ResiRCA: A resilient energy harvesting ReRAM crossbar-based accelerator for intelligent embedded processors

Keni Qiu[1], Nicholas Jao[3], Mengying Zhao[2], Cyan Subhra Mishra[3], Gulsum Gudukbay[3], Sethu Jose[3], Jack Sampson[3], Mahmut Taylan Kandemir[3] and Vijaykrishnan Narayanan[3]

[1]Capital Normal University, Beijing, China, (Email: qiukn@cnu.edu.cn)
[2]Shandong University, Qingdao, China , (Email: zhaomengying@sdu.edu.cn)
[3]Pennsylvania State University, USA, (Email: {naj5075, cyan, gulsum, sethu}@psu.edu, {sampson, mtk2, vijay}@cse.psu.edu)

***Abstract*—Many recent works have shown substantial efficiency boosts from performing inference tasks on Internet of Things (IoT) nodes rather than merely transmitting raw sensor data. However, such tasks, e.g., convolutional neural networks (CNN), are very compute intensive. They are therefore challenging to complete at sensing-matched latencies in ultra-low-power and energy-harvesting IoT nodes. ReRAM crossbar-based accelerators (RCAs) are an ideal candidate to perform the dominant multiplication-and-accumulation (MAC) operations in CNNs efficiently, but conventional, performance-oriented RCAs, while energy-efficient, are power hungry and ill-optimized for the intermittent and unstable power supply of energy-harvesting IoT nodes.**

This paper presents the *ResiRCA* architecture that integrates a new, lightweight, and configurable RCA suitable for energy harvesting environments as an opportunistically executing augmentation to a baseline sense-and-transmit battery-powered IoT node. To maximize ResiRCA throughput under different power levels, we develop the *ResiSchedule* approach for dynamic RCA reconfiguration. The proposed approach uses loop tiling-based computation decomposition, model duplication within the RCA, and inter-layer pipelining to reduce RCA activation thresholds and more closely track execution costs with dynamic power income. Experimental results show that ResiRCA and ResiSchedule achieve average speedups and energy efficiency improvements of 8× and 14× respectively compared to a baseline RCA with intermittency-unaware scheduling.

***Keywords*-Energy harvesting, ReRAM crossbar, CNN, Reconfigurable hardware, Loop tiling, Computation scheduling**

I. Introduction

In recent years, inference tasks, such as convolutional neural networks (CNNs), have been integrated into an increasing number of embedded applications to process edge-device collected data locally [1]. Such integration grants IoT devices an important degree of independence from remote servers, which can be critical in deployments with challenging communication environments. However, continuing this trend onto ultra-low-power (ULP) IoT nodes presents clear design challenges due to the mismatch between the performance and computation requirements of CNNs and the limited resources of ULP platforms. Such platforms often already operate at their limits just in order to transmit sensed data at acceptable quality of service (QoS) rates for deployment-viable battery lifetimes, and may not have additional resources available for further computation.

For many inference tasks, it is known that multiplication-and-accumulation (MAC) is the dominant operation type. In CNNs, for instance, MACs between the feature map data and kernel weights comprise nearly 90% of the total operations [2], [3]. Resistive random-access memory (ReRAM) crossbars are regarded as a promising mechanism for accelerating CNNs with high energy-efficiency as they can perform MAC operations through analog current summation and can retain model parameters in memory during inactive periods with extremely low power overheads [3], [4], [5], [6], [7], [8], [9], [10]. In the remainder of the paper, we may shorten the term *ReRAM crossbars* to *ReRAMs*.

Despite the obvious potential synergy between ReRAM crossbar-based CNN accelerators (RCAs) and IoT applications needing CNN inference, there can remain substantial challenges in efficiently performing inference on an IoT device if it does not have either a high power or high stability power source. Given form factor constraints on energy storage, the former may be challenging, and energy-harvesting from sources such as solar, thermal, kinetic and radio frequency [11], [12], [13], [14], [15], [16] is notoriously unstable. While unstable power sources have been successfully utilized for applications in the IoT space [17], [18], [19], their use has not been heavily explored for RCA design. Current RCA approaches can be divided into two categories. The approaches in the first category employ precision-conservative high power consuming ReRAM circuits and organize numerous large scale ReRAMs [3], [4], [5], whereas those in the second category adopt simple ReRAM organizations that constrain their execution style (e.g., parallelism granularity), which disadvantages them in coping with both variances across different ReRAMs and changing power supply [6], [8]. However, neither of them is a good fit for energy-harvesting scenarios.

To address these challenges, this paper proposes and experimentally evaluates *ResiRCA*, a resilient ReRAM crossbar-based CNN accelerator. Supported by a reconfigurable lightweight hardware design, ResiRCA is able to activate scalable computations via a multi-dimension tuning strategy. ResiRCA is designed as an auxiliary co-processor, powered by energy-harvesting, that augments a baseline, battery-powered MCU-style IoT node that would otherwise transmit its data without performing inference. In this design paradigm, the basic low power, lightweight MCU system can enjoy the advantage of continuous operation without suffering power outages, while the compute-heavy inference tasks can be offloaded to the RCA during periods when power income is sufficiently high and to external systems otherwise. Such a system is capable of both continuously collecting data and computing CNNs locally near data. ResiRCA allows an RCA to adapt to changing harvested energy and, with our co-designed scheduling approach, ResiSchedule, it can achieve very high throughput. To the best of our knowledge, this is the first work that focuses on low power and reconfigurable RCA design from both the hardware and software angles targeting energy harvesting systems. This paper makes the following key **contributions**:

- **Low power, reconfigurable hardware design:** We propose a novel architecture that implements a lightweight and low power RCA to adapt to time-varying power resources. Furthermore, the proposed hardware is reconfigurable at a fine grain, to be able to dynamically activate different scaled computations, which can fit to the changing features of the underlying power resources.
- **Resilient computation scheduling:** We provide three knobs to schedule computation blocks in the proposed architecture: (i) loop tiling which decomposes MAC operations in a given layer (ReRAM) into small blocks, (ii) ReRAM duplication which provides opportunity to perform one-layer operations with multiple weight copies, and (iii) pipelining that can organize multiple ReRAM tiles to further exploit the harvested power. These knobs can be integrated to form sequential or pipelined computation modes. For each computation mode, we can derive the optimal activation solutions under each power level directed by the *power model* and *throughput model* offline. We propose ResiSchedule, which combines the advantages of the two computation modes to cope with different power levels during the course of execution.
- **Smooth schedule transitioning:** We identify smooth transition conditions to transfer as many partial results as possible from the last incomplete inference in one power cycle to the next power cycle with a different power level. In addition, we discuss how to keep the partial results in appropriate computation points with or without power prediction.

II. Motivation

To avoid negatively impacting the underlying system's QoS, we consider RCA-based acceleration for ULP IoT nodes as an opportunistic computation knob, operating solely on ambiently harvested energy, when available. In energy harvesting systems, there are two critical features, namely, *power strength* and *power window length*. First, the variance of input power strength can be quite large: peak power can be hundreds or thousands of times larger than average power. Second, the variance of the input power window, i.e., how long the power input stays at a given level, can be large as well.

It is known that RCAs achieve their highest efficiency when every cell participates in the MAC computations simultaneously [20]. However, naively integrating such an RCA renders its activation power requirement so high that the system will likely have very low duty-cycle on an intermittent supply and may never activate at all for weaker power sources unless a substantial energy store were added, which could be burdensome for form factor constraints in a system that already employs a battery for sensing and other non-inference tasks.

TABLE I
An example of different activation schemes for an eight-cycle power trace.

Power cycle	Harv. Power (μW)	Power consumption with full-size acti.(μW)/ Thr. (GiGa MACs/s)/ Power utilization	Power consumption with resilient acti. (μW)/ Thr. (Giga MACs/s) /Power utilization
1	50	0/Power failure/0/0%	0/Power failure/0/0%
2	100	0/Power failure/0/0%	80/25×1×1/0.312/80%
3	500	480/25×6×1/1.872/96%	480/25×6×1/1.872/96%
4	200	0/Power failure/0/0%	160/25×2×1/0.624/80%
5	250	0/Power failure/0/0%	240/25×3×1/0.936/96%
6	750	480/25×6×1/1.872/64%	720/25×3×3/2.808/96%
7	650	480/25×6×1/1.872/74%	640/25×2×4/2.496/98%
8	350	0/Power failure/0/0%	320/25×2×2/1.248/91%

