ADVANCED INTELLIGENT SYSTEMS
Open Access
www.advintellsyst.com

A Fully Integrated System-on-Chip Design with Scalable Resistive Random-Access Memory Tile Design for Analog in-Memory Computing

Fuxi Cai, She-Hwa Yen, Apurva Uppala, Luke Thomas, Tianchi Liu, Peter Fu, Xiaofeng Zhang, Ambrose Low, Deepak Kamalanathan, Joe Hsu,* and Buvna Ayyagari-Sangamalli*

As the demands of big data applications and deep learning continue to rise, the industry is increasingly looking to artificial intelligence (AI) accelerators. Analog in-memory computing (AiMC) with emerging nonvolatile devices enable good hardware solutions, due to its high energy efficiency in accelerating the multiply-and-accumulation (MAC) operation. Herein, an Applied Materials custom-designed system-on-chip (SoC) targeting AI applications with analog in-memory computing using resistive random-access memory (ReRAM) as the compute element is demonstrated. The first silicon achieves high energy efficiency in MAC operations. This chip is implemented with LeNet-1 neural network on ReRAM tiles and demonstrated by Modified National Institute of Standards and Technology (MNIST) classification with accuracy matching that predicted in the simulations. A simulation framework, AI Sim, is also developed to evaluate the system performance for large-scale application and guide the bitcell development and design choices.

1. Introduction

Artificial intelligence (AI) and deep learning have fundamentally changed our way of life, and the need of high-performance and energy-efficient computing has grown rapidly in this big data era.[1] However, conventional digital systems based on CMOS computing devices and von Neumann architectures are facing several challenges these days, including the limitation of Moore's law scaling and frequent data movement between processing and memory unit, known as the von Neumann Bottleneck.[2] Although CMOS-based digital solutions such as graphic processing units (GPU) and application-specific integrated circuits (ASIC) can be designed as dedicated AI chips for fast computation, due to the limitation in power and footprint scaling, new computing devices and architectures are being developed to reduce the size and power requirements for AI accelerators and edge computing.[3,4]

Novel computing technologies have arisen in this More-than-Moore era, and analog in-memory computing with emerging nonvolatile memory arrays have become strong contenders for AI accelerators. The vector-matrix multiplication (VMM) computation is performed in situ within the memory array, eliminating extensive data movements for matrix multiplication. The VMM computation is highly parallel and requires low power by directly using Ohm's law for multiplication and Kirchhoff's law for accumulation, leading to much improved computation efficiency for MAC operations.[5,6] **Figure 1**a illustrates an example of a VMM operation calculated with analog in-memory computing through a parallel read process. Neural weights are stored in memory cells as different conductance levels. The input vector is encoded as voltage through digital-to-analog converters (DACs) and fed to the memory array. The current of each column is measured in parallel with analog-to-digital converters (ADC) and decoded as dot-product results.

Many prior works have demonstrated using in-memory computing for AI acceleration, with different computing elements such as digital memory devices like NAND Flash,[7] SRAM[8], and emerging analog devices such as phase-change memory (PCM)[9,10] and resistive random access memory (ReRAM).[11] ReRAM is a promising candidate due to its demonstrated capability to maintain multiple conductance levels,[12,13] ability to scale down to <10 nm node, and low programming voltage.[14,15] Also, the nonvolatility of ReRAM enables it to store trained weights for long-term computation better than a volatile memory like SRAM. A hybrid system with ReRAM and SRAM can combine the merits of nonvolatility of ReRAM with the fast speed of SRAM.[16]

F. Cai, S.-H. Yen, A. Uppala, L. Thomas, T. Liu, P. Fu, X. Zhang, A. Low, J. Hsu, B. Ayyagari-Sangamalli
Corporate Strategy and Development: Design Technology
Applied Materials Inc.
3325 Scott Blvd., Santa Clara, CA 95054, USA
E-mail: fuxi_cai@amat.com; j_hsu@amat.com

D. Kamalanathan
Semiconductor Products Group: Speciality Devices R&D
Applied Materials Inc.
3325 Scott Blvd., Santa Clara, CA 95054, USA

The ORCID identification number(s) for the author(s) of this article can be found under https://doi.org/10.1002/aisy.202200014.

DOI: 10.1002/aisy.202200014

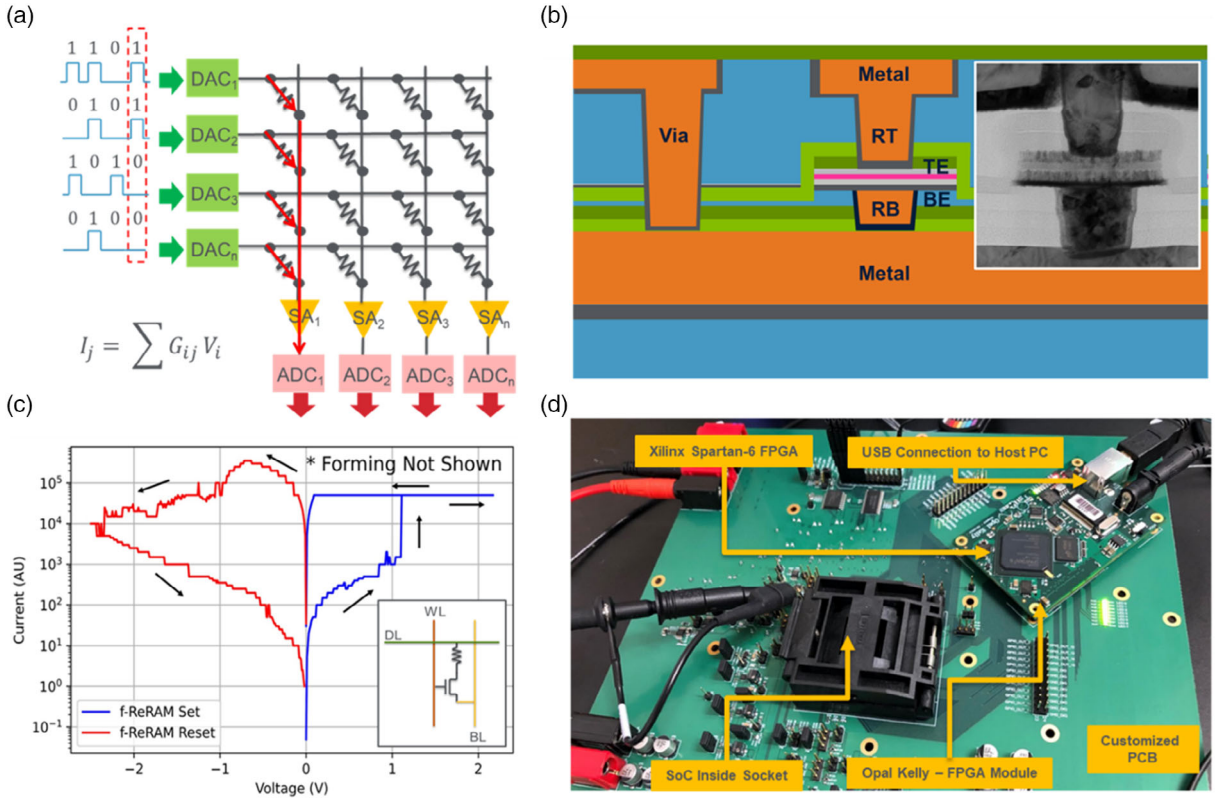

Figure 1. a) Analog in-memory computing with memory array. The VMM is performed by a parallel read operation. Input bits are converted to a pulse train by bit-serial DACs and sensed by binary-weighted TIA and ADC to achieve bit-wise MAC computation. b) An illustration of ReRAM integrated into the 65 nm BEOL process. Inset: transmission electron microscopy of ReRAM stack with contacts to both top and bottom electrodes. c) DC *I–V* curves for f-ReRAM devices. Arrows show the sweep direction for set and reset. Currents are normalized to arbitrary units and forming is not shown. Inset: The schematic of 1T1R bitcell. d) The custom-designed PCB for silicon testing and AI application demonstration.

Various AI-related applications have been demonstrated with analog in-memory computing using ReRAM based systems, such as image compression,[17,18] convolutional kernel,[13,19] motion detection,[20] online training,[21–24] and combinatorial optimization.[25] Some system designs use ReRAM array as standalone memory macro driven by essential analog peripherals and control signals from external components on the printed-circuit board (PCB). The ReRAM may be either a passive array[17,21] or one-transistor-one-resistor (1T1R) configuration with decoding blocks.[18,22,26] An alternative approach directly integrates the ReRAM with essential analog residing circuitry and decoding blocks[27] or additional digital processors[24] designed on chip. The tight integration will minimize the RC delay and signal loss during transmission and enable high-level program control for complicated tasks.

In this work, we custom designed a system on chip (SoC) with fully integrated ReRAM tiles, analog peripheral circuitry, and a RISCV processor in a single die. The ReRAM tile is designed as a configurable intellectual property (IP) block and serves as the primary component to construct various neural networks. The tiled approach with optimized analog periphery and digital system enables the scalability of the system to implement larger-scale deep neural networks (DNNs).[28,29]

We have implemented the first layer of the LeNet-1 on our ReRAM tile, with 2-bit per bitcell for weight storage. A benchmark with 1000 test images demonstrated in hardware a test accuracy of 96.8%. In addition, our *AI Sim* platform is developed for large-scale neural network implementations. The *AI Sim* can evaluate the optimal bitcell properties and tile architecture before hardware, providing guidelines in making design decisions. By correlating simulated and actual silicon results, the performance of the simulator can be tuned to scale to larger networks and applications.

