AGNI: In-Situ, Iso-Latency Stochastic-to-Binary Number Conversion for In-DRAM Deep Learning

Supreeth Mysore Shivanandamurthy, Sairam Sri Vatsavai, Ishan Thakkar, and Sayed Ahmad Salehi
Department of Electrical and Computer Engineering, University of Kentucky, Lexington, KY 40506, USA

2023 24th International Symposium on Quality Electronic Design (ISQED) | 979-8-3503-3475-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/ISQED57927.2023.10129301

***Abstract*—Recent years have seen a rapid increase in research activity in the field of DRAM-based Processing-In-Memory (PIM) accelerators, where the analog computing capability of DRAM is employed by minimally changing the inherent structure of DRAM peripherals to accelerate various data-centric applications. Several DRAM-based PIM accelerators for Convolutional Neural Networks (CNNs) have also been reported. Among these, the accelerators leveraging in-DRAM stochastic arithmetic have shown manifold improvements in processing latency and throughput, due to the ability of stochastic arithmetic to convert multiplications into simple bit-wise logical AND operations. However, the use of in-DRAM stochastic arithmetic for CNN acceleration requires frequent stochastic to binary number conversions. For that, prior works employ full adder-based or serial counter-based in-DRAM circuits. These circuits consume large area and incur long latency. Their in-DRAM implementations also require heavy modifications in DRAM peripherals, which significantly diminishes the benefits of using stochastic arithmetic in these accelerators. To address these shortcomings, this paper presents a new substrate for in-DRAM stochastic-to-binary number conversion called AGNI. AGNI makes minor modifications in DRAM peripherals using pass transistors, capacitors, encoders, and charge pumps, and re-purposes the sense amplifiers as voltage comparators, to enable in-situ binary conversion of input statistic operands of different sizes with iso latency. Our evaluations, based on detailed SPICE simulations (https://github.com/uky-UCAT/AGNI_SPICE.git), show that AGNI can achieve savings of at least 8× in area, at least 28× energy-delay product (EDP), and at least 21× in $area \times latency$, compared to two in-DRAM stochastic-to-binary conversion circuits from prior works. These circuit-level benefits are demonstrated to propagate at the system-level to achieve at least 3.9× gain in performance across four deep CNN models.**

***Index Terms*—convolutional neural networks, processing-in-memory, stochastic to binary conversion.**

I. Introduction

Convolutional Neural Networks (CNNs) have gained immense popularity in recent times and are extensively used in many real-world applications related to machine learning (ML) and Artificial Intelligence (AI) [1] [2]. These CNNs mimic the structure and behavior of the human brain in visual recognition tasks. In general, CNNs use a large number of computationally complex arithmetic operations such as multiply-accumulate (MAC), nonlinear activation, and pooling. Although these CNN functions can be accelerated due to their high degree of compute parallelism, their acceleration using conventional ASIC platforms (e.g., Dadiannao [1], EIE [3]) is challenging due to the memory wall problem while accessing their large number of operands [2].

In modern deep CNNs, such as RESNET-50 [4] and GoogLeNet [5], MAC operations are the most compute and memory intensive operations. Therefore, to accelerate MAC operations, several prior works have explored Processing-In-Memory (PIM) designs. Among these, some PIM designs are based on the emerging non-volatile memory (NVM) crossbar technologies (e.g., ISAAC [2], PRIME [6], XNOR-RRAM [7]), some are based on the traditional DRAM technology (e.g., DRISA [8], SCOPE [9], DRACC [10], LACC [11]), and some are based on the SRAM technology (e.g., [12] [13] [14]). These PIM solutions work to prevent data migration in order to balance memory performance and computational efficiency while processing CNNs locally.

Among these PIM designs from prior works, the DRAM-based PIM designs are more favorable. This is because, compared to NVM, DRAM is more dominant memory technology for main memory in current computing systems, which makes adopting the DRAM-based PIM accelerators in current computing systems naturally more appealing. Moreover, compared to NVM, DRAM provides lower latency. DRAM is also more tolerant to frequent writing of partial results. On the other hand, SRAM is also prevalent in current computing systems, but the high area cost and low capacity of SRAM makes SRAM-based PIM accelerators less suitable for modern large-scale, deep CNNs. Due to these reasons, DRAM-based PIM accelerators are preferred by the industry as well [15] [16].

