%, 10%) noise($Analog_{channel}$,$Digital_{channel}$)
$ACC_{calculated}$ = **Inference(**$Digital_{channel}$,$Analog_{channel}$**)**
while $ACC_{calculated} \approx ACC_{desired}$ **do**
  if $ACC_{calculated} > ACC_{desired}$ **then**
    $channel$ = Added to $Digital_{channel}/2$.**pop()** $Analog_{channel}$.**push**($channel$)
  end
  if $ACC_{calculated} < ACC_{desired}$ **then**
    $channel$ = Added to $Analog_{channel}/2$.**pop()** $Digital_{channel}$.**push**($channel$)
  end
  Apply $noise$ in $Analog_{channel}$ $ACC_{calculated}$=**Inference(**$Digital_{channel}$,$Analog_{channel}$**)**
end

$ACC_{calculated}$ is almost equal to $ACC_{desired}$ (e.g., less than 1% difference [49]), it selects the half of newly added top channel to digital (analog) of the sorted list of channel-wise sensitive values and puts them in the analog (digital) cores.Although computing the Hessian matrix and weight sensitivities happen in the compile time, our proposed optimizations help to run the whole procedure faster.

The output of this procedure would distinguish the channels that need to be mapped to the analog and digital cores and hence the corresponding channels of input data as shown in the left side of Fig. 2.1. Compared to the individual weight selection [32, 33], the proposed solution reduces extra data movement and avoids irregular and expensive sparsity.

2.0.4 Hybrid Quantization

Quantization is a well-known hardware-friendly solution to reduce computation efforts and increase energy efficiency. After the quantization, the original weights and/or activations are represented with lower precision as fixed-point or integer representation while the accuracy is preserved. If we represent the original floating-point tensor by x_f, its quantized by x_q, the

zero-point by zp_x, the scaling factor by s_x, and the number of bits used for quantization by n, we will have [50]:

$$x_q = round((x_f - minx_f)\frac{2^n - 1}{max_{x_f} - min_{x_f}}) = \\ round(x_f s_x - min_{x_f} s_x) = round(x_f s_x - zp_x) \tag{2.3}$$

As shown, the scaling factor is a parameter that depends on the desired bit-precision, n, and the range of tensor values ($max - min$). In a typical CNN, each output feature map for a particular layer is generated by convolving the input feature maps with a number of filters.

$$y = \sum_{0}^{M-1} w_m * x_m \tag{2.4}$$

where M is the number of input channels for the kernel of that layer and $*$ denotes convolution operation between input x_m and w_m as 2D tensors.

The linear quantization approximates the results as follows:

$$y_q = \text{round}(\frac{s_y}{s_x s_w}(\sum(x_q + zp_x)(w_q + zp_w))) \tag{2.5}$$

where $y, x, w,$ represent the output activation, input activation, and weight respectively. The other parameters that are s, q, z_p represent the scaling factor, quantized output, and the zero-point quantization offset, respectively.

In ***Harmonica***, the input channel with important weights are steered to digital cores while the rest of them are fed into analog ones. Since analog and digital cores do not share any weights, they can have a different number of quantization bits and different scaling factors. Although digital and analog cores do not share activations, we consider a similar quantization for all activations (i.e., 8-bit). The partial results of the analog and the digital

cores are only added once at the end for each layer:

$$y = y_d + y_a = \sum_{d=0}^{D-1} w_d * x_d + \sum_{a=0}^{A-1} w_a * x_a \tag{2.6}$$

where A and D represent the number of input channels computed inside analog and digital cores and $D+A=M$ is total number of input channels. **Harmonica** quantizes the weights in analog with n_1 bits and weights in digital with n_2 bits, where $n_2 > n_1$ as important channels are processed in digital cores.

To combine the analog and digital partial results, they need to be in the same scaling factor. So, before adding them together, partial results are converted back to floating points, as follows.

$$y_{fd} = \frac{s_y}{s_x s'_w}(\sum_{d=0}^{D-1}(x_{qd} + zp_x)(w'_{qd} + zp'_w)) \tag{2.7}$$

$$y_{fa} = \frac{s_y}{s_x s''_w}(\sum_{a=0}^{A-1}(x_{qa} + zp_x)(w''_{qa} + zp''_w)) \tag{2.8}$$

where s'_w and s''_w are digital and analog scaling factors, w'_{qd} and w''_{qd} are digital and analog quantized weights, and zp'_w and zp''_w are zero points in digital and analog domains. Instead of rounding y_{fd} and y_{fa} and then summing them up, **Harmonica** first adds the floating-point values and then applies the rounding operation in order to reduce the quantization error. Note that converting from quantized value to floating-point happens in all typical DNNs in the return path before feeding the results into the next layer [51, 9, 33]. Here an extra floating-point addition is required compared to the linear quantization described in equation Equation 2.5. The results are stored in the eDRAM buffer. We consider the area and power overhead of the quantization circuits in the results.

We use FP16 data type for y_{fd} and y_{fa} as prior work shows that FP16 accumulator is enough to preserve the accuracy during inference [52]. Our simulation results show that if the weights in the analog and digital accelerators are 6-bit (4-bit in the case of differential cells) and 8-bit respectively, there would be negligible accuracy loss compared with the case

that both of them are 8-bit.

CHAPTER 3

ARCHITECTURE BUILDING BLOCKS

3.0.1 Distribution of the Sensitive Weights

To have a better understanding of how the architecture of ***Harmonica*** should be designed, we need to have an understanding of distribution of the selected weights mapped to the digital accelerator among all the layers. Fig. 3.1 illustrates these results for ResNet18 over CIFAR10 data set. We observe the deviation of important weights per layer in ***Harmonica*** is less than the IWS method [32, 33].

As seen, for the first and the last layers, both IWS and ***Harmonica*** weight selection methods map a significant portion of the weights of these layers to digital cores. The reason is that these weights are usually more sensitive to variation [53, 54]. The number of weights in the first and last layers usually is not considerable compared to the other layers. For instance, for ResNet18 over CIFAR10, the number of parameters in the first and last layers are 1728 and 5120 respectively while for a middle layer (i.e., the seventeenth layer) the number of parameters is 2359296. Even recent studies show that the last linear layers can be compressed significantly (e.g., 49× for VGG16) [55, 56]. We observed that if we dedicate the first and the third last tiles to digital cores, it leads to better performance and energy

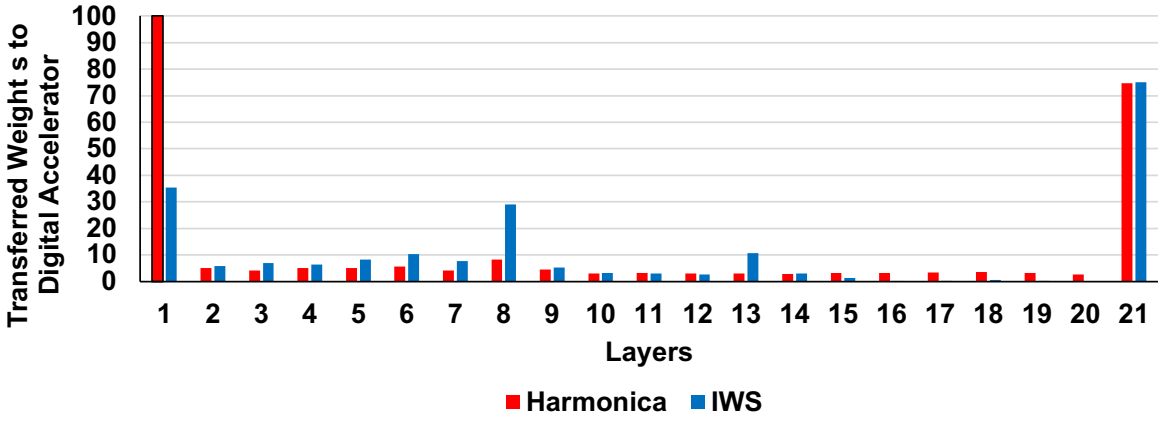

Figure 3.1: Comparing distribution of the selected important weights in ResNet18/CIFAR10 (***Harmonica*** vs. IWS [32])

Another feature of the proposed weight selection method is that the deviation among the percentage of protected weights per layer (except first and last layers that have dedicated tiles) in ***Harmonica*** is lesser than the IWS method. For example, in the case of ResNet18, the standard deviation of the percentage of protected weights per layer is 4.8× less than individual weight selection (1.37 vs 6.69). We found the same trend for all the evaluated DNNs over different data sets. That means that the proposed approach provides a more uniform weight-selection distribution rather than the IWS [32, 33]. This property simplifies the hardware of ***Harmonica*** and increases resource utilization.

3.0.2 Analog Accelerator Architecture

As the right side of Fig. 2.1 shows, the analog/digital units, called analog/digital tiles, are connected with an on-chip mesh fashion. Each analog tile is composed of several multiply-accumulate units (MACs), an eDRAM buffer to store input values, shift-and-add units that compute final results, output registers to store computed results, max-pooling and non-linearity activation units that are connected with a shared bus. MAC units include inverters, multiple crossbar subarrays, ADCs, and shift-and-add units.

In the proposed architecture, channels are mapped to the rows of crossbar arrays. Since sensitive input channels are mapped to the digital accelerator cores, those rows will be filled with zeros in the analog cores. [1]. On the other hand, ADCs are located per bitlines. Although only a small portion of the weights are mapped to the digital accelerator, they play crucial roles in the final accuracy. Removing rows with important weights causes the value of the accumulated current per column be less sensitive to imperfections.

Our experimental studies show that if we use ADCs with a lower precision per crossbars where none of the rows are removed, there would be a considerable accuracy drop. Zeroing out (a) row(s) will have a uniform impact on the whole bitlines of a crossbar and hence ADC resolution. Hence, it allows us to use ADCs with a lower resolution that enables a

[1] We can eliminate those zero-out rows. Removing those rows can also be combined with hybrid pruning. However, due to space limitation, we leave them to future work

considerable reduction in the area/power consumption [57]. For the IWS method [32, 33], we are not able to have the same ADC for all bitlines. The reason is that sensitive weights have a random distribution. As a result, the number of important weights may change dramatically by going from one bitline to another one.

Since partial sum values over the bitlines are reduced, we are now able to employ smaller sample-and-hold and shift-and-add units.