2. System-on-Chip (SoC) Design

2.1. ReRAM Integration

The SoC chip is designed using the 65 nm technology node and fabricated using split-fab mechanism, where all CMOS circuits and back-end-of-line (BEOL) metal layers up to the ReRAM bottom metal layer are deposited by the foundry, and the wafer is transferred to the Applied Materials META R&D foundry for ReRAM integration along with the rest of the metal layers. ReRAM devices are designed in a 1T1R architecture, where the filamentary ReRAM (f-ReRAM) is inserted between two

metal layers in the BEOL (see Experimental Section for details), which is illustrated in Figure 1b.

The resistive switching mechanism of f-ReRAM is the formation and rupture of conductive filaments, which are controlled by a programming voltage applied across the two metal electrodes. When the external voltage is removed, the filaments remain and the device can retains its current resistance.[30] Although the f-ReRAM usually requires an extra high voltage for the one-time forming step, it has an advantage of a larger memory window.[31] The series transistor provides precise current compliance and control filament geometry and stochastic behavior during the forming and programming steps. A typical DC I–V curve for the f-ReRAM is shown in Figure 1c. It demonstrates bipolar switching with distinct set and reset operations (more switching curves can be found in Figure S1, Supporting Information). Drive line (DL) is defined as the signal line connected to the top electrode, with bit line (BL) and word line (WL) connecting the source and gate of the series transistor, respectively (Figure 1c).

After the ReRAM integration and the deposition of the top layers' metal, the chips were then diced and packaged in pin-grid-array (PGA) for silicon testing (Figure S2, Supporting Information). We designed a custom PCB with a socket to test our SoC chip and an field programmable gate array for digital control signals and transfer of data (Figure 1d). Some additional peripherals are designed on the board to support test features (see Experimental Section for details).

2.2. System-on-Chip Design

The SoC design is based on the open-source single-core microcontroller system PULPino, based on 32-bit RISCV cores.[32,33] We customized the digital design and connected four ReRAM tiles to the PULPino system, which can be scaled to various applications by adding additional tiles. Each ReRAM tile is an independent IP module containing a 64×64 1T1R ReRAM array, along with the essential analog peripherals such as digital-to-analog converter (DAC), transimpedance amplifier (TIA), analog-to-digital converter (ADC), and digital controller that is used to configure the analog circuit and transmit data. The chip also contains a 32 kB instruction memory and a 512 kB data memory for storing the instructions and data for the RISCV core. A phase-locked loop (PLL) is used for generating a high-frequency clock for the SoC system, and IO peripherals like serial peripheral interface (SPI), universal asynchronous receiver-transmitter (UART), and general-purpose input/output (GPIO) are included in the design to communicate to the host machine. A microscopic photo of the die is shown in **Figure 2**a.

An illustration of the ReRAM tile design is shown in Figure 2b. The 1T1R array has 64 separate DLs, WLs, and BLs, with each line addressing a specific bitcell. Each DL is driven by 8-bit DACs, which holds the input data for MAC operations. The BL of each column is connected to a sense amplifier (SA) circuitry. Groups of eight BLs share one 8-bit ADC; thus, eight MAC operations are required to readout the entire 64×64 array MAC results. More details of the analog peripherals will be discussed in Section 3.2.

Figure 2c shows a block diagram of the SoC architecture. The design is scalable using standard AXI interface to connect all four ReRAM tiles, which allows additional tiles and different ReRAM array sizes for a wide range of neural networks and portability to future designs.

3. Results and Discussion

3.1. Bitcell Programming

Finite-state machines (FSM) designed in the digital blocks of each tile design control the bitcell programming. Several modes are defined for different ReRAM array operations, such as "form," "reset," "set/POT (potentiation)," "DEP" (depression), and "MAC" modes. **Figure 3**a shows the array configuration for a "set" mode, where the selected WL is connected to the 10-bit on-chip DAC_{WL}, which can be set from 0.5 to 3 V to control programming current, and the selected DL is connected to the 8-bit DAC_{DL}, which varies from 0.5 to 3 V to provide the programming voltage for the ReRAM device. All unselected rows and columns are connected to ground. By controlling the MUX on the WL, DL, and BLs, pulses with target voltage levels can be sent to corresponding signal lines (illustration of other modes can be found in Figure S4, S6, S7, Supporting Information).

Finite state machines control the timing and the sequence of the DL and WL. The DL and WL pulse width, as well as the delay between the WL and DL edges, are programmable through the configuration registers. This enables flexibility in tuning programming conditions (Figure S3, S5, S8, Supporting Information).

With the combination of "Form", "Set", and "Reset" mode, the device conductance can be formed and programmed to different levels (see Figure S9 and S10, Supporting Information). Particularly, the adaptive set and reset pulse trains are used to fine tune to target current levels, which is a method that has been reported in our previous work[34] (see Figure S11, Supporting Information). The adaptive set pulse train will first be applied to the device under test (DUT), with the voltage across 1T1R bitcell starting from 0 to 3 V and a fixed gate voltage around 1.1 V. Each set pulse is followed by a read pulse to sample the current of ReRAM device (Figure S12, Supporting Information). If the current reaches the target threshold, the set process will halt, and the rest of set pulse will be skipped. The adaptive reset process is similar, except that the voltage across bitcell steps from 0 to −3 V, and gate is connected to V_{dd} to minimize serial resistance of the transistor (Figure 3b). Using the double-sided adaptive programming mechanism, we can demonstrate a 3-bit storage per cell in a 32×32 subarray on one tile, with device current spanning from 5 to 25 μA. Each conductance level is programmed into four columns (Figure S13, Supporting Information), and the cumulative distribution function (CDF) plot is shown in Figure 3c.

3.2. Multiply and Accumulation in ReRAM Tile

The fundamental feature of the analog MAC function is the analog multiplier. The multiplier leverages Ohm's law: $V \times G = I$, where V is the voltage across the ReRAM bit cell and G is the

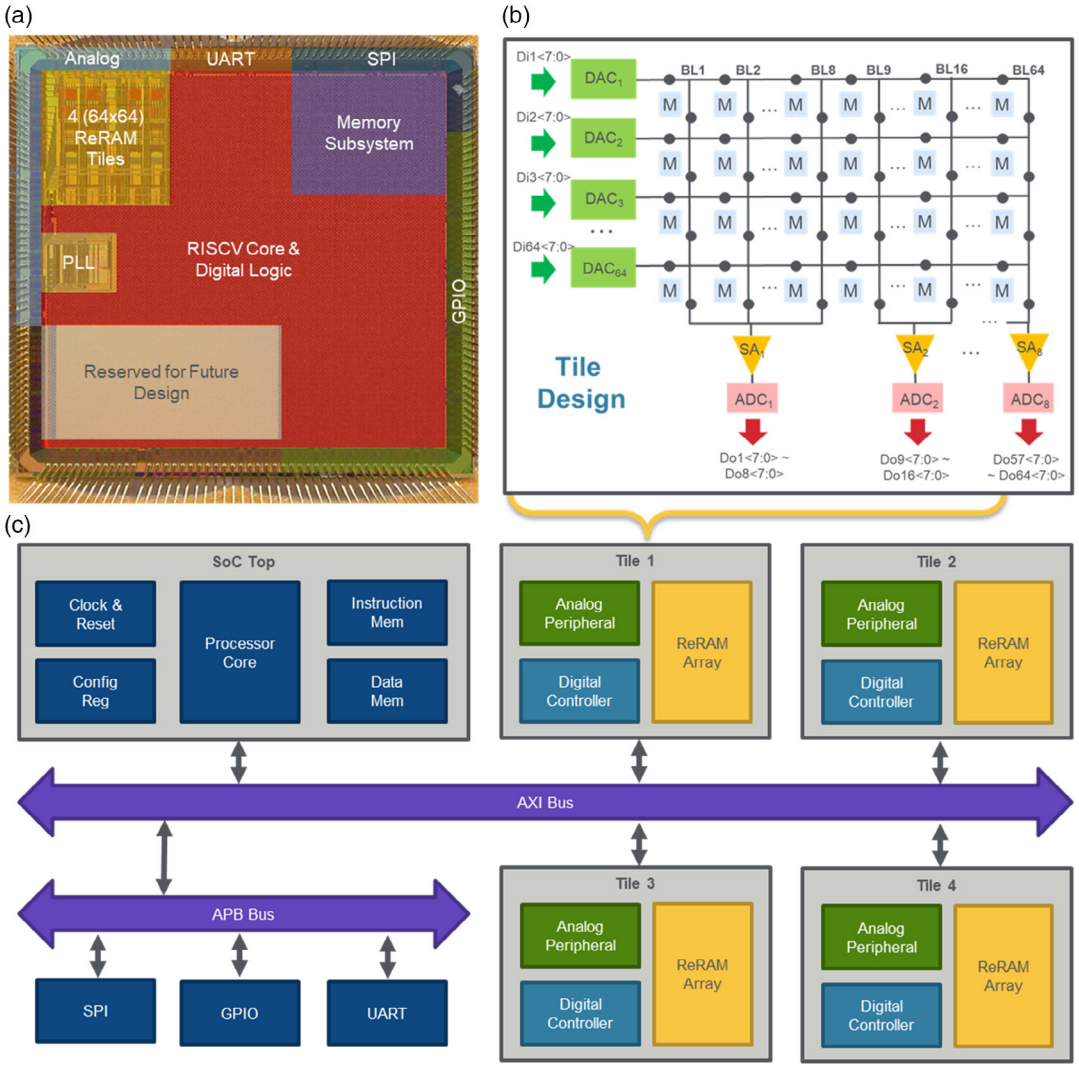

Figure 2. a) The SoC die photo. Design blocks are labeled in their corresponding layout sections. b) Simplified illustration of the ReRAM tile design. Each tile contains a 64 × 64 1T1R array, with 64 8-bit input DACs and eight 8-bit ADCs for MAC operation. Each ADC is shared by every eight column. c) Block diagram of the SoC architecture. Four Tile IPs are connected through AXI bus to the SoC as peripheral, making the design scalable for connecting more tiles.

conductance of the ReRAM bit cell. **Figure 4**a shows the signal path of the analog MAC operation (only shows one column as an example). During a MAC operation, array is configured into the "MAC" mode to allow multiple rows to be turned on and simultaneously aggregate multiply operations (Figure S14, Supporting Information). The 8-bit input is converted to a serial pulse train by the bit-serial DAC and a bitwise MAC is performed (see Experimental Section for more details).

