Leakage Reuse for Energy Efficient Near-Memory Computing of Heterogeneous DNN Accelerators

Md Shazzad Hossain, *Member, IEEE*, and Ioannis Savidis, *Senior Member, IEEE*

***Abstract*— The exploration of custom deep neural network (DNN) accelerators for highly energy constrained edge devices with on-device intelligence is gaining traction in the research community. Despite the superior throughput and performance of custom accelerators as compared to CPUs or GPUs, the energy efficiency and versatility of state-of-the-art DNN accelerators is constrained due to a) the storage and movement of a large volume of data and b) the limited scope of monolithic architectures, where the entire accelerator executes only a single model at any given time. In this paper, a multi-voltage domain heterogeneous DNN accelerator is proposed that executes multiple models simultaneously with different power-performance operating points. The proposed architecture concurrently implements near-memory computing and leakage reuse, where the leakage current of idle memory banks within each processing element is utilized to deliver current to the adjacently placed multiply-and-accumulate (MAC) units. The proposed architecture and circuit techniques are evaluated with SPICE simulation in a 65 nm CMOS technology. The simulation results indicate that the proposed heterogeneous architecture with leakage reuse results in an energy efficiency of 3.27 tera-operations per second per watt (TOPS/W) as compared to a conventional monolithic and single voltage domain architecture that exhibits an energy efficiency of 0.0458 TOPS/W. In addition, the proposed accelerator that implements the leakage reuse technique on only half of the memory elements storing the weights reduces the power consumption of the sub-arrays of processing elements by 26% (99.4 mW) as compared to an accelerator that does not apply leakage reuse.**

***Index Terms*— Leakage reuse, near-memory computing, on-chip SRAM, heterogeneous DNN accelerator, ubiquitous DNN applications, edge devices.**

I. Introduction

ON DEVICE artificial intelligence (AI) is the primary driving force for edge devices, where the global market for edge computing is projected to increase to $43.4 billion dollars by 2027 from the 2021 estimated market value of $4.60 billion dollars [1]. Recent advances in deep neural network (DNN) models and DNN accelerators (customized hardware architectures optimized for DNN inference) have provided significant improvement in incorporating intelligence into ubiquitous edge devices that are constrained by highly stringent energy efficiency requirements. The use of edge devices for applications including computer vision, augmented reality (AR), face recognition, image processing, and speech applications require DNNs with varying specifications [2], [3]. The state-of-the-art DNN models customized for resource constrained edge devices are capable of *inference* with as few as 2-bits of arithmetic operation [4], while *training* is demonstrated with as few as 4-bits of arithmetic operation [5]. As the bit precision is reduced, the execution of both inference and training is more feasible on edge devices. However, hardware level implementations of optimized DNNs has not sufficiently progressed in the research community as compared to the breakthroughs made in model and algorithm development due to a) the lack of efficient circuits and architectures implementing the DNNs, and b) the higher power consumption of the DNN accelerators as a result of executing a large number of computations and the inclusion of large off- and on-chip memories.

Regardless of the model, application, and hardware architecture, all DNN accelerators require a sufficiently large data set, where the data is primarily categorized into three types: input activation, output activation, and weight (or filter). The DNN accelerators are efficient in performing convolution operations on an array of processing elements (PEs), where each PE is composed of single or multiple multiply-and-accumulate (MAC) units and local memories [6]. However, storing and moving a large amount of data poses a key challenge to improving the energy efficiency of a DNN accelerator [6], [7]. Prior research that analyzed the execution of Google workloads on consumer devices indicates that more than 60% of the total energy consumed by a system is due to data movement [8]. The large data sets required for DNN inference are stored in a combination of off-chip and on-chip memory, while the choice between off-chip and on-chip memory is based on the fundamental trade-off between latency and energy consumption [3], [7], [9].

The use of on-chip memory such as SRAM or embedded DRAM (eDRAM) reduces the latency of read and write operations at the cost of increased circuit area. In contrast, off-chip memory such as DRAM does not occupy on-chip area, while resulting in a significant increase in the latency and energy consumption of the overall system [3], [7], [9], [10]. The use of off-chip memory for edge devices with stringent energy budgets is not optimal as off-chip memory consumes orders of magnitude more energy than on-chip memory [3], [7], [9]. For example, a single off-chip DRAM access consumes 200× more energy than a MAC operation, while a single on-chip SRAM access consumes approximately 6× more energy than a MAC operation [7], [11]. Therefore, many current DNN accelerators store a significant portion of the data in on-chip SRAM to avoid expensive (increased latency and energy per access) off-chip data traffic and to improve the

Manuscript received May 1, 2021; revised September 5, 2021; accepted October 4, 2021. Date of publication October 21, 2021; date of current version December 13, 2021. This article was recommended by H. Homayoun. *(Corresponding author: Md Shazzad Hossain.)*

The authors are with the Department of Electrical and Computer Engineering, Drexel University, Philadelphia, PA 19104 USA (e-mail: msh89@drexel.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JETCAS.2021.3121687.

Digital Object Identifier 10.1109/JETCAS.2021.3121687

overall power-performance tradeoff [3], [7], [9], [11], [12]. For example, a 10-bit DNN accelerator implemented to operate at a near-threshold voltage (NTV) of 0.4 V utilizes separate memories of 400 KB each for the activation and weight parameters [12]. In addition, a 16-bit Eyeriss accelerator includes 181.5 KB of on-chip memory, where 0.5 KB of local memory is dedicated to each PE [13]. Furthermore, the DaDianNao accelerator utilizes separate on-chip memory spaces of 4 MB and 32 MB for, respectively, the input/output activations and weights [14]. The size of the on-chip SRAM as a percentage of the total circuit area is 37%, 32%, 38%, 20%, and 67.97% for, respectively, a TPU [15], Eyeriss V2 [6], DiaNano [16], Mythic [17], and the NTV [12] accelerator architecture. A large on-chip memory, however, results in a significant amount of leakage energy, where the leakage loss is further increased with technology scaling [18]–[20]. For example, the power consumption of an idle TPU that consists of 29.3 MB of on-chip memory is 28 W, which is equivalent to 70% of the 40 W of total power consumed while in active mode [15]. In addition, the power consumed by the on-chip memory corresponds to 86.64% of the total system power consumption of the NTV accelerator [12]. More recently, architectural techniques were proposed to address the power overhead due to data traffic including 1) in-memory computing, where computing is performed within the memory array [21] and 2) near-memory computing, where computing units are placed adjacent to the memory array [22].

The DNN accelerators are typically composed of an array of monolithic processing elements (a homogeneous accelerator architecture where all PEs are tasked with executing a single model at any given time), and all PEs are tied to a single power domain [3], [6], [7], [9], [11], [12], [15]–[17]. The implementation of a monolithic DNN architecture limits any improvement in the energy efficiency and performance of the system as i) all PEs are allocated for a particular model regardless of the actual hardware requirements of the executing model and ii) the same supply voltage is applied to all PEs regardless of the specific latency and throughput requirements of the executing application.

In this paper, a novel power management technique is proposed that applies *leakage reuse (LR)* [23], [24], a technique that recycles leakage current from idle circuit blocks and provides the leaked current to active circuit blocks, to implement near-memory computing of DNN accelerators. The proposed technique improves the energy efficiency of a heterogeneous and multi-voltage domain DNN accelerator, where multiple PEs are grouped to form N_{SA} number of sub-arrays that perform simultaneous execution of N_{SA} number of models.

The leakage current of idle SRAM banks or blocks is reused to supply current to the computing units of the DNN accelerator. More recently, variants of custom DNN ASICs have been proposed, where regardless of the underlying architecture, the generic structure of each accelerator is composed of 1) an array of processing elements, 2) on-chip memory, 3) off-chip memory, and 4) a communication network such as a network on chip (NoC) [11], [16]. The proposed multi-voltage domain heterogeneous DNN accelerator that implements near-memory computing with leakage reuse is shown in Fig. 1. The leakage current from idle on-chip memory (SRAM) is reused to deliver power to the computing units within the processing elements.

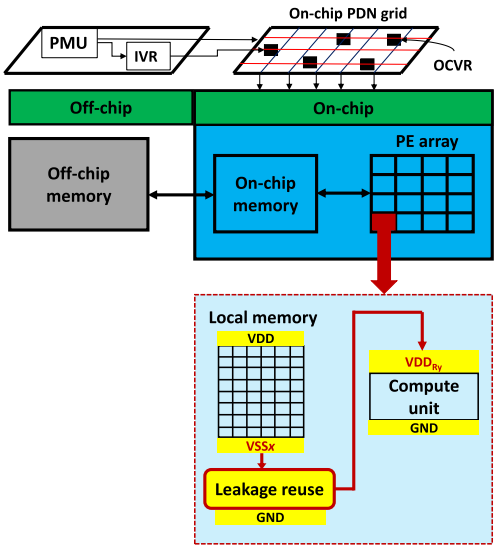

Fig. 1. Architecture of the proposed multi-voltage domain DNN accelerator implementing both near-memory computing and leakage reuse.

A conventional power delivery system is assumed for the memory banks, where the supply voltage *VDD* is generated and distributed through integrated voltage regulators (IVRs) and/or distributed on-chip voltage regulators (OCVRs) controlled by a power management unit (PMU) [25].

The primary contributions of this paper include

- a design space exploration analyzing the computational demand and execution time of the DNN models, which includes the characterization of multiple sub-layers of the models,
- a novel leakage reuse (LR) technique applied to on-chip SRAM, where the leakage current from idle memory banks (memory banks that are in the hold state) is recycled to deliver power to the adjacent processing elements of a DNN accelerator for near-memory computing, and
- a novel multi-voltage domain heterogeneous DNN accelerator architecture that executes multiple models simultaneously with different power-performance operating points, where each sub-array of PEs is tied to an independent power domain.

The rest of the paper is organized as follows. The leakage reuse technique is described in Section II. The challenges of state-of-the-art DNN accelerators and the proposed heterogeneous multi-voltage domain DNN accelerator architecture are discussed in Section III. The framework to evaluate the proposed techniques is described in Section IV, which is followed by a discussion of simulation results in Sections V, VI, and VII. In addition, a novel technique to maintain a stable supply voltage from the recycled leakage current of the memory cells is discussed in Section VIII. Some concluding remarks are provided in Section IX.