DRAM-based PIM accelerators for CNNs break a MAC operation into multiple functionally complete memory operation cycles (MOCs). However, these accelerators require huge number of MOCs per MAC, e.g., DRISA requires 222 MOCs per MAC [8]. Each MOC can incur up to 49ns latency and consume up to 4nJ of energy. Therefore, to reduce the required number of MOCs per MAC, SCOPE [9] and ATRIA [17] employed stochastic arithmetic. In ATRIA and SCOPE, the use of stochastic arithmetic could reduce the multiplication operations into simple bit-wise logical AND operations, which in turn reduced the per-MAC MOCs to 5/16 for ATRIA [17] and 25 for SCOPE [9].

Despite these advantages, the use of stochastic arithmetic in ATRIA and SCOPE for CNN acceleration requires frequent stochastic-to-binary (StoB) number conversions; one StoB conversion is required for every point in the per-layer output tensor. In these accelerators, StoB conversions consume substantial latency and energy, even though ATRIA's StoB operations are hidden from the critical path to some extent. SCOPE and ATRIA use a parallel pop counter (Parallel PC)

and Serial pop counter (Serial PC)-based StoB conversion, respectively. Parallel PC circuits require full adders, which can take up non-trivial area in DRAM peripherals [18]. On the other hand, Serial PC circuits require high-speed counters. The implementations of full adders and counters in DRAM cannot be optimized for area, latency and energy, due to the constraints of DRAM processes which are significantly different from the standard CMOS logic processes [19]. As a result, the advantages of using stochastic arithmetic are severely diminished in ATRIA and SCOPE.

To address these shortcomings, this paper presents a new substrate for in-DRAM StoB number conversion called AGNI. AGNI makes minor modifications in DRAM peripherals using pass transistors, capacitors, encoders, and charge pumps, to divide the StoB conversion process into four distinct steps: (i) DRAM row activation, (ii) stochastic to analog conversion, (iii) analog to transition-coded unary conversion, and (iv) transition-coded unary to binary conversion. Each step utilizes a distinct set of DRAM timing signals to orchestrate charge-sharing among various DRAM components, such as sense amplifiers, DRAM cells, bitlines, and capacitors. Moreover, for the "analog to transition-coded unary conversion" step, AGNI re-purposes the sense amplifiers as voltage comparators. Through all these steps, AGNI enables in-situ binary conversion of input statistic operands of different sizes.

The organization of this paper is as follows: Section II provides preliminaries; Section III describes the structure of our AGNI substrate; Section IV explains the four operational steps of AGNI substrate in detail, using the results of our conducted SPICE simulations (https://github.com/uky-UCAT/AGNI_SPICE.git); Section V discusses the ***overheads, SPICE simulations-based and CNN benchmarks-driven performance analysis, error analysis, and comparison with prior works***, for AGNI; Section VI concludes the paper.

II. Preliminaries

A. Stochastic versus Transition-Coded Unary Numbers

In the unipolar format of unary computing [20], a unary number *W* is a bit-stream of *N* bits that represents a real-valued variable $\upsilon \in [0,1]$ by encoding υ through the ratio N_1/N, where N_1 is the number of 1's in *W*. As shown in Fig. 1, a unary number (e.g., υ=0.5) can be presented in the stochastic format (also known as rate-coded unary format) (Fig. 1(left)) or in the transition-coded unary format (Fig. 1(right)). As evident from the figure, in the stochastic format the '1's in the bit-stream do not appear in a group, whereas in the transition-coded unary format the '1's in the bit-stream appear in group.

Fig. 1: The stochastic (rate-coded unary) representation (left) and the transition-coded unary representation (right) of a real value υ=0.5.

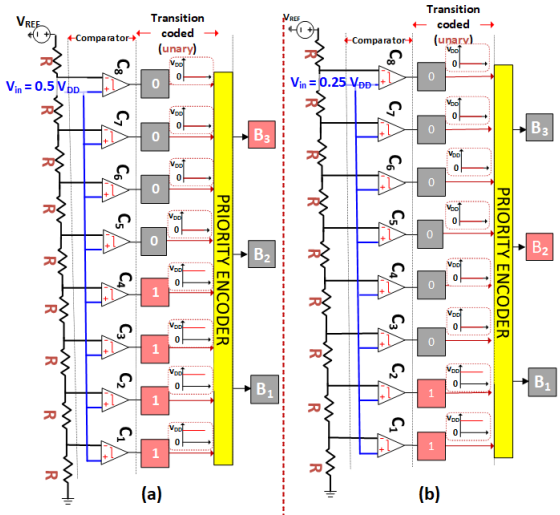

Fig. 2: Flash ADC with (a) $V_{in} = 0.5V_{DD}$, and (b) $V_{in} = 0.25V_{DD}$.