3.0.3 Digital Accelerator Architecture

The right side of Fig. 2.1 shows the building blocks of the digital accelerator. The digital units of the tiles are located face to face to reduce wire length communication. Unlike conventional accelerators like Eyeriss that has PEs one one side and large buffer/memories on another side of the chip, ***Harmonica*** distributes tiny SRAMs throughout the chip and put them besides computation units. It is important since the sensitive channels are distributed among the DNN layers that are mapped to the tiles. The proposed architecture is based on WAX [58] where each tile is tiny SRAM bank coupled with a MAC array , and three register units for activations, weights, and partial sums, respectively. In ***Harmonica***, we utilize short interconnections, where it connects small SRAMs. These small SRAMs have been distributed among tiles and bring high data reuse of both weights and activations.

In ***Harmonica***, we replace 6KB SRAM buffer of WAX [58] with 1KB buffer access (5.2$\times$ energy reduction). Like WAX architecture [58], we replace 12- and 24-entry register files and 224-byte scratchpad in Eyeriss with single register access.

Unlike WAX, these units are connected through a grid structure rather than H-tree. Using H-tree makes the distance between these two neighbors as bad as $log(chipwidth)$. Although far tiles can communicate together thorough the grid structure, each tile usually needs to access each local SRAM or its neighbors (e.g., unlike WAX or Eyeriss-like designs that need to access far-distant SRAMs). WAX architecture also needs additional muxing at each split point of its H-tree [59, 60]. These hierarchical muxing is essential so that data from a neighboring sub-array is guided either to the other adjacent sub-array or to a central

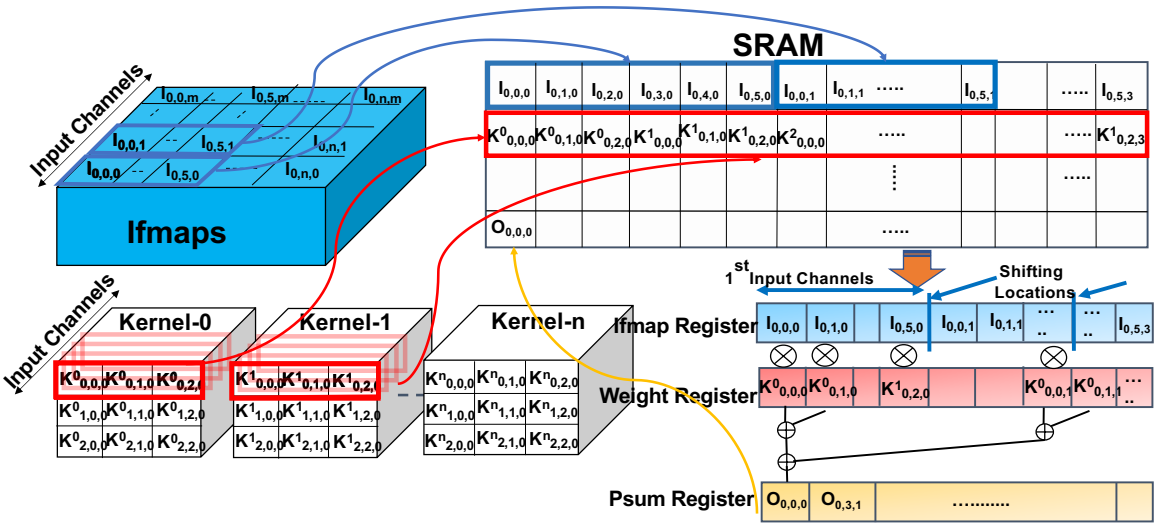

Figure 3.2: Data flow of the proposed digital accelerator.

complex controller. This complex controller unit is eliminated in our proposed architecture. This brings the advantages of more bandwidth, less area, and better energy efficiency as demonstrated in the previous works [61, 62]. Besides, unlike WAX, each SRAM has 1 row for activations, 24 rows for weights, and 7 rows for partial sums (6X size reduction). Additionally, the number of required units is almost 20% of WAX because only small portions of the weights are mapped to digital cores.

3.0.4 Large vs distributed small SRAMs

It has been shown that having small distributed memories leads to significant energy savings compared to the large memory due to less data traveling [58]. For instance, replacing 224-byte scratchpad access with single register access yields 46X energy reduction [58]. Large memories need more rows and columns that increase the size of muxes and decoders. In addition, more flip-flops share the same address, read, and write signals, which result in larger load and parasitic capacitance. Our results show that the access energy of a 54KB buffer is 9X greater than accessing a 1KB buffer.

3.0.5 ***Harmonica*** Mapping and Dataflow

Harmonica requires the following modification to the mapping and data flow of existing mixed-signal accelerators. First, the tile allocated to each layer is based on the maximum

number of required analog and digital cores. Second, the weight tensor is reshuffled such that the resulting output channels are clustered into analog and digital groups to simplify the data path. Finally, we propose novel dataflow in the digital cores and describe how the digital and analog cores' partial results are added up.

In ***Harmonica*** one or several tile(s) are programmed to store the weights of each layer, where the input channels of a layer are mapped to the row of crossbars or digital cores. The tiles are connected to each other in a pipelined manner.

In ***Harmonica***, digital channels are allocated at the end of each tensor. To achieve such arrangements, we rely on the fact that reordering filters in a layer only changes the order of channels in the resulting tensor. Thus, by mapping the filters associated with digital channels in the next layer to the end of the wight tensor, we guarantee digital channels come after analog ones. This simplifies reading corresponding input activation and routing them to either digital or analog cores in the next layer.

Fig. 3.2 shows the data flow of the digital accelerator. Superscript shows the kernel number and subscript for both kernel and activation shows the dimension (height, width, depth). As Fig. 3.2 shows, the SRAM is 24-bit wide. The first row of the SRAMs is filled with the input feature maps of different input channels (i.e., 6 consecutive inputs of 4 different input channels). A row of weights from the first two kernels is placed in the first row dedicated to the weights in SRAM (e.g., the second row of SRAM). The remaining 23 weight rows of the SRAM are filled using the weights of the same position of other input channels. (e.g., 3 successive weights of 4 input channels for kernel1 and the same weights for kernel2).

Once the inputs and weights are loaded from SRAM to registers and computation is started, loading of the next input feature maps is started (i.e., overlapping computation and communication). When the weights are loaded into the SRAM subarray, they remain there until all weights are fully exploited. Since we load weights and activations from different input channels (i.e. four different input channels), the corresponding activation and

weights registers are split into four parts where shifting for each channel happens only in its corresponding partition.

The multiplications happen using MAC units by multiplying weights and their corresponding activation. In the first cycle, 24 multiplications of corresponding weights and activation are performed. Each partial sum output is computed using adding four different outputs of multiplications using a 3-level adder tree. In 12 cycles, the 24 partial sum registers are populated. Then, the results are written into the first row of SRAM dedicated to partial sum results (e.g., row 26 of SRAM). The same flow continues for the remaining activations and weights. Finally, the results of the digital and analog portions for each layer are merged in the output register of each tile. We discuss more about the load balancing of ***Harmonica*** in Section Sec. 5.0.6.

CHAPTER 4

SIMULATION SETUP

We developed an in-house simulator to obtain the area, power, energy, and throughput of the ***Harmonica*** and the evaluated baselines. Fig. 4.1 shows the overall flow of the developed simulator with the main input/output of each component. ***Harmonica*** employs Pytorch API to get the Hessian matrix of pretrained weights and then the accuracy and weight distribution. The simulator utilizes NVSIM [63], Cacti [64], and PIM primitives library [65] as the backbones to perform design space exploration and gets required inputs of the architectural simulator. For the architectural simulation, we modify MNSIM[66] to model the hardware of analog units. The MNSIM is enriched with a digital simulator that mimics the proposed dataflow.

Previous works [34, 32] suggest that conductance variation of the individual devices in a PIM design can be modeled by a Gaussian distribution of noise with 0 mean and a standard deviation proportional to the stored parameter value. Eq. (Equation 4.1) shows the noise model considered in our work:

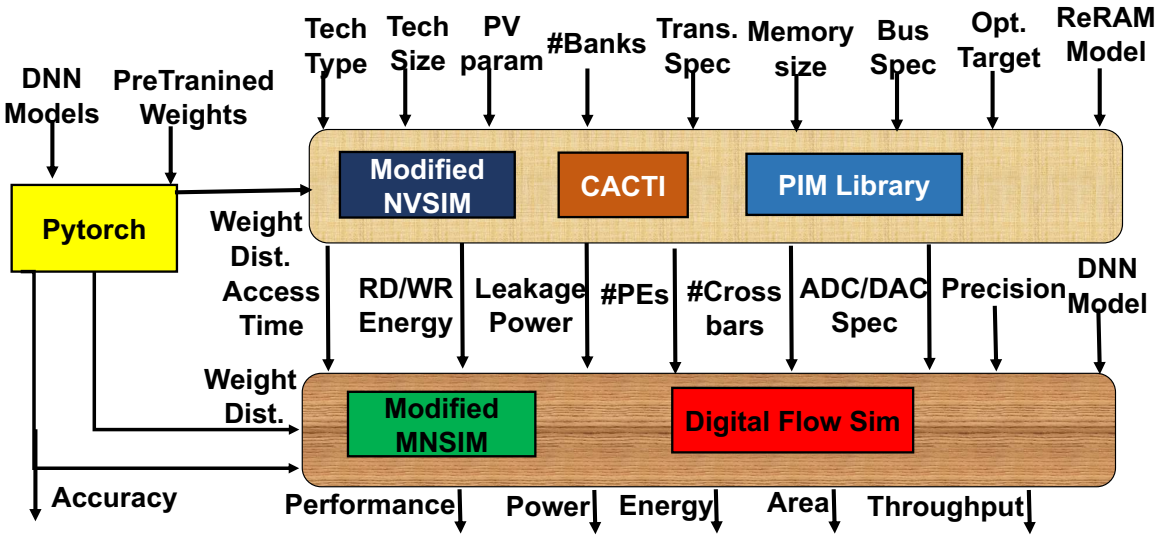

Figure 4.1: Overall view of the developed in-house tool.

We consider $\sigma = 50\%$ and $\sigma = 10\%$ for the weights in analog and digital cores, respectively. We employ the VTEAM ReRAM model [67] and follow the methodology of ISAAC to model max-pooling, shift-and-add, ADC, and activation functions. The power and area of the shift-and-add, max-pool, and ReLU are taken from ISAAC [9] and PRIME [10].