II. Leakage Reuse in Memory

Leakage reuse is a technique where the leakage current from an idle circuit block or core in a system-on-chip (SoC) is reused to deliver power to an active circuit block or core within the same SoC [23], [24], [26]. Prior research applied the leakage reuse technique to logic circuits [23], [24]. Recently,

circuits and algorithms for simultaneous implementation of leakage reuse and power gating were proposed [26]. In this paper, however, the leakage reuse technique is applied to on-chip memory (SRAM). Idle memory banks from which leakage current is reused are described as *donors* and computing units (PEs or MACs) to which reused charge is delivered as *receivers*. Unlike the prior techniques, in this paper the leakage current from memories (donors) is reused to supply power to computing units (receivers) that implement a near-memory computing architecture. The stored data is not affected when leakage reuse is applied, which is validated through SPICE simulation (more discussion is provided in Section II-B). The primary advantages of applying the leakage reuse technique include a) a reduction in the leakage current of the donors, which improves the overall energy-efficiency of the system, b) the leakage energy, which is otherwise considered wasted energy, is recycled to deliver power to computing units, and c) voltage regulators are not required to generate and regulate the supply voltages of the receivers, which reduces the area and power overhead of the IC.

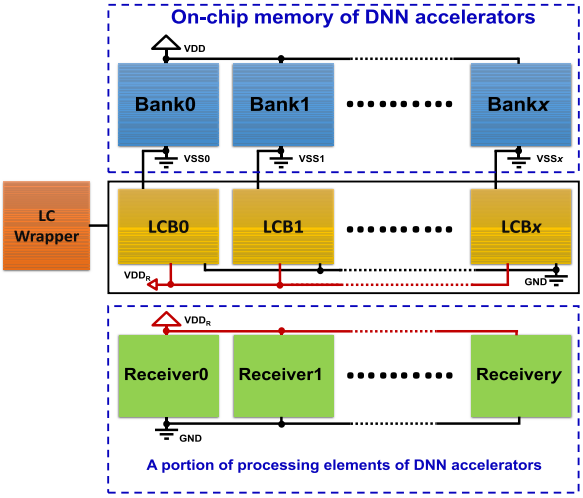

Fig. 2. An overview of applying the leakage reuse technique at the memory bank level, where leakage current from x number of SRAM banks (donors) provide power to y number of computing elements (receivers).

A. Bank-Level Leakage Reuse

Large state-of-the-art on-chip SRAM memories are composed of many smaller banks. The smaller distributed memory blocks allow for selective activation of banks that are required for specific workloads, which results in a significant reduction in the overall power consumption of the SoC [10], [27]. In addition, for conventional topologies, the maximum allowed bit cells per row or column in an SRAM array is limited to no more than 300 due to the increased *RC* delays on the wordlines and bitlines [27]. Furthermore, the size of the decoders is correspondingly increased, which results in greater delay as the size of the array is increased. The greater decoder delay is in addition to the large distributed *RC* load on the wordlines and bitlines. The use of banks also increases the throughput as *memory interleaving* is permitted, which is a technique where the addresses of large on-chip or off-chip memory are evenly distributed across many smaller memory banks [10]. The benefit of memory interleaving is that the execution of read/write operations and the waiting time to re-execute read/write operations on a specific memory address is reduced, which increases the data throughput [10].

Given the significant benefits of using SRAM banks to improve power consumption, latency, and throughput, in this paper the leakage reuse technique is applied to the SRAM banks, which are the smallest units the technique is implemented on. An overview of applying the leakage reuse technique to SRAM banks is provided through the block level representation shown in Fig. 2, where the leakage current from x number of donors (memory banks) is recycled to generate supply voltages for y number of receivers (computing units). Each donor consists of a switching fabric described as a leakage control block (LCB) that controls the flow of leakage current between the donor and receiver(s), while the leakage control wrapper (LC wrapper) provides the control signals to the LCBs.

The circuit implementation of the LCB and the LC wrapper is shown in Fig. 3. Each LCB consists of PMOS and NMOS switches that pass current in leakage reuse and active mode, respectively. Multiple PMOS switches are utilized to allow for finer control of the generated voltage for the receiver. The LC wrapper sets the control bits based on the activity of the donor(s) and receiver(s) and the amount of current required by the receiver to generate a given supply voltage. A binary 4-to-16 decoder is implemented that sets the idle donor memory bank(s) for leakage reuse. The NMOS transistor N of the LCB connects the virtual ground node *VSS0* of Bank0 to the true ground node *GND* when Bank0 is in active mode. The PMOS transistors $P0$ through $P2$ tie the *VSS0* node to the power network of the receiver when Bank0 is idle (hold state). The control signal $C0$, which is connected to the gates of the PMOS and NMOS transistors in Fig. 3, is set to 0 or 1 when Bank0 is operating in either idle mode or active mode, respectively. The transistors $P0$ to $P2$ must be large enough to sufficiently conduct the leakage current through the virtual ground (receiver power network) within a given clock period and without resulting in significant resistive voltage drop, while the threshold voltage of transistor N must be large enough to prevent leakage loss during the idle mode operation of the memory banks.

The circuit representation of Bank0 is shown in Fig. 4, where each bank has a separate virtual ground node. Note that the circuit structure of all remaining banks is identical to that of Bank0. Each bank consists of an $m \times n$ SRAM cell array, row/column decoders, sense amplifiers, and driver circuits as shown in Fig. 4, where in this work 16 rows ($m = 16$) and 16 columns ($n = 16$) are implemented in each bank.

Existing techniques that minimize the leakage current of SRAM memory cells are also considered in this work. The leakage current of a single 6T SRAM cell, a 16×16 cell array, and a 16×16 SRAM macro are characterized, with results as listed in Table I. Three scenarios are evaluated: a) an SRAM cell that includes transistors with standard threshold voltage V_t and standard gate length, b) an SRAM cell that includes high-V_t transistors, and c) an SRAM cell that includes transistors with 25% longer gate length. Note that high-V_t transistors are utilized in both the pull-up and the pull-down networks of both inverters of the 6T SRAM cell to produce the maximum reduction in the leakage current as reported in [28], [29]. Despite the reduction in the leakage current of the

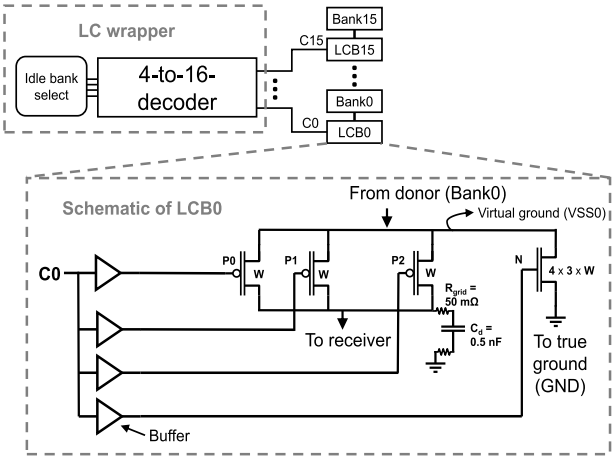

Fig. 3. Schematic representation of the leakage control blocks and the LC wrapper.

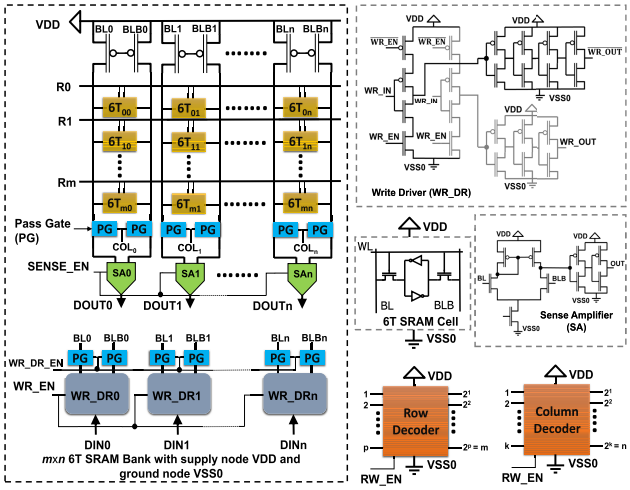

Fig. 4. Circuit representation of an implemented SRAM bank (Bank0) with a cell array of m rows and n columns, where the supply and ground node of Bank0 is, respectively, *VDD* and *VSS0*.

SRAM cells that include transistors with high-V_t and long gate length, a significant amount of power is dissipated as leakage, as indicated by the results listed in Table I. Specifically, the leakage power of a 6T SRAM is 95.25 pW, 60.08 pW, and 40.51 pW when implemented with transistors of, respectively, standard V_t and gate length, 25% longer gate length, and higher-V_t. In addition, SRAM cells that are implemented with transistors of higher threshold voltage and longer gate length result in greater access time for both read and write operations as indicated by the results listed in Table I, which negatively impacts the overall performance of the accelerator. Therefore, SRAM cells with both standard V_t and gate length are used in this work. However, the leakage reuse technique is applicable to the SRAM banks that are implemented with transistors of higher threshold voltage and longer channel length by adjusting the ratio of the donor area to receiver area at design time. Note that the leakage power of the SRAM banks is significantly larger than the total power of an 8-bit MAC even with transistors of higher-V_t and longer channel length.

Despite the reduction in the leakage current in more advanced process nodes, leakage current loss is not eliminated, and a significant amount of power is still dissipated as leakage.

TABLE I

CHARACTERIZATION OF LEAKAGE POWER AND ACCESS TIME OF SRAM WITH STANDARD-V_t TRANSISTORS, HIGH-V_t TRANSISTORS, AND TRANSISTORS WITH 25% LONGER CHANNEL LENGTH

	One 6T bit-cell	16×16 cell array	16×16 memory bank	Access time of a 6T memory cell	
				Read	Write
Standard-V_t and L	95.25 pW	104.5 μW	113.3 μW	0.179 ns	0.152 ns
High-V_t	40.51 pW	39.11 μW	47.91 μW	0.195 ns	0.172 ns
25% longer gate length (L)	60.08 pW	77.74 μW	86.54 μW	0.192 ns	0.166 ns
Total power of a 8b MAC (@0.45 V)	14.53 μW				

Specifically, the leakage power of an 8T SRAM cell is 114.44 pW and 27.5 pW for a 6T SRAM cell in, respectively, a 32 nm tri-gate CMOS and 7 nm FinFET process [30], [31]. The proposed technique generates the receiver supply voltages by tuning the size and configuration of the leakage control blocks, the LC wrappers, and the ratio of the donor area to the receiver area at design time. Therefore, even at advanced technology nodes, the leakage reuse technique remains applicable. Moreover, the complex circuit techniques available in state-of-the-art on-chip SRAM architectures including the sharing of decoders, the sharing of sense amplifiers, and the use of higher order address bits to apply chip select and bank addressing are not considered in this paper as 1) the design of the SRAM architecture is not a primary objective of this work and 2) the use of the SRAM architecture shown in Fig. 4 provides sufficient circuit details for the proof of concept described, where the inclusion of more complex techniques only increases the area and leakage current.