As the multiplication result of each bitcell is current, the accumulation function of MAC connects the output of each bitcell on the same column (BL). Current electrically sums in a shared channel which computes the accumulated partial sum of a certain input bit for all the rows. To read out the MAC current value (I_{mac}), a TIA followed by an ADC is used to convert the analog current back to digital code by calculating the partial sum from each cycle and combining the binary-weighted sum as the output value (Figure S15, Supporting Information).

In this architecture, the TIA serves two functions: to convert the current (I_{mac}) to voltage with programmable gain and provide the common-mode reference voltage (V_{cm}) for the ReRAM bit-cells. A successive-approximation ADC (SAR ADC) shares multiple columns to save power consumption and silicon area.

Figure 4b-e shows the example of performing a multiple-and-accumulation operation in the programed ReRAM tile on the chip, using its "MAC" mode. To have a more intuitive interpretation from the MAC result, we programmed certain patterns in the ReRAM array, with all bitcells programmed to relatively similar conductance levels. In Figure 4b, the logo "AMAT" is programmed in a 26 × 7 subarray, with most bitcells programmed to 8–10 μA. By sending a read voltage of 0.3 V for the five rows of the subarray and 0 V to all other 59 rows in the ReRAM tile, a MAC operation of all five rows is performed and the total current of corresponding 26 columns is shown in Figure 4c. A qualitative connection can be observed between the value the total MAC

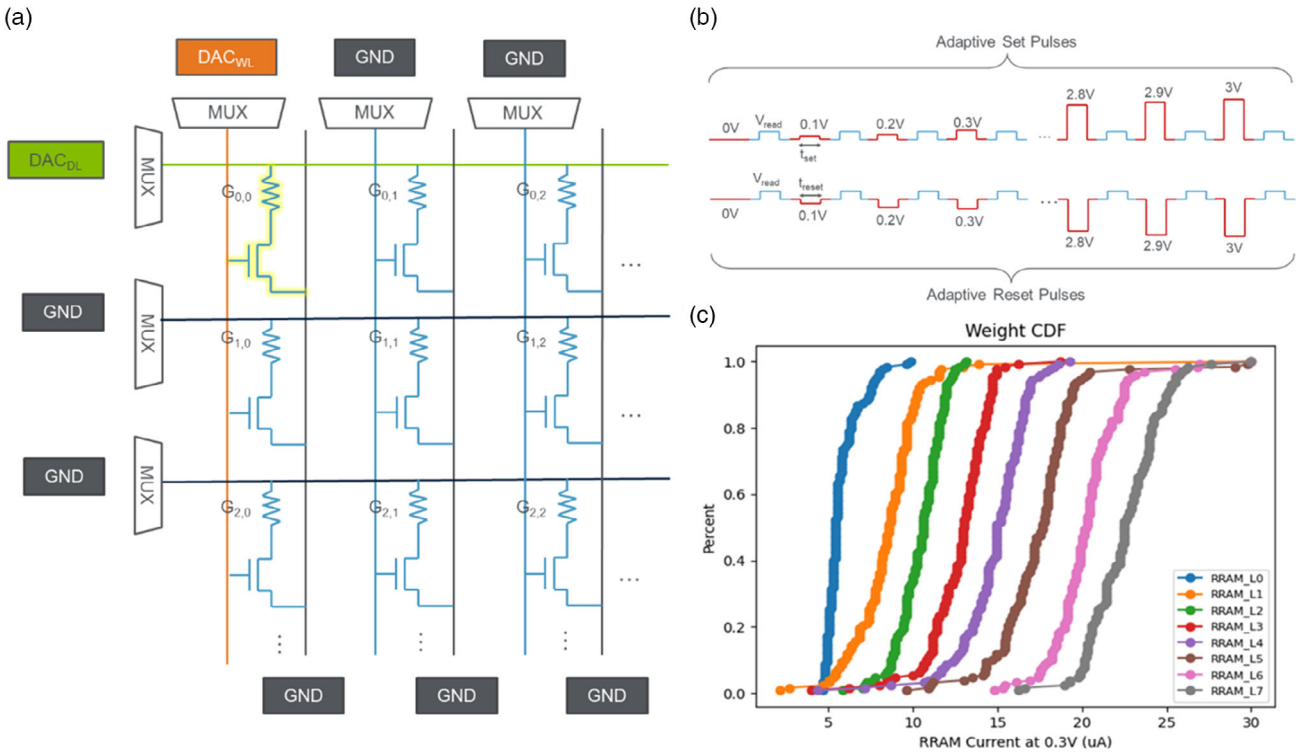

Figure 3. a) Array voltage configuration at "Set" mode. b) The adaptive set and adaptive reset pulse schemes to fine tune ReRAM conductance. The voltage labeled in waveform represents the voltage drop across the 1T1R bitcell. c) ReRAM conductance CDF of eight levels (3-bit) stored in 32×32 subarray, with increments of four columns programmed to distinct levels.

current and the number of programmed cells in the column. For example, the first column of the first letter "A" has double current of its second column, as it contains four programmed bitcells compared with two in the second column.

A more quantitative experiment is shown in Figure 4d-e to demonstrate the functionality of the MAC circuitry. An incremental staircase-shaped pattern is programmed into a 5×5 subarray with each bitcell programmed to $\approx 6\,\mu A$, and the MAC operation is activated with the read voltages applied to the five selected rows. It can be observed from Figure 4e that the result currents are perfectly proportional to the number of cells in the corresponding column, demonstrating good linearity of the ADC and sensing circuitry.

3.3. Neural Network Implementation

To further demonstrate the capability of this SoC on accelerating neural network applications, we have implemented a LeNet-1 network on our ReRAM tiles to demonstrate performance on the MNIST dataset.[35] More is discussed in Experimental Section about the MNIST dataset.

LeNet-1 is one of the earliest convolutional neural networks (CNN) invented by Professor Yann LeCun and others in the late 1990s, which is designed to achieve better performance in classifying handwritten digits.[36] It is a three-layer neural network with two convolutional layers and one fully connected layer (**Figure 5**a), making it a slimmer model with less weights compared with the more popular LeNet-5 and more hardware-friendly to implement on ReRAM array.

In this experiment, we implement the first convolutional layer on our ReRAM tile, which contains four 5×5 filters. The weights of the LeNet-1 are trained in PyTorch with quantization-aware training for 3-bit weights and 8-bit activations (see Experimental Section for details). As ReRAM conductance can only store positive-valued information, the 5×5 filter weights are flattened and stored in a 25×2 differential pair, with one column storing only the positive weights and the other one storing the additive inverse of negative weights, so that the difference of the two columns equals the original weight (Figure 5b,c).

The original 25×4 weight matrix size is doubled after using the positive–negative pair and contains only four levels from 0 to 3 (equivalent 2-bit) and stored onto a 25×8 subarray in one ReRAM tile and then mapped to four equally spaced conductance levels (Figure 5d, S17, Supporting Information). Figure 5e shows the CDF plot of the ReRAM stored weights, showing the majority of the weights confined within the target levels (also see Figure S16, Supporting Information, for retention).

During the inference of MNIST images, the convolution is calculated in a weight-stationary fashion. The 5×5 sliding window will move through the grayscale input image (the handwritten digit "6" in Figure 5f), and the windowed region is flattened and set by DACs on appropriate DLs. As each filter is implemented in two adjacent columns containing positive and