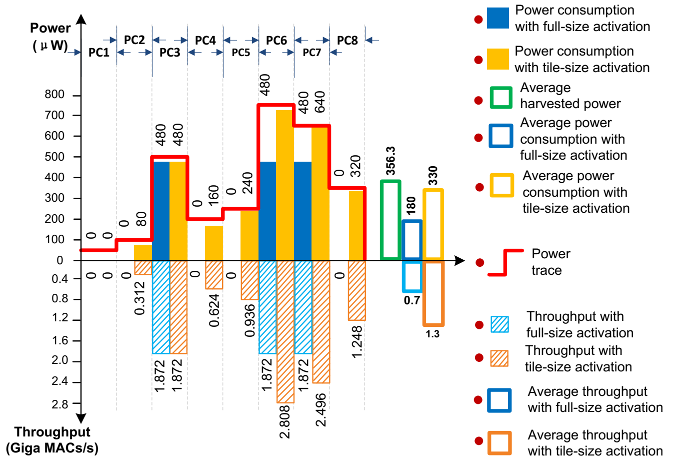


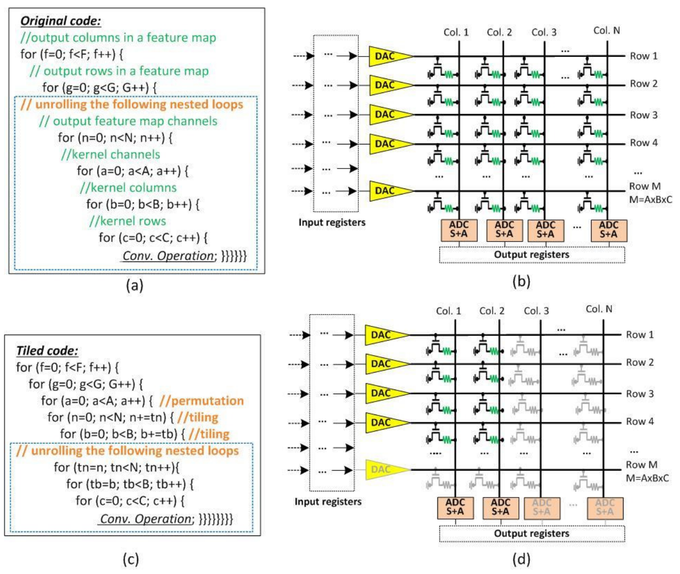
Fig. 1. Comparisons on power consumption and throughput with tile-size over full-size activation

From the perspective of an intelligent embedded system, the dominant power consuming part, the RCA, exhibits a highly parallel and uniform execution property. Under this context, if the power dominant RCA works in a fixed high-power mode, as in traditional RCA designs, there would be large mismatches between the harvested power and the consumed power. These mismatches can lead to the following two "nonideal" working scenarios:

(i) Unutilized energy: As long as the harvested power is less than the activation power requirement of one ReRAM, it is regarded as a power failure because the RCA is inactive. In this case, the harvested energy will leak away and cannot be recovered.

(ii) Underutilized energy: When the harvested power is much higher than the activation power of the RCA, the RCA can only work in the default lower energy consuming level. In this case, the unused energy will be wasted, resulting in low energy efficiency.

Considering the simple RCA working under a harvested power trace shown in Table II and Figure 1, the RCA consists of four 25×6 ReRAM crossbars, each can be mapped to six kernels, all sized $5 \times 5 \times 1$. In the default case, the RCA works under an either ON or OFF mode with a power threshold of *$80\mu W$*. During the eight harvested power cycles, the ReRAM can be ON during power cycles *PC3, PC6* and *PC7* and OFF with the other five power cycles. However, even when the system goes through the three power cycles, only some portion of the harvested power is consumed. As a result, the gap between the harvested power source and the consuming trace indicates a large energy waste from an RCA designed for efficiency under stable, high-power scenarios.



```
Original code:
//output columns in a feature map
for (f=0; f<F; f++) {
  // output rows in a feature map
    for (g=0; g<G; G++) {
// unrolling the following nested loops
    // output feature map channels
      for (n=0; n<N; n++) {
        //kernel channels
          for (a=0; a<A; a++) {
            //kernel columns
              for (b=0; b<B; b++) {
                //kernel rows
                  for (c=0; c<C; c++) {
                     Conv. Operation; }}}}}}
```

(a)

```
Tiled code:
for (f=0; f<F; f++) {
  for (g=0; g<G; G++) {
    for (a=0; a<A; a++) { //permutation
      for (n=0; n<N; n+=tn) { //tiling
        for (b=0; b<B; b+=tb) { //tiling
// unrolling the following nested loops
      for (tn=n; tn<N; tn++){
        for (tb=b; tb<B; tb++) {
          for (c=0; c<C; c++) {
             Conv. Operation; }}}}}}}}
```

(c)

Fig. 2. Comparisons on loop code and ReRAM activation with tile-size activation over full-size activation. (a) The original MAC codes; (b) The kernel loops are mapped to a full-size ReRAM activation; (c) The tiled MAC codes; (d) The tiled kernel loops are mapped to a tiled-size ReRAM activation

If we tentatively use loop tiling to decompose the MAC operations at the kernel level as shown in Figure 2 and perform the MAC operations on the ReRAM tile one by one sequentially, the system can achieve "continuous progress" under lower power supply. This is because the starting power requirement of the RCA is reduced, and the system can thus get through power failures and translate even the low input energy into forward progress. If only one tile is activated to perform the MAC operations at one time, the system can still make progress during time windows of power cycles *PC2, PC4, PC5* and *PC8* under limited power budget, as depicted in Table II and Figure 1. The annotations indicate the consumed power (μW), tile size and duplication count (e.g., 25x2x1) and power efficiency. With the resilient activation approach supported by loop tiling and ReRAM duplication, it can be seen that the power exploitation is increased from an average of 180μW to 330μW, and the throughput is increased by 85.7%. Note that the partial activation of computation cells can be realized by partially activating the peripheral circuits of the corresponding rows and columns in the ReRAM crossbar. *This resilient activation approach can effectively combat **Nonideal scenario 1***.

The underlying reason of encountering so many power failures in the case of conventional working mode is that the power threshold of the system to remain alive is set too high. With the loop tiling technique, the power failure threshold can be dropped to the requirements of the minimum activation tile of a ReRAM. With this, the RCA can be active in a very large power range and find more opportunities to make execution progress. Further, if the power supply is larger than the starting power threshold of one entire ReRAM, we can even arrange multiple ReRAMs to work in a parallel fashion, as seen in power cycles *PC6, PC7* and *PC8*. Parallel computations across multiple ReRAMs and loop tiling-based computation for each ReRAM are orthogonal optimizations.

Figure 2 shows the codes and ReRAM mapping schemes under full-size activation mode over tile-size activation mode. In this example, the full size of ReRAM (or loop nest) is $M \times N = A \times B \times C \times N$, and the tile size of ReRAM (or loop nest) is $m \times n = 1 \times tb \times C \times tn$. Since each single ReRAM can be activated at a finer granularity with tiling, the parallelism can be achieved under a flexible range of power consumption to match a variable power supply. Furthermore, the better the high-power supply can be aggressively utilized, the more ambient energy can be continuously extracted without increasing the energy storage capacity for the energy-harvesting power delivery system. *Therefore, this weight duplication-based execution style built upon fine-granularity activation can effectively combat **Nonideal scenario 2***.

Figure 1 also shows the throughput under the full-size activation mode and the tile-size activation mode. The flexible working mode based on the loop tiling technique can achieve more forward progress and higher power utilization. Extending this single-ReRAM toy architecture to a practical multi-ReRAM architecture to process multi-layer convolutions for real-world CNNs introduces a new source of power variation in terms of the different time and power costs of different convolution layers. In this scenario, the idea of integrating tiling on ReRAMs and paralleling ReRAMs, can also achieve high energy efficiency.

III. System level flow

This section presents the system level design of ResiRCA from both the hardware and software perspectives. Below, we provide an overview of the RCA interfaces and our system integration model and then discuss the two main steps to map a CNN to ResiRCA: *offline compilation* and *runtime execution*.

A. ResiRCA overview

Figure 3 shows the conceptual architecture of an intelligent embedded system where an RCA is added to an existing MCU system. The baseline, battery-powered MCU system samples data at a fixed rate, supported by the provisioned battery, and transmits either sensor data or the results of RCA processing

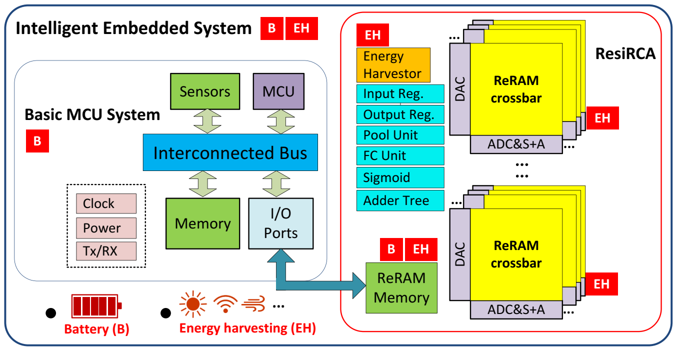


Fig. 3. ResiRCA architecture overview

to the network. The RCA is powered by harvesting ambient energy and employs a separate, very small capacitor as its only energy storage medium, primarily for power smoothing, similar to prior energy-harvesting NVP designs [17], [21], rather than as a task-scaled energy reservoir [22]. Note that the ReRAM memory depicted in Figure 3 functions as both data storage for the sensors and input/output storage for the RCA; so, it must be able to operate from both the battery and harvested power sources. Similar hybrid arrangements have been explored in the NVP literature [23] and impose minimal design overheads. The baseline MCU system is also augmented with a power-level and RCA activity feedback mechanism from the energy-harvesting portion of the platform to allow initial MCU programming of the RCA control registers and model parameters and RCA completion notifications.

B. Mapping inference tasks to ResiRCA

To achieve both generally low power and intermittency-compatible execution, the proposed ResiRCA architecture has the following two features that impact the software management of the RCA:

Lightweight: From the perspective of the ReRAM circuit at the core of the RCA, the precision and resolution of inputs, weights and outputs are kept low to yield low power. This entails that only models trained or adapted to low-precision implementations can be used with ResiRCA. Similarly, total model size, including any granularity overheads (e.g., from the partitioning used to store both positive and negative weights by having the kernels of one layer mapped to two crossbars, one each for positive and negative weights, which share the same input port) must fit within the allocated RCAs of a particular ResiRCA design.