B. Operation of Donors During Leakage Reuse

The functionality of the SRAM banks (donors) and the computing units (receivers) when implementing the leakage reuse technique is analyzed. Four banks are implemented to characterize the functionality of the donors and receivers when implementing the proposed technique. The 6T SRAM bit cell array and the peripheral circuits of the memory bank are developed in a 65 nm CMOS technology. The results from transient analysis of Bank0, which include the implementation of the leakage reuse technique, are obtained through SPICE simulation and are shown in Fig. 5. A 4-bit address is used to access a smaller sub-array (4×4) of the memory, as shown in Fig. 5. At any given time, two banks are assumed active, while the remaining two are assumed idle. Bank0 and Bank1 (Bank2 and Bank3) share the same control signals. A 4-bit carry look-ahead adder is implemented as the receiver. A 0.45 V supply voltage is generated for the receiver from the leakage current of the two idle SRAM banks. The control signals for the SRAM banks are listed in Table II, where signals for only Bank0 are provided. At the beginning of the control cycle, Bank0 and Bank1 are active ($C0 = C1 = 1$) and the leakage current from the two idle banks (Bank2 and Bank3) is provided to the adder. A logic high is written into a memory cell located at address 0100, which is followed by a control pulse that changes the state of Bank0 to idle (hold state). Therefore, Bank0 and Bank1 are now available for leakage reuse. During the third control pulse, where Bank0 is in an active state again, the stored data is retrieved correctly. Similarly, a logic low is written

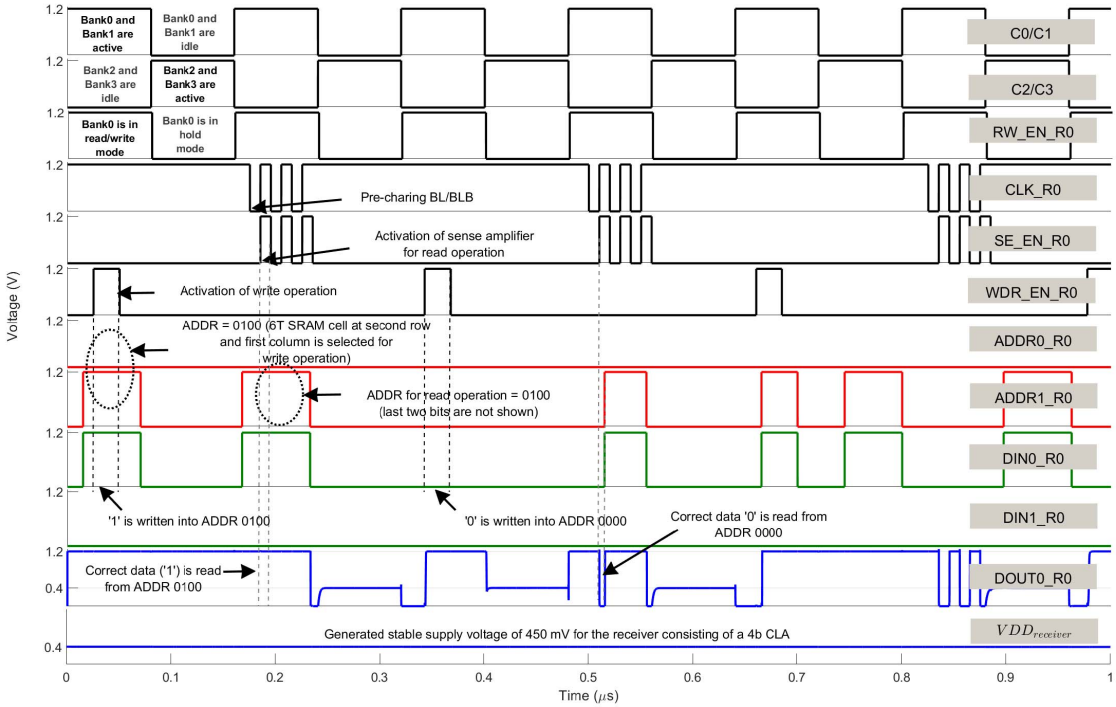

Fig. 5. Analysis of the functionality of SRAM Bank0 (donor) as a 4×4 cell array before and after implementing leakage reuse.

TABLE II
CONTROL SIGNALS CORRESPONDING TO SRAM BANK0

Signal	Operation
CLK_R0	= 0 : pre-charge = 1 : evaluate
RW_EN_R0 (read/write enable)	= 0 : hold mode = 1: read/write mode
SENSE_EN_R0 (enable sense amplifier)	= 0 : write mode = 1 : read mode
WR_DR_EN_R0 (write driver enable)	= 0 : read/hold mode = 1 : write mode
C0 (leakage control)	= 0 : LR mode = 1 : donor is active

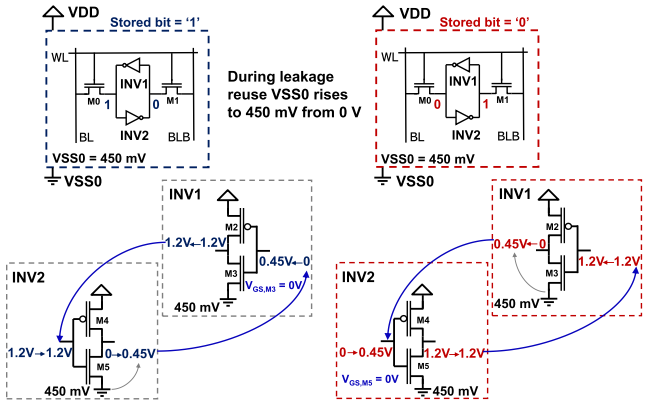

Fig. 6. Intrinsic data retention of SRAM cells when accounting for the implementation of the leakage reuse technique.

to memory address 0000 and is retrieved correctly. Therefore, the proposed technique recycles leakage current from memory banks without causing functional errors and data loss.

The implementation of the leakage reuse technique does not result in a perturbation of the stored bits of the SRAM cell arrays. As an example, a bit cell from Bank0 is analyzed as shown in Fig. 6. As the leakage reuse technique is only applied during the hold state of the memory bank, data must be retained when the bank is returned to the active operating mode (read/write). When the bit cell stores a logic '1', the input and output of INV2 is, respectively, 1.2 V and 0 V. As the bit cell enters the leakage reuse mode, the output node of INV2 rises to 0.45 V, which results in a corresponding change to the input of INV1 to 0.45 V from 0 V. However, the rise in the input voltage of INV1 does not perturb the output state of INV1 as the NMOS transistor M3 is in cut-off mode ($V_{GS,M3} = 0\ V$). Therefore, the 1.2 V input to INV2 remains unchanged. Similarly, when a logic '0' is stored on the bit cell, the output node of INV1 rises to 0.45 V from 0 V. In this case, the NMOS transistor M5 remains in cut-off, which prevents any perturbation of the output node of INV2. Therefore, stored data on the SRAM cells is retained even with implementation of the leakage reuse technique.

C. Operation of Receivers During Leakage Reuse

The functionality of the computing units (receivers) when implementing the leakage reuse technique is analyzed. A 4-bit carry look-ahead (CLA) adder that operates at a supply voltage of 0.45 V is implemented as the receiver. The supply voltage of the adder is generated from the leakage current of two memory banks. The results from transient analysis obtained through SPICE simulation are shown in Fig. 7, where two 4-bit numbers (A and B) are added to produce the output ADD_OUT. Despite the need to change donors while applying

leakage reuse, a stable voltage of 0.45 V is maintained at the receiver.

In addition, changes to the adder inputs, which represent variation in workload, result in minor voltage drops, as shown through the magnified view of the $VDD_{receiver}$ signal. A 0.5 nF on-chip decoupling capacitor is added as shown in Fig. 3 to maintain the $VDD_{receiver}$ signal close to the 0.45 V target. The total area of the 0.5 nF decoupling capacitor in a 65 nm fabrication process is 100 μm × 500 μm, which is composed of five units of capacitors connected in parallel. Each unit capacitance is implemented through a five metal layer (metal 4 to metal 8) stack that provides 0.1 nF. The adder is implemented with a single-stage pipeline and, therefore, requires a one clock cycle delay to provide a logical output.

III. Heterogeneous DNN Accelerator With Multiple Voltage Domains

The development of an optimized DNN accelerator is an area of on-going research effort [3], [7], [9], [11], [12], [15], [17], [21], [22]. Several custom architectures were proposed in the past five years that implement DNN accelerators with improved performance, throughput, and energy efficiency as compared to CPU, GPU, and FPGA based implementations [3], [7], [9], [11], [12], [15], [17], [21], [22]. However, as neural networks are implemented with a greater number of layers and with increasingly diverse layer structures, new challenges emerge due to a) the storing and management of a large volume of data sets, and b) the requirement of efficient dataflows for layer by layer processing of DNN models [3], [9]. In addition, the implementation of a DNN accelerator that offers both optimized energy efficiency and optimized performance across the various DNN models is a challenge as the architecture of the DNN accelerators is often fixed and only provides optimized results for a sub-set of DNN models [2], [3]. The primary challenges of implementing optimized DNN accelerators are discussed in this section. The challenges due to the management of the large number of network parameters are discussed in Section III-A. The challenges of executing DNN models on a target hardware platform in a sequential layer-by-layer flow is discussed in Section III-B. The limited scope of monolithic DNN accelerators is discussed in Section III-C. Finally, the proposed multi-voltage domain DNN accelerator that includes the implementation of the leakage reuse technique is described in Section III-D.

A. Storing a Large Number of Network Parameters On-Chip

Most prior research has focused on the development of efficient hardware that computes target DNN models, where an accurate characterization of the on-chip and off-chip memory requirements is not satisfactorily performed [2], [3], [6], [9]. Recent research proposed maximizing the size of the on-chip memory of DNN accelerators as a means to improve the performance and energy efficiency by avoiding expensive off-chip data transfers [3], [9]. In addition, several techniques are explored to reduce the memory footprints of deep neural networks [3]. The memory requirements of DNN accelerators for several neural network models are characterized with an emphasis on the size of the on-chip memory and the off-chip memory bandwidth, where results indicate that increasing the size of the on-chip memory improves the performance, bandwidth, and energy efficiency [3]. Recently, a memory management technique for DNN accelerators was proposed, where the activation memory of two sequentially adjacent layers are intentionally overlapped, as opposed to ping-pong buffering where the input and output activations of two consecutive layers are temporarily stored into two dedicated memory blocks for layer-by-layer processing [9]. By overlapping the activation memory, the energy overhead due to the on-chip memory of the DNN accelerators is reduced [9].