B. Flash ADC via Transition-Coded Unary Values

Fig. 2 shows a schematic of flash ADC (analog-to-digital converter) with 3-bit binary output. The figure shows the conversion of two example input values. As evident, a B-bit flash ADC employs one voltage divider, a total of 2^B comparators and one 2^B:B priority encoder (B=3 in Fig. 2). Thus, each circuit in Fig. 2 has eight voltage comparators (i.e., $C_1, C_2, C_3, C_4, C_5, C_6, C_7, and\ C_8$). The positive terminals of all comparators are connected to the analog input V_{in}. The negative terminal of the comparators are connected to the V_{REF} derived from the resistor-ladder based voltage divider. Suppose a scenario where V_{in} = 0.5 V_{DD} (Fig. 2(a)). In this case, the output of the priority encoder is binary four, and upon observation, the output of the comparators that is input to the priority encoder (i.e., the bit sequence 00001111) represents 0.5 in the transition-coded unary format. Similarly, if V_{in} = 0.25 V_{DD} (Fig. 2(b)), the output of the priority encoder is binary two, and the output of the comparators (i.e., the bit sequence 00000011) represents 0.25 in the transition-coded unary format. Thus, a flash ADC undertakes analog to binary conversion in two phases: first, analog to unary connversion through the comparators, and second, unary to binary conversion through the priority encoders. *Note that, as discussed later in the paper, our AGNI substrate re-purposes the sense amplifiers in DRAM tiles to implement this first phase of analog to binary conversion.*

III. Overview of Our AGNI Substrate

The purpose of our AGNI substrate is to enable in-situ conversion of input stochastic operands (bit-vectors) into binary numbers. To fulfill this purpose, the AGNI substrate employs a few modifications in the structure of each tile of a commodity DRAM module. These modifications are highlighted in Fig. 3(a). Evidently, our AGNI substrate logically groups the bitlines of each DRAM tile into multiple bitline groups (BLgroups). Each BLgroup corresponds to an input stochastic

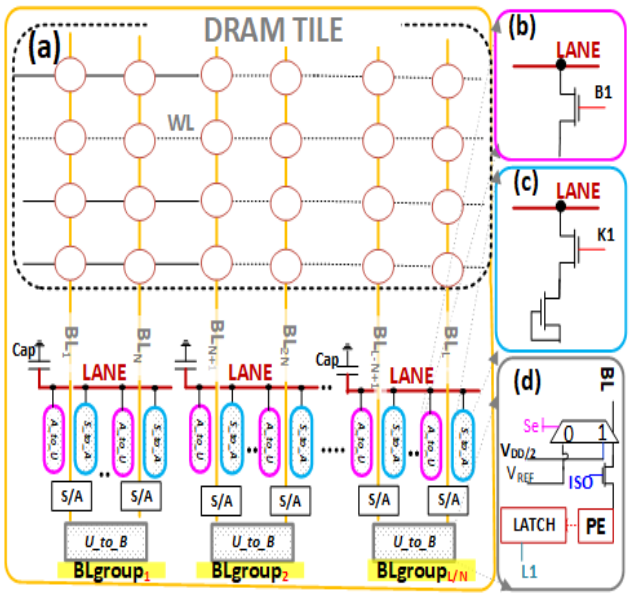

Fig. 3: Schematic layout of AGNI substrate and employed peripherals. Illustration of (a) an AGNI-modified DRAM tile, (b) an A_to_U peripheral unit, (c) an S_to_A peripheral unit, and (d) a U_to_B peripheral unit.

operand. Therefore, the number of bitlines in a BLgroup equals the number of bits in an input stochastic operand (i.e., the size of the input stochastic operand's bit-vector). Consequently, if the size of each input stochastic operand is N bits, and if each DRAM tile has a total of L bitlines (L is 256 or 512 typically), then each DRAM tile contains a total of L/N logical BLgroups, with each BLgroup having a total of N bitlines.