Fine-grained reconfiguration: The ResiRCA architecture supports not only partial activation for one ReRAM or multiple ReRAMs, but also sequential and pipelining execution modes. This flexible reconfigurability enables fine-grained activations to exploit the harvested power. While execution is relatively straightforward when maintaining a specific configuration of tiling and pipelining strategy, transitions between configurations require additional management and power-intermittency awareness to preserve progress from partial executions after power level transitions and failures. The hardware design details will be presented in Section IV.

As part of compiling a CNN to ResiRCA, we build a profiling table relating each potential tiling and pipeline configuration that might be used with the target CNN with its ReRAM model resources, activation requirements, and power draw. This profiling collects data used to determine the best activation solution for each power level.

At runtime, each time when entering a new power cycle, we first check the statically determined solution tables and pick up the corresponding activation solution for the ReRAMs for the current power level. Then, the execution process the "*data loading$\rightarrow$ Mac computing$\rightarrow$ data storing*" steps in a sequential way to perform convolution operations. The hardware design details will be presented in Section V.

IV. A HARVESTING-COMPATIBLE, LOW-POWER RESIRCA

Supporting the necessary features for adapting RCAs to a harvested power supply will require optimizations in both RCA circuit design and the development of variable-power-optimized loop-tiling strategies. First, feasible implementations of flexible activation options require a low power and reconfigurable RCA. Second, a dynamic loop tiling strategy alongside a coordinated parallelism scheme should be devised to match execution power consumption as closely as possible to power income to maximize efficiency. This section addresses the first of these challenges, and Section V discusses our approach to the second.

Challenge 1: *Achieving low-power, reconfigurable RCA*

Although recent works have presented systems [4], [3], [5] and circuits [24], [6] for inference-oriented RCAs, they are not directly suitable for adoption in our target scenario because of either their high power consumption or their stringent execution parameters (e.g., computation granularity). In general, these designs are not optimized for enabling the small-scale partial activation on ReRAM that would allow for power tracking in an energy-harvesting environment.

Figure 4 shows five harvested power sources with the maximum, mean and median values and their ratios indicated. It has been shown that the power requirement to fully activate a 128×8 sized ReRAM and obtain 8 outputs concurrently is more than 24mW [3]. With this design, the presented power sources can hardly activate even a small ReRAM. In order for the ResiRCA to operate on harvested power, it must reduce minimum ReRAM activation power. Therefore, the the RCA should be built on the basis of a low power hardware design that is upwardly reconfigurable to higher power scenarios rather than the reverse.

One approach to achieve lower RCA power is to limit precision. The impact of different precisions on CNN accuracy has been extensively studied [24], [25], [26], [27], [28], [29], [30], [31], [32], [33]. To meet our power constraints while preserving reasonable accuracy, we adopt a 4-bit input with a resolution of 1-bit, a cell resolution of 1-bit and a 4-bit output. With this design setting, the ReRAM size only needs to be equal to the kernel size, and the ReRAM scale does not need to be extended using a bit composing scheme (e.g. as in ISAAC [3]). In addition, it consumes very low power to handle the Successive-Approximation Register (SAR)-ADC referencing with the 4-bit output resolution. Moreover, we adopt the 1T1R technique to build high resistance and low

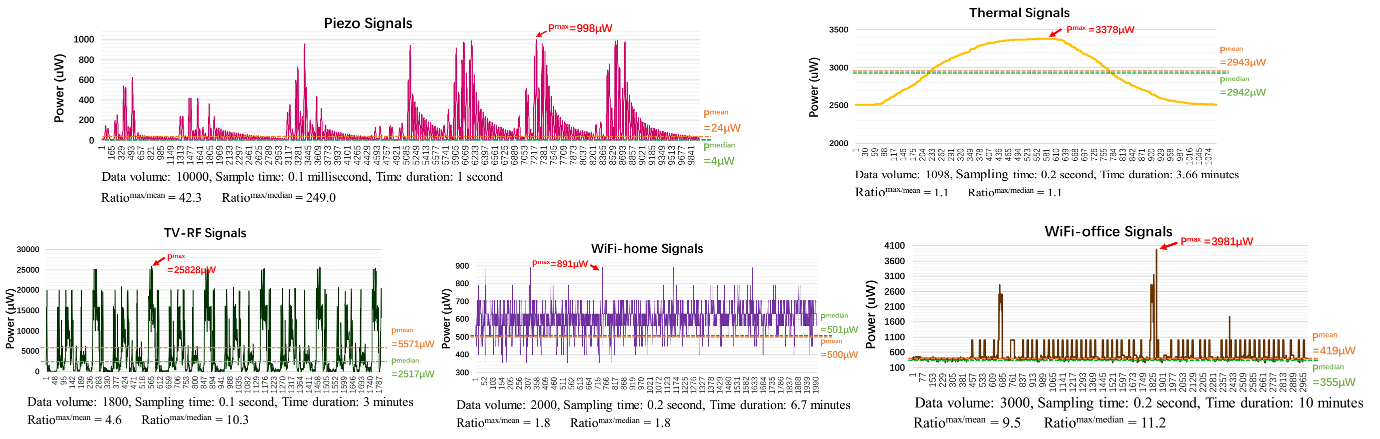


Fig. 4. Variance feature of different power sources

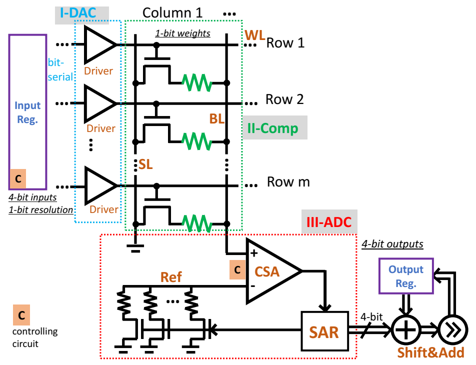


Fig. 5. Lightweight ReRAM circuit design

power ReRAM cells. Figure 5 shows the proposed peripheral circuit design for one ReRAM crossbar. This design is more concise even than the SINWP [6], because we target low power as the primary goal.

To increase efficiency further our design supports aggressive power gating and other circuit techniques to dynamically reconfigure active tile sizes and shut-off inactive ReRAMs. Specifically, we employ clock gating and input vector control (IVC) techniques to further reduce leakage in inactive rows. We modify the column multiplexers to enable variable active columns and turn off the ADCs of inactive channels.

Lastly, we apply coarse-grain power gating to configure the number of duplicated ReRAMs. This reconfiguration ability can enable scaled activation of the circuits such that small tile-size computation can be enabled while yielding very low power consumption.

V. Power-dynamic RCA scheduling

Given a viable RCA architecture for energy-harvesting IoT nodes, the other key issue is the design of a software scheduling mechanism to choreograph resilient execution on this architecture.

Challenge 2: ***Software controlled dynamic RCA activation and scheduling***

The idea of loop tiling has been widely leveraged in RCA design to either increase system throughput by smoothing the pipelining or reduce memory accesses by improving data locality [34], [3]. In this work, we re-purpose loop tiling to perform computation decomposition on ReRAM accelerated MACs. Moreover, we allow parallelism along different dimensions to seamlessly integrate it with loop tiling, and as a result, a range of scalable computations that can fit in different power supplies are achieved. With this design idea, the system can keep making forward progress over a large range of power incomes. Sections V-A-V-C develop a dynamic activation strategy for different power levels and Section V-D discusses the transition strategy between dynamic activation solutions.

A. Computation decomposition and parallelism

If the harvested power P^{budget} is larger than the power requirement of activating the smallest size of ReRAM, it implies that the RCA is active and can make computation progress. For active RCAs, one option is to use loop tiling to decompose computations, and the other is to parallelize computations. The parallelism in this context is of two types: *intra layer parallelism via layer duplication* and *inter layer parallelism via layer pipelining*. Further, tiling and parallelization can also be combined to generate fine-grained scales of computations to efficiently fit into the changing harvested power.

1) Computation tiling: In this work, we use loop tiling [35] to decompose large parallel MAC operations into smaller parallel blocks and execute the resulting blocks one by one. As shown in Figure 2, if loop tiling is applied to the unrolled MAC operations, only a tile of ReRAM cells along with their peripheral circuits is enabled to perform MAC operations. After traversing all the tiles one by one, one batch of MAC operations on the entire ReRAM is completed. Note that, if the row-wise tiling factor is less than the ReRAM row number, this tiled execution strategy will introduce partial sums. When the traversal completes, an *Adder Tree* will be used to merge the partial sums for ReRAM columns and obtain the final MAC result.

2) Computation parallelism: Intra-layer parallelism means overlapping the layer computations on duplicated copies of ReRAMs that store the same weights for one layer. We use parallelism granularity G to denote the duplication count as defined in [5]. G can be determined considering the tradeoff between energy efficiency and chip area during the design phase.

In this work, the parallelism granularity G of a layer is determined by the ratio between 50% of the peak harvested power during profiling and the power consumption of the full-size ReRAM corresponding to this layer. That is, if the 50% peak harvested power by profiling is twice (G=2) of the power consumption with a ReRAM size of Layer 1 of 25x6, the RCA will be designed to offer two sets of ReRAMs sized 25x6 for Layer 1. The actual parallelism granularity $aG \leq G$ for a layer is decided by the harvested power level. If we allow aG ReRAMs to perform the concerned layer's computations in parallel, the input data should be divided into aG partitions. In this way, the data in the same partition are processed in a sequential fashion whereas the data in different partitions are processed in parallel. This offers a flexible way to tune the power consumption in a large design space, even though the kernel size, convolution count and power consumption of different layers can significantly vary.