As the DNNs are implemented with deeper and larger networks to improve accuracy, the storage of a greater number of parameters in a combination of on-chip and off-chip memory is required. The increasing need for additional on-chip memory poses challenges on power and resource constrained DNN accelerator SoCs, where all or a majority of the network data (activations and weights) are stored in on-chip memory [3], [9]. Both the energy per access and the latency are reduced by storing the activations (inputs and outputs) and weights in on-chip memory [3]. For example, the DaDianNao accelerator includes a large on-chip memory to store all activations (4 MB) and weights (32 MB) of the executed models [14]. The SCNN accelerator stores all activations in on-chip memory (1.2 MB), while the weights are fetched from off-chip memory through a 32 KB FIFO [32]. The Eyeriss V2 architecture includes a total of 246 KB of on-chip storage for both activations and weights [6], while the EIE accelerator includes separate on-chip memory locations for activations (128 KB) and weights (10.2MB) [33].

The performance and energy efficiency of state-of-the-art DNN accelerators are, therefore, constrained by the on-chip memory, where the energy consumed by a unit operation of the on-chip memory is 6× greater than the energy required by a unit operation of computation [7], [9]. Activations and weights are conventionally stored in either a global on-chip memory and/or within each PE in separate local memory [3]. As recent research has targeted improving the breadth of models executed on a DNN accelerator, the minimum and maximum memory requirements vary by orders of magnitude for different neural network models [2], [3], [9]. Therefore, as the number of DNN models and the size of the neural networks increases, a greater portion of the total power consumption of a DNN accelerator is consumed by the on-chip memory [2], [6], [15]. In addition, as the size of the on-chip memory increases, the amount of unused memory for the implementation of smaller models at any given time results in a significant amount of energy loss due to leakage current.

B. Layer-by-Layer Processing of Neural Network Models

DNN models are composed of several convolution layers, fully connected layers, and activation layers, where the convolution layers are the most computation and memory intensive [34]. For an n-layer convolution operation, each weight kernel is convolved with the input feature map by sliding in the x and y directions [9]. Once the computation of the first layer is complete, the output is provided to the second layer, where the same computation is repeated [9]. Therefore, the convolutional neural network (CNN) performs layer-by-layer processing, where the output from a given layer is applied to the input of the subsequent layer [3], [9]. The

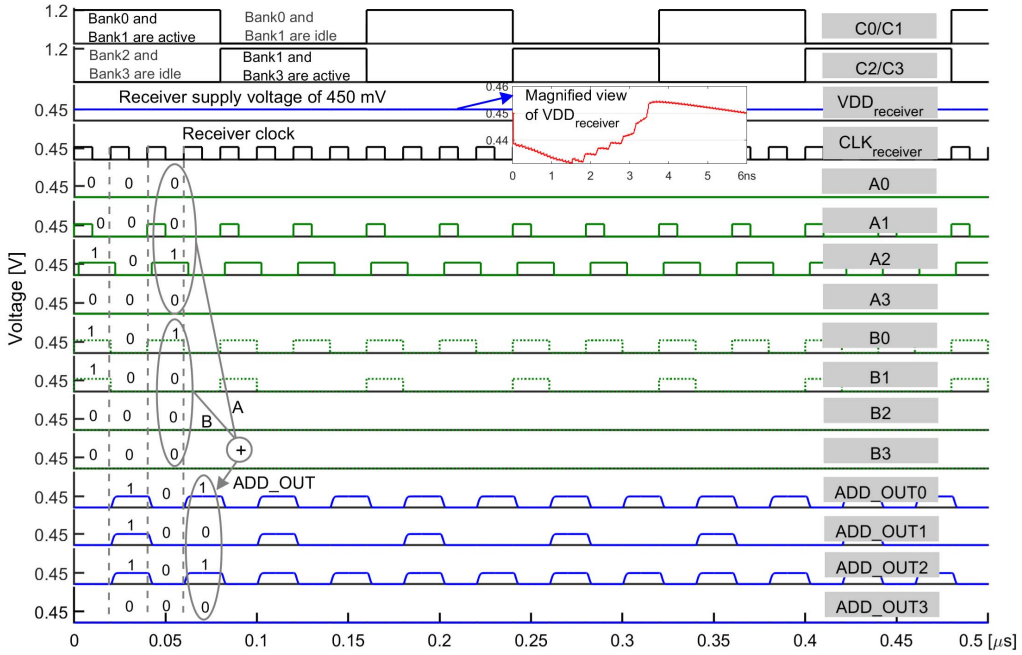

Fig. 7. Functional verification of a 4-bit carry look-ahead adder (receiver) that is supplied current from the implementation of the leakage reuse technique.

layer-by-layer computation and data access pattern is common to all CNNs regardless of the implemented model, executing application, and implemented hardware architecture. However, the layer-by-layer computation poses challenges when optimizing the utilization of the processing elements and the on-chip memory.

For a given CNN model, multiple layers exhibit different shapes (stride, padding, number of channels, and dimension of inputs, outputs, and weights), which results in disparate configurations of the PE array and memory. Therefore, the execution of models on an accelerator SoC must be dynamically assigned through an efficient dataflow [11]. Several optimized dataflows including row stationary (RO), output stationary (OS), and weight stationary (WS) were proposed to maximize the utilization of PEs across all layers [11]. However, a given dataflow optimizes the target hardware only for a sub-set of layers and models. Therefore, the challenge of efficiently utilizing on-chip memory resources and PE arrays remains. Specifically, available PEs are not best utilized for all layers, which results in the need for dynamic memory allocation for each layer.

In addition, the on-chip memory utilization at any given time changes when executing layer-by-layer processing. As noted in Section I and Section III-A, the size of the on-chip memory of current hardware accelerators is larger [3], [7], [9], [11], [12]. If the weights and/or activations of all layers are stored in on-chip memory, only a fraction of the total on-chip memory is utilized when executing a particular layer [9]. Therefore, a significant amount of energy is lost due to the leakage current through the idle memory banks, as leakage energy increases proportionally to the size of the circuits.

C. Limited Scope of Monolithic DNN Accelerators

The computational requirements, size of the on-chip memory, and the memory bandwidth vary by multiple orders of magnitude for different neural network models as well as

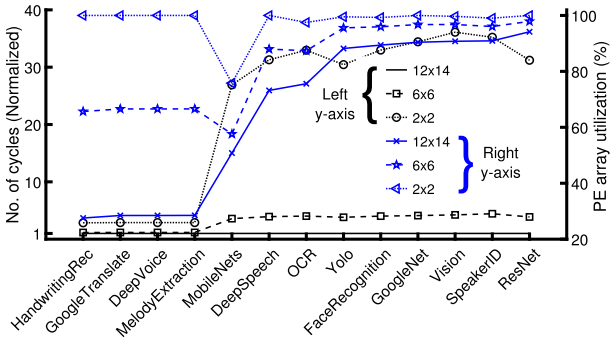

Fig. 8. Characterization of the utilization of the PEs and the average number of cycles to execute a set of DNN models for array sizes of 12×14, 6×6, and 2×2.

across layers within a given model [3], [9]. Maintaining a high energy efficiency when implementing a monolithic DNN accelerator is a challenge as a given dataflow does not map diverse layers and models optimally to the available hardware resources. In this work, an Eyeriss-like architecture is characterized using a cycle accurate neural processing unit (NPU) simulator [11], [35]. The architecture is analyzed across a set of DNN application models and for three PE array sizes (12×14, 6×6, and 2×2). A weight stationary (WS) dataflow is applied to all array sizes and models [7]. The average utilization of the PE array and the average number of cycles required to complete execution of a diverse set of models used for applications including vision, object detection, and speech recognition are characterized, with results as shown in Fig 8. The number of cycles of execution is normalized to the number of cycles required by the 12×14 array. For the 12×14 PE array, the average number of cycles required to complete execution of HandwritingRec, GoogleTranslate, DeepVoice, MelodyExtraction, MobileNet, DeepSpeech, OCR, Yolo, FaceRecognition, GoogleNet, Vision, SpeakerID, and ResNet is, respectively, 0.245K, 2.84K, 2.48K, 3.627K, 206K, 1639K,

127K, 1735K, 423K, 174K, 1823K, 6007K, and 462K. The average PE utilization is less than 90% for most of the models when the PE array size is 12×14, while the PE utilization significantly increases for the smaller array sizes of 6×6 and 2×2. Underutilization of PE arrays across hundreds to thousands of cycles results in a significant reduction in the energy efficiency of the accelerator due to increased leakage current. Conversely, an increase in the utilization of PEs results in a significant improvement of the total energy efficiency of the DNN accelerator. However, the reduction in the size of the array results in an increase in the average number of cycles needed to complete execution of the models by $1.16\times$ to $4.43\times$ and $2.86\times$ to $36\times$ for, respectively, an array of size 6×6 and 2×2. The overall power-performance trade-off is, therefore, improved when the different models and layers are optimally mapped to a heterogeneous PE sub-array.

State-of-the-art edge devices execute multiple applications that concurrently run in the background. As an example, edge devices performing augmented and virtual reality require concurrent execution of object detection, classification, hand tracking, depth estimation, and pose estimation, which require the MobileNet, ResNet50, UNet, DepthNet, and HandposeNet models, respectively [2]. In addition, due to the increasing complexity and the greater variety of DNN based workloads executed on edge devices, dynamic allocation of resources and computational loads is required [2], [34], [36]. Traditional DNN accelerators with monolithic architectures optimized to efficiently execute only a sub-set of models are not suitable for current trends in applications that require the use of a diverse set of DNN models. Recent research proposed flexible and heterogeneous DNN accelerators to improve the performance and energy efficiency of edge devices simultaneously running a diverse set of DNN models [2], [36]. The heterogeneous DNN accelerators are composed of multiple sub-arrays of PEs each optimized for different layer shapes and operations [2], [36]. Each sub-array of PEs is mapped to a dataflow that optimizes the utilization of resources and improves the overall power-performance trade-off.

In addition, monolithic DNN accelerators, where all PEs share a common power domain, limit any improvement in the energy efficiency of the circuit provided by techniques such as fine-grain dynamic voltage scaling, adaptive voltage scaling, and power gating [14], [15], [32], [33]. For example, more recently, an inference processor was implemented for improved energy efficiency, where all of the PEs operate at a near-threshold voltage of 0.4 V for the entire execution of the DNN [12]. While the power consumption is lower at 0.4 V as compared to 1.2 V, the operating frequency of 60 MHz is also significantly lower, which limits the inference processor to the implementation of only a sub-set of DNN models [12]. The highly constrained monolithic architecture is, therefore, not well suited for the execution of a diverse set of DNN models as the throughput, energy efficiency, and execution time of the accelerator are not dynamically adjustable.