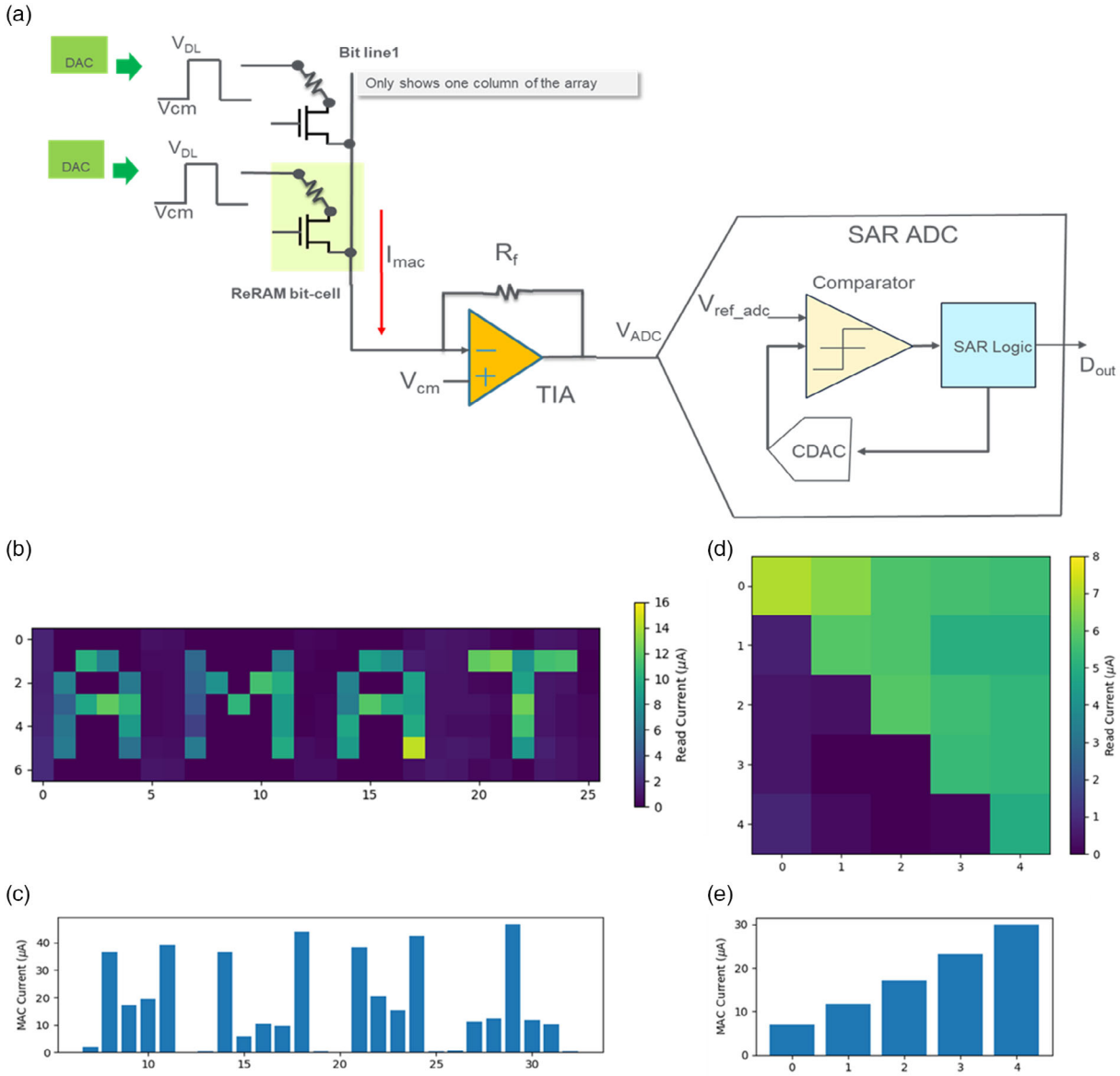

Figure 4. a) Simplified diagram of the current-sensing circuitry. The complete signal path from the input DAC through the 1T1R bitcell to the SAR ADC is illustrated. b) "AMAT" letters programmed in the ReRAM array. Most ReRAM cells are programmed around 8≈10μA c) The MAC current from (b), with 0.3 V read voltage only sent to the five rows where ReRAM cells are programmed. d) A staircase pattern programmed in the ReRAM array. Most ReRAM cells are programmed around 6μA. e) The MAC current from (d), with 0.3 V read voltage only sent to the five rows where ReRAM cells are programmed.

negative weights, their MAC results will be stored separately in the memory. After the sliding window computes the entire input image, 24 × 24 positive and negative feature maps (Figure 5g,h) remain and the net output is generated by subtracting the negative map from the positive one. (Figure 5f).

As the eight columns of the MAC are computed in parallel and stored separately, all four feature maps of the first layer can be computed from a similar approach shown in Figure 5f–h. Hardware-to-software correlation is illustrated in **Figure 6**a,b, between the ideal first-layer output feature map simulated from Python software and our experimental MAC result measured from on-chip ADC and ReRAM stored weights.

With the first-layer activation output from the hardware, we can verify the accuracy performance through a hybrid approach, by plugging the first-layer activation hardware results into our Python model to finish predictions in PyTorch model (Figure 6c). The hardware output feature maps in Figure 6b will be scaled to the similar range of the simulation results in Figure 6a.

The fully connected layer output before SoftMax activation (i.e., the "Logits") is shown in Figure 6d. It can be observed that although the results of the hardware hybrid approach differ slightly from the python-only approach, the winning neurons both point to the sixth digit, and both approaches can finally

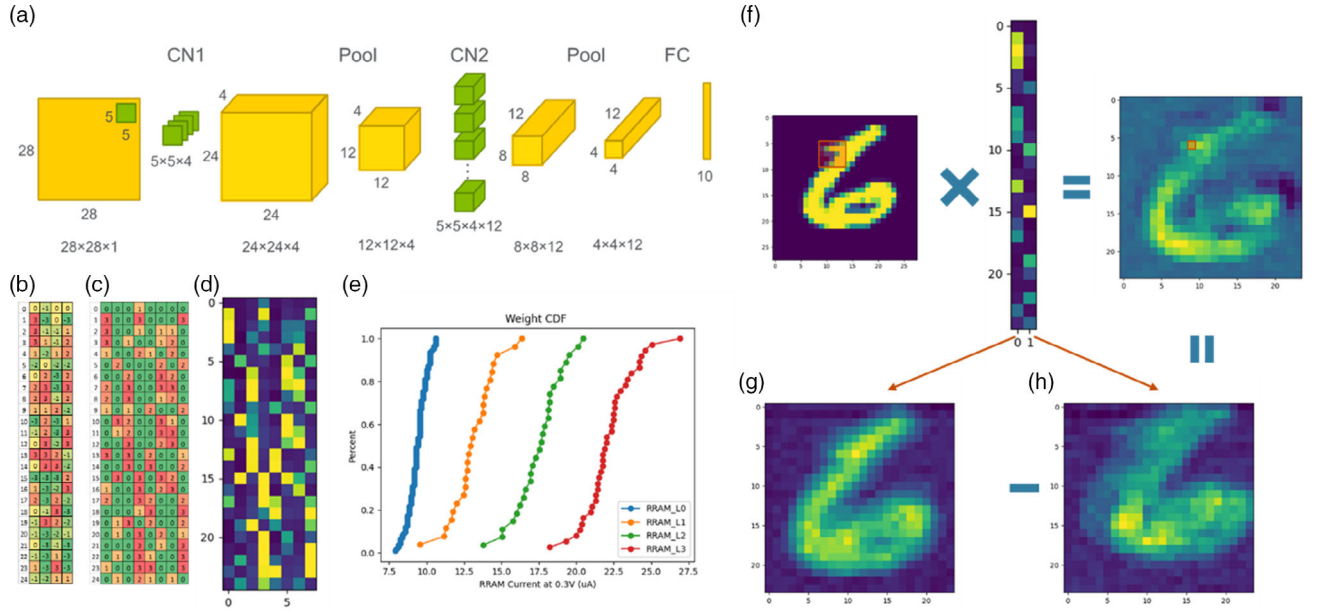

Figure 5. a) The LeNet-1 neural network illustration. b) The weight of the first filter of first layer in LeNet-1 trained by quantization-aware training. c) The expanded weight converted from (b), with differential pair of positive- and negative-weight columns. d) The programmed weight in 25 × 8 ReRAM array. e) The CDF plot of bitcell conductances in (d), grouped in ideal weight levels. f–h) An illustration of the 2D convolutional with the original 28 × 28 input image with the 25 × 2 ReRAM weight array. Two 24 × 24 output feature maps (the positive feature map g and negative feature map h) are generated by scanning the 5 × 5 window through the entire input image and performing MAC operations. The two feature maps are subtracted by processor and generate the final result in (f).

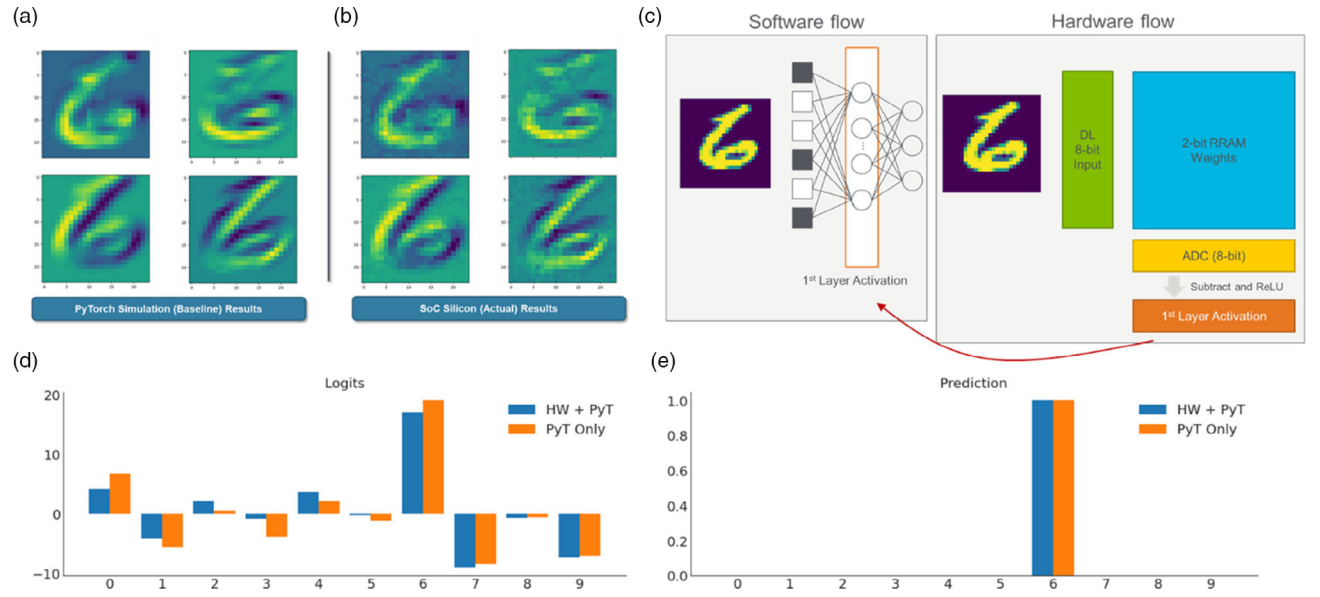

Figure 6. a–b) The correlation of Lenet-1 first-layer output from the ideal PyTorch simulation in (a) and the silicon results measurement with ReRAM programmed weights and SoC MAC operation in (b). c) The illustration of the hardware hybrid approach. The first-layer activation from the hardware measurement is normalized and plugged into the second layer in software model for accuracy evaluation. d) The comparison of the LeNet-1 logits - between the PyTorch-only flow and the hardware hybrid flow. e) The comparison of the LeNet-1 prediction for input image in (c) between the PyTorch-only flow and the hardware hybrid flow. Both approaches predict the outcome "6" with > 99% confidence.

predict the number "6" with ≈99% probability in Figure 6e (See Figure S18, Supporting Information, for another example).