Inter-layer parallelism means overlapping ReRAM computations for different convolution layers in a pipelined fashion. This pipeline parallelism provides us with another dimension to aggressively exploit the harvested energy. We can naturally integrate the duplication based parallelism into the pipeline parallelism to build a parallelization strategy where the pipeline stages are composed of ReRAMs mapped from different convolution layers. Previous work [3] has noted vulnerabilities to pipeline bubbles and execution stalls in CNNs because of the large variance in weight and feature map scales across different layers. In this work, the pipeline imbalance issue is addressed by tuning the activation degrees, duplication degrees and even the pipeline execution style in a very fine-grain fashion.

3) Execution strategies: Figure 6 shows five different execution strategies for a two-layer convolution execution experiencing two power cycles with different levels. In Figure 6(a), a naive scheduling strategy is employed on a *Simple architecture*. In this scheduling strategy, the RCA is only active when the harvested power is adequate to support the maximal power requirement among all the convolution layers and, in the "simple" architecture, one convolution layer can only be mapped to one ReRAM (no ReRAM duplication).

In the remainder of this paper, this execution strategy is referred to as *Naive1*. We assume that *Naive1* is also designed with the proposed lightweight circuits.

In Figure 6(b), a naive scheduling scheme is applied, but this time on the proposed ResiRCA architecture, which supports ReRAM duplication. This execution strategy is referred as *Naive2*. None of *Naive1* and *Naive2* executions can go through power cycle *PC-i* and the power utilization is very low, as there is a significant mismatch between the power producer and consumer.

Figure 6(c) presents a flexible scheduling strategy applied to ResiRCA. In this strategy, the loop tiling technique integrated with the ReRAM duplication is enabled to obtain resilient MAC computation blocks. The layers are scheduled in a sequential fashion. This execution style is called *Sequential*. In the example, we allow activating L1 ReRAMs with aG=4 and one partial L2 ReRAM in a sequential fashion.

In Figure 6(d), the loop tiling technique integrated with both duplication and pipelining is used to schedule all layers on the

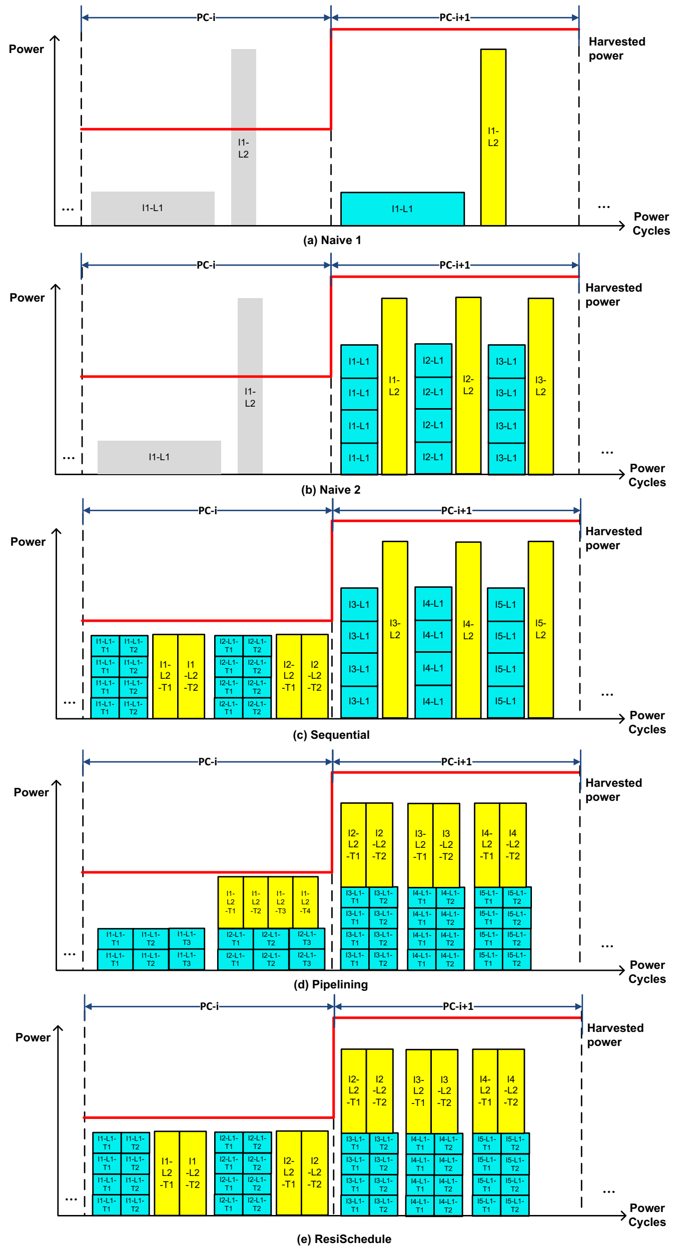


Fig. 6. Five layer scheduling schemes: (a) Naive execution @Simple architecture; (b) Naive execution @ResiRCA architecture; (c) Sequential resilient execution @ResiRCA architecture; (d) Pipelining resilient execution @ResiRCA architecture; and (e) Hybrid resilient execution @ResiRCA architecture

ResiRCA architecture; we call this execution style *Pipelining*. For ease of explanation and simulation, we only allow full pipelining in this execution, which means MAC operations of all the layers are included in each pipeline stage.

Finally, Figure 6(e) shows the loop tiling technique integrated with a hybrid parallelism scheme. We refer to this as *ResiSchedule*. At runtime, ResiSchecule dynamically selects activation solutions from either *Sequential* or *Pipelining* in each power cycle, depending on which can provide a better throughput. With *ResiSchedule*, we can cover a large tuning range commensurate with power supply variation. Section V-C will further present quantitative analysis and solution on how to figure out the optimal activation size, duplication degree and execution style to achieve an ideal ResiSchedule for ResiRCA.

B. Power model and latency model

Power supply is a significant constraint for ResiSchedule. By analyzing the power cost of each step of the convolution operations, we can build a power model related to the activation solution $\langle m,\ n,\ aG\rangle$ where *m, n,* and *aG* denote row factor and column factor of the ReRAM tiling and the actual parallelism granularity of ReRAM duplication copies. ResiRCA power consumption divides into three major parts from an architectural viewpoint, P^{load}, P^{comp} and P^{store}, and they are performed in sequence.

1) Load and store: P^{load} and P^{store} denote the power consumed by loading the data from the pure ReRAM memory into the input registers and storing the data from the output registers into the ReRAM memory. $P^{load} = aG \times (Bits^{input}/BN^{in}) \times P^{ld-bit}$ models load power operation. Here, the terms $Bits^{input}$ and BN^{in} denote the number of input bits to serve MAC operations for a full-sized ReRAM and the batch number of transferring these input bits respectively. Therefore, Bit^{input}/BN^{in} means the actual loaded data bits each batch. The term P^{ld-bit} denotes the power consumption of loading one bit from ReRAM memory to the input register. The input batch number BN^{in} is determined by the power budget because $P^{load} <= P^{budget}$ should always be satisfied.

The latency model for data load for one convolution operation is $Lat^{load} = Bits^{input}/BW^{ld}$. The term Lat^{load} represents the latency to load the data required by the convolution operations for one-cycle MAC operations for a full-size ReRAM. The term BW^{ld} denotes the bandwidth of each load operation. The models of P^{store} and Lat^{store} can be derived in a similar fashion.

2) Computation on ReRAMs: P^{comp} is the dominant and most complicated part where the analog and digital signals are mixed. The energy of one-cycle MAC operations for an activation size of $m \times n$ and actual duplication *aG*, $P^{comp-tile}$ divides into the following parts: 1) E^{DAC} denotes the energy consumed for converting the digital input signal to the analog signal in a bit-serial fashion; 2) E^{MAC} denotes the energy for performing MAC operations on ReRAMs; and 3) E^{ADC} consists of three parts as shown in Figure 5: 3i) E^{BL} denoting the energy for activating bit lines; 3ii) E^{SA-Ref} denoting the energy for sensing and amplifying the MAC result signal and then referencing analog signals to digital signals; and, 3iii) E^{S+A} denoting the energy of Shift&Add part to compose the final output.

In the ResiRCA design, the time for performing one-cycle of MAC operations on one ReRAM tile is fixed as $Lat^{comp} = T^{comp}$, and is independent of the activation size. Therefore, we can build the power model for the computation part in terms of a tile as shown in Equation 1.

$$\begin{aligned} E^{comp} =& E^{comp\ \ tile}/Lat^{comp} \\ =& (E^{DAC} + E^{MAC} + E^{ADC})/T^{comp} \end{aligned} \tag{1}$$

The power of each part is taken to be linear to the tiling factors of *m* or *n* or the actual parallelism granularity *aG.* The energy for one ReRAM row (e^{DAC}), one ReRAM cell (e^{MAC}) and one ReRAM column (e^{BL}, e^{SA-Ref}, e^{S+A}) are the worst-case values from the simulation. Table V-B2 presents the relationship of energy and the ReRAM tiling size and ReRAM copies.

TABLE II
RELATIONSHIP OF ENERGY AND THE RERAM TILING SIZE AND RERAM COPIES.

Component		Energy equation
DAC		$E^{DAC} = e^{DAC} \times m \times aG$
Computation		$E^{MAC} = e^{MAC} \times m \times n \times aG$
ADC	BL	$E^{BL} = e^{BL} \times n \times aG$
	SA-Ref	$E^{SA-Ref} = e^{SA-Ref} \times n \times aG$
	S+A	$E^{S+A} = e^{S+A} \times n \times aG$

3) Partial sums: The computation decomposition across ReRAMs by loop tiling may produce partial sums for the activated tiles when each column in the tile is not fully activated. As a result, these partial sums need to be merged once the (tile) traversal of an entire ReRAM is complete. The sum merging operation is performed by an *Adder Tree* as illustrated in Figure 3. We can provide tree topology candidates for different power levels, keeping in mind that the constraint of $P^{merg} < P^{budget}$ should be always met. Therefore, the power P^{merge} and latency Lat^{merge} of the partial sum merging operation under different merging cases can be obtained offline.