Further, to enable stochastic-to-binary number conversion of input operands, AGNI employs additional peripherals in each DRAM tile atop the already existing sense amplifiers (SAs). As shown in Fig. 3(a), these peripherals include per-bitline S_to_A units, per-bitline A_to_U units, per-BLgroup U_to_B units, and per-BLgroup analog lanes (see LANEs in the figure). Each LANE is horizontally laid out across a BLgroup and has a capacitor at the end. From Fig. 3(c), Each S_to_A unit contains two transistors (Fig. 3(c)), whereas each A_to_U unit contains one transistor (Fig. 3(b)). Each U_to_B unit contains one isolation transistor (ISO) per bitline (i.e., N ISOs per BLgroup) along with one priority encoder (PE) and a latch (Fig. 3(d)). All S_to_A and A_to_U units belonging to a BLgroup connect their corresponding bitlines and SAs to the corresponding LANE and analog lane capacitor. The N S_to_A units of a BLgroup enable stochastic-to-analog number conversion; the N A_to_U units of the same BLgroup enable analog-to-unary (transition coded unary) number conversion; and the U_to_B unit of the same BLgroup enables unary-to-binary number conversion. Thus, the additional peripherals of AGNI enable one stochastic-to-analog-to-unary-to-binary number conversion per BLgroup, thereby enabling a total of L/N such conversions in-parallel per DRAM tile. The operation of our AGNI substrate that enables such conversions is discussed next.

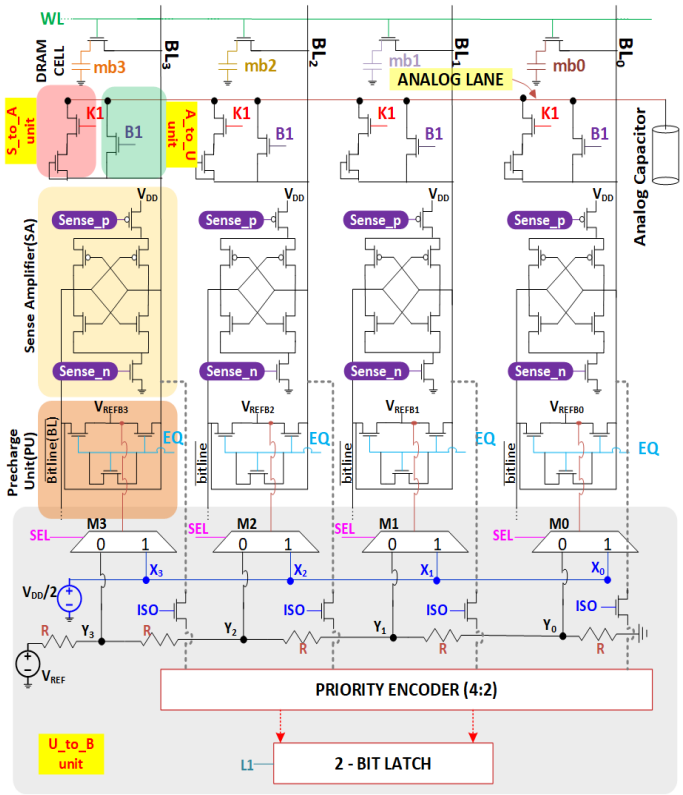

Fig. 4: Schematic of AGNI substrate for N = 4, consisting of peripherals such as S_to_A units, A_to_U units and U_to_B unit.

IV. Operation of Our AGNI Substrate

As implied from the previous section, AGNI substrate undertakes stochastic-to-binary conversion of input operands in the following three sequential steps: (i) stochastic to analog (S_to_A) conversion, (ii) analog to transition-coded unary (A_to_U) conversion, and (iii) transition-coded unary to binary (U_to_B) conversion. For these steps to work for an input stochastic operand, the operand needs to be read into the SAs of its corresponding BLgroup, which can be achieved by activating the DRAM row that contains the stochastic operand. Thus, a DRAM row activation must precede the above three steps, to constitute a sequence of a total of four steps for the operation of AGNI substrate for achieving stochastic-to-binary number conversion.