3.4. Benchmark Results

3.4.1. Test Accuracy

To achieve the test accuracy of the MNIST data, benchmarks with hundreds of images need to be run during the inference. As the on-chip data memory size is limited to 512 kB and cannot store a large image dataset, we developed a flow that utilizes an external flash on the PCB for larger-scale benchmarks.

In this test, 1000 images from the original 10 000 test dataset are stored in the on-board flash before the benchmark. During the inference benchmark, the processor will communicate to the external flash through on-chip SPI master and grab one image at each time (Figure S19, Supporting Information). The 28×28 input image is sent to the ReRAM tile to perform the first-layer convolution, as Section 3.3 discussed. The output feature maps from all eight columns are then stored back to external flash through the SPI interface, and the processor will continue to load the next image. After the 1000 image inference is completed, all the output feature maps are read from flash and passed to the software model to generate the final prediction.

The inference results are illustrated in the confusion matrix in **Figure 7**a. From the 1000 images benchmark, a test accuracy of 96.8% can be achieved, which is within 2% of baseline accuracy of the 98.7% simulated in software.

3.4.2. Energy Efficiency

Besides the test accuracy, another important benchmark we investigated is energy efficiency of the SoC, which is usually evaluated by TOPS/W. TOPS/W is defined as the ratio of throughput and power, where throughput is calculated by the number of Tera Operations per Second. Peak TOPS/W is important to evaluate in-memory computing hardware. Peak TOPS/W is computed at top clock frequency and 100% array utilization.

$$\frac{\text{TOPS}}{\text{W}}\,(\text{Peak}) = \frac{\text{Number of operations/Sec}}{\text{Power consumption}} \tag{1}$$

Some debate TOPS/W as a metric and if it can reflect actual system energy efficiency, due to the ambiguity of how the throughput and power are defined and measured.[37] Also, the conventional peak TOPS/W focuses more on the system architecture design, but is inadequate in representing the quality of the bitcell and array used for MAC operation, which is an important factor in analog in-memory computing. To take into consideration the bitcell memory elements, we defined an "AiMC" *TOPS/W* as

$$\text{AiMC } TOPS/W = T_{\text{p}} \times M \times N \times Y \tag{2}$$

where T_{p} stands for peak TOPS/W of a single tile, with throughput and power measured at the maximum clock frequency. T_{p} is mainly constrained by the system design limitations such as peak clock frequency, system power, as well as the tile array size.

M represents bit precision of input defined by the m-bit DAC and N represents the bits per bitcell defined by the logarithm N that represents number of stable levels in the analog ReRAM array. This serves as a normalization factor for the TOPS, as different systems use different precisions for weights and activations. Many have discussed multiplying the TOPS with the bit precision of input and weight as a scaling factor to normalize different systems with various precisions all to a 1-bit precision metric, assuming that an n-bit MAC will require $n\times$ more operations than a 1-bit precision system.[6,26]

Y stands for array yield, which is an important scaling factor for the AiMC peak TOPS, considering that a failing bitcell in the array cannot be programmed to the target conductance level and cannot contribute to the MAC operation.

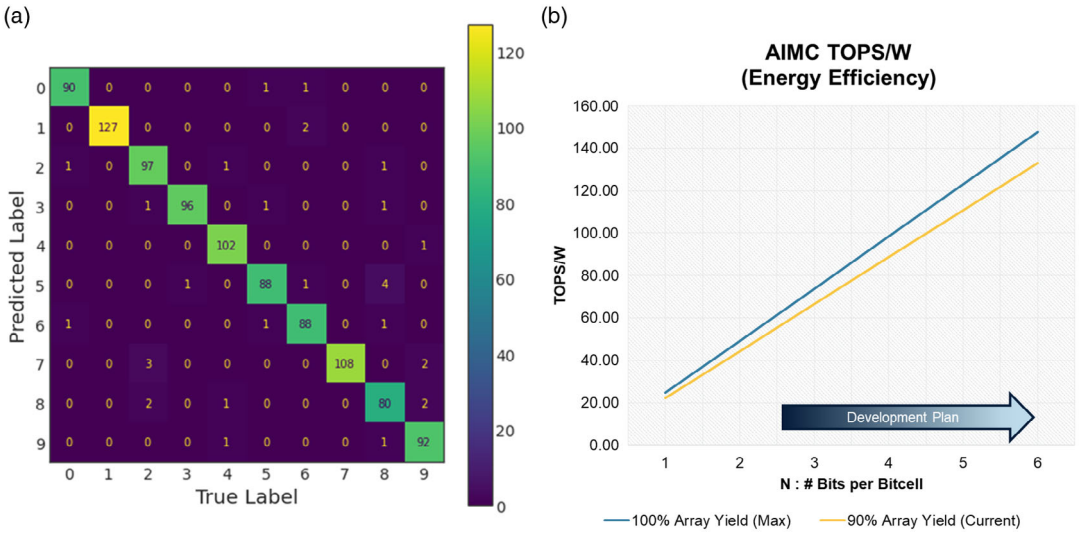

Figure 7. a) Confusion matrix of the 1000 test image benchmark. The true label is presented in the *x*-axis and the predict label is presented in *y*-axis. b) The "AiMC" TOPS/W for the SoC. A linear relation of TOPS/W against the number of bits per bitcell and the array yield is observed.

Table 1. Design considerations for AiMC-based AI hardware accelerators.

Analog in-memory computing system design considerations			
ReRAM Physics	Bitcell	Array	Analog peripheral circuit
ReRAM switching physics	CMOS device selection	Array yield	ADC precision
Endurance	Operating current range and on/off ratio	Device-to-device variation	ADC multiplexing
Retention	Read and write voltages	Parasitics, IR drop	DAC precision
Read speed	Number of conductance levels	Size of arrays	Partial sum quantization accuracy loss
Read disturb	–	–	Peripheral circuit-induced noise
RTN noise	–	–	PVT sensitivity

From the illustration in Figure 7b, it can be observed that improving the number of bits per bitcell and array yield can increase the AiMC TOPS/W, which guides our future development plan. With an example of the 3-bit per cell (Figure 3c), we can measure energy efficiency of ≈73.9 TOPS/W per single tile at the clock frequency of 100 MHz. By further optimizing the ReRAM characteristics using improved structure and recipe, and targeting 8-bit per cell bitcell, we can extrapolate to >197 TOPS/W per single tile. Furthermore, in a larger system of multiple tiles integrated and configured together, with further optimization of the array utilization, dataflow, and power management, much higher energy efficiency can be achieved.

4. Simulator for DNN Acceleration

While the discussion so far focuses mainly on LeNet-1, a useful but relatively small neural network, DNNs that find applications in the industry today have millions of trainable weights. And so, scaling analog in-memory computing (AiMC) technology to real-world applications requires several hundred ReRAM tiles to store these weights.

To design an efficient AiMC-based hardware accelerator system, several ReRAM-related nonidealities affecting system accuracy need to be considered, such as device-to-device variation, limited retention, read disturb, etc. These ReRAM device characteristics need to be parameterized with a