To get the power and area of ADC, we use the most updated dataset [68]. To extract the area and power of the same ADC but with a lower resolution, we scale down the power and area of the memory, clock, and *vref* buffer linearly, and the capacitive DAC exponentially as explained in [57]. We choose this methodology to model peripheral circuitry and make a fair comparison at 32nm with state-of-the-art works. We examine both offset subtraction-based designs [9, 28, 69], (i.e., ***Harmonica***) and differential cell-based designs [10, 70, 71] (i.e., ***HarmonicaDi***) to show ***Harmonica*** mechanism is independent of the underlying architecture.

We implement the proposed digital units and baselines in RTL using Verilog for precise power and area computation. Synopsys Design Compiler at 28nm, a Low Leakage library, and a 1GHz clock frequency are used for digital design synthesis. For SRAM results, Synopsys memory compiler is employed. Innovus is used for digital layout, with extracted SRAM from the memory compiler as a hard macro. The results are scaled to 32nm and back-annotated in our simulator for increased accuracy.

CHAPTER 5

RESULT

5.0.1 Weight Selection Acceleration

In Fig. 5.1, we demonstrate the effectiveness of our acceleration technique compared to the exhaustive search method (used in IWS). By employing binary search and reducing the number of Eigen pairs from five to four, we achieve comparable accuracy to IWS. We conducted evaluations on ResNet18 and DenseNet121 across various datasets. Notably, our acceleration method is more pronounced in larger datasets and models, reducing weight selection execution time by 1% to 17.8% compared to IWS. Furthermore, reducing the number of Eigen pairs further decreases execution time by 21.9% to 48.7% compared to IWS.

5.0.2 Accuracy vs Protected Weight Percentage

Table 5.1 illustrates the accuracy results of CIFAR10/100 datasets of the proposed approach and compares it with the IWS method [32, 33]. We repeat the experiments 50 times and get the average of the results. The third column shows the accuracy when there is no variation. The fourth column shows the results when there are 50% variations for all the weights. The "%Selected Weights" columns show the percentage of the weights that need to be

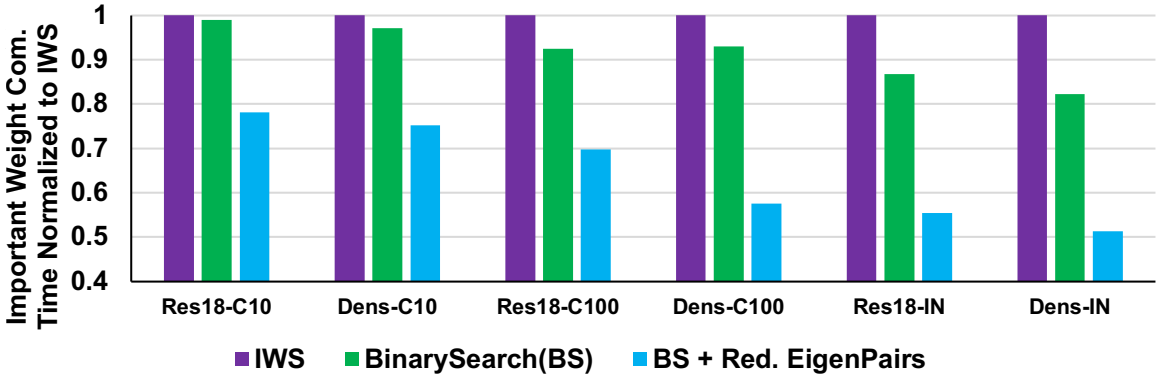

Figure 5.1: Speeding up weight sensitivity computation.

original accuracy [49] as columns "Accuracy IWS" and "Accuracy ***Harmonica***" show. Plots Fig. 5.5 demonstrate the results along with their trends over the ImageNet dataset. The x-axis shows the protected weight percentage and the y-axis shows the accuracy. For ImageNet, the accuracy without any protection mechanism is 7.25%, 17.13%, and 4.4% for ResNet18, ResNet34, and DenseNet121, respectively.

As the results illustrate, the amount of the weights that needs to be protected to reach the desired accuracy varies depending on the DNNs, dataset, number of layers, weight parameters, and their distributions. In general, when the DNN and dataset become more complex, the amount of protected weight is increased. For example, the percentage of protected weight for Imagenet is more than CIFAR100 and for CIFAR100 is more than CIFAR10 for the same DNN. On the other hand, for the same dataset, in general, DenseNet121 needs the highest weight percentage to be protected among evaluated networks.

Compared with the IWS, the ***Harmonica*** pushes more weights to the digital units since it selects the whole channel rather than choosing the individual weights. Fortunately, this mechanism leads to better load balancing and higher hardware utilization. We discuss this feature in Sec. 5.0.6.

Table 5.1: Comparing the impact of IWS and ***Harmonica*** approaches on accuracy over CIFAR10/100 datasets

		***Harmonica*/IWS [32, 33]**					
DataSet	**DNN**	**Original Acc No Noise**	**Acc with Var Both**	**%Select Weights IWS [32]**	**Accuracy IWS [32]**	**%Select Weights *Harmonica***	**Acc *Harmonica***
CIFAR10	VGG16	92.01%	27.53%	4.00%	91.81%	10.00%	91.69%
	ResNet18	94.76%	25.66%	3.00%	94.53%	10.00%	94.26%
	ResNet34	93.93%	55.01%	6.00%	92.98%	9.00%	92.94%
	DenseNet121	94.96%	22.77%	6.00%	94.32%	11.00%	94.04%
	EfficientNetB3	95.22%	23.88%	5.00%	94.46%	10.00%	94.36%
CIFAR100	VGG16	69.29%	12.3%	4.00%	68.35%	12.00%	68.30%
	ResNet18	74.07%	13.86%	8.00%	73.73%	13.00%	73.51%
	ResNet34	74.78%	24.28%	8.00%	74.06%	14.00%	73.99%
	DenseNet121	73.9%	10.16%	10.00%	73.12%	16.00%	72.80%
	EfficientNetB3	82.45%	13.20%	8.00%	81.61%	14.00%	81.50%

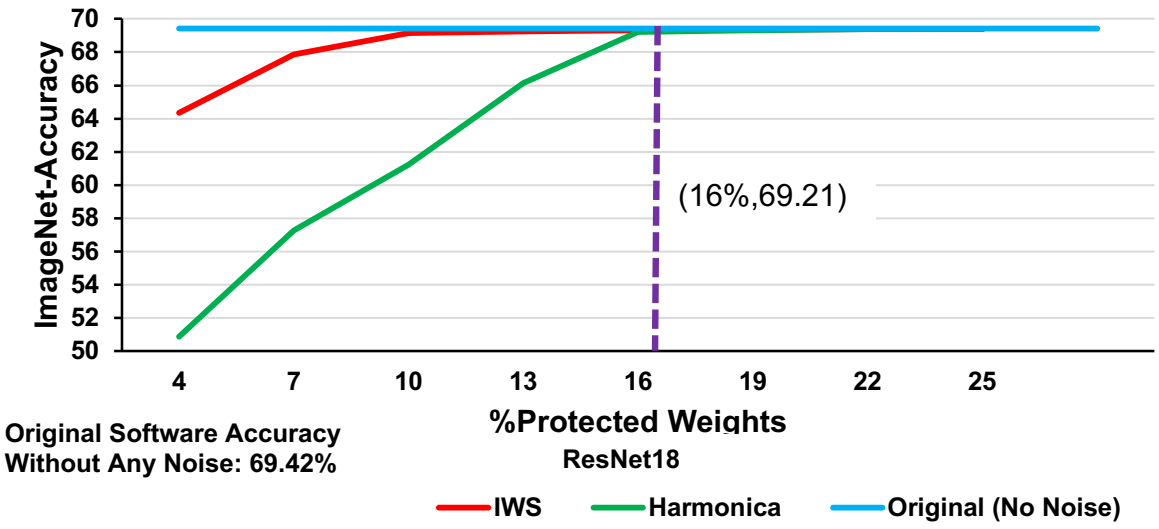

Figure 5.2: Comparison of various approaches. ResNet18 over ImageNet.

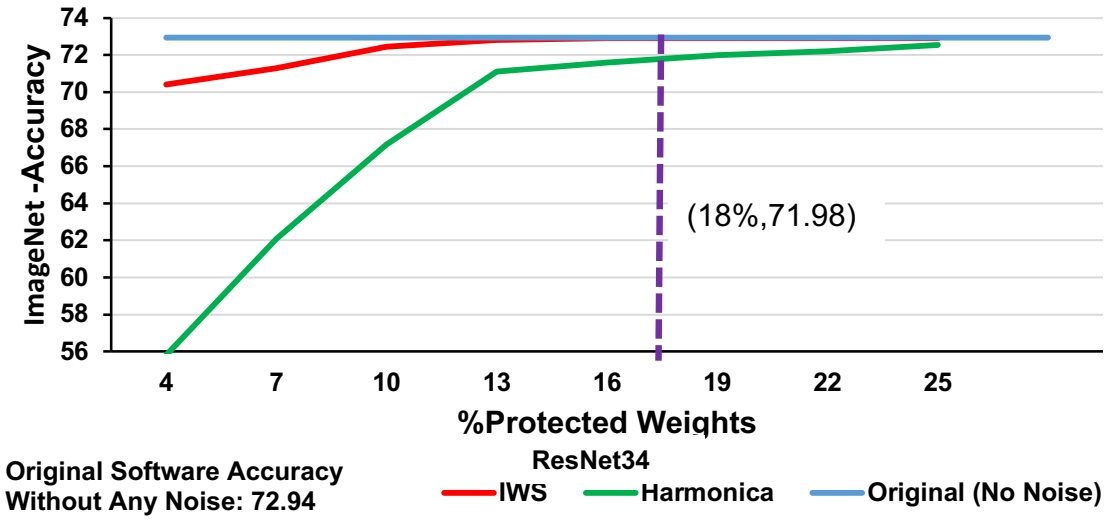

Figure 5.3: Comparison of various approaches. ResNet34 over ImageNet.

5.0.3 ADC Resolution

For the evaluated architecture, we consider 2 bits per cell while 128 rows can be activated at the same time and the number of bits per input is 1. We employ the same bias and offset encoding techniques used in ISAAC to save 1-bit ADC. Accordingly, for the mentioned configuration, the required ADC resolution with zero accuracy loss assuming there is no source of variation is 8-bit [9].