To realize these four steps, our AGNI substrate utilizes several timing signals. The timing signals required for the first step (i.e., DRAM row activation) include the standard DRAM operation signals [21] [22]. The remaining three steps require additional new timing signals to control the added peripherals of the AGNI substrate. The definitions and exact uses of these signals are summarized in Table I. These signals affect various hardware units of the AGNI substrate. This is illustrated in Fig. 4 for an example BLgroup of AGNI substrate with N = 4.

The BLgroup illustrated in Fig. 4 has 4 bitlines, i.e., $BL_0, BL_1, BL_2, and\ BL_3$. These bitlines correspond to four DRAM bit-cells, i.e., mb0, mb1, mb2, and mb3, respectively. From Fig. 4, each bitline is connected to a SA (highlighted

in light yellow) and a pre-charge unit (highlighted in light orange). Additionally, each bitline connects to one S_to_A unit (highlighted in light red) and a A_to_U unit (highlighted in light green). Moreover, all four bitlines of the BLgroup (i.e., $BL_0, BL_1, BL_2, and\ BL_3$) connect to one U_to_B unit (highlighted in grey), which consists of one N:$\log_2 N$ priority encoder, one $\log_2 N$-bit latch, N isolation transistors (ISOs), one resistor ladder based voltage divider, and N multiplexers that select the V_{REF} values for corresponding SAs. A V_{REF} value is either $V_{DD}/2$ or an appropriate level from the voltage divider. The selection of V_{REF} values from the voltage divider enables the SAs to operate as voltage comparators that can provide analog-to-unary conversion (just like flash ADC; Fig. 2). In the following subsections, we explain how the toggling of various timing signals listed in Table I has to be AGNI signals orchestrated to realize the four operational steps of AGNI substrate.

The exact time stamps for the toggling of these signals are summarized in Table II. The time evolution of these signals are also depicted in Fig. 5. Note that at the initialization, these signals are in the OFF state. The time evolution of these signals triggers the voltage levels corresponding to various DRAM structures (e.g., bitlines, analog capacitor, bit-cells) to evolve, which is also illustrated in Fig. 5. We have evaluated various timing and voltage signals depicted in Fig. 5 by modeling and simulating the circuit shown in Fig. 4 using LTSpice.

A. DRAM Row Activation (Step 1)

The DRAM row activation step employs EQ, WL, and sense_n (sense_p) signals to read the input stochastic operands into the SAs of their corresponding BLgroups. For this step, EQ is first toggled ON (to a higher voltage level in Fig. 5) at 0 ns. At 0 ns, we consider that SEL has been ON; therefore, at 0 ns, V_{REF} for the SAs has already been selected to be $V_{DD}/2$. As a result, the voltage on the bitlines swiftly settles to $V_{DD}/2$ after 0 ns time-stamp (see the evolution of voltage on the bitlines in Fig. 5(d)). This step is conventionally known as bitline pre-charging. We are able to achieve swift bitline pre-charging in AGNI because we consider short bitline DRAM architecture with only 8 cells per bitline [21]. After pre-charging, EQ is toggled OFF at 5 ns. Then, at 7 ns, WL is toggled ON. As a result, the DRAM cells (see mb0, mb1, mb2, and mb3 in Fig. 4) connect to their respective bitlines (see BL_0, BL_1, BL_2, and BL_3 in Fig. 4) to start sharing their charge with the bitlines. Due to this charge sharing, the voltage on the DRAM cells dips (see Fig. 5(e) at 7 ns) and the voltage on the bitlines is perturbed (see Fig. 5(d) at 7 ns).

Then, at 9ns, sense_n (sense_p) is toggled ON (see Fig. 5(f)), which enables the SAs to sense the perturbed bitline voltage and amplify it to the full swing. In Fig. 5(d), since the bitline voltage perturbation is in the positive direction (corresponding to logic '1' stored in the DRAM cells), the bitline voltage is swung to V_{DD} by the SAs. After this full-swing amplification of bitline voltage perturbation, the SAs complete replenishing the DRAM cell voltage at 11 ns. The perturbation amplification and cell replenishing both occur swiftly because we consider the short bitline DRAM architecture for AGNI. Then, at 12 ns, WL is toggled OFF, to disconnect the DRAM cells from the bitlines, to mark the end of the DRAM row activation step.

Note, during this step, the bitline voltage evolution encounters transient noise at two events, due to parasitic effects. First, at 5 ns when EQ is toggled OFF. This event is labeled as *glitch 1* in Fig. 5(d). Second, at 12 ns when WL is toggled OFF (labeled as *glitch 2* in Fig. 5(d)).