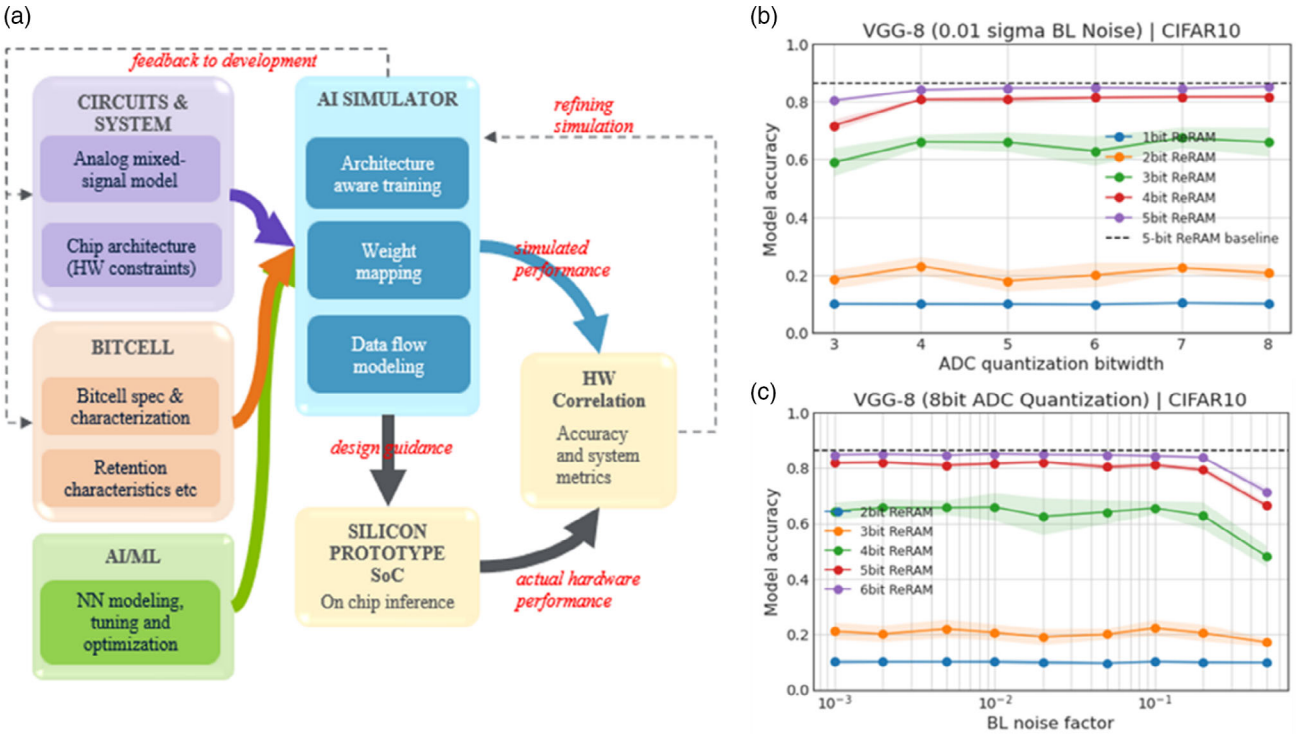

Figure 8. a) *AI Sim*, DNN system simulator usage model. b,c) Example-simulated results showing accuracy dependence on quantization and noise for VGG-8 neural network on CIFAR-10 dataset.

software tool so that they can be updated in the evolving ReRAM fabrication process.[38]

In addition, the array design and its analog peripheral circuit are also critical factors to be considered. A software tool is needed to incorporate these parameters when engineering ReRAM bitcells and exploring analog design choices, which can help designers and engineers understand these complex interactions and help make some broad tradeoffs and aid chip design.[29,39] **Table 1** shows a detailed list of design considerations discussed earlier.

With the goal of modeling realistic performance at a system level, we built a simulation platform named AI Sim using the PyTorch framework.[40] **Figure 8**a illustrates its usage model at a high level. AI Sim models DNN accuracy considering different quantization schemes, ReRAM bitcell nonidealities (like finite on/off ratio, read disturb, drift), mixed signal circuit precision, and noise. It also estimates system performance metrics like power, performance, area (PPA), inference time, and TOPS/W by considering various mapping and dataflow strategies, component circuit characteristics, and process technology assumptions. It serves to accelerate R&D by providing feedback for bitcell development and providing operation specification for the circuit design.

Figure 8b,c shows examples of *AI Sim* analyses: VGG-8 model accuracy degrades sharply with ReRAM bitcell precisions below 4 bits. Higher ADC bitwidth and lower bit line (BL) noise are also required for maintaining high inference accuracy. These analyses can provide important guidelines in the early stage of system design and ReRAM development to guarantee high inference accuracy in the finished system.

The *AI Sim* also helps in making architecture tradeoffs and exploring algorithms which are tolerant to nonidealities. With programmed bitcell data from actual silicon measurement, the AI Sim can extrapolate inference performance of a large DNN implemented on these devices by extracting the variance of target bitcell levels. In addition, the AI Sim is also capable of investigating inference accuracy degradation over time or after cycling by extrapolating bitcell reliability data such as retention characteristics (**Figure 9**) and read disturb (**Figure 10**), using the approach discussed in prior works.[41,42]

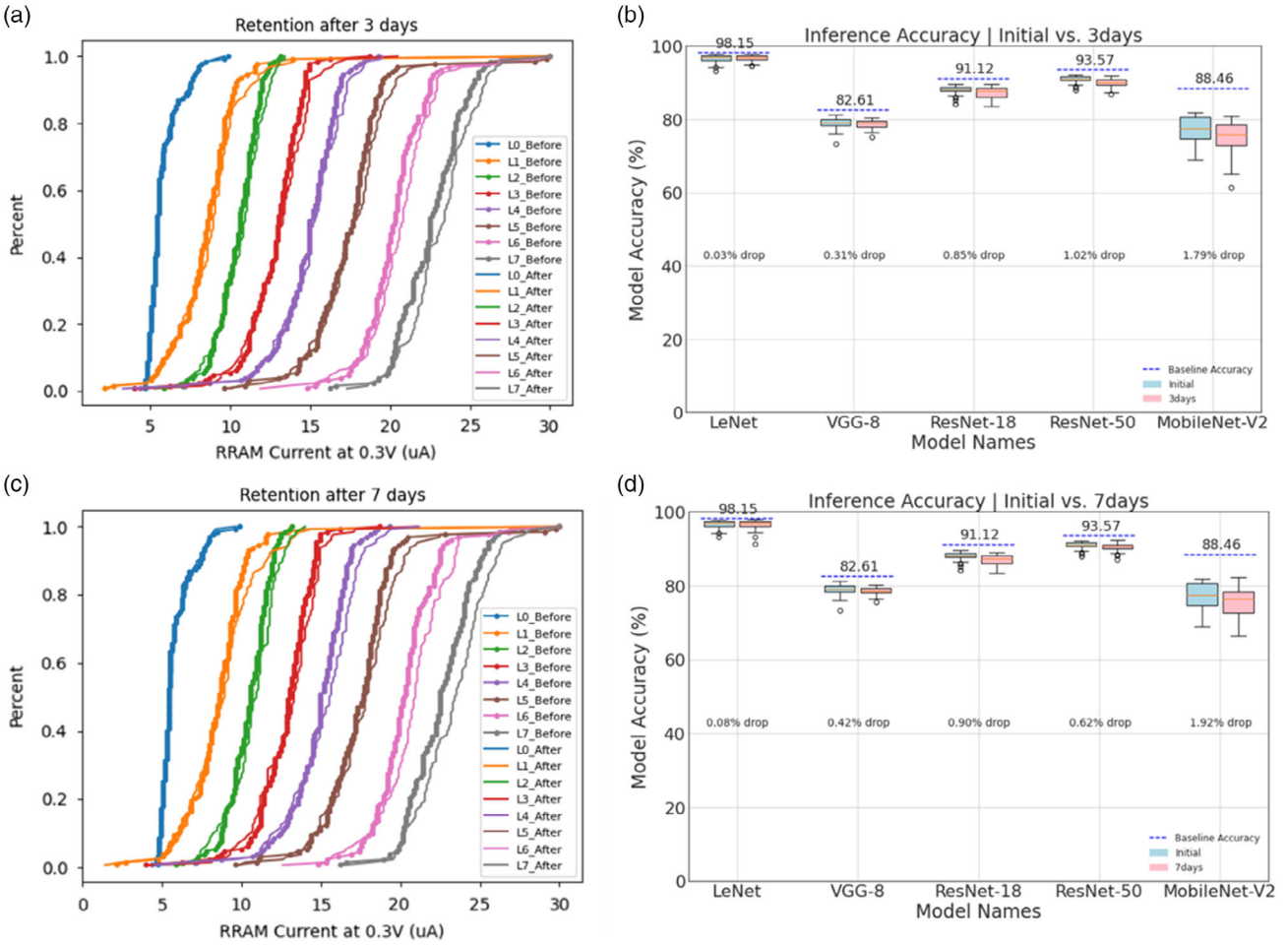

Figure 9. Comparison of eight-level ReRAM conductance distribution a) before and after three days and c) before and after seven days. Simulated Impact of Retention on inference accuracy of five models with ideal weights (blue dotted line) and with weights of extracted Gaussian variations b) before and after three days and d) before and after seven days (box plot is plotted from 50 trials, with each trial sample having different variation from the distribution).