Table 5.2 shows accuracy results when ADCs with different resolutions are used for ***Harmonica*** with bias and offset subtraction architecture (i.e., Har.), baseline(i.e.,

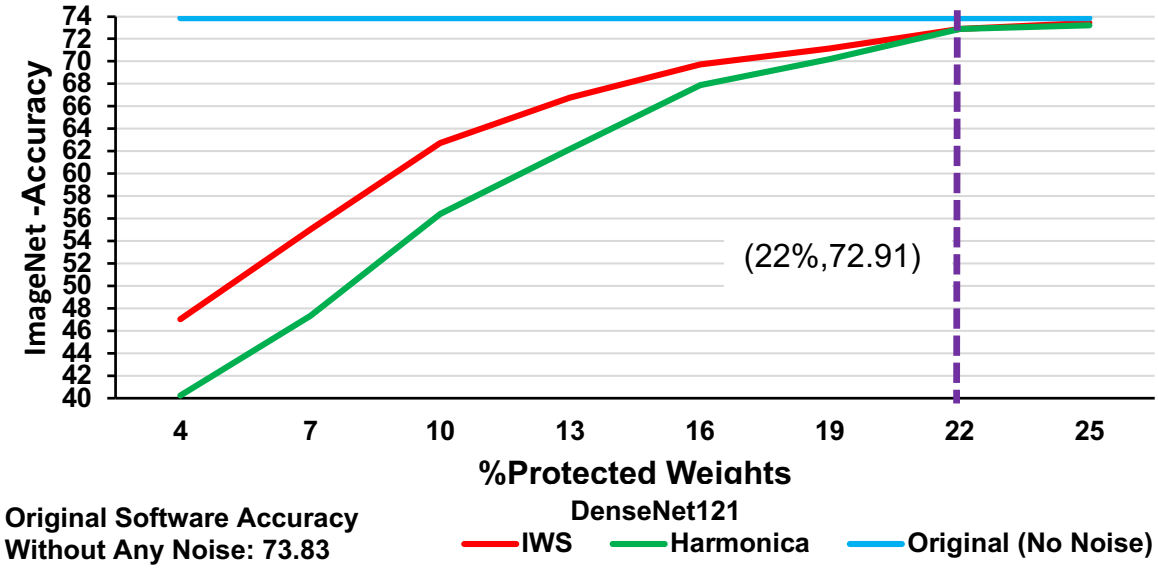

Figure 5.4: Comparison of various approaches. DensNet over ImageNet.

Figure 5.5: Comparing the impact of IWS and ***Harmonica*** approaches on accuracy over ImageNet dataset.

IWS [32])), and also differential cell architectures (i.e., HarDi, IWSDi).

As the results illustrate, for 7-bit ADC, in the case of CIFAR10/100 the accuracy results of IWS are slightly higher than ***Harmonica*** because its initial accuracy is higher (The first two columns). The reason is that due to channel-wise weight selection, the accuracy is slightly worse than IWS method. However, by reducing ADC resolution, the gains of the proposed solution become more apparent. For instance, in the worst case, 0.29%, 0.45%, and 0.66% accuracy drops happen for CIFAR10, CIFAR100, and Imagenet datesets, respectively in the ***Harmonica*** framework. Nonetheless, using IWS method [32], the accuracy drops are 0.98%, 1.95%, and 3.87%, respectively.

In the last two columns, we assess more extreme situations where we use 4-bit ADCs and reach almost the same accuracy as 6-bit ADC but through different architecture. The reason is that, here, we examine differential cells [10, 70, 71] where two separated crossbar categories are used for positive and negative weights. Unlike bias and offset subtraction cells [9, 28, 69], this mapping does not add biases to the weights, and accordingly their corresponding conductance are reduced. This is helpful as in many DNNs, zero and low-value weights contribute significantly in distribution of weight values [72, 73, 74].

Accordingly, the conductance (weight) variation and parasitic voltage drop across the crossbar columns (rows) are decreased, that results in having a higher accuracy. This fact shows that the proposed method is independent of the underlying architecture. Note that the differential cell-based designs have more hardware overheads and generally less throughput than offset-based designs.

Reducing ADC resolution is important as ADC contributes more than 30% of tile area and 50% of tile power in many architecture [9]. Having 7-bit ADC instead of 8-bit will save 7% area and 14% power of the tile. In the case of 6-bit ADC, the saving area and power of the tile would be 13% and 29%. By employing 6-bit ADC and pushing a percentage of weights to digital units as shown in Table 5.1 and Fig. 5.5 with the acceptable accuracy drop, the power/area saving is considerable. Obviously, by pushing more weights to digital accelerator, we can increase the accuracy at the cost of having lower throughput and performance.

5.0.4 Hybrid Quantization

Table 5.3 demonstrates the effects of hybrid quantization and different ADC resolutions on accuracy. In the second column, we use an 8(6)-bit quantization for weights in digital(analog) cores. In the best case (ResNet18/CIFAR10), accuracy drops by 0.09%, and in the worst case (DenseNet121/ImageNet), it drops by 0.53% compared to an 8-bit scheme. The last column shows the impact of hybrid quantization with 6-bit ADCs. In the best case, there's a 0.30% accuracy loss (ResNet18/CIFAR10), and in the worst case, a 1.11% loss (DenseNet121/ImageNet) happens compared to no smaller ADC and hybrid quantization. We observe a bit of accuracy drop when using 8(6) bits quantized values and 4-bit ADC with differential architecture cells [10] compared to bias and offset subtraction cells [9].

5.0.5 Power and Area

Table 5.5 shows the power and the area of different components of ***Harmonica*** and its comparison with ideal-ISAAC as a baseline architecture in detail. Table 5.6 summarize the

Table 5.2: Comparing the impact of using ADC with smaller resolution on the accuracy of both ***Harmonica*** and IWS [32, 33].

		***Harmonica*/IWS [32, 33]**							
DSet	**DNN**	**8-bit**	**8-bit**	**7-bit**	**7-bit**	**6-bit**	**6-bit**	**4-bit**	**4-bit**
		Har.	**IWS**	**Har.**	**IWS**	**Har.**	**IWS**	**HarDi**	**IWSDi**
	VGG16	91.69%	91.81%	91.57%	91.60%	91.45%	91.15%	91.39%	91.05%
	ResNet18	94.26%	94.53%	94.20%	94.31%	94.05%	93.89%	93.98%	93.82%
C10	ResNet34	92.94%	92.98%	92.90%	92.88%	92.77%	92.47%	92.68%	92.41%
	DenseNet121	94.04%	94.32%	93.89%	94.09%	93.75%	93.34%	93.62%	93.29%
	EfficientNetB3	94.36%	94.46%	94.26%	94.39%	94.12%	93.66%	94.01%	93.61%
	VGG16	68.30%	68.35%	68.11%	68.01%	67.98%	66.66%	67.88%	66.57%
	ResNet18	73.51%	73.73%	73.40%	73.43%	73.23%	72.17%	73.17%	72.09%
C100	ResNet34	73.99%	74.06%	73.85%	73.85%	73.60%	72.54%	73.52%	72.48%
	DenseNet121	72.80%	73.12%	72.69%	72.75%	72.35%	71.17%	72.29%	71.06%
	EfficientNetB3	81.50%	81.61%	81.37%	81.42%	81.25%	81.08%	81.13%	80.95%
	ResNet18	69.21%	69.21%	68.99%	67.55%	68.65%	66.08%	68.56%	65.95%
IN	ResNet34	71.98%	72.20%	71.84	70.06%	71.53%	68.95%	71.42%	68.81%
	DenseNet121	72.91%	72.91%	72.61%	72.28%	72.25%	69.04%	72.12%	68.92%

Table 5.3: Comparing the impact of hybrid quantization on accuracy.

DSet	**DNN**	**(8-6) 8-ADC**	**(8-6) 6-ADC**
	VGG16	91.53%	91.30%
	ResNet18	94.17%	93.96%
C10	ResNet34	92.83%	92.61%
	DenseNet121	93.89%	93.58%
	EfficientNetB3	94.29%	93.97%
	VGG16	68.10%	67.71%
	ResNet18	73.37%	73.04%
C100	ResNet34	73.78%	73.40%
	DenseNet121	72.69%	72.10%
	EfficientNetB3	81.46%	81.07%
	ResNet18	68.83%	68.29%
IN	ResNet34	71.49%	71.07%
	DenseNet121	72.40%	71.80%

numbers of Table 5.5 considering the required number of MCUs and tiles. These tables also compare the power and area of ***Harmonica*** with assorted baselines. As we discussed in Sec. 3.0.2, because of moving critical weights to the digital cores, ***Harmonica*** needs fewer crossbars, smaller peripheral circuitry such as ADC, and sample-and-hold logic. ***Harmonica*** needs bigger quantization circuitry as we explained in Sec. 2.0.4. However, its overhead is negligible compared to other components of the chip. ***Harmonica*** requires only 8 MCUs per tile, while other designs usually need 12 MCUs per tile. In addition, the total number of required tiles reduces from 168 (in ISAAC) to 148. The bottom of Table 5.5 provides the power and area of the digital accelerator of ***Harmonica***.

As demonstrated, ***Harmonica*** provides 28% better area and 57% better power compared to ideal-ISAAC. For IWS [32, 33], we consider two different approaches. In the

first one (IWS-1), there is only 1 tile and the weights are written into the same crossbar when the current layer is finished. Although this approach saves the area and power of the analog accelerator, the performance and the throughput drop significantly. The reason is that writing the weights into ReRAM cells takes a considerable amount of energy. Moreover, we need to write into a cell multiple times to make sure it has a correct value [75]. Using the same tile for writing the weights of all layers makes the endurance issue of ReRAM technology worse [76]. Besides, this method limits the amount of parallelism to only one tile, which drops the throughput remarkably.

Accordingly, most of the existing works [9, 10, 77] consider another approach, where all (or most) weights are written into the different crossbars in various tiles only once. IWS-2 follows this methodology.

IWS-2 considers 3-bit quantization for individual weights in analog cores and 8-bit for individual weights in digital cores [33]. With this scheme and the same percentage of weight protection as ***Harmonica***, the accuracy drops for ResNet18 over CIFAR10 is 4% while over ImageNet is 24.8%. This scheme reduces the number of required crossbars for IWS-2, but with a considerable accuracy degradation. ***Harmonica*** can use the same quantization bits, and reduces the number of crossbars and tiles even more, while the accuracy drop for ResNet18 over CIFAR10 (ImageNet) is 1.5% (12.2%).