4) Activation transition cost: The execution transition from one tile to another inside one power cycle or from one activation solution to another in different power levels also costs power P^{trans} and latency Lat^{trans}. Activation transition implies that we need to enable the corresponding circuits of the to-be-activated rows and columns while shutting down the others. This function is supported by the gating circuits described in Section IV. This cost will be only counted at the beginning of a power cycle when an activation transition occurs.

5) Power and latency models: The above analysis captures the power consumption and execution latency of processing one convolution layer. It is assumed that all of these steps are performed in sequence. Putting them all together, for convolution layer *Lk*, the power and latency pair can be modeled as in Equation 2 and 3.

$$\begin{aligned} P_{LK} =& (P^{ld} \times \sum Lat^{ld}_{Lk} + P^{comp} \times \sum Lat^{comp}_{Lk} \\ &+ P^{st} \times \sum Lat^{st}_{Lk} + P^{merge} \times \sum Lat^{merge}_{Lk})/Lat_{Lk} \end{aligned} \tag{2}$$

$$Lat_{LK} = \sum Lat^{ld}_{Lk} + \sum Lat^{comp}_{Lk} + \sum Lat^{st}_{Lk} + \sum Lat^{merge}_{Lk} \tag{3}$$

Considering processing multiple inferences with multiple convolutional layers LC ($LC \geq 2$), the layers can be scheduled for either *sequential* or *pipelined* computation mode, as shown in the examples in Figures 6(c) and (d), respectively. Based on the models capturing one layer in Equations 2 and 3, we can build the models for each convolution layer of a CNN application. Under the sequential computation mode, the convolution layers are executed one by one in a sequential fashion, and as a result, the power model and latency model in Equations 2 and 3 can be directly used. We also model the pipelined computation mode shown in Equations 4 and 5. Note that the pipelined computation mode means that all the *LC* convolution layers are executed fully parallel.

$$P^{pipe} = \sum_{Lk=1}^{LC} P_{Lk} \tag{4}$$

$$Lat^{pipe} = max(Lat_{L1}, Lat_{L2}...Lat_{LC}) \tag{5}$$

C. Dynamic activation strategy

1) Problem formulation: In this section, we focus on figuring out the ResiSchedule solution to achieve the maximal throughput. Although we can arrange more hardware resources with the pipelined computation mode, it does not mean this mode will always yield greater computation progress than the sequential mode due to the constraints of tile size and parallelism granularity. Therefore, in order to achieve optimal progress, we need to select the best activation solution offered by both the computation modes. Given a power supply level, we can derive the optimal tile size and actual duplication granularity to form the activation solution $\langle m,\ n,\ aG\rangle$ for sequential or pipelined computation modes, respectively. Then, a global activation strategy can pick up the best one of these two and generate a hybrid solution for the concerned power level.

Throughput model Achieving the maximal computation progress under the harvested energy has two implications. The first one is that we expect more energy can be used for program progress. The other is more subtle in that we expect the power can be consumed quickly in order to receive more energy from outside. In this regard, the metric *throughput* measured by computations (convolutional MACs) per second is a useful proxy for ResiRCA in energy-harvesting scenarios. We use the number of convolutional MAC operations to represent the computations. For the sequential computation mode, the throughput for Layer *Lk* can be expressed as below:

$$Thr_{Lk}^{sequ} = \frac{(m \times n)_{Lk} \times aG_{Lk}}{Lat_{Lk}} \tag{6}$$

The average throughput with a *LC*-convolution CNN inference can be expressed as shown below.

$$Thr_{ave}^{sequ} = \frac{\sum_{Lk=1}^{Lk=LC}(m \times n)_{Lk} \times aG_{Lk}}{\sum_{Lk=1}^{Lk=C} Lat_{Lk}} \tag{7}$$

For the pipelining computation mode, all the *LC* layers are executed in parallel. The throughput can be expressed as follows:

$$Thr_{ave}^{pipe} = \frac{\sum_{Lk=1}^{Lk=LC}(m \times n)_{Lk} \times aG_{Lk}}{Lat^{pipe}} \tag{8}$$

2) Activation strategy formulation : The activation strategy for the sequential mode can be described as shown below. In order to formulate the problem in a concise way, the tiling factors, *m* and *n*, are constrained to be divisors of the ReRAM weight matrix M× N using the annotations of $m \mid M;\ n \mid N$.

Objective: Maximize Thr_{ave}^{sequ}
Subjected to: for each layer Lk,
$P_{Lk}^{load},\ P_{Lk}^{store},\ P_{Lk}^{comp},\ P_{Lk}^{trans},\ P_{Lk}^{merge} < P^{budget};$
$aG_{Lk} < G_{Lk};$
$m_{Lk} \mid M_{Lk};\ n_{Lk} \mid N_{Lk};$
Solution output: $\langle m_{Lk},\ n_{Lk},\ aG_{Lk}\rangle$ for each layer Lk

Similarly, the activation strategy under the pipeline execution mode can be described as below.

Objective: Maximize Thr_{ave}^{pipe}
Subjected to:
$\sum P^{load},\ \sum P^{store},\ \sum P_{pipe}^{comp},\ \sum P^{trans},\ \sum P^{merge} < P^{budget};$
$aG_{Lk} < G_{Lk};$
$m_{Lk} \mid M_{Lk};\ n_{Lk} \mid N_{Lk};$
Solution output: $\langle m_{Lk},\ n_{Lk},\ aG_{Lk}\rangle$ for each layer Lk

By solving the above problems, we can obtain activation solution $\langle m,\ n,\ aG\rangle$ for each power level under the sequential and pipelining computation modes, referred to as SOL^{sequ} and SOL^{pipe} respectively. Then the optimal solution can be selected from among them. The solution space is actually very small because of the constraints that tile size candidates and *aG* are all bounded in the integer domain. It may happen that multiple equivialent solutions can be obtained either for sequential computing mode or pipelining computing mode. In this case, we choose the activation solution with larger tiling size by taking into account the transition cost. If the tiling sizes of output solutions are the same, we choose larger *m* because larger *m* implies fewer partial sum adds.

D. Activation transition

The power instability of energy harvesting implies that the activation solution needs to change dynamically as power level changes. Figure 7 shows the finite state machine (FSM) directing transition strategy. An FSM transition can happen between any pair of power levels.

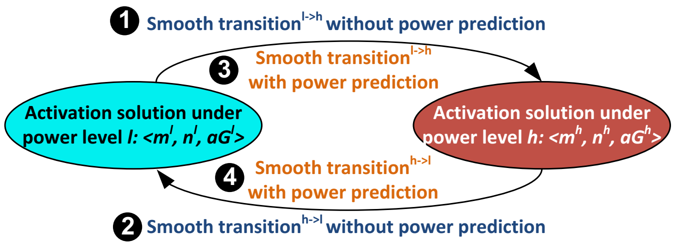

❶ Smooth transition^l->h without power prediction
❸ Smooth transition^l->h with power prediction
Activation solution under power level l: <m^l, n^l, aG^l>
Activation solution under power level h: <m^h, n^h, aG^h>
❹ Smooth transition^h->l with power prediction
❷ Smooth transition^h->l without power prediction

Fig. 7. Activation solution transition FSM

The convolution computations of one inference may not be completed while transitioning to a new power level. For small-scale applications with strong harvested power supply, discarding the incomplete execution may have only modest overheads. However, for large-scale applications with weak harvested power supply, it is highly desirable to maintain the already-obtained results and smoothly transfer them to the next power cycle.

Fortunately, there exist opportunities to keep and transfer the intermediate computation results of the last incomplete inference to the next power cycle. Considering a transition for one layer from an activation solution $\langle m1, n1, aG1 \rangle$ to $\langle m2, n2, aG2 \rangle$, we find that, if the expression $Condition^{trans}$: $(m1 = m2)\&(n2 \mid (Tile^{count1} \times n1))\&(aG1 = aG2)$ is true for each convolution layer, the activation solution $\langle m1, n1, aG1 \rangle$ with power level PL1 can be transferred to be equivalent to an execution of activation solution $\langle m2, n2, aG2 \rangle$ with PL2. The execution equivalency can be transferred by $Tile^{count2} = Tile^{count1} \times n1/n2$.

The next power cycle's level information cannot be known with certainty. In order to not discard the acquired results, we can search the maximal $^{*}Tile^{count1}$ where $^{*}Tile^{count1} \leq Tile^{count1}$ to make the expression $n2 \mid (^{*}Tile^{count1} \times n1)$ conservatively true for each layer, and then transfer the tile count $^{*}Tile^{count1}$ to accommodate to the new activation solution. This means that only a portion of computation results from $^{*}Tile^{count1}$ to $Tile^{count1}$ will be discarded. The discussion on this transition without power prediction is applied for transitions ❶ and ❷ in the Figure 7. We call this smooth transition strategy as $Transition^{Keep}$.

With a power predictor [36], we can estimate the power level of the next power cycle in advance. In this way, we can, when accurate, make a much smoother transition when transferring the results of an incomplete inference. Still, with $Condition^{trans}$ satisfied, the smooth transition can be achieved in a similar way. Otherwise, if the expression $Condition^{trans}$ is not satisfied, with power prediction, we can still speculatively attempt to perform a smooth transition. There are two strategies to achieve this.