The operating frequency of most state-of-the-art DNN accelerators is limited by the memory bandwidth despite the opportunity of running the computational units at much higher frequencies [6], [11]–[13]. For example, the Eyeriss accelerator implemented in a 65 nm CMOS fabrication process operates at a clock frequency of 200 MHz, where each PE consists of either a 16-bit MAC [11], [13] or two 8-bit MACs [6]. In addition, an inference processor implementing 848 KB of SRAM memory operates at a 120 MHz frequency for a supply voltage of 0.7 V in a 65 nm CMOS technology [12]. In both cases, the 65 nm technology permits much higher operating frequency. Therefore, there are opportunities to improve the overall system energy efficiency without significantly impacting the performance of the accelerator SoC by operating the computational units (MACs) at a lower supply voltage while operating memory at a higher supply voltage.

D. Proposed Heterogeneous DNN Accelerator Architecture Implementing Near-Memory Computing Through Leakage Reuse

A multi-voltage domain heterogeneous DNN accelerator architecture is proposed to address the limitations of monolithic DNN accelerators and the energy loss due to the leakage current of on-chip memory. The proposed architecture implements both near-memory computing and leakage reuse. The architecture of the DNN accelerator is shown in Fig. 9, where the accelerator is composed of multiple sub-arrays each operating in a separate voltage domain, as opposed to established architectures that implement one large PE array with a single voltage domain V0 as shown for the conventional DNN accelerator of Fig. 9. The input and output activations are stored in separate global on-chip memory, where the total size of each memory block is 108 KB. The weights required by the multiple layers are stored in an on-chip memory of 0.5 KB within each PE. The total memory to store weights for 168 PEs is 84 KB.

The proposed technique implements near-memory computing within a given PE by generating the supply voltage of the MAC units from the leakage current of the adjacent idle memory blocks that store the weight parameters. Within a processing element, the idle memory banks from all non-active layers of the DNN accelerator are utilized for leakage reuse, where at any given time a single layer and the corresponding weights are active. When any of the sub-arrays execute a given layer L, the weights associated with only layer L are in use, while the remaining memory cells storing weights are idle within each PE of a given sub-array.

The total number of PEs, the number of MACs per PE, the size of the memory storing the input and output activation values, and the size of the on-chip memory per PE are listed in Table III, where the hardware specifications are similar to that required by the Eyeriss architecture [6], [13]. However, unlike the Eyeriss architecture, the PEs for the proposed heterogeneous DNN accelerator are clustered into multiple sub-arrays with independent voltage domains. Therefore, each sub-array of PEs executes a separate DNN model with an optimized performance-energy operating point. The block level overview of a single PE is shown in Fig 10, where each PE includes 0.5 KB of on-chip SRAM and two fixed-point 8-bit MACs with two-stage pipelines that produce two multiply-and-accumulate results per cycle.

IV. Evaluation Framework

The proposed multi-voltage domain heterogeneous DNN accelerator consists of a total of 168 PEs and 300 KB of

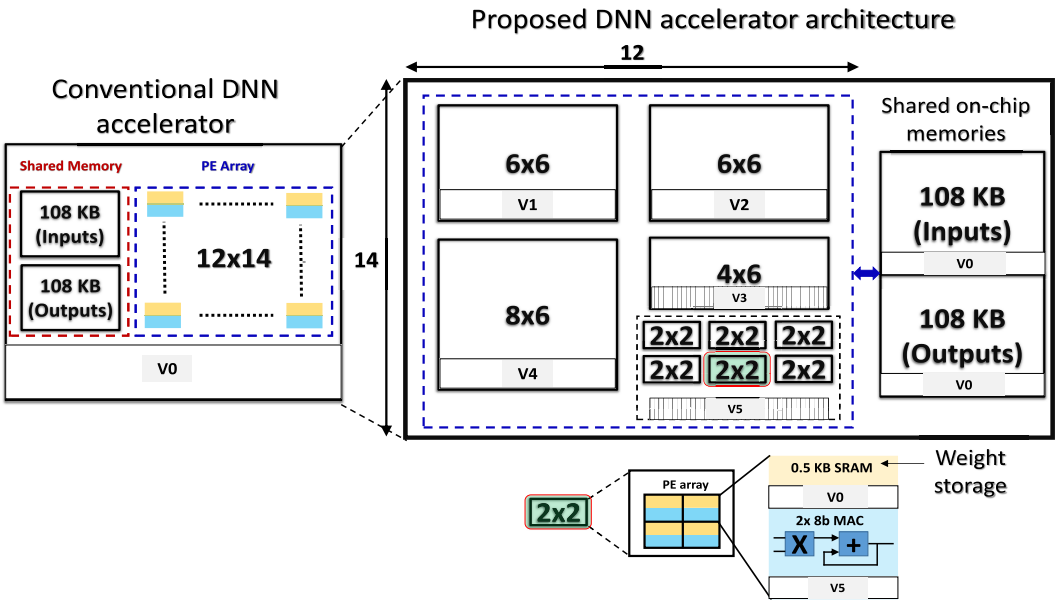

Fig. 9. Block level representation of the proposed multi-voltage domain heterogeneous DNN accelerator based on the Eyeriss architecture.

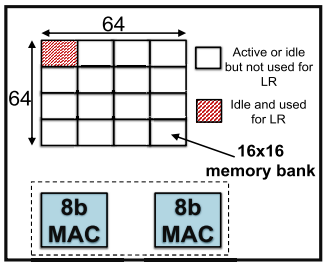

Fig. 10. A single processing element with a 0.5 KB on-chip memory and two 8-bit MACs. The 0.5 KB memory is implemented as 16 16 × 16 SRAM banks, where at any given time one bank is utilized for leakage reuse.

TABLE III

HARDWARE SPECIFICATION OF THE PROPOSED HETEROGENEOUS MULTI-VOLTAGE DOMAIN ACCELERATOR

Process technology	TSMC 65 nm LP
Total on-chip SRAM	300 KB
Total number of PEs	168
Number of MAC units per PE	2
Number of PE sub-array	10
Number of voltage domain	6
Shared memroy	216 KB
On-chip SRAM per PE	0.5 KB
Clock frequency	200 MHz
Combined peak throughput when all PE sub-arrays operate at 0.45 V	14.686 GOPS
Arithmetic precision	8-bit fixed point
Operating voltage	Memory: 1.2 V MAC: 0.45 V

on-chip SRAM memory. The accelerator is implemented and characterized through SPICE simulation in a 65 nm CMOS technology. Each 8-bit MAC unit consists of a radix-4 booth multiplier and a carry look-ahead adder. Six voltage domains (V0 to V5) are implemented as shown in Fig. 9 for the selection of the optimal power-performance point. However, two voltages are applied for the characterization of the heterogeneous accelerator, where voltage domain V0 is set to 1.2 V, and voltage domains V1, V2, V3, V4, and V5 are all set to 0.45 V. Therefore, ultra-low voltage scaling is not applied to the on-chip SRAM as 1) both the performance and throughput of DNN accelerators are already limited due to the on-chip memory and any further reductions in the operating voltage of the SRAM leads to significant degradation in the performance [6], [12], [18] and 2) operating the SRAM cells at ultra-low voltages requires specialized techniques that include write assist circuits or dual-supply rail architectures [37], [38], which are not objectives of this work.

The characterization of the proposed heterogeneous DNN accelerator is performed with 168 PEs that are clustered into ten sub-arrays consisting of a) one 8 × 6 sub-array with a throughput of 3.67 giga operations per second (GOPS), b) two 6 × 6 sub-arrays each with a throughput of 2.75 GOPS, c) one 4 × 6 sub-array with a throughput of 1.84 GOPS, and d) six 2 × 2 sub-arrays each with a throughput of 0.306 GOPS. The number of sub-arrays and the size of each sub-array are chosen such that the different throughput requirements of the DNN models and corresponding layers are met.

Three scenarios are considered to characterize the throughput and energy efficiency of the proposed accelerator, which are the 1) baseline, 2) proposed architecture without leakage reuse, and 3) proposed architecture with leakage reuse. The three scenarios differ based on the implemented power management technique. The baseline includes a single voltage domain V0 set to 1.2 V that provides current to both the memory and MAC units. For the second scenario, the heterogeneous DNN architecture includes two independent voltage domains, one for the on-chip memory and the other for the MAC units, which are set to, respectively, 1.2 V and 0.45 V. For the third scenario, the memory banks are supplied by a 1.2 V source, while the 0.45 V voltage for the MAC units is generated from the leakage current of idle memory banks within each PE. The 0.5 KB of SRAM memory in each PE that stores the weights consists of 16 banks of 32 bytes each in a 16 × 16 cell array. The on-chip SRAM of each PE is considered as the donor blocks that supply current, while the MAC

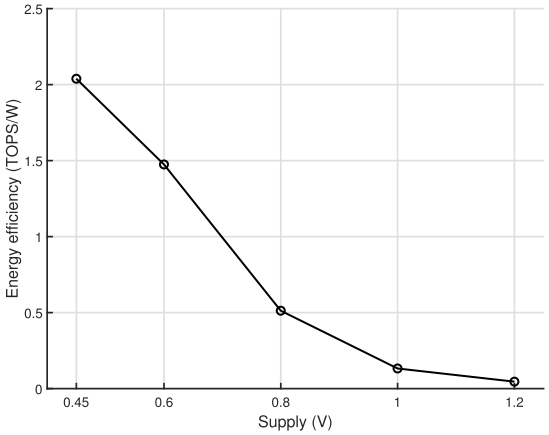

Fig. 11. Energy efficiency of the accelerator with a monolithic 12×14 PE array across five voltages.