Harmonica improves the power and the area of the IWS-2 [32] by 65% and 2.1× respectively. IWS-1 has a slightly better power (e.g., 1%) and worse area than ***Harmonica*** (47%) due to using sparse-centralized architecture (e.g., SIGMA [41]) that has high power and area. Generally, accelerators that handle irregular sparse architectures have significant overheads. ***Harmonica*** outclasses FORMS [29] by 44%(29%) in terms of power and area, while the improvement over SRE [28] is 45%(27%). Although digital accelerators show better power consumption, the area and power efficiency of the ***Harmonica*** outperforms the existing accelerators as Table 5.4 shows.

5.0.6 Power/Area Efficiency

Table 5.4 presents the area and power efficiency of ***Harmonica*** (e.g., using 6-bit ADC) and ***HarmonicaDi*** (using 4-bit ADC) and also assorted digital and analog accelerators normalized to ideal-ISAAC. The area- and power-efficiency of Ideal-ISAAC (e.g., it idealistically activates 128 wordlines considering it is immune to the variation while, in practice, it is not possible) where both weight and activation are 8 bits are 1912 $\frac{GOPS}{s \times mm^2}$ and 2510 $\frac{GOPS}{s \times w}$. The WAX-Hybrid employs the proposed solution to separate channels and apply 8-bit and 6-bit mixed-precision for important and non-important channels.

Harmonica and ***HarmonicaDi*** outperform existing works and offer immunity against variation. ***Harmonica***'s efficiency results from its innovative design with smaller ADCs,

Table 5.4: Peak area- and power-efficiency of different architectures normalized to Ideal-ISAAC.

Architecture	$\frac{GOPs}{s \times mm^2}$	$\frac{GOPs}{s \times W}$
Ideal-ISAAC [9]	1	1
PUMA [69]	0.70	0.79
SRE[28]	0.19	0.26
FORMS8(not pruned)[29]	0.54	0.61
FORMS16(not pruned)[29]	0.77	0.84
DaDianNao [11]	0.13	0.45
TPU [78]	0.08	0.48
WAX[58]	0.33	2.3
WAX-Hybrid[58]	0.45	2.4
SIMBA[61]	0.34-0.62	0.08-2.4
IWS-1[32, 33]	0.13	0.15
IWS-2[32, 33]	0.38	0.41
Harmonica	1.43	1.91
HarmonicaDi	1.75	2.65

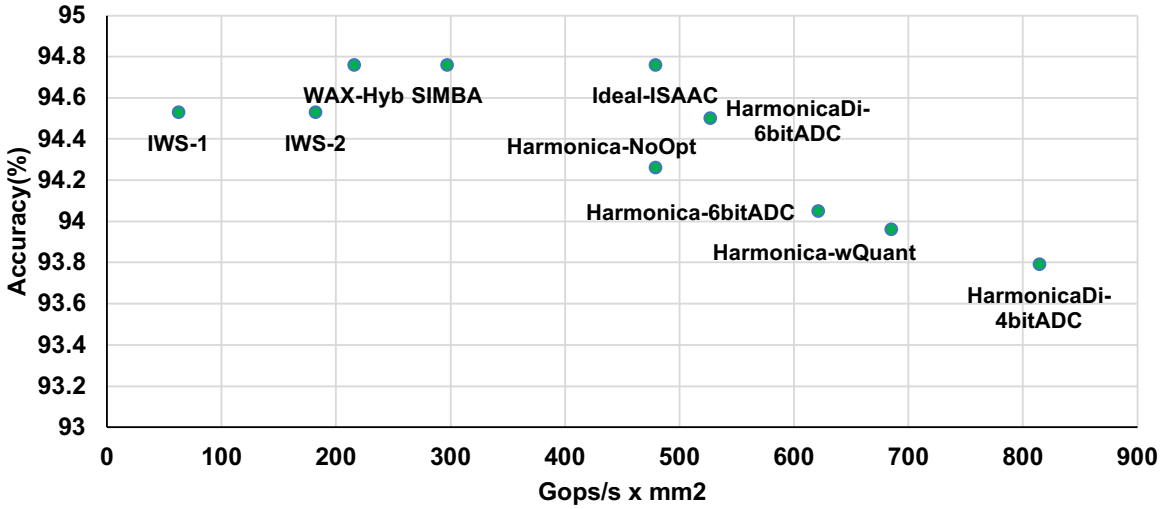

Figure 5.6: Accuracy vs area-efficiency comparison for different architectures for ResNet18 inference on CIFAR10.

hybrid quantization, fewer crossbars, and a high-throughput digital accelerator. Fig. 5.6 displays accuracy vs. area-efficiency for ResNet18 architectures on CIFAR10, showcasing how ***Harmonica*** optimizations bring us closer to the ideal point.

In the case of ***Harmonica***, we aim to simplify the design by using a common clock frequency of 1 GHz for both digital and analog accelerators. Our results indicate that the analog cores achieve a peak area-efficiency of 2549 $\frac{GOPS}{s \times mm^2}$, while the digital cores have a throughput of 434 $\frac{GOPS}{s \times mm^2}$. This means that the area-efficiency of the analog cores is $\frac{2549}{434} = 5.87\times$ greater than that of the digital cores.

This implies that, in an ideal load-balanced scenario where both digital and analog cores finish simultaneously, the analog cores need to perform 5.87× more work than the digital ones. For example, nearly 16% of the workload must be handled by the digital accelerator. This applies to all DNNs except DenseNet121 on ImageNet. Note that the results in Table 5.1 and Fig. 5.5 are the minimum digital weight percentages required to maintain in digital cores to reach the high accuracy (less than 1% loss, as assumed in many applications[49]). We will show how load balancing have an impact on energy and execution time results in Section Sec. 5.0.7.

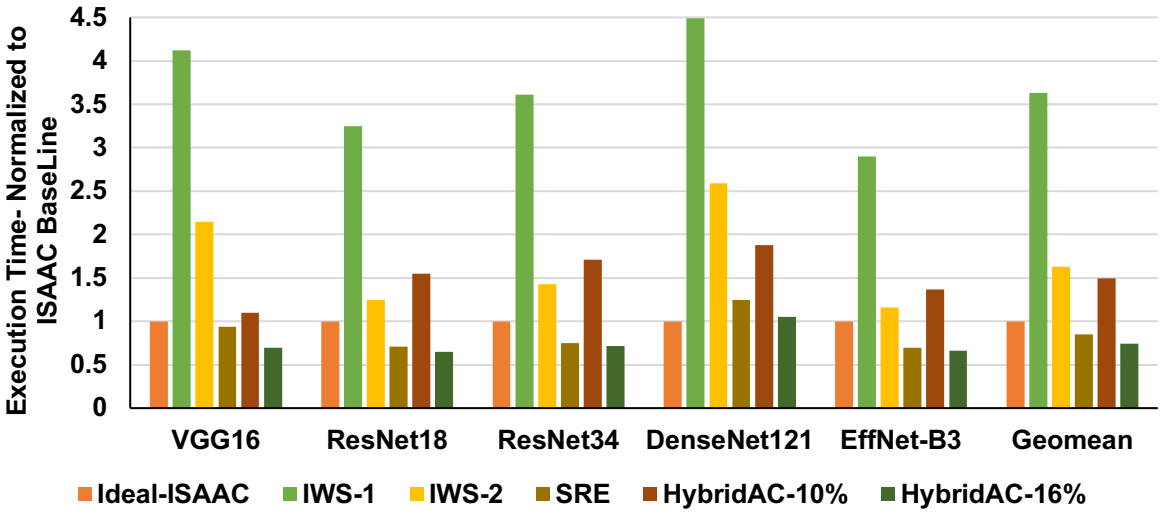

Figure 5.7: Execution time of different DNNs using different architectures

		Harmonica			**Ideal-ISAAC [9]**		
Component	Parameter	Spec	Power(mW)	Area(mm^2)	Spec	Power(mW)	Area(mm^2)
eDRAM Buffer	size num-bank bus-width	32KB 2 256	11.2	0.041	64KB 2 256	20.7	0.08
eDRAM to MCU	num-wire	384	7	0.09	384	7	0.09
Router			10.5	0.037		10.5	0.037
Activation	number	2	0.364	0.00042	2	0.364	0.00042
S+A	number	1	0.035	0.000042	1	0.035	0.000042
Max-pool	number	1	0.28	0.000016	1	0.28	0.000016
Quant. Cir.	number	1	0.0065	0.00098	1	0.0025	0.00040
Output Reg.	size	3KB	1.176	0.00224	3KB	1.176	0.00224
ADC	resolution frequency number	6-bit 1.2GHz 32	9.6	0.0062	8-bit 1.2GHz 8	16	0.0096
Inverter	resolution number	1-bit 8×128	4	0.00017	1-bit 8×128	4	0.00017
S&H	number	8×128	0.007	0.00003	8×128	0.01	0.00004
Crossbars	number size bit/cell	8 128×128 2	2.4	0.00024	8 128×128 2	2.4	0.00024
S+A	number	4	0.2	0.000024	4	0.2	0.000024
local SRAM	number	152	303.71	0.88	–	—	—
MACs	number	152	480.36	1.11	—	—	—
Weight Reg	number	152	111.22	0.37	—	—	—
Activation Reg	number	152	150.26	0.42	—	—	—
PSum Reg	number	152	95.23	0.39	—	—	—

Table 5.5: Power/area of different components of ***Harmonica*** and ISAAC

Table 5.6: Comparison of total power and area of ***Harmonica*** with different baselines

	Harmonica			**IWS-1 [32, 33]**			**IWS-2 [32, 33]**				**SIGMA [11]**	
	Spec	Power(mW)	Area(mm^2)	Spec	Power(mW)	Area(mm^2)	Spec	Power(mW)	Area(mm^2)		Power(mW)	Area(mm^2)
Dig unit		30.055	0.17		40.85	0.213		49.8	0.225	Adder	2679.6	7.812
MCUs per tile	8	140.6	0.076	12	288.96	0.158	6	144.48	0.078	Multiplier	10846.1	31.62
1 Tile (MCUs + Dig unit)		170.655	0.24		329.81	0.37		185.33	0.29	Local Memories	255.2	0.744
Tiles	148	25256.94	36.69	1	329.81	0.37	142	26316.86	41.37	Dist Noc	3700.4	10.788
HyperTransport freq	4/1.6GHz	10400	22.88	4/1.6GHz	10400	22.88	4/1.6GHz	10400	22.88	Layout Redundancy	6890.4	20.088
HyperTransport bw	6.4GB/s			6.4GB/s			6.4GB/s			Read NoC	765.6	2.232
Analog Chip Total		35656.94	59.577		10738.81	23.265		36716.86	64.25	FAN Controller	382.8	1.116
Digital Chip Total		1788.1	6.81		25520.1	74.4		25520.1	74.4	Analog Chip Total	—	—
Chip Total		37444.94	66.39		36258.81	97.665		61936.96	138.65	Chip Total	25520.1	74.4