- We can search for an intermediate power level, where we first switch to the activation solution corresponding to this intermediate power level and then switch to the solution of the actual power level – this strategy is called *Multi-step Transition*.
- If the power predictor reports a power transition from a high level to a low level, we can move to the new activation solution before performing the last incomplete inference – this strategy is referred to as *Eager Transition*.

The discussion on this transition with power prediction is applied for transitions ❸ and ❹ in the Figure 7. Since a power predictor itself consumes power, it makes sense to employ it for large-scale applications under weak power sources where discarding a portion of computations may impose a big loss or for the scenarios where power level transitions happen frequently.

VI. Experiments

To evaluate ResiRCA, we have extended the Gem5 [37] simulator with RCA modeling. The basic MCU is built on an ARM core, and the entire system runs on a 200MHz clock. The energy harvesting mechanism is supported by the power management unit, which can record power production and consumption at an execution cycle level. The added RCA module consists of ReRAM crossbars, the activation solution table, and other function units (e.g., Pooling, FC, sigmoid) of the CNNs. We perform cycle-accurate simulation for the MAC computations based on tile activation and the data load/store process. However, for other functional units, we assign a fixed latency. For the ReRAM circuit simulation, we quantify power and performance of our design in HSPice [38] using 20nm FinFET ReRAM parameters from [39]. Load/store parameters of the ReRAM memory are from NVSim [40]. The main parameters of our simulations are given in Table VI. Four practical CNNs listed in Table VI are evaluated on the five power traces illustrated in Figure 4 for each of the five execution strategies from Section V-A3: *Naive1*; *Naive2*; *Sequential*; *Pipelining*; and *ResiSchedule*.

TABLE III
ReRAM parameters

ReRAM computing crossbar	DAC(150×N×1 title)	5.4 pJ
	ADC(M×1×1 title)	749 fJ
	S+A(M×1×1 title)	41.6 fJ
	area (all peripherals included)	2950.47 μm^2
ReRAM memory (16KB)	bandwidth	128 bit/s
	read energy	37.993 pJ
	read latency	1.577 ns
	write energy	95.412 pJ
	write latency	20.09 ns

TABLE IV
IoT-Practical CNN workloads

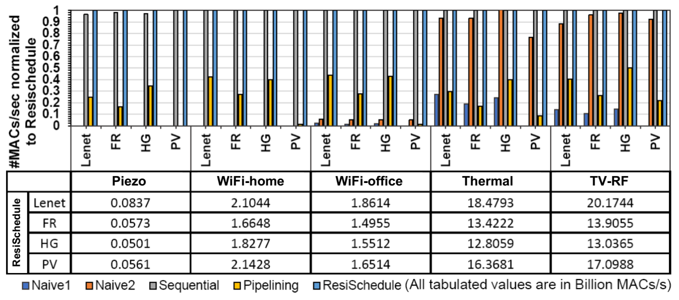
CNN	Layer	Kernel	RRAM Size	Acti. power	Input
PV [41]	Input				1@50×50
	Conv1	8@6×6×1	36×8	752.2μW	8@45×45
	Conv2	12@3×3×8	72×12	1125.6μW	12@20×20
	Conv3	16@3×3×12	108×16	1526μW	16@8×8
	Conv4	10@3×3×16	144×10	1114μW	10@6×6
	Conv5	6@3×3×10	90×6	676.2μW	6@4×4
FR [42]	Input				1@32×32
	Conv1	4@5×5×1	25×4	377.4μW	4@28×28
	Conv2	16@4×4×4	64×16	1433.6μW	16@10×10
LeNet [43]	Input				1@32×32
	Conv1	6@5×5×1	25×6	539.7μW	6@28×28
	Conv2	16@5×5×6	150×16	1614.2μW	16@10×10
HG [44]	Input				1@28×28
	Conv1	6@5×5×1	25×6	539.7μW	6@24×24
	Conv2	12@4×4×6	96×12	1176μW	12@8×8

For each application on each power trace, we report the throughput and energy efficiency under the five different execution strategies. We then demonstrate the benefits from the proposed smooth transition strategy and power prediction. We also study the sensitivity of our proposed approach to available ReRAM hardware resources.

A. Throughput

Figure 8 shows the throughput comparison of the five execution strategies. The bars are all normalized to *ResiSchedule*. The included table gives the absolute values of throughput by ResiSchedule. The results show that ResiRCA and ResiSchedule combine to achieve an average throughput improvement of 8× compared to a baseline RCA with intermittency-unaware scheduling. One can make the following observations and analyses from these results:

- For each workload with each power source, *ResiSchedule* always achieves the highest throughput because it combines the best activation solution in each power cycle. The results of *Naive1* are the worst because it lacks both adequate hardware resources and scheduling flexibility. Although *Naive2* is based on the *ResiRCA* architecture, the throughput is still relatively low because it lacks scheduling adaptation to fit to the changing harvested power.
- The results of *ResiSchedule* are very close or equal to that of *Sequential* under most cases. When we track the simulation cycles, it is found that the throughput of *Sequential* solution



		Piezo	WiFi-home	WiFi-office	Thermal	TV-RF
ResiSchedule	Lenet	0.0837	2.1044	1.8614	18.4793	20.1744
	FR	0.0573	1.6648	1.4955	13.4222	13.9055
	HG	0.0501	1.8277	1.5512	12.8059	13.0365
	PV	0.0561	2.1428	1.6514	16.3681	17.0988

(All tabulated values are in Billion MACs/s)

Fig. 8. Throughput of CNNs across the power sources normalized to ResiSchedule

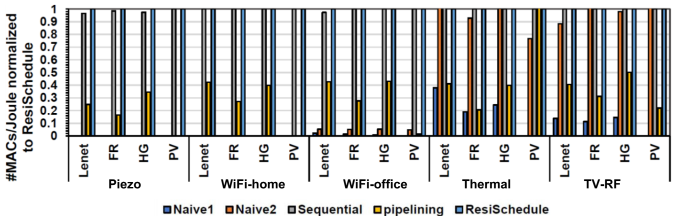


Fig. 9. Energy efficiency of CNNs across the power sources normalized to ResiSchedule

is higher than that of *Pipelining* solution for a significant fraction of the active power cycles. That is, the selection ratio of *Sequential* is much higher than the ratio of *Pipelining* in *ResiSchedule* solutions in the whole power trace.

• Consistent with the above observation, the entire *Sequential* strategy competes with the entire *Pipelining*. The reason is that the active power threshold of *Pipelining* is much higher than that of *Sequential*. As a result, fewer power cycles of *Pipelining* are available than that of *Sequential*. However, we believe that this is highly related to the default ReRAM duplication assignment in the experiments. When we change to a smaller ReRAM duplication granularity *G*, we find that the throughput of *Pipelining* is better than that of *Sequential* for many power levels. The duplication sensitivity results are presented in Section VI-F.

• Regarding the throughput absolute values, the results with the power sources of *Thermal* and *TV-RF* are much higher than those with the others, which is constant with the power strength illustrated in Figure 4.

B. Energy efficiency

We evaluate energy efficiency by measuring MAC operations per Joule, as shown in Figure 9. Note that this includes the energy overheads of data movements and other functional units in addition to MACs. Overall, the normalized results of energy efficiency are very similar to those of the throughput evaluation. The results show that ResiRCA and ResiSchedule achieve average energy efficiency improvements of 14× compared to a baseline RCA with intermittency-unaware scheduling.

The only difference that can be observed is that, regarding *LeNet* and *PV* with the power source of *Thermal*, relatively speaking, the results of energy efficiency with *Pipelining* strategy are higher than that appearing in the throughput evaluation. One possible reason for this is that *Pipelining* requires loading several inputs from ReRAM memory to perform the parallel operations, which is power-expensive. This results in a behavior where, albeit less frequently having enough power to activate at all, the energy efficiency when active is high.

Compared to the cloud for online processing, the preference for local compute over offload can stem from security, connectivity and latency concerns as well as power and energy constraints. In our work, local computation across the CNN applications is ∼50x more efficient than transmission over Bluetooth with 3Mbps and 2.5mW.

C. Power utilization

In order to further understand the power utilization, we use a two-dimensional plot that illustrates the features of power consumption with the*ResiSchedule* strategy, as shown in Figure 10. The x-axis (power efficiency) denotes the percentage of power cycles where the RCA can activate. The y-axis (power utilization), on the other hand, denotes the percentage of valid power during activations which is actually utilized for computation and data transfer. An ideal system would be at the point (1,1). It can be observed from these results that the proposed *ResiSchedule* strategy can make good use of the *Piezo* source when it does exceed the minimal activation thresholds, though the very low duty cycle yields very low throughput.

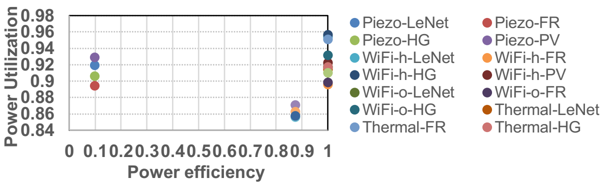


Fig. 10. ResiSchedule power efficiency analysis

D. Transition efficiency

Table VI-D shows the ratio of inferences using smooth-transitioned partial results and total inference count number. These results indicate that the smooth transition strategy $Transition^{Keep}$ enables a significant fraction of the inferences for all workloads on *Piezo*. However, a very small fraction is observed with the other, stronger power sources. For *Piezo*, saving the intermediate results of one incomplete inference is meaningful. However, one power cycle of the other power sources can usually process thousands or hundreds of inferences.