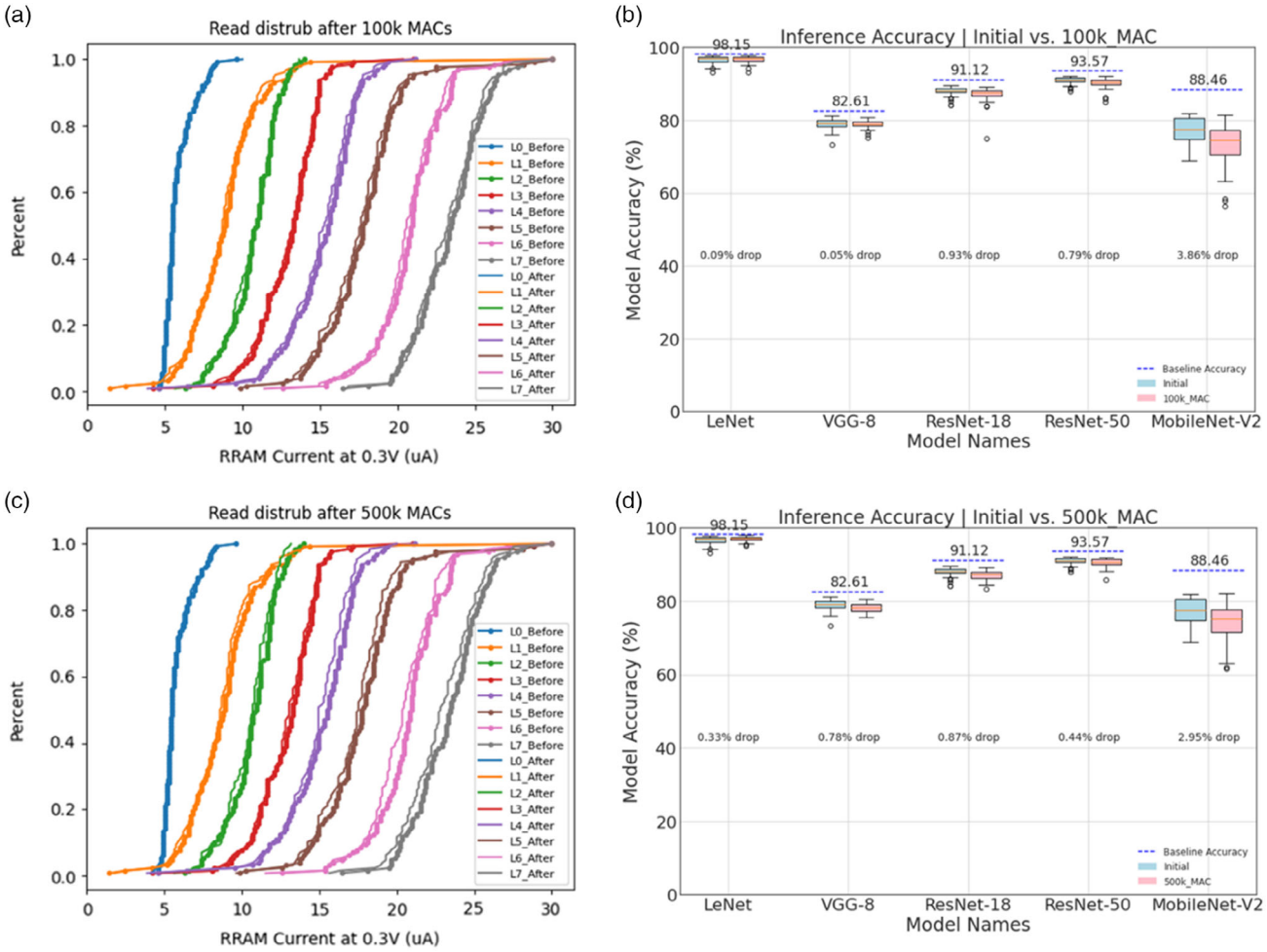

Figure 10. Comparison of eight-levels ReRAM conductance distribution a) before and after 100 k MAC operations and c) before and after 500 k MAC operations. Simulated Impact of Retention on inference accuracy of five models with ideal weights (blue dotted line) and with weights of extracted Gaussian variations b) before and after 100 k MAC operations and d) before and after 500 k MAC operations (box plot is plotted from 50 trials, with each trial sample having different variations from the distribution).

We first investigated the retention of our programmed ReRAM conductance at room temperature. The bitcell in a 32×32 subarray of ReRAM tile is programmed to eight levels, as shown in Figure 3c. We fitted each of the eight levels using a Gaussian distribution and extract its mean conductance value and standard deviation (Figure S20, Supporting Information).

The network then runs inference with resampled weights. Inference accuracy is evaluated for 50 trials of resampling across several neural networks such as VGG-8, ResNet, and MobileNet. The simulated results show that running inference with devices similar to those from the 32×32 subarray would lead to less than 4% accuracy degradation for LeNet, VGG-8, and ResNet (Figure S21, Supporting Information).

To investigate the impact of retention characteristics, the programmed bitcell data is extracted again after 3 and 7 days. The comparison of the eight-level distribution is plotted in Figure 9a,c.

We repeated the extrapolation above by resampling weights from the newly computed Gaussian variables after 3 and 7 days. The simulated results show that device retention remains stable within a week and all five models would not degrade by more than 2% mean accuracy within 7 days (Figure 9b,d).

Similarly, to investigate the impact of read disturbance on inference performance, the programmed bitcell data is extracted again after 100 000 MAC operations and 500 000 MAC operations. The MAC operation is designed to turn all odd rows or all even rows simultaneously and read the current, meaning all the bitcells are read once every two MACs. The comparison of the eight-level distribution is plotted in Figure 10a,c.

Inference accuracy of five models is simulated again by resampling weights from the newly computed Gaussian variables. Based on simulation, there is less than 1% accuracy degradation for four models after 100 000 MAC (Figure 10b), and there is no further significant impact on inference after 500 000 MACs. (Figure 10d).

5. Conclusion

In this work, we have designed and fabricated a SoC with multiple ReRAM tile arrays integrated with an RISCV processor. We demonstrated the standard MNIST benchmark with LeNet implemented on hardware with 96.8% test accuracy and ≈73.9 TOPS/W energy efficiency per tile at 65 nm technology. The ReRAM tile design can be treated as an independent design IP and a building block, making the system design expandable and flexible for different ReRAM technologies and also for larger-scale neural networks and different applications. With careful optimization of the larger system integrated with many tiles, higher energy efficiency is possible to be achieved. The *AI Sim*, which was correlated to silicon, allows explorations and trade-offs between bitcell, end applications, neural networks, and design choices.

6. Experimental Section

ReRAM Devices: The ReRAM stack was built using a traditional metal–oxide dielectric and a metal-containing oxygen exchange layer, both of which were deposited using an Impulse physical vapour deposition process using AMAT's Endura platform.

PCB Design and Test Setup: We designed a customized PCB for silicon bring-up as well as AI application demonstrations. Besides the essential components, the PCB included additional on-board DACs and ADCs to provide extra testing and calibration capabilities. An SPI flash was also included on board to enable large dataset storage for benchmark.

A Xilinx Opal Kelly XEM6010 module was connected to the PCB, making connection to the host machine. It controls digital signals and loads programs from the host machine to the SoC chip and receives data results from the chip, through FrontPanel SDK.

Modified National Institute of Standards and Technology Database (MNIST): The MNIST dataset is a collection of images of handwritten digits, which is a standard dataset used in computer vision and machine learning. It contains a training set of 60 000 samples and a testing set of 10 000 samples. It was created from a subset of NIST's larger dataset, with all digits normalized and cropped to fit in to 28×28-pixel grayscale images.

Quantization Aware Training: The training of the neural network was performed using Applied's AI Sim with high-performance GPU. As ReRAM array stores weights in four levels, we limited the weight precision to 3-bit using quantization-aware training by incorporating Xilinx Brevitas library in PyTorch.[43]

The 3-bit weight after training contained only integers from −3 to 3 (in fact only seven distinct levels), but as we stored the weight on ReRAM tiles using positive and negative differential pairs, only four levels (0, 1, 2, 3) were needed to map weights to conductance. This explains why we trained the weights in 3-bit but only require 2-bit per bitcell in hardware.

Bit-Serial DAC and Binary-Weighted Multiplier Accumulator: In our system, we used bit-serial DAC input and binary-weighted TIA and ADC to achieve MAC computation (Figure S15, Supporting Information).

In a MAC operation, the input is converted to an 8-bit serial pulse train by DAC and bitwise MAC is performed. A small-fixed read voltage (0.1–0.3 V) represents an input digit "1"to minimize the read disturbance, and 0 V is for input digit "0."

TIA and ADC will calculate the partial sum from each cycle and combine the binary-weighted sum as the output value. The MAC computation requires eight cycles from input plus several reset-and-ready cycles in total.

More details about our bit-serial DAC and binary-weighted multiplier accumulator can be founded in our previous published work.[44]

AI Sim Model Parameter Selection: In a comprehensive simulation framework such as AI Sim, parameter selection in the neural network model can be quite challenging given the various considerations shown in Table 1.

There are mainly two types of evaluation parameters our AI Sim selected.

The first type was neural network related, which was mainly the model size and hyperparameters used during training. We followed the original size and layers of neural network models (i.e., LeNet and ResNet-18) and swept the hyperparameters to achieve the best test accuracy.

The second type was hardware related, such as variations, retention, read disturbance from ReRAM devices, circuit noise from analog peripheral circuits, and tile size from system architectures. Those were extracted from actual hardware measurements or by design and entered into our AI Sim to help align experimental results with simulated results.

Our current approach used the original model parameters as baseline and modified them when including the hardware nonidealities, which required trial and error. A more efficient approach was to use a generic search approach to automate the model parameter selection, which will be our goal in the future work.[39]

Supporting Information

Supporting Information is available from the Wiley Online Library or from the author.

Acknowledgements

The authors would like to acknowledge all collogues and team members who are involved in this work. The authors would like to thank V. Reddy, F. Guo, C. Chevallier, J. Chung, V. Bhavikatti, J. He, B. Alexander, and R. Gupta from Applied DT group for their hard work on system-on-chip design, tape out, and testing. The authors also appreciate D. Kamalanathan, J. Correll, A. Nourbakhsh, F. Nardi, E. Smith, R. Smith, J. Hebding, A. Pang, J. Holt, A. Kumar, R. Someshwar, G. Saheli, J. Anthis, R. Quon, M. Pesic, L. Larcher, W. Cho, B. Juang, S. Krishnan, and M. Chudzik from Applied speciality devices R&D group for their tremendous contribution on ReRAM device development and integration.

Conflict of Interest

The authors declare no conflict of interest.

Data Availability Statement

Research data are not shared.