	FORMS[29]			**SRE [28]**			**Ideal-ISAAC [9]**		
	Spec	Power(mW)	Area(mm^2)	Spec	Power(mW)	Area(mm^2)	Spec	Power(mW)	Area(mm^2)
Dig unit		53.052	0.25		49.27	0.23		40.85	0.213
MCUs per tile	12	280.05	0.152	12	212.16	0.1171	12	288.96	0.16
1 Tile (MCUs + Dig unit)		333.1	0.39		262.01	0.34		329.81	0.37
Tiles	168	55960.8	66.27	168	44017.68	57.88	168	55408.08	62.21
HyperTransport freq	4/1.6GHz	10400	22.88	4/1.6GHz	10400	22.88	4/1.6GHz	10400	22.88
HyperTransport bw	6.4GB/s			6.4GB/s			6.4GB/s		
Analog Chip Total		66360.8	89.15		54417.68	80.76		65808.08	85.09
Digital Chip Total		—	—	Index overhead	28.2	4.23		—	—
Chip Total		66360.8	89.15		54445.88	84.99		65808.08	85.09

5.0.7 Execution Time and Energy Results

The results of execution time and energy of different DNNs using different methods are demonstrated in Fig. 5.7 and Fig. 5.8 where idealistically assuming even ISAAC and SRE are immune to imperfections (ISO-accuracy). As we discussed, in practice, their accuracy

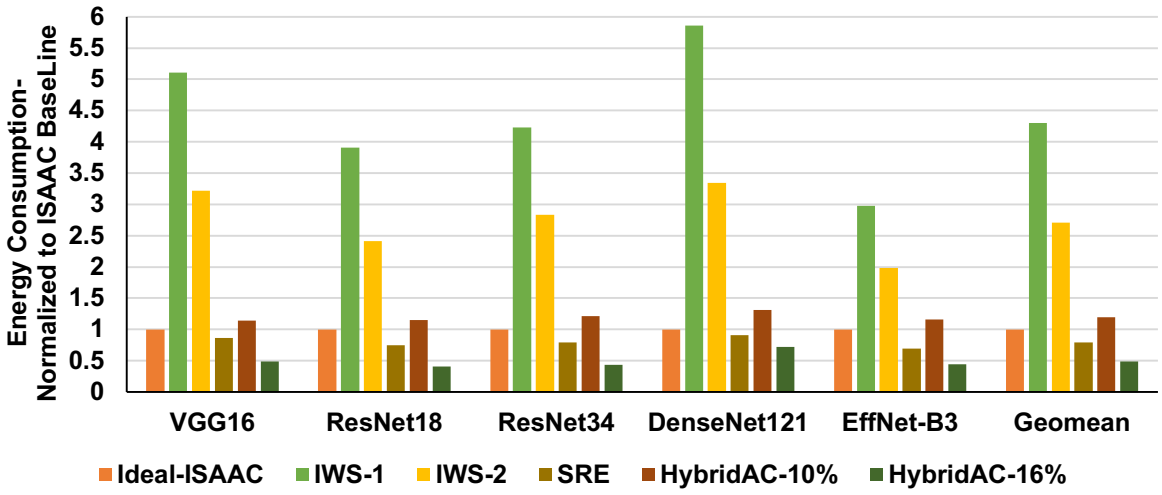

Figure 5.8: Energy consumption of different DNNs using different architectures

drops significantly.

On average, the execution time of Ideal-ISAAC is 3.6× and 1.6× better than IWS-1 and IWS-2 [32, 33], respectively. Due to exploiting activation and weight sparsity, SRE [28] can outperform ISAAC by 15×; however, by having weight and activation as 8-bit rather than 16-bit, the sparsity exploitation is reduced. In addition, SRE has substantial column/row indexing overhead and relatively lower throughput as it activates only 16 rows. In ***Harmonica***-10%, we allocate a maximum of 10% of weights to digital cores. If a higher percentage is required, it must wait for available digital core resources.

As the plots show, due to load unbalancing issue, the timing (energy) results of ***Harmonica***-10% are worse than ISAAC and SRE. In contrast, it offers a distinct immunity solution, setting it apart from the baselines. Following our discussion in Section Sec. 5.0.6, by moving more weights to the digital cores (up to 16%), we have a better load balancing that leads to improved timing and energy results. This allocation is well-suited for the evaluated DNNs, except for DenseNet121 on ImageNet as the results in Table 5.1 and Fig. 5.5 show. ***Harmonica***-16% improves the ISAAC timing by 26% and SRE by 14%. It is worthy to note perfect mapping and load balancing for mixed-signal designs remain an open challenge, beyond the scope of this study.

As we discussed, ***Harmonica*** has significantly lesser data movement than IWS. ***Harmonica***-16% consumes less energy than the corresponding baselines. Since ***Harmonica***-

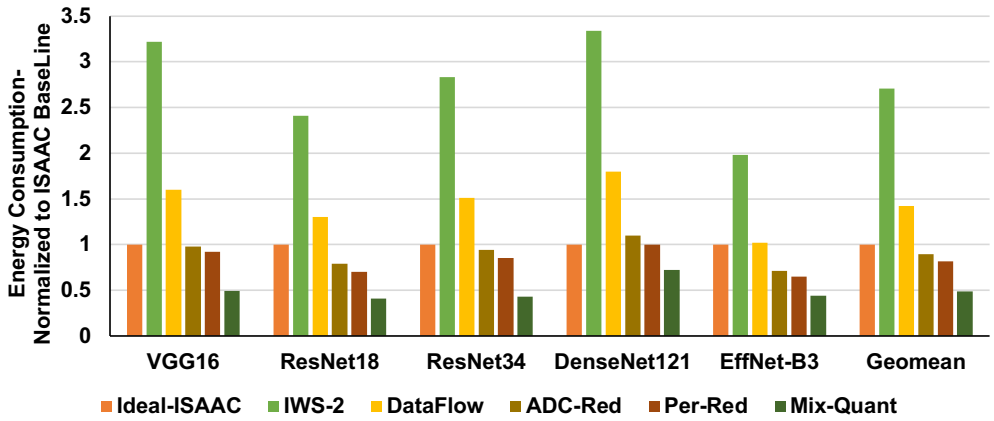

Figure 5.9: Break down energy results.

10% has a longer execution time, its energy consumption is more than ***Harmonica***-16% and SRE. In average, ***Harmonica***-16% improves energy consumption over Ideal-ISAAC, SRE, IWS-1, and IWS-2 by 52%, 40%, 8.9×, and 5.6×, respectively. Being faster and consuming less energy than the evaluated baselines leads to a better energy-delay product as well. As discussed, both ISAAC and SRE are not immune to imperfections, rendering them impractical for real-world applications regardless of weight distribution.

Fig. 5.9 illustrates the impact of optimization techniques (dataflow, ADC and peripheral circuitry reduction, and mixed-quantization) on energy consumption. Moving from left to right, each optimization already includes the impact of the previous one. Results reveal that, in descending order, avoiding activation replication, optimized data flow, ADC reduction, and mixed-quantization have the greatest impact on energy reduction.

The energy consumption per layer of ResNet18 over ImageNet in logarithmic scale is shown in Fig. 5.10, where the results of each layer are normalized to ISAAC results. As seen, except for the first layer, IWS-1 and IWS-2 consume significantly more energy than ***Harmonica*** (13.2× and 7.5×, respectively).

5.0.8 Activating More Wordlines Simultaneously

As we discussed, existing mixed-signal designs especially those that rely on bias and offset subtraction methods cannot activate many wordlines at the same time. This problem can be mitigated if we can make the important weights immune against variations. Fig. 5.11 shows