TABLE V
THE RATIO OF ADDITIONAL INFERENCES ENABLED BY THE SMOOTH TRANSITION STRATEGY VS. TOTAL INFERENCES

	Piezo	WiFi-h	WiFi-o	Thermal	TV-RF
LeNet	0.978632	0.000574	0.000782	0.000096	0.000068
FR	0.927445	0.000538	0.000594	0.000067	0.000059
HG	0.862620	0.000319	0.000416	0.000062	0.000049
PV	0.980769	0.002529	0.003181	0.000335	0.000266

E. Power predictor

With an accurate power predictor [45], [36], we can make more smooth transitions among different power levels. The benefit is that we can keep more MAC results of the last incomplete inference when switching from a higher power level to a lower power level, even if $Condition^{trans}$ is not satisfied. However, to be valuable the power predictor must have high accuracy. For both *Piezo* and *Thermal* power sources the prediction accuracy when using a multi-power-level-optimized extension of the power predictor in [36] are above 80%. Figure 11 shows the percentages of additional inferences enabled by power prediction over all inferences and additional inferences with $Transition^{keep}$ for all the workloads with these power sources.

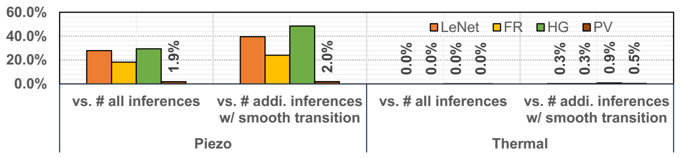


Fig. 11. Percentages of additional inferences with power prediction over all inferences and additional inferences with the $Transition^{keep}$ strategy

The portion of inferences added with power prediction are significant for *Piezo* for most workloads. This is because, the *Piezo* source is very weak and and the total completed number of inferences is quite small. Speculative action supported by power prediction can keep quite a few incomplete inferences to be completed in the next power cycle. This can also explain why the portions with the power source of *Thermal* are very small. However, for both *Piezo* and *Thermal*, the portions for *PV* are very small. The underlying reason is that the smooth $Transition^{keep}$ strategy can already handle the smooth transitions with no need of power prediction support for this workload.

F. Sensitivity study on duplication copy

We vary the ReRAM duplication granularity *G* for each layer and evaluate with *TV-RF* source. Throughput results are plotted in Figure 12 and area costs for $G1 \sim G5$ can be found in Figure 13. For each benchmark, all the numbers are normalized to that of the *G4* setting with the *Naive2* policy. As expected, the throughput increases as *G* grows for every benchmark. Another interesting observation is that the results of *ResiSchedule* policy can be competitive to that of *Sequential* policy when *G* is small and vice versa. The main reason for this is that the *ResiSchedule* policy can efficiently organize more hardware resources than the *Sequential* policy when hardware resources are limited.

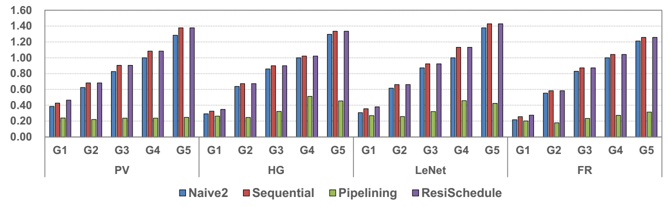


Fig. 12. Throughput normalized to G4 with *Naive2* vs. ReRAM duplication granularity

The RCA area is impacted from the parallelism granularity *G*, as shown in Figure 13. It also demonstrates that the proposed ResiRCA has total area requirements smaller than previously proposed "smart dust" solutions [8]. The ideal parallelism granularity determination for a particular deployment should consider the balance between throughput and area.

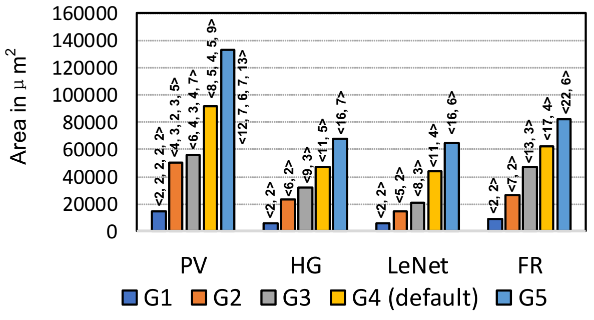


Fig. 13. Area with different duplication granularity

VII. Related Work

The previous RCA related work can be divided into the following two categories:

High Performance RCA Architectures: PRIME [4] uses 6-bit inputs and 8-bit weights and targets 6-bit output precision. A composition scheme is proposed, which uses two 3-bit input signals to construct one 6-bit input signal and two 4-bit cells representing one 8-bit synaptic weight. In the PRIME configuration, the ReRAM-based Full Function (FF) subarrays have both computation and data storage capabilities. To achieve the dual modes of FF subarrays and maximize reusability, custom peripheral circuits are designed.

In the ISAAC design [3], the inputs, weights and outputs are all 16 bits, where the DAC, ReRAM cell and ADC resolutions are, respectively, 1-bit, 2-bit and 8-bit. Also, a similar composition scheme is employed to organize the input, weight and output data. The ISAAC architecture is composed of 16 tiles and each tile consists of 8 IMAs which includes 4 ReRAMs along with 4 sets of peripheral circuits. An intra-tile pipeline is formed to boost the dot-product throughput.

In the PipeLayer design [5], a spike-based scheme, instead of a voltage-level based scheme, is used for input to eliminate the power overhead of DACs and ADCs. The underlying idea is to use spike counts to represent the data value. They propose intra-layer and inter-layer parallelism to support the training phase by reducing potential stalls.

ReRAM MAC circuits for the IoT: The nonvolatile intelligent processor (NIP) [8] is designed for accelerating fully-connected layers in energy harvesting IoT scenarios, in contrast to the convolutional layers ResiRCA targets. It includes four ReRAMs, each 32x32. The inputs and weights are binary and the output is adaptive between 1-3 bits. The serial-input non-weighted product (SINWP) structure [6] is the first work to propose multi-bit input/weight and output design from the circuit level, adopting a 2-bit input, 3-bit weight, and 4-bit output scheme. Note that each 3-bit signed weight needs a single-level-cell (SLC) ReRAM cell and is processed with a

3-bit resolution, which is a high performance but also a high power consuming design.

Although the above designs provide different approaches to achieve high throughput, high energy efficiency and low power, they cannot be directly applied or combined to be applied in the energy harvested edge devices due to the following reasons.

• The architecture-centric works [3], [4], [5] conservatively maintain high precision data and high resolution circuit signals, leading to high power consumption. Furthermore, the hierarchy they adopt with multiple ReRAMs targets primarily high throughput, leading to high power consumption on the whole RCA. For example, the total 168 *Tiles* and one *IMA* element of the ISAAC architecture [3] collectively consume 55.4W and 27.5mW respectively, while the peak harvested power for edge devices often lies in the range from hundreds of micro-watts to a few milli-watts in our collection sets. It can be seen that those designs are *not* suitable for an RCA supplied with harvested unstable power.

• Although the spike-based scheme [5] eliminates the power consuming part of the ReRAM peripheral circuits, it introduces very long latency to input/output data. It is known that the energy harvesting system often suffers from power failures and works in an intermittent mode; so, the spike-based data injection scheme is not favorable.

• For the ReRAM circuit concerned works [6], [8], although they are lightweight, they *cannot* be dynamically reconfigured to adapt changing power levels. In addition, such works have not presented any software level solution to maximize the utilization of the hardware platform. In order to accommodate the RCA to the changing harvested power supply, we need a "lightweight" and "fine-grain controllable" design from both the hardware and software angles.

VIII. Conclusion

MAC operations are the dominant computations in CNN applications which play a key role in intelligent edge devices such as smart sensors in IoTs. Considering the application scenarios where the accelerator is supported by harvested energy, we find that the previous designs cannot well accommodate the RCA to the changing power sources. This paper proposes ResiRCA, a resilient energy harvesting accelerator. We propose a lightweight and flexibly tuning RCA architecture and a ResiSchedule scheme to dynamically activate various scaled MAC operations so as to fully translate the "harvested energy" into "computation progress". ResiRCA supports smooth transitions among different activation solutions against computation loss. The experiment results show that the proposed ResiRCA along with the ResiSchedule scheme can achieve much higher speedups and energy efficiency compared to the baselines. ResiRCA for the first time supports harvested energy, expecting to initialize deeper researches on intelligent energy harvesting IoTs in the future.

IX. Acknowledgements

This work was supported in part by Semiconductor Research Corporation (SRC), Center for Brain-inspired Computing (C-BRIC), Center for Research in Intelligent Storage and Processing in Memory (CRISP), NSF Grants #1822923 (SPX: SOPHIA), #1763681, #1629915, #1629129, #1317560, #1526750, National Natural Science Foundation of China [NSFC Project No. 61872251] and Beijing Advanced Innovation Center for Imaging Technology.

This work was completed when Dr. Keni Qiu was visiting the Pennsylvania State University. The authors also greatly appreciate Dr. Yongpan Liu, Dr. Kaisheng Ma, Dr. Xulong Tang and Mr. Challapalle Nagadastagiri Reddy's useful discussion.

References

[1] C. Xia, J. Zhao, H. Cui, and X. Feng, "Characterizing DNN models for edge-cloud computing," in *2018 IEEE International Symposium on Workload Characterization (IISWC)*, pp. 82–83, 2018.

[2] L. Xia, T. Tang, W. Huangfu, M. Cheng, X. Yin, B. Li, Y. Wang, and H. Yang, "Switched by input: Power efficient structure for RRAM-based convolutional neural network," in *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2016.