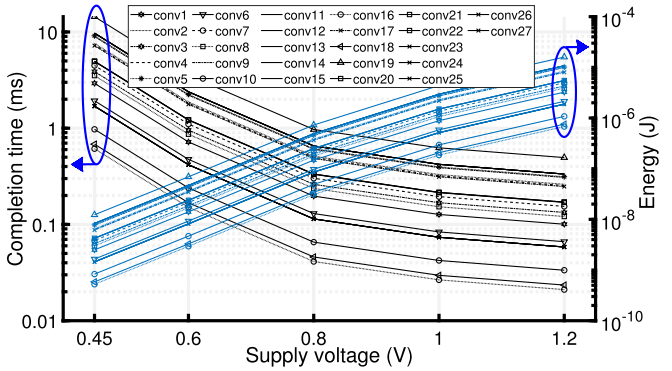

Fig. 12. A monolithic 12×14 PE array is characterized for energy consumption and completion time of each layer when executing the 27 convolutional layers of the MobileNets model and operating at each of five different supply voltages.

units are considered as the receivers. Therefore, for the third scenario, the 0.45 V supply voltage for domain V5 is generated from the implementation of the leakage reuse technique, while OCVRs provide the 1.2 V supply to the on-chip SRAM as shown in Fig. 1. Through simulation, the leakage current from an idle 16×16 SRAM bank (donor) is determined to sufficiently generate a stable supply voltage of 0.45 V for two 8-bit MAC units (receiver). Therefore, at any given time, only one idle memory bank (16×16) of a PE is utilized for leakage reuse. Note that the leakage reuse technique is capable of generating supply voltages that are either larger or smaller than 0.45 V for domains V1 to V4 [23], [26]. In addition, the ratio of the donor area to the receiver area is determined at design time as opposed to run-time due to the significant energy overhead of reconfiguration needed at run-time. Recent research has demonstrated that heterogeneous DNN accelerators exhibit superior energy efficiency as compared to reconfigurable DNN accelerators due to the requirement of per layer reconfiguration and the energy overhead of the switches, interconnect, and controllers needed to implement the reconfiguration [2].

V. Characterization of a Monolithic Accelerator Architecture Using the MobileNets Model

An accelerator that consists of 336 MAC units within 168 PEs is characterized for energy efficiency, which is represented as tera operations per second per watt (TOPS/W), across the five voltages of 1.2 V, 1 V, 0.8 V, 0.6 V, and 0.45 V with results as shown in Fig. 11. The energy efficiency increases as the supply voltage is reduced, with the PE array being 44.5× more efficient at 0.45 V (2.04 TOPS/W) as compared to 1.2 V (0.0458 TOPS/W). Therefore, the leakage reuse technique is evaluated at a supply voltage of 0.45 V.

In addition, the monolithic accelerator architecture is characterized through the implementation of the MobileNets model using a cycle accurate neural processing unit simulator to obtain the number of MAC operations required to execute each of 27 convolutional layers [35], [39]. The simulation provides an analysis of the energy consumed by each layer at voltages of 1.2 V, 1 V, 0.8 V, 0.6 V, and 0.45 V, with results as shown in Fig. 12. The number of MAC operations is directly proportional to the completion time of each layer as shown in Fig. 12, where the completion time at each voltage is calculated based on the minimum cycle time of the two 8-bit MACs. For example, the Conv27 layer is the most computationally expensive layer of the MobileNets model and requires 3282 MAC operations when executed on the 12×14 PE array. The energy consumption (completion time) of Conv27 is 15.96 µJ (0.5 ms), 4.37 µJ (0.63 ms), 0.73 µJ (0.97 ms), 0.07 µJ (3.6 ms), and 0.01 µJ (14.4 ms) when operating at a supply voltage of, respectively, 1.2 V, 1 V, 0.8 V, 0.6 V, and 0.45 V. Conv24 is the least computationally expensive layer and, therefore, requires only 139 MAC operations to execute, resulting in an energy consumption and a completion time of, respectively, 0.68 µJ and 0.021 ms at a supply voltage of 1.2 V. Among the 27 convolutional layers, the standard deviation (SD) of the number of MAC operations required is 845. The completion time and energy consumed per layer is also characterized for additional models including GoogleNet, ResNet50, Yolo, and AlexNet. The average mean and average standard deviation across all layers of the five models evaluated in this paper are provided through Fig. 13. Note that the results for the completion time and energy consumption are obtained for a supply voltage of 0.45 V. Similar to the MobileNets model, large variation in the MAC operations across all layers is observed for the other models, where the standard deviation of the number of MAC operations required across all layers of Googlenet, Resnet50, Yolo, and Alexnet is, respectively, 1782, 1236, 8813, and 414809. There is significant variation in the required computational resources and completion time across the multiple layers of a single model. The variation further increases when implementing multiple models as discussed in Section III-C. Therefore, there is benefit in computing multiple models on a heterogeneous PE array that accounts for the required number of MAC operations and the latency of each layer of the neural network.

VI. Characterization of the Energy Efficiency and Throughput of the Proposed Heterogeneous Architecture That Includes Leakage Reuse

The total power consumption and the throughput of the MAC arrays are characterized for the baseline and the proposed multi-voltage domain heterogeneous DNN accelerator architecture with and without applying leakage reuse, with results as shown in Fig. 14. The supply voltage is set to 1.2 V and 0.45 V for, respectively, the baseline and the proposed

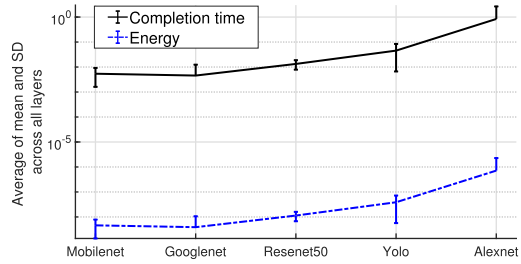

Fig. 13. Characterization of the mean and standard deviation of the completion time and energy consumed per layer, which are averaged across all layers of each respective model.

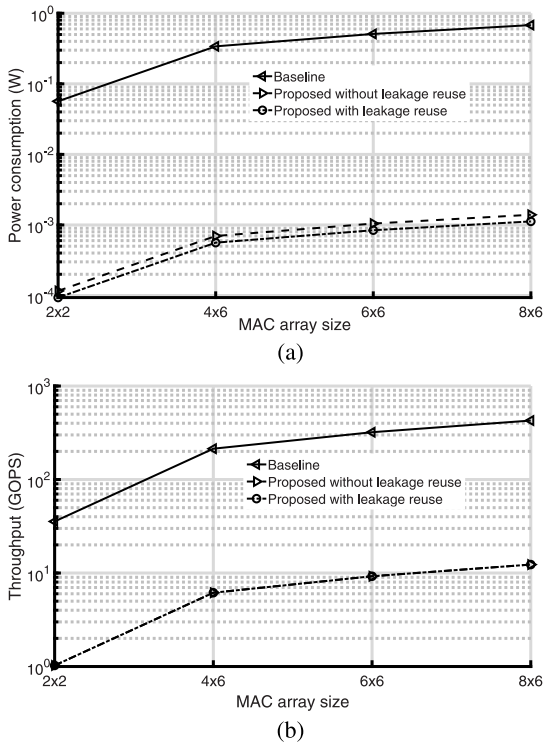

Fig. 14. Comparison of a) total power consumption and b) throughput for the baseline, proposed technique without leakage reuse, and proposed technique with leakage reuse.

technique (with and without leakage reuse) across the four sub-array sizes of 2×2, 4×6, 6×6, and 8×6. The relative difference between the three configurations in total power consumption and energy efficiency scales with sub-array size. The power consumption of the baseline is $605.5\times$ and $487.2\times$ that of, respectively, the proposed technique with leakage reuse and without leakage reuse. The throughput of the baseline MAC units operating at 1.2 V is $35\times$ the throughput of the proposed technique with or without leakage reuse as the supply voltage in both cases is set to 0.45 V.

The energy efficiency of the proposed architecture, with and without leakage reuse, is compared with a baseline homogeneous implementation, with results as shown in Fig. 15. Note that 1) the total power consumption and delay of the MAC arrays is considered when calculating the energy efficiency for each topology and 2) the leakage reuse technique is applied to 6.25% of the memory of a processing element. The implementation of leakage reuse on the proposed heterogeneous architecture exhibits a maximum energy efficiency

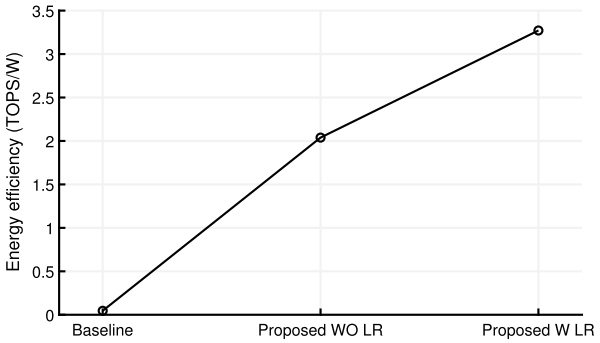

Fig. 15. Characterization of the energy efficiency of all MAC sub-array sizes in TOPS/W when assuming all processing elements in each topology are equally loaded and at the same voltage.

of 3.27 TOPS/W, which is $71.44\times$ and $1.60\times$ greater than the baseline and the proposed architecture without leakage reuse, respectively.

VII. Characterization of Total Power Consumption

The total power consumption of a 2×2 sub-array is characterized, where each processing element within the sub-array consists of two 8-bit fixed-point MACs and a 0.5 KB memory as shown in Fig 10. In addition, the total power consumption of a group of six 2×2 sub-arrays (shown in Fig. 9) is also characterized. The simulation results of both a single 2×2 sub-array and six 2×2 sub-arrays for the baseline, proposed architecture without leakage reuse, and proposed architecture with leakage reuse are shown in Fig. 16. The total power consumption of one 2×2 sub-array is 65.57 mW, 9.04 mW, and 8.73 mW for, respectively, the baseline, proposed architecture without leakage reuse, and proposed architecture with leakage reuse. Note that the energy loss due to the voltage regulators is not considered when characterizing the power consumption of the accelerator without leakage reuse. The relative ratio of the power consumption across the three topologies, which are described in Section IV, is maintained when characterizing the six 2×2 PE sub-arrays. Note that application of the leakage reuse technique to only six 2×2 sub-arrays, which includes a total of 24 PEs, results in a 1.9 mW (3.5%) reduction in the total power consumption. For the 24 out of 168 PEs that the leakage reuse technique is applied to, only 24 16×16 memory banks (one bank from each PE) are used for leakage reuse from a total of 2688 (16×168) memory banks in the entire accelerator. Therefore, 6.25% of the memory storing the weights in 24 PEs (24 banks out of 384 banks) is used for leakage reuse, where the 24 memory banks correspond to only 0.89% of the entire 2688 memory banks of the accelerator.

The total power consumption of all PE sub-arrays is characterized for seven different percentages (1%, 5%, 10%, 20%, 30%, 40%, and 50%) of idle on-chip memory used for leakage reuse as shown in Fig. 17, where all MACs operate at 0.45 V. The total power consumption includes the on-chip memory (84 KB) and MAC units within each of the 168 PE. The power savings increases as more of the memory used for the weights is included for leakage reuse. For example, the total power consumption of the accelerator is reduced by 6 mW and 99.4 mW when, respectively, 1% and 50% of the memory is used for leakage reuse.