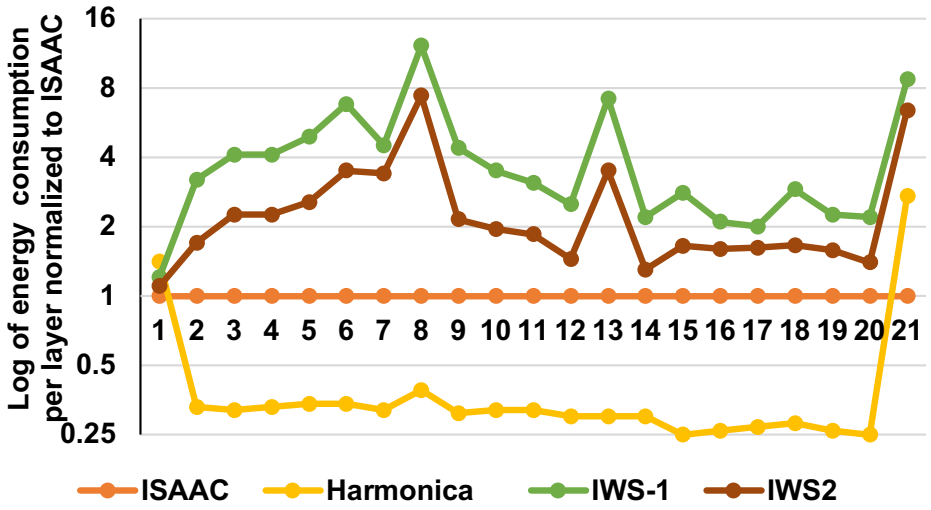

Figure 5.10: Comparing IWS and ***Harmonica*** of energy consumption per layer

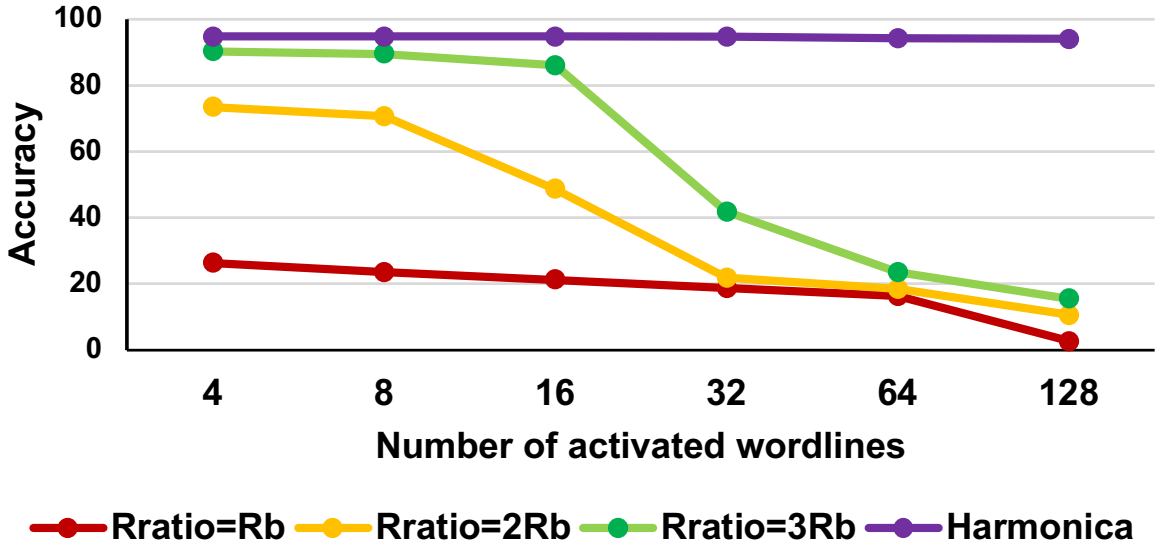

Figure 5.11: Activating more number of wordlines with different scenarios

our simulation results on different scenarios for ResNet18/ CIFAR10 where we increase the $R-ratio = \frac{R_{on}}{R_{off}}$ of memory cell and decrease conductance deviation (σ). The first plot (e.g., red one) considers $R-ratio$ of VTEAM model [67] and $\sigma = 50\%$ as a baseline ($R-ratio = R_b$). The second and third plots are for the situations when $R_{ratio} = 2R_b$ $\sigma = \sigma_b/2$ and $R_{ratio} = 3R_b$ $\sigma = \sigma_b/3$, respectively. The last one shows the behaviour of ***Harmonica***. These figures show that the amount of accuracy degradation is remarkable if there is no protection mechanism, while with ***Harmonica*** the accuracy drop is less than 1%.

CHAPTER 6
RELATED WORK

Feinberg et al. propose an error correction scheme using AN-code to restore the accuracy loss [27]. AN codes rectify errors in linear functions and only consider the random telegraph noise. Moreover, it adds hardware overheads, which makes it less scalable [38]. Prior works activate fewer wordlines and rely on ADC quantization to surpass the propagated errors [79, 29]. However, this solution cannot completely resolve the issue. Besides, their methods show lower throughput and area/power efficiency. The method proposed by Liu [37] requires fine-tuning the network. This work replaces the final linear classification layer with a set of collaborative logistic classifiers to overcome the performance degradation, which needs retraining and adding hardware overheads. In addition, modern image classifier architecture (e.g., SqueezeNet) does not incorporate any linear layer for making the final decision. In such a scenario, their method results in additional compute and hardware overhead to maintain the original performance. Long et al. consider a Gaussian noise model and propose an algorithmic solution that needs retraining by incorporating the modeled noise [34].

Retraining is an expensive process in terms of time, energy, and hardware resource consumption. In addition, exact modeling a particular noise is a hard task. Training data is also not available all the time. Nevertheless, this solution is not able to capture all variations that occur in a real system if that noise and/or its complete demeanor is not modeled during retraining comprehensively. However, retraining-based approaches are less effective when models of variations assumed during retraining do not accurately match the variations experienced by the hardware during inference. In general, exact and comprehensive modeling of noise is a hard task that makes retraining-based methods less effective.

CHAPTER 7

CONCLUSION

In ***Harmonica***, we introduce a novel hybrid accelerator that combines digital and analog components through algorithm-hardware co-design. This addresses imperfections like noise, weight variations, and programming errors in the mixed-signal accelerators. ***Harmonica*** presents a method for input channel-wise weight selection and their corresponding activation, which are then processed by a novel tiny digital accelerator.

Harmonica enables us to use a fewer number of crossbars, smaller peripheral circuitry, and ADCs with lower precision. The proposed solution also unlocks hybrid quantization based on input channels' weight distillation per layer. In our experiments, even with imperfections up to 50%, ***Harmonica*** reduces accuracy degradation from 60%-90% in designs without protection to 1%-2% for various DNNs and datasets. Additionally, compared to ISAAC (SRE), ***Harmonica*** improves execution time, energy, area, power, area-efficiency, and power-efficiency by 26% (14%), 52% (40%), 28% (28%), 57% (45%), 43% (7.5×), and 91% (7.3×), respectively. Using architecture-based differential cells ***Harmonica*** outperforms ISAAC (SRE) by 75% (9.2×) and 2.65× (10.2×) in area- and power-efficiency.

REFERENCES

[1] H. N. Khan, D. A. Hounshell, and E. R. Fuchs, "Science and research policy at the end of moore's law," *Nature Electronics*, vol. 1, no. 1, pp. 14–21, 2018.

[2] K. T. Malladi, F. A. Nothaft, K. Periyathambi, B. C. Lee, C. Kozyrakis, and M. Horowitz, "Towards energy-proportional datacenter memory with mobile dram," in *2012 39th Annual International Symposium on Computer Architecture (ISCA)*, IEEE, 2012, pp. 37–48.

[3] T.-J. Yang and V. Sze, "Design considerations for efficient deep neural networks on processing-in-memory accelerators," in *2019 IEEE International Electron Devices Meeting (IEDM)*, IEEE, 2019, pp. 22–1.

[4] B. Kazemivash and V. D. Calhoun, "Bparc: A novel spatio-temporal (4d) data-driven brain parcellation scheme based on deep residual networks," in *2020 IEEE 20th International Conference on Bioinformatics and Bioengineering (BIBE)*, IEEE, 2020, pp. 1071–1076.

[5] L. Deng and D. Yu, "Deep learning: Methods and applications," *Foundations and trends in signal processing*, vol. 7, no. 3–4, pp. 197–387, 2014.

[6] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[8] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE journal of solid-state circuits*, vol. 52, no. 1, pp. 127–138, 2016.

[9] A. Shafiee *et al.*, "Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 14–26, 2016.

[10] P. Chi *et al.*, "Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 27–39, 2016.

[11] Y. Chen *et al.*, "Dadiannao: A machine-learning supercomputer," in *MICRO*, 2014.

[12] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th annual international symposium on computer architecture*, 2017, pp. 1–12.

[13] V. Sze, Y.-H. Chen, J. Emer, A. Suleiman, and Z. Zhang, "Hardware for machine learning: Challenges and opportunities," in *2017 IEEE Custom Integrated Circuits Conference (CICC)*, IEEE, 2017, pp. 1–8.

[14] Y. K. Rupesh, P. Behnam, G. R. Pandla, M. Miryala, and M. N. Bojnordi, "Accelerating k-medians clustering using a novel 4t-4r rram cell," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 12, pp. 2709–2722, 2018.

[15] Y. Luo *et al.*, "Codg-reram: An algorithm-hardware co-design to accelerate semi-structured gnns on reram," in *2022 IEEE 40th International Conference on Computer Design (ICCD)*, IEEE, 2022, pp. 280–289.

[16] P. Behnam, A. P. Chowdhury, and M. N. Bojnordi, "R-cache: A highly set-associative in-package cache using memristive arrays," in *2018 IEEE 36th International Conference on Computer Design (ICCD)*, IEEE, 2018, pp. 423–430.

[17] G. Yuan *et al.*, "Tinyadc: Peripheral circuit-aware weight pruning framework for mixed-signal dnn accelerators," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, 2021, pp. 926–931.

[18] B. Kazemivash and V. Calhoun, "A novel 5d brain parcellation approach based on spatio-temporal encoding of resting fmri data from deep residual learning," *Journal of Neuroscience Methods*, vol. 369, p. 109 478, 2022.

[19] B. Kazemivash and V. D. Calhoun, "Bparc: A novel spatio-temporal (4d) data-driven brain parcellation scheme based on deep residual networks," in *IEEE 20th International Conference on Bioinformatics and Bioengineering*, ser. BIBE, 2020, pp. 1071–1076.

[20] W.-H. Chen *et al.*, "A 65nm 1mb nonvolatile computing-in-memory reram macro with sub-16ns multiply-and-accumulate for binary dnn ai edge processors," in *2018 IEEE International Solid-State Circuits Conference-(ISSCC)*, IEEE, 2018, pp. 494–496.

[21] C.-X. Xue *et al.*, "15.4 a 22nm 2mb reram compute-in-memory macro with 121-28tops/w for multibit mac computing for tiny ai edge devices," in *2020 IEEE International Solid-State Circuits Conference-(ISSCC)*, IEEE, 2020, pp. 244–246.

[22] Y. Luo, X. Han, Z. Ye, H. Barnaby, J.-S. Seo, and S. Yu, "Array-level programming of 3-bit per cell resistive memory and its application for deep neural network inference," *IEEE Transactions on Electron Devices*, vol. 67, no. 11, pp. 4621–4625, 2020.

[23] K. Ueyoshi *et al.*, "Diana: An end-to-end energy-efficient digital and analog hybrid neural network soc," in *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, IEEE, vol. 65, 2022, pp. 1–3.

[24] G. Li *et al.*, "Understanding error propagation in deep learning neural network (dnn) accelerators and applications," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2017, pp. 1–12.

[25] P.-Y. Chen, X. Peng, and S. Yu, "Neurosim: A circuit-level macro model for benchmarking neuro-inspired architectures in online learning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 12, pp. 3067–3080, 2018.

[26] S. Rashidi, M. Jalili, and H. Sarbazi-Azad, "Improving mlc pcm performance through relaxed write and read for intermediate resistance levels," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 15, no. 1, pp. 1–31, 2018.

[27] B. Feinberg, S. Wang, and E. Ipek, "Making memristive neural network accelerators reliable," in *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, IEEE, 2018, pp. 52–65.