B. S_to_A Conversion (Step 2)

S_to_A conversion step (*Step 2*) employs sense_n (sense_p), and K1 signals to conduct the conversion of the stochastic operands (which are read into the SAs at the end of *Step 1*) into analog quantities (analog voltage levels). These analog quantities are accrued on respective analog capacitors; one analog voltage level per capacitor per BLgroup. For that, the analog capacitor of each BLgroup is forced to accrue charge incoming from respective SAs by having K1 signal to operate the S_to_A peripheral units of the BLgroup. Each S_to_A unit consists of a pass transistor and a diode (realized as the back-biased nmos transistor shown in Fig. 3(c) and Fig. 4). The presence of diode enables the connection of the SA to the pass transistor only if the SA has latched logic '1', i.e., if the corresponding bitline is at V_{DD}. Consequently, when K1 signal turns ON the pass transistors of all S_to_A

TABLE I: Definitions and uses of various timing signals employed by AGNI substrate.

Standard DRAM Operation Signals	
WL	Signal to turn on a DRAM wordline to enable charge sharing between a row of DRAM cells and corresponding bitlines
sense_p sense_n	Complementary signals that are used with each SA to enable the sensing and amplification of the bitline voltage perturbation
EQ	Signal for the precharge unit to equalize the BL voltages
Newly Added Timing Signals	
K1	Signal to turn on S_to_A units to enable charge flow from the SAs of a BLgroup to the analog LANE capacitor
B1	Signal to turn on A_to_U units to enable charge flow from the analog LANE capacitor of a BLgroup to the bitlines
ISO	Signal to turn on/off the isolation transistors, to connect/disconnect the priority encoder from a BLgroup
SEL	Signal to the MUXEs that enables the selection of a SA reference voltage (V_{REF})
L1	Signal to enable the latch for the binary result

TABLE II: Toggle time stamps (↑ or ↓) for various timing signals to realize the four operational steps of our AGNI substrate.

Activate	0ns ($EQ\uparrow$) 5ns ($EQ\downarrow$) 7ns ($WL\uparrow$) 9ns ($sense\_n\uparrow$) 12ns ($WL\downarrow$)
S_to_A	13ns ($K1\uparrow$) 37ns ($K1\downarrow sense\_n\downarrow$)
A_to_U	38ns ($EQ\uparrow$)($SEL\downarrow$) 42ns ($EQ\downarrow$) 43ns ($B1\uparrow$) 45ns ($sense\_n\uparrow$)
U_to_B	45ns ($ISO\uparrow$) 51ns ($L1\uparrow$) 52ns ($L1\downarrow$) 55ns ($B1\downarrow ISO\downarrow$)