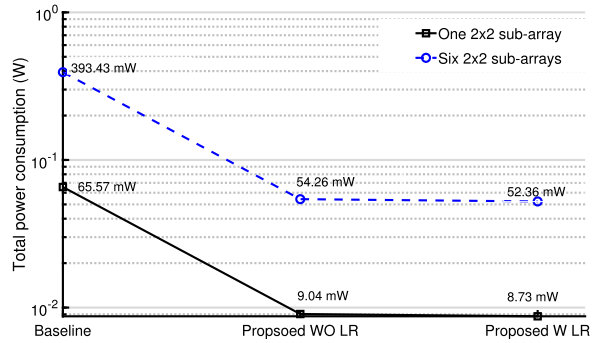

Fig. 16. Characterization of the total power consumption of a 2×2 sub-array, where a single PE within each 2×2 sub-array is composed of 0.5 KB memory and two 8-bit MACs.

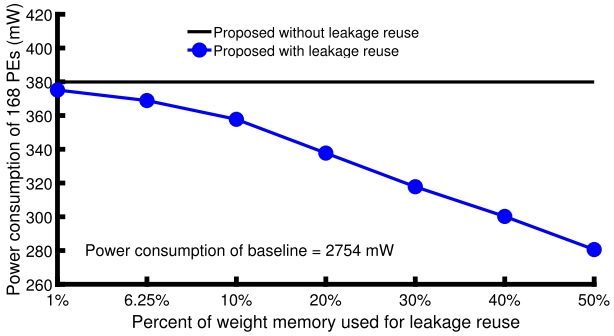

Fig. 17. Characterization of total power consumption of the accelerator SoC for four different percentage of weight memory utilization for leakage reuse.

TABLE IV
POWER CONSUMPTION OF THE BASELINE AND THE PROPOSED ACCELERATORS

Breakdown of total power consumption	Baseline	Proposed without leakage reuse	Proposed with leakage reuse
One PE including 0.5 KB of weight memory and two 8-bit MACs	16.39 mW	2.26 mW	1.21 mW
168 PEs	2754 mW	380 mW	203.35 mW
Overhead due to leakage reuse including 2688 LCBs, 168 LC wrappers, and 168 PVT mitigation circuit blocks	N/A	N/A	75.71 mW + 0.63 mW + 0.88 mW
Total power consumption	2754 mW	380 mW	280.57 mW

The components of the total power consumption of the baseline and the proposed accelerators is listed in Table IV, where the power is characterized for 50% utilization of the weight memory when leakage reuse is implemented. The proposed accelerator implemented with the leakage reuse technique reduces the power consumption by 26% (99.4 mW) as compared to an accelerator that does not apply leakage reuse. The simulated results are obtained from the execution of the accelerators across 8000 clock cycles and for a clock frequency of 200 MHz. Note that the active area of one 16×16 memory bank is 345.3 μm^2. The total area of the LC wrapper is equivalent to 4.75% of the area of a 16×16 memory bank. The total area of the leakage control block (LCB) when process, voltage, and temperature (PVT) mitigation circuits are included, as discussed in Section VIII, is equivalent to 2.93% of the area of a 16×16 memory bank.

VIII. CIRCUIT TECHNIQUES FOR ROBUST LEAKAGE REUSE UNDER PROCESS, VOLTAGE, AND TEMPERATURE VARIATIONS

The results described in Sections VI and VII are obtained for the typical-typical (TT) process corner and a temperature of 27°C. As noted, for the leakage reuse technique proposed in this paper, the supply voltage of the MAC units is generated from the leakage current of the SRAM banks. However, the leakage current is highly sensitive to process and temperature variations, which results in a deviation in the generated supply voltage of the MAC units from the target of 0.45 V.

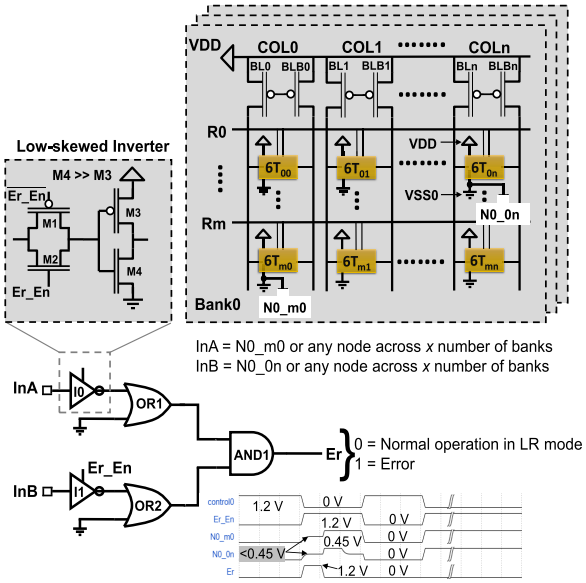

Fig. 18. Proposed circuit that maintains a stable supply voltage for the receiver in the presence of process and/or temperature variation.

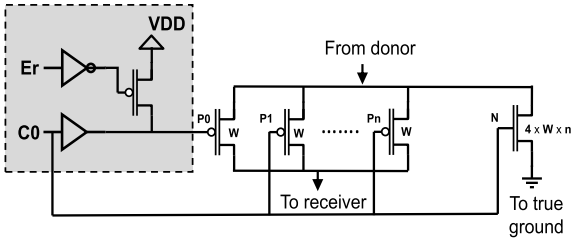

Fig. 19. Modified leakage control block that dynamically increases the supply voltage for the receiver.

In this paper, a novel circuit technique is developed to generate a stable supply voltage when applying leakage reuse in the presence of process, voltage, and temperature (PVT) variations. The proposed circuit is shown in Fig. 18, which dynamically detects any variation in the generated supply voltage from the leakage current of Bank0 and maintains a stable supply voltage of 0.45 V. The error detection circuit consists of two tri-state low-skewed inverters, two OR gates, and one AND gate. The term "error" refers to any increase or decrease in the supply voltage from the target value of 0.45 V. The two error detection circuits are placed on the virtual ground node of two memory cells of Bank0 to avoid any invalid activation of the error signal. When the voltage fluctuation on the virtual ground node of both cells *m0* (row *m*, column 0) and *0n* (row 0, column *n*) exceeds a set threshold, the error signal is asserted. The threshold for generating the error signal is tuned by the transistor widths of the error detection circuit, which is set to 0.43 V in this paper. The leakage control block (LCB), which is shown in Fig. 19, is modified to implement the proposed error detection and correction technique. Once the supply voltage deviates from the target value of 0.45 V,

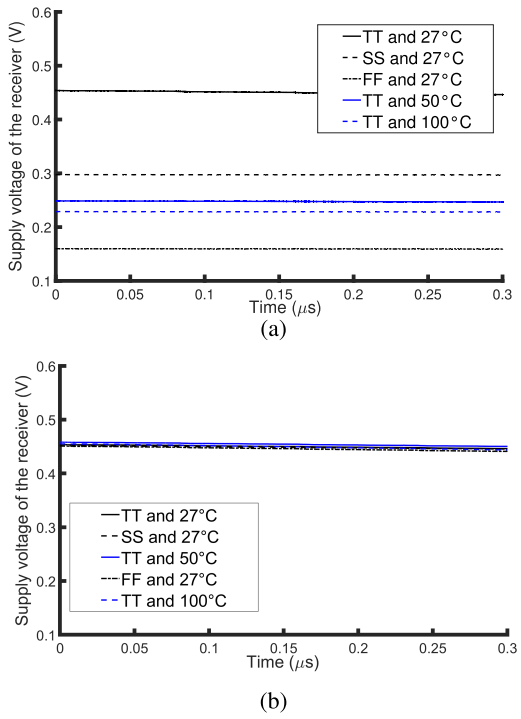

Fig. 20. Characterization of the supply voltage of the receiver for the TT, FF, and SS process corners and at temperatures of 27°C, 50°C, and 100°C, where a) includes the results without variation mitigation and b) depicts the results with the proposed process and temperature variation mitigation technique.

the error signal E_r is asserted as logic high. The error signal is connected to the modified leakage control block as shown in Fig. 19. The modified LCB consists of n number of PMOS transistors and one NMOS transistor that allow for the flow of current in both the idle and active mode of operation of the donor SRAM, respectively. However, the PMOS transistors are placed in parallel to enable fine-grain control of the generated supply voltage for the receiver. In this paper, three PMOS transistors (P0, P1, and P2) are used to recycle the leakage current of the idle memory units with a 1-bit control signal C0 that is applied to all three transistors. The gate of transistor P0 is also controlled by the E_r signal. Transistor P0 is forced to operate in the cut-off mode when E_r is set to logic high, which increases the supply voltage of the receiver. Note that the tuning of the supply voltage of the receiver to a different voltage than 0.45 V is possible at run-time by applying the multi-bit control signal Cn to the n number of PMOS transistors.

The supply voltage of the MAC units, which is generated through leakage reuse, is characterized for the typical-typical (TT), the fast-fast (FF), and the slow-slow (SS) process corners and at temperatures of 27°C, 50°C, and 100°C. The simulation results are shown in Fig. 20. The generated supply voltage significantly deviates from the target voltage of 0.45 V across process corners and temperatures. Note that in all cases, the supply voltage is lower than the target voltage due to the sizing of the PMOS and NMOS transistors within the leakage control block. The size of the NMOS transistor is 4× the size of the PMOS transistors to minimize the resistance along the path to ground during the active mode operation of the SRAM. The current through the NMOS transistor is the primary cause of any variation that results from operating the circuit in the SS corner, FF corner, and at higher temperatures (50°C and 100°C). Therefore, tri-state low-skewed inverters are used, which produce a logic high for an input voltage less than 430 mV and a logic low at the output when the input voltage is greater than 430 mV. Note that a logic high is required on the *Er_En* input signal to enable the inverters. The results from transient simulation of the mitigation circuit across process and temperature corners are shown in Fig. 20(b). At all process corners and temperatures, the proposed technique generates a stable supply voltage of 0.45 V for the MAC units.

IX. Conclusion

A heterogeneous multi-voltage domain DNN accelerator architecture is proposed that implements near-memory computing within a given PE. The leakage current from memory banks operating at 1.2 V is recycled to generate a supply voltage of 0.45 V for the adjacent MAC units. Therefore, a separate voltage source is not required for the MAC units operating at 0.45 V. In addition, a novel circuit to mitigate process and temperature variation effects is proposed that maintains a stable supply voltage of 0.45 V for the MAC units. The proposed architecture improves the energy efficiency by 71.4× (3.27 TOPS/W) as compared to a baseline topology that includes both memory and MAC units operating within a single voltage domain of 1.2 V. However, the throughput is reduced by 35×. Applying the leakage reuse technique to only half of all memory cells within each PE reduces the total power consumption by 26% (99.4 mW) as compared to an accelerator that does not apply leakage reuse. Therefore, the proposed architecture and techniques allow for more energy efficient means of inference for edge devices.