[28] T.-H. Yang *et al.*, "Sparse reram engine: Joint exploration of activation and weight sparsity in compressed neural networks," in *Proceedings of the 46th International Symposium on Computer Architecture*, 2019, pp. 236–249.

[29] G. Yuan *et al.*, "Forms: Fine-grained polarized reram-based in-situ computation for mixed-signal dnn accelerator," 2021.

[30] Y. N. Wu, V. Sze, and J. S. Emer, "An architecture-level energy and area estimator for processing-in-memory accelerator designs," in *2020 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, IEEE, 2020, pp. 116–118.

[31] M. Imani, S. Gupta, Y. Kim, and T. Rosing, "Floatpim: In-memory acceleration of deep neural network training with high precision," in *2019 ACM/IEEE 46th Annual International Symposium on Computer Architecture (ISCA)*, IEEE, 2019, pp. 802–815.

[32] S. Dash and S. Mukhopadhyay, "Hessian-driven unequal protection of dnn parameters for robust inference," in *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, IEEE, 2020, pp. 1–9.

[33] S. Dash, Y. Luo, A. Lu, S. Yu, and S. Mukhopadhyay, "Robust processing-in-memory with multi-bit reram using hessian-driven mixed-precision computation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2021.

[34] Y. Long, X. She, and S. Mukhopadhyay, "Design of reliable dnn accelerator with un-reliable reram," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, 2019, pp. 1769–1774.

[35] C.-C. Chang *et al.*, "Mitigating asymmetric nonlinear weight update effects in hardware neural network based on analog resistive synapse," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 8, no. 1, pp. 116–124, 2017.

[36] A. Kazemi, C. Alessandri, A. C. Seabaugh, X. S. Hu, M. Niemier, and S. Joshi, "A device non-ideality resilient approach for mapping neural networks to crossbar arrays," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, IEEE, 2020, pp. 1–6.

[37] T. Liu, W. Wen, L. Jiang, Y. Wang, C. Yang, and G. Quan, "A fault-tolerant neural network architecture," in *2019 56th ACM/IEEE Design Automation Conference (DAC)*, IEEE, 2019, pp. 1–6.

[38] S. Jain and A. Raghunathan, "Cxdnn: Hardware-software compensation methods for deep neural networks on resistive crossbar systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 18, no. 6, pp. 1–23, 2019.

[39] X. Yang *et al.*, "Multi-objective optimization of reram crossbars for robust dnn inferencing under stochastic noise," *arXiv preprint arXiv:2109.05437*, 2021.

[40] M. S. Ali *et al.*, "Erdnn: Error-resilient deep neural networks with a new error correction layer and piece-wise rectified linear unit," *IEEE Access*, vol. 8, pp. 158 702–158 711, 2020.

[41] E. Qin *et al.*, "Sigma: A sparse and irregular gemm accelerator with flexible interconnects for dnn training," in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, IEEE, 2020, pp. 58–70.

[42] A. Grossi *et al.*, "Resistive ram endurance: Array-level characterization and correction techniques targeting deep learning applications," *IEEE Transactions on Electron Devices*, vol. 66, no. 3, pp. 1281–1288, 2019.

[43] E. Kondratyuk *et al.*, "Automated testing algorithm for the improvement of 1t1r reram endurance," *IEEE Transactions on Electron Devices*, vol. 68, no. 10, pp. 4891–4896, 2021.

[44] Y. Chen, "Reram: History, status, and future," *IEEE Transactions on Electron Devices*, vol. 67, no. 4, pp. 1420–1433, 2020.

[45] J. H. Ko, D. Kim, T. Na, J. Kung, and S. Mukhopadhyay, "Adaptive weight compression for memory-efficient neural networks," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, IEEE, 2017, pp. 199–204.

[46] D. Kim, J. Kung, and S. Mukhopadhyay, "A power-aware digital multilayer perceptron accelerator with on-chip training based on approximate computing," *IEEE Transactions on Emerging Topics in Computing*, vol. 5, no. 2, pp. 164–178, 2017.

[47] B. Hassibi, D. Stork, and G. Wolff, "Optimal brain surgeon: Extensions and performance comparisons," *Advances in neural information processing systems*, vol. 6, 1993.

[48] Z. Yao, A. Gholami, K. Keutzer, and M. W. Mahoney, "Pyhessian: Neural networks through the lens of the hessian," in *2020 IEEE international conference on big data (Big data)*, IEEE, 2020, pp. 581–590.

[49] B. Darvish Rouhani *et al.*, "Pushing the limits of narrow precision inferencing at cloud scale with microsoft floating point," *Advances in Neural Information Processing Systems*, vol. 33, pp. 10 271–10 281, 2020.

[50] N. Zmora, G. Jacob, L. Zlotnik, B. Elharar, and G. Novik, "Neural network distiller: A python package for dnn compression research," Oct. 2019.

[51] B. Jacob *et al.*, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2704–2713.

[52] H. Abdelaziz, J. H. Shin, A. Pedram, J. Hassoun, *et al.*, "Rethinking floating point overheads for mixed precision dnn accelerators," *Proceedings of Machine Learning and Systems*, vol. 3, 2021.

[53] R. Yu *et al.*, "Nisp: Pruning networks using neuron importance score propagation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9194–9203.

[54] T. Hoefler, D. Alistarh, T. Ben-Nun, N. Dryden, and A. Peste, "Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks.," *J. Mach. Learn. Res.*, vol. 22, no. 241, pp. 1–124, 2021.

[55] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[56] S. Han *et al.*, "Deep compression and eie: Efficient inference engine on compressed deep neural network.," in *Hot Chips Symposium*, 2016, pp. 1–6.

[57] M. Saberi, R. Lotfi, K. Mafinezhad, and W. A. Serdijn, "Analysis of power consumption and linearity in capacitive digital-to-analog converters used in successive approximation adcs," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2011.

[58] S. Gudaparthi, S. Narayanan, R. Balasubramonian, E. Giacomin, H. Kambalasubramanyam, and P.-E. Gaillardon, "Wire-aware architecture and dataflow for cnn accelerators," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, 2019, pp. 1–13.

[59] S. Rashidi *et al.*, "Enabling compute-communication overlap in distributed deep learning training platforms," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, IEEE, 2021, pp. 540–553.

[60] M. N. Bojnordi and E. Ipek, "Desc: Energy-efficient data exchange using synchronized counters," in *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture*, 2013, pp. 234–246.

[61] Y. S. Shao *et al.*, "Simba: Scaling deep-learning inference with multi-chip-module-based architecture," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, 2019, pp. 14–27.

[62] Y.-H. Chen, T.-J. Yang, J. Emer, and V. Sze, "Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 2, pp. 292–308, 2019.

[63] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi, "Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory," *IEEE T-CAD*, 2012.

[64] R. Balasubramonian, A. B. Kahng, N. Muralimanohar, A. Shafiee, and V. Srinivas, "Cacti 7: New tools for interconnect exploration in innovative off-chip memories," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 14, no. 2, pp. 1–25, 2017.

[65] Y. N. Wu, V. Sze, and J. S. Emer, "An architecture-level energy and area estimator for processing-in-memory accelerator designs," in *2020 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, IEEE, 2020, pp. 116–118.

[66] Z. Zhu *et al.*, "Mnsim 2.0: A behavior-level modeling tool for memristor-based neuromorphic computing systems," in *Proceedings of the 2020 on Great Lakes Symposium on VLSI*, 2020, pp. 83–88.

[67] S. Kvatinsky, M. Ramadan, E. G. Friedman, and A. Kolodny, "Vteam: A general model for voltage-controlled memristors," *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2015.

[68] B. Murmann, *Adc performance survey 1997-2019,[online]. available: Http://web.stanford.edu/ murmann/adcsurvey.html*.

[69] A. Ankit *et al.*, "Puma: A programmable ultra-efficient memristor-based accelerator for machine learning inference," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2019, pp. 715–731.

[70] P. Yao *et al.*, "Fully hardware-implemented memristor convolutional neural network," *Nature*, vol. 577, no. 7792, pp. 641–646, 2020.

[71] V. Joshi *et al.*, "Accurate deep neural network inference using computational phase-change memory," *Nature communications*, vol. 11, no. 1, pp. 1–13, 2020.

[72] T. Zhang *et al.*, "A systematic dnn weight pruning framework using alternating direction method of multipliers," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 184–199.

[73] R. Zhao, Y. Hu, J. Dotzel, C. De Sa, and Z. Zhang, "Improving neural network quantization without retraining using outlier channel splitting," in *International conference on machine learning*, PMLR, 2019, pp. 7543–7552.

[74] T. Han, D. Li, J. Liu, L. Tian, and Y. Shan, "Improving low-precision network quantization via bin regularization," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5261–5270.

[75] C. Xu *et al.*, "Overcoming the challenges of crossbar resistive memory architectures," in *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, IEEE, 2015, pp. 476–488.

[76] S. Mittal, "A survey of reram-based architectures for processing-in-memory and neural networks," *Machine learning and knowledge extraction*, vol. 1, no. 1, pp. 75–114, 2019.

[77] W. Li, P. Xu, Y. Zhao, H. Li, Y. Xie, and Y. Lin, "Timely: Pushing data movements and interfaces in pim accelerators towards local and in time domain," in *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, IEEE, 2020, pp. 832–845.

[78] Google supercharges machine learning tasks with TPU custom chip, https://cloudplatform.googleblog Google-supercharges-machine-learning-tasks-with-custom-chip.html.

[79] F. Zhang and M. Hu, "Mitigate parasitic resistance in resistive crossbar-based convolutional neural networks," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 16, no. 3, pp. 1–20, 2020.

VITA

Payman Behnam received his bachelor's degree in Computer Engineering from Shiraz University and two master's from the University of Tehran, Tehran, Iran, and the University of Utah, Salt Lake City, Utah in Computer Engineering and Computer Engineering. He joined Georgia Tech in the Fall 2020 to start his Ph.D. in Electrical & Computer Engineering.

His research interests center on high-performance/energy-efficient design at the intersection of machine learning and systems. As a graduate student, his experience spans different levels of abstraction from fabrication up to the system and algorithmic levels. Currently, he is working on efficient DNN computing, model compression, and resource management for machine learning frameworks and applications. His research has been honored with several awards such as the Nvidia Graduate Fellowship, the Qualcomm Innovation Fellowship, Les Palukti Best Student Poster Paper Award at GOMACTech23 conference, and the Featured paper of the Month at TC'21 Journal.