[3] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar, "ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pp. 14–26, 2016.

[4] P. Chi, S. Li, C. Xu, T. Zhang, J. Zhao, Y. Liu, Y. Wang, and Y. Xie, "PRIME: A novel processing-in-memory architecture for neural network computation in reram-based main memory," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pp. 27–39, 2016.

[5] L. Song, X. Qian, H. Li, and Y. Chen, "Pipelayer: A pipelined ReRAM-Based accelerator for deep learning," in *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 541–552, 2017.

[6] C. Xue, W. Chen, J. Liu, J. Li, W. Lin, W. Lin, J. Wang, W. Wei, T. Chang, T. Chang, T. Huang, H. Kao, S. Wei, Y. Chiu, C. Lee, C. Lo, Y. King, C. Lin, R. Liu, C. Hsieh, K. Tang, and M. Chang, "24.1 a 1mb multibit ReRAM computing-in-memory macro with 14.6ns parallel mac computing time for CNN based AI edge processors," in *2019 IEEE International Solid- State Circuits Conference - (ISSCC)*, pp. 388–390, 2019.

[7] W. Chen, K. Li, W. Lin, K. Hsu, P. Li, C. Yang, C. Xue, E. Yang, Y. Chen, Y. Chang, T. Hsu, Y. King, C. Lin, R. Liu, C. Hsieh, K. Tang, and M. Chang, "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)*, pp. 494–496, 2018.

[8] F. Su, W. Chen, L. Xia, C. Lo, T. Tang, Z. Wang, K. Hsu, M. Cheng, J. Li, Y. Xie, Y. Wang, M. Chang, H. Yang, and Y. Liu, "A 462gops/j RRAM-based nonvolatile intelligent processor for energy harvesting ioe system featuring nonvolatile logics and processing-in-memory," in *2017 Symposium on VLSI Technology*, pp. T260–T261, 2017.

[9] Y. Ji, Y. Zhang, X. Xie, S. Li, P. Wang, X. Hu, Y. Zhang, and Y. Xie, "FPSA: A full system stack solution for reconfigurable reram-based NN accelerator architecture," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pp. 733–747, 2019.

[10] K. Qiu, W. Chen, Y. Xu, L. Xia, Y. Wang, and Z. Shao, "A peripheral circuit reuse structure integrated with a retimed data flow for low power rram crossbar-based cnn," in *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1057–1062, March 2018.

[11] X. Jiang, J. Polastre, and D. Culler, "Perpetual environmentally powered sensor networks," in *Fourth International Symposium on Information Processing in Sensor Networks (IPSN)*, pp. 463–468, 2005.

[12] S. Sudevalayam and P. Kulkarni, "Energy harvesting sensor nodes: Survey and implications," *IEEE Communications Surveys Tutorials*, vol. 13, no. 3, pp. 443–461, 2011.

[13] M. Mangrulkar and S. G. Akojwar, "A simple and efficient solar energy harvesting for wireless sensor node," in *2016 Second International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, pp. 95–99, 2016.

[14] R. Grezaud and J. Willemin, "A self-starting fully integrated auto-adaptive converter for battery-less thermal energy harvesting," in *2013 IEEE 11th International New Circuits and Systems Conference (NEWCAS)*, pp. 1–4, 2013.

[15] V. Leonov, T. Torfs, P. Fiorini, and C. Van Hoof, "Thermoelectric converters of human warmth for self-powered wireless sensor nodes," *IEEE Sensors Journal*, vol. 7, pp. 650–657, May 2007.

[16] X. Li, U. Dennis Heo, K. Ma, V. Narayanan, H. Liu, and S. Datta, "RF-powered systems using steep-slope devices," in *2014 IEEE 12th International New Circuits and Systems Conference (NEWCAS)*, pp. 73–76, 2014.

[17] K. Ma, Y. Zheng, S. Li, K. Swaminathan, X. Li, Y. Liu, J. Sampson, Y. Xie, and V. Narayanan, "Architecture exploration for ambient energy harvesting nonvolatile processors," in *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, pp. 526–537, 2015.

[18] K. Ma, X. Li, M. T. Kandemir, J. Sampson, V. Narayanan, J. Li, T. Wu, Z. Wang, Y. Liu, and Y. Xie, "NEOFog: Nonvolatility-exploiting optimizations for fog computing," in *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems ASPLOS*, pp. 782–796, 2018.

[19] M. Zhao, K. Qiu, Y. Xie, J. Hu, and C. J. Xue, "Redesigning software and systems for non-volatile processors on self-powered devices," in *2016 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, pp. 1–6, Sep. 2016.

[20] L. Ni, Z. Liu, H. Yu, and R. V. Joshi, "An energy-efficient digital ReRAM-crossbar-based cnn with bitwise parallelism," *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, vol. 3, pp. 37–46, Dec 2017.

[21] M. Zhao, C. Fu, Z. Li, Q. Li, M. Xie, Y. Liu, J. Hu, Z. Jia, and C. J. Xue, "Stack-size sensitive on-chip memory backup for self-powered nonvolatile processors," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 36, pp. 1804–1816, Nov 2017.

[22] A. Colin, E. Ruppel, and B. Lucia, "A reconfigurable energy storage architecture for energy-harvesting devices," in *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS 2018, Williamsburg, VA, USA, March 24-28, 2018*, pp. 767–781, 2018.

[23] X. Sheng, C. Wang, Y. Liu, H. G. Lee, N. Chang, and H. Yang, "A high-efficiency dual-channel photovoltaic power system for nonvolatile sensor nodes," in *2014 IEEE Non-Volatile Memory Systems and Applications Symposium (NVMSA)*, pp. 1–2, Aug 2014.

[24] X. Sun, S. Yin, X. Peng, R. Liu, J. Seo, and S. Yu, "XNOR-RRAM: A scalable and parallel resistive synaptic architecture for binary neural networks," in *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1423–1428, 2018.

[25] A. K. Mishra and D. Marr, "WRPN & apprentice: Methods for training and inference using low-precision numerics," *CoRR*, vol. abs/1803.00227, Apr 2018.

[26] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," *CoRR*, vol. abs/1510.00149, 2015.

[27] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," in *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pp. 1737–1746, 2015.

[28] S. Jain, S. Venkataramani, V. Srinivasan, J. Choi, P. Chuang, and L. Chang, "Compensated-dnn: Energy efficient low-precision deep neural networks by compensating quantization errors," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2018.

[29] N. Wang, J. Choi, D. Brand, C. Chen, and K. Gopalakrishnan, "Training deep neural networks with 8-bit floating point numbers," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pp. 7686–7695, 2018.

[30] Z. Cai, X. He, J. Sun, and N. Vasconcelos, "Deep learning with low precision by half-wave gaussian quantization," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5406–5414, 2017.

[31] G. Venkatesh, E. Nurvitadhi, and D. Marr, "Accelerating deep convolutional networks using low-precision and sparsity," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2861–2865, 2017.

[32] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *J. Mach. Learn. Res.*, vol. 18, pp. 6869–6898, Jan. 2017.

[33] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, pp. 3123–3131, 2015.

[34] M. Alwani, H. Chen, M. Ferdman, and P. Milder, "Fused-layer cnn accelerators," in *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 1–12, 2016.

[35] M. D. Lam, E. E. Rothberg, and M. E. Wolf, "The cache performance and optimizations of blocked algorithms," in *Proceedings of the Fourth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pp. 63–74, 1991.

[36] K. Ma, X. Li, S. R. Srinivasa, Y. Liu, J. Sampson, Y. Xie, and V. Narayanan, "Spendthrift: Machine learning based resource and frequency scaling for ambient energy harvesting nonvolatile processors," in *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 678–683, 2017.

[37] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The Gem5 Simulator," *SIGARCH Comput. Archit. News*, vol. 39, pp. 1–7, Aug. 2011.

[38] Synopsis, "HSPICE." https://www.synopsys.com/verification/ams-verification/hspice.html/.

[39] H. Lv, X. Xu, P. Yuan, D. Dong, T. Gong, J. Liu, Z. Yu, P. Huang, K. Zhang, C. Huo, C. Chen, Y. Xie, Q. Luo, S. Long, Q. Liu, J. Kang, D. Yang, S. Yin, S. Chiu, and M. Liu, "BEOL based RRAM with one extra-mask for low cost, highly reliable embedded application in 28 nm node and beyond," in *2017 IEEE International Electron Devices Meeting (IEDM)*, pp. 2.4.1–2.4.4, 2017.

[40] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi, "NVSim: A circuit-level performance, energy, and area model for emerging nonvolatile memory," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 31, pp. 994–1007, July 2012.

[41] R. Wang and Z. Xu, "A pedestrian and vehicle rapid identification model based on convolutional neural network," in *Proceedings of the 7th International Conference on Internet Multimedia Computing and Service*, ICIMCS '15, pp. 32:1–32:4, 2015.

[42] S. A. Dawwd and B. S. Mahmood, "A reconfigurable interconnected filter for face recognition based on convolution neural network," in *2009 4th International Design and Test Workshop (IDT)*, pp. 1–6, 2009.

[43] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, Nov 1998.

[44] H. Lin, M. Hsu, and W. Chen, "Human hand gesture recognition using a convolution neural network," in *2014 IEEE International Conference on Automation Science and Engineering (CASE)*, pp. 1038–1043, 2014.

[45] K. Ma, X. Li, Y. Liu, J. Sampson, Y. Xie, and V. Narayanan, "Dynamic machine learning based matching of nonvolatile processor microarchitecture to harvested energy profile," in *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 670–675, 2015.