References

[1] Grand View Research. (Mar. 2020). *Edge Computing Market Size, Share and Trends Analysis Report*. [Online]. Available: https://www.grandviewresearch.com/industry-analysis/edge-computing-market

[2] H. Kwon, L. Lai, M. Pellauer, T. Krishna, Y.-H. Chen, and V. Chandra, "Heterogeneous dataflow accelerators for multi-DNN workloads," in *Proc. IEEE Int. Symp. High-Performance Comput. Archit. (HPCA)*, Feb. 2021, pp. 71–83.

[3] K. Siu, D. M. Stuart, M. Mahmoud, and A. Moshovos, "Memory requirements for convolutional neural network hardware accelerators," in *Proc. IEEE Int. Symp. Workload Characterization (IISWC)*, Sep. 2018, pp. 111–121.

[4] J. Choi, S. Venkataramani, V. Srinivasan, K. Gopalakrishnan, Z. Wang, and P. Chuang, "Accurate and efficient 2-bit quantized neural networks," in *Proc. Syst. Mach. Learn. (SysML) Conf.*, Apr. 2019, pp. 1–12.

[5] X. Sun *et al.*, "Ultra-low precision 4-bit training of deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, Dec. 2020, pp. 1–12.

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

[7] Y.-H. Chen, J. Emer, and V. Sze, "Using dataflow to optimize energy efficiency of deep neural network accelerators," *IEEE Micro*, vol. 37, no. 3, pp. 12–21, Jun. 2017.

[8] A. Boroumand *et al.*, "Google workloads for consumer devices: Mitigating data movement bottlenecks," in *Proc. ACM Int. Conf. Architectural Support for Program. Lang. Operating Syst.*, pp. 316–331, Mar. 2018.

[9] P. Jokic, S. Emery, and L. Benini, "Improving memory utilization in convolutional neural network accelerators," *IEEE Embedded Syst. Lett.*, vol. 13, no. 3, pp. 1–4, Sep. 2021.

[10] S. Thoziyoor, J. H. Ahn, M. Monchiero, J. B. Brockman, and N. P. Jouppi, "A comprehensive memory modeling tool and its application to the design and analysis of future memory hierarchies," *ACM SIGARCH Comput. Archit. News*, vol. 36, no. 3, pp. 51–62, Jun. 2008.

[11] Y.-H. Chen, J. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," *ACM SIGARCH Comput. Archit. News*, vol. 44, no. 3, pp. 367–379, Jun. 2016.
[12] J. Sim, S. Lee, and L.-S. Kim, "An energy-efficient deep convolutional neural network inference processor with enhanced output stationary dataflow in 65-nm CMOS," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 1, pp. 87–100, Sep. 2019.
[13] Y.-H. Chen, T. Krishina, J.-S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Nov. 2016.
[14] Y. Chen *et al.*, "DaDianNao: A machine-learning supercomputer," in *Proc. 47th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Dec. 2014, pp. 609–622.
[15] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proc. 44th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2017, pp. 1–12.
[16] T. Chen *et al.*, "Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning," *SIGARCH Comput. Archit. News*, vol. 42, no. 1, pp. 269–284, Feb. 2014.
[17] D. Fick and M. Henry. (Aug. 2018). *Analog Computation in Flash Memory for Datacenter-Scale AI Inference in a Small Chip*. Accessed: Apr. 15, 2021. [Online]. Available: https://www.hotchips.org
[18] M. Qazi, M. Sinangil, and A. Chandrakasan, "Challenges and directions for low-voltage SRAM," *IEEE Design Test. Comput.*, vol. 28, no. 1, pp. 32–43, Jan. 2011.
[19] A. C. Cabe, Z. Qi, and M. R. Stan, "Stacking SRAM banks for ultra low power standby mode operation," in *Proc. 47th Design Automat. Conf. (DAC)*, Jun. 2010, pp. 699–704.
[20] H. Qin, Y. Cao, D. Markovic, A. Vladimirescu, and J. Rabaey, "SRAM leakage suppression by minimizing standby supply voltage," in *Proc. IEEE 5th Int. Symp. Quality Electron. Design*, Mar. 2004, pp. 55–60.
[21] Q. Dong *et al.*, "A 351 TOPS/W and 372.4 GOPS compute-in-memory SRAM macro in 7 nm FinFET CMOS for machine-learning applications," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2020, pp. 242–244.
[22] J. Wang *et al.*, "A 28-nm compute SRAM with bit-serial logic/arithmetic operations for programmable in-memory vector computing," *IEEE J. Solid-State Circuits*, vol. 55, no. 1, pp. 76–86, Jan. 2020.
[23] M. S. Hossain and I. Savidis, "Reusing leakage current for improved energy efficiency of multi-voltage systems," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2019, pp. 1–5.
[24] M. S. Hossain and I. Savidis, "Recycling of unused leakage current for energy efficient multi-voltage systems," *Microelectron. J.*, vol. 101, pp. 1–16, Jul. 2020.
[25] E. A. Burton *et al.*, "FIVR-fully integrated voltage regulators on 4th generation Intel Core SoCs," in *Proc. IEEE Appl. Power Electron. Conf. Expo. (APEC)*, Mar. 2014, pp. 432–439.
[26] M. S. Hossain and I. Savidis, "Dynamic idle core management and leakage current reuse in MPSoC platforms," in *Proc. ACM/IEEE Int. Symp. Low Power Electron. Design*, Aug. 2020, pp. 49–54.
[27] E. Karl *et al.*, "A 4.6 GHz 162 Mb SRAM design in 22 nm tri-gate CMOS technology with integrated active VMIN-enhancing assist circuitry," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2012, pp. 230–232.
[28] G. Pasandi, R. Mehta, M. Pedram, and S. Nazarian, "Hybrid cell assignment and sizing for power, area, delay-product optimization of SRAM arrays," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 66, no. 12, pp. 2047–2051, Dec. 2019.
[29] B. Amelifard, F. Fallah, and M. Pedram, "Leakage minimization of SRAM cells in a dual-V_t and dual-T_{ox} technology," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 16, no. 7, pp. 851–860, Jun. 2008.
[30] S. Paul *et al.*, "A sub-cm^3 energy-harvesting stacked wireless sensor node featuring a near-threshold voltage IA-32 microcontroller in 14-nm tri-gate CMOS for always-ON always-sensing applications," *IEEE J. Solid-State Circuits*, vol. 52, no. 4, pp. 961–971, Apr. 2017.
[31] M. U. Mohammed, A. Nizam, and M. H. Chowdhury, "Performance stability analysis of SRAM cells based on different FinFET devices in 7nm technology," in *Proc. IEEE SOI-3D-Subthreshold Microelectron. Technol. Unified Conf. (S3S)*, Oct. 2018, pp. 1–3.
[32] A. Parashar *et al.*, "SCNN: An accelerator for compressed-sparse convolutional neural networks," *SIGARCH Comput. Archit. News*, vol. 45, no. 2, pp. 27–40, May 2017.
[33] S. Han *et al.*, "EIE: Efficient inference engine on compressed deep neural network," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit.*, vol. 44, no. 3, Jun. 2016, pp. 243–254.
[34] J. Lee, S. Kang, J. Lee, D. Shin, D. Han, and H.-J. Yoo, "The hardware and algorithm co-design for energy-efficient DNN processor on edge/mobile devices," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 10, pp. 3458–3470, Oct. 2020.
[35] A. Samajdar, Y. Zhu, P. Whatmough, M. Mattina, and T. Krishna, "SCALE-sim: Systolic CNN accelerator simulator," 2018, *arXiv:1811.02883*. [Online]. Available: https://arxiv.org/abs/1811.02883
[36] L. Yang *et al.*, "Co-exploration of neural architectures and heterogeneous asic accelerator designs targeting multiple tasks," in *Proc. 57th ACM/IEEE Design Automat. Conf. (DAC)*, Jul. 2020, pp. 1–6.
[37] M. Clinton *et al.*, "A low-power and high-performance 10 nm SRAM architecture for mobile applications," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2017, pp. 210–211.
[38] A. Banerjee, M. E. Sinangil, J. Poulton, C. T. Gray, and B. H. Calhoun, "A reverse write assist circuit for SRAM dynamic write VMIN tracking using Canary SRAMs," in *Proc. 15th Int. Symp. Qual. Electron. Design*, Mar. 2014, pp. 1–8.
[39] A. G. Howard *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*. [Online]. Available: https://arxiv.org/abs/1704.04861

Md Shazzad Hossain (Member, IEEE) received the B.Sc. degree in electrical and electronic engineering from American International University Bangladesh, Bangladesh, in 2011, and the M.Sc. degree in computer engineering from Drexel University, PA, USA, in 2016. Before joining at Drexel, he worked at Ulkasemi Private Ltd., in the areas of ASIC verification, physical design, PCB design, and IC layout design. He was a Research Intern with Nokia Bell Labs, Murray Hill, NJ, USA, in 2019 and 2020, where he was involved in designing energy-efficient AI accelerator for deep neural networks. His current research interests include energy-efficient and ultra-low voltage circuits and systems, and power management.

Ioannis Savidis (Senior Member, IEEE) received the B.S.E. degree in electrical and computer engineering and biomedical engineering from Duke University, Durham, NC, USA, in 2005, and the M.Sc. and Ph.D. degrees in electrical and computer engineering from the University of Rochester, Rochester, NY, USA, in 2007 and 2013, respectively.

He joined the Department of Electrical and Computer Engineering, Drexel University, Philadelphia, PA, USA, in 2013, where he is currently an Associate Professor and directs the Integrated Circuits and Electronics Design and Analysis Laboratory. His current research interests include analysis, modeling, and design methodologies for high performance digital and mixed-signal integrated circuits, power management for SoC and microprocessor circuits, hardware security, including digital and analog obfuscation and Trojan detection, and electric and thermal modeling and characterization, signal and power integrity, and power and clock delivery for heterogeneous 2-D and 3-D circuits.

Dr. Savidis is a member of the Association of Computing Machinery, the IEEE Circuits and Systems Society, the IEEE Communications Society, and the IEEE Electron Devices Society. He was a recipient of the 2018 National Science Foundation Early Faculty (CAREER) Award. He serves on the organizing committees for the IEEE International Symposium on Hardware Oriented Security and Trust, the ACM Great Lakes Symposium on VLSI, and the International Verification and Security Workshop. He also serves on the editorial boards for the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS, *Microelectronics Journal*, and the *Journal of Circuits, Systems and Computers*.