Keywords

analog in-memory computing, one-transistor-one-resistor, resistive random-access memory

Received: January 12, 2022
Revised: April 1, 2022
Published online:

[1] Y. LeCun, Y. Bengio, G. Hinton, *Nature* **2015**, *521*, 436.
[2] M. A. Zidan, J. P. Strachan, W. D. Lu, *Nat. Electron.* **2018**, *1*, 22.
[3] C. Sung, H. Hwang, I. K. Yoo, *J. Appl. Phys.* **2018**, *124*, 151903.
[4] O. Krestinskaya, A. P. James, L. O. Chua, *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *31*, 4.
[5] Q. Xia, J. J. Yang, *Nat. Mater.* **2019**, *18*, 309.

[6] S. Yu, H. Jiang, S. Huang, X. Peng, A. Lu, *IEEE Circuits Syst. Mag.* **2021**, *21*, 31.
[7] F. Merrikh-Bayat, X. Guo, M. Klachko, M. Prezioso, K. K. Likharev, D. B. Strukov, *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *29*, 4782.
[8] Y.-H. Chen, T. Krishna, J. S. Emer, V. Sze, *IEEE J. Solid-State Circuits* **2017**, *52*, 127.
[9] G. W. Burr, R. M. Shelby, S. Sidler, C. di Nolfo, J. Jang, I. Boybat, R. S. Shenoy, P. Narayanan, K. Virwani, E. U. Giacometti, B. N. Kurdi, H. Hwang, *IEEE Trans. Electron Devices* **2015**, *62*, 3498.
[10] V. Joshi, M. Le Gallo, S. Haefeli, I. Boybat, S. R. Nandakumar, C. Piveteau, M. Dazzi, B. Rajendran, A. Sebastian, E. Eleftheriou, *Nat. Commun.* **2020**, *11*, 2473.
[11] D. Ielmini, H.-S. P. Wong, *Nat. Electron.* **2018**, *1*, 333.
[12] X. Sheng, C. E. Graves, S. Kumar, X. Li, B. Buchanan, L. Zheng, S. Lam, C. Li, J. P. Strachan, *Adv. Electron. Mater.* **2019**, *5*, 1800876.
[13] P. Yao, H. Wu, B. Gao, J. Tang, Q. Zhang, W. Zhang, J. J. Yang, H. Qian, *Nature* **2020**, *577*, 641.
[14] C. H. Ho, C. L. Hsu, C. C. Chen, J. T. Liu, C. S. Wu, C. C. Huang, C. Hu, F. L. Yang, in *Int. Electron Devices Meeting*, IEEE, Piscataway, NJ, **2010**, pp. 19.1.1–19.1.4.
[15] B. Govoreanu, G. S. Kar, Y.-Y. Chen, V. Paraschiv, S. Kubicek, A. Fantini, I. P. Radu, L. Goux, S. Clima, R. Degraeve, N. Jossart, O. Richard, T. Vandeweyer, K. Seo, P. Hendrickx, G. Pourtois, H. Bender, L. Altimime, D. J. Wouters, J. A. Kittl, M. Jurczak, in *Int. Electron Devices Meeting*, IEEE, Piscataway, NJ **2011**, pp. 31.6.1–31.6.4.
[16] J. Feng, Y. Wang, X. Hu, G. Wen, Z. Wang, Y. Lin, D. Wu, Z. Ma, L. Zhao, Z. Lu, Y. Xie, in *Silicon Nanoelectronics Workshop (SNW)*, IEEE **2021**, pp. 1–2, https://ieeexplore.ieee.org/abstract/document/9499323.
[17] P. M. Sheridan, F. Cai, C. Du, W. Ma, Z. Zhang, W. D. Lu, *Nat. Nanotech.* **2017**, *12*, 784.
[18] C. Li, M. Hu, Y. Li, H. Jiang, N. Ge, E. Montgomery, J. Zhang, W. Song, N. Dávila, C. E. Graves, Z. Li, J. P. Strachan, P. Lin, Z. Wang, M. Barnell, Q. Wu, R. S. Williams, J. J. Yang, Q. Xia, *Nat. Electron.* **2018**, *1*, 52.
[19] L. Gao, P.-Y. Chen, S. Yu, *IEEE Electron. Dev. Lett.* **2016**, *37*, 870.
[20] W. Wang, E. Covi, A. Milozzi, M. Farronato, S. Ricci, C. Sbandati, G. Pedretti, D. Ielmini, *Adv. Intell. Syst.* **2021**, *3*, 2000224.
[21] M. Prezioso, F. Merrikh-Bayat, B. D. Hoskins, G. C. Adam, K. K. Likharev, D. B. Strukov, *Nature* **2015**, *521*, 61.
[22] C. Li, D. Belkin, Y. Li, P. Yan, M. Hu, N. Ge, H. Jiang, E. Montgomery, P. Lin, Z. Wang, W. Song, J. P. Strachan, M. Barnell, Q. Wu, R. S. Williams, J. J. Yang, Q. Xia, *Nat. Commun.* **2018**, *9*, 2385.
[23] A. Fumarola, S. Sidler, K. Moon, J. Jang, R. M. Shelby, P. Narayanan, Y. Leblebici, H. Hwang, G. W. Burr, *IEEE J. Electron Dev. Soc.* **2018**, *6*, 169.
[24] F. Cai, J. M. Correll, S. H. Lee, Y. Lim, V. Bothra, Z. Zhang, M. P. Flynn, W. D. Lu, *Nat. Electron.* **2019**, *2*, 290.
[25] F. Cai, S. Kumar, T. Van Vaerenbergh, X. Sheng, R. Liu, C. Li, Z. Liu, M. Foltin, S. Yu, Q. Xia, J. J. Yang, R. Beausoleil, W. D. Lu, J. P. Strachan, *Nat. Electron.* **2020**, *3*, 409.
[26] C.-X. Xue, T.-Y. Huang, J.-S. Liu, T.-W. Chang, H.-Y. Kao, J.-H. Wang, T.-W. Liu, S.-Y. Wei, S.-P. Huang, W.-C. Wei, Y.-R. Chen, T.-H. Hsu, Y.-K. Chen, Y.-C. Lo, T.-H. Wen, C.-C. Lo, R.-S. Liu, C.-C. Hsieh, K.-T. Tang, M.-F. Chang, in *IEEE Int. Solid- State Circuits Conf.*, IEEE, Piscataway, NJ **2020**, pp. 244–246.
[27] Q. Liu, B. Gao, P. Yao, D. Wu, J. Chen, Y. Pang, W. Zhang, Y. Liao, C.-X. Xue, W.-H. Chen, J. Tang, Y. Wang, M.-F. Chang, H. Qian, H. Wu, in *IEEE Int. Solid- State Circuits Conf.*, IEEE, Piscataway, NJ **2020**, pp. 500–502.
[28] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, V. Srikumar, *SIGARCH Comput. Archit. News* **2016**, *44*, 14.
[29] Q. Wang, X. Wang, S. H. Lee, F.-H. Meng, W. D. Lu, in *IEEE Int. Electron Devices Meeting*, IEEE, Piscataway, NJ, **2019**, p. 14.4.1–14.4.4.
[30] F. Zahoor, T. Z. Azni Zulkifli, F. A. Khanday, *Nanoscale Res. Lett.* **2020**, *15*, 90.
[31] H.-S. P. Wong, H.-Y. Lee, S. Yu, Y.-S. Chen, Y. Wu, P.-S. Chen, B. Lee, F. T. Chen, M.-J. Tsai, *Proc. IEEE* **2012**, *100*, 1951.
[32] M. Gautschi, P. D. Schiavone, A. Traber, I. Loi, A. Pullini, D. Rossi, E. Flamand, F. K. Gurkaynak, L. Benini, *IEEE Trans. VLSI Syst.* **2017**, *25*, 2700.
[33] F. Conti, D. Rossi, A. Pullini, I. Loi, L. Benini, *J. Sign. Process. Syst.* **2016**, *84*, 339.
[34] D. Kamalanathan, S. Krishnan, F. Cai, F. Nardi, in *Soft Reset For Multi-Level Programming Of Memory Cells In Non-Von Neumann Architectures*, **2021**, https://patents.google.com/patent/US20210280247A1/en.
[35] L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, L. D. Jackel, Y. LeCun, U. A. Muller, E. Sackinger, P. Simard, V. Vapnik, in *Proc. of the 12th IAPR Int. Conf. on Pattern Recognition (Cat. No.94CH3440-5)*, IEEE Comput. Soc. Press, Jerusalem, Israel **1994**, pp. 77–82.
[36] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, *Proc. IEEE* **1998**, *86*, 2278.
[37] V. Sze, Y.-H. Chen, T.-J. Yang, J. S. Emer, *IEEE Solid-State Circuits Mag.* **2020**, *12*, 28.
[38] P.-Y. Chen, X. Peng, S. Yu, *IEEE Trans. Comput.-Aid. Des. Integr. Circuits Syst.* **2018**, *37*, 3067.
[39] O. Krestinskaya, K. N. Salama, A. P. James, *Adv. Intell. Syst.* **2020**, *2*, 2000075.
[40] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, *arXiv:1912.01703 [cs, stat]* **2019**.
[41] W. Shim, J. Meng, X. Peng, J. Seo, S. Yu, in *IEEE Int. Reliability Physics Symp.*, IEEE, Piscataway, NJ **2021**, pp. 1–4.
[42] W. Shim, Y. Luo, J. Seo, S. Yu, in *IEEE Int. Reliability Physics Symp.*, IEEE, Piscataway, NJ **2020**, pp. 1–5.
[43] A. Pappalardo, Xilinx/brevitas, Zenodo, **2021**, https://github.com/Xilinx/brevitas.
[44] S.-H. Yen, F. T.-W. Guo, *Bit-Ordered Binary-Weighted Multiplier-Accumulator*, https://patents.google.com/patent/US11194886B2/en.