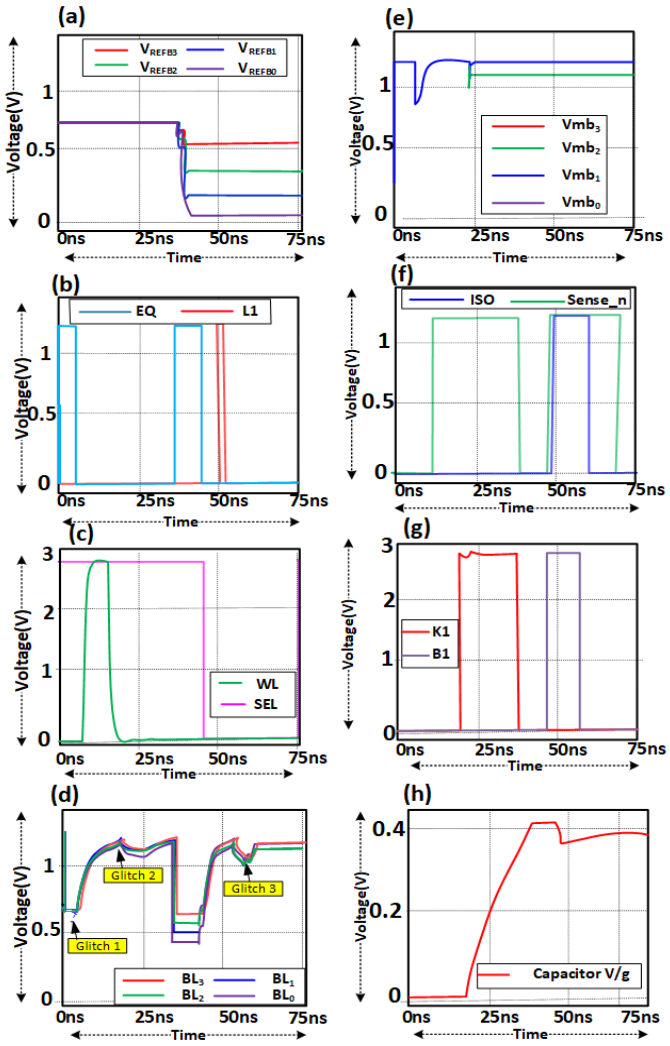

Fig. 5: Signal evolution traces from SPICE simulations of our AGNI substrate for N = 4. (a) voltages of the precharge units (V_{REF}), (b) equalizer (EQ) and Latching (L1) signals, (c) wordline (WL) and SEL signals, (d) bitline (BL) voltages, (e) DRAM cell capacitor voltage, (f) sense_n and isolation (ISO) signals, (g) K1 and B1 signals, and (h) analog capacitor voltage.

units of a BLgroup, the SAs of the BLgroup that are storing V_{DD} connect to the corresponding analog capacitor via the bitlines and analog LANE (Fig. 4). The SAs, if ON, then can force the analog capacitor to accrue some charge, and consequently, some analog voltage level. If SAs are kept ON for a fixed period of time, the accrued analog voltage level on the capacitor would be proportional to the number of SAs that are connected to the capacitor. Since only the SAs that are storing V_{DD} can connect to the analog capacitor, the accrued voltage level on the capacitor would be proportional to the number of logic '1's in the stochastic operand read into the SAs in *Step 1*. This is because only the SAs corresponding to logic '1' bits of the stochastic operand would be storing V_{DD} after *Step 1*). Thus, the voltage level accrued on the analog capacitor provides the analog representation of the stochastic operand.

Discussing the signal time-stamps, K1 is toggled ON at 13 ns (Fig. 5(g)). At this time, sense_n and sense_p are already ON, as they were toggled ON during *Step 1* at 9 ns. We then keep K1 and sense_n (sense_p) ON for 24 ns, during which the SAs accrue a voltage level on the analog capacitor (see Fig. 5(h)). In equilibrium, the analog LANE also accrues the same voltage level as the analog capacitor. Then, at 37 ns, both K1 and sense_n (sense_p) are toggled OFF (see Figs. 5(f) and 5(g)), to mark the end of Step 2. At the end, the voltage level accrued on the analog capacitor and LANE provides the analog representation of the stochastic operand.

During this step, for how long to keep K1 and sense_n (sense_p) ON is really a design choice. But since a thorough exploration of this design choice is beyond the scope of this paper, we decided the duration of 24 ns based on a coarse observation. We observe that our chosen duration of 24 ns is appropriate to provide sufficient noise margin so that different analog voltage levels accrued on the analog capacitor are unerringly distinguishable. We made this observation for our example AGNI substrate with N=4 shown in Fig. 4. For N=4, the total number of logic '1's in the input stochastic operand can take a total of four distinct values, i.e., 1, 2, 3, and 4. These four distinct values, respectively, correspond to {mb0=0, mb1=0, mb2=1, mb3=0}, {mb0=1, mb1=0, mb2=0, mb3=1}, {mb0=1, mb1=0, mb2=1, mb3=1}, and {mb0=1, mb1=1, mb2=1, mb3=1} in Fig. 4. For these, we evaluate how the analog voltage level accrued on the analog capacitor evolves during the 24 ns period; the evolution traces are shown in Fig. 6. For {mb0=mb1=mb2=mb3=1}, the accrued voltage at 37 ns reaches the maximum value V_{MAX} = 514 mV. For other cases, it is evident that the accrued voltage level is proportional to the number of '1's in the set {mb0, mb1, mb2, mb3} (i.e., in the input stochastic operand). We extend this analysis further and evaluate V_{MAX} for N of 16, 32, 64, 128, and 256 (these N values respectively correspond to binary number precision of 4-bit, 5-bit, 6-bit, 7-bit, and 8-bit). The results of V_{MAX} are presented in Table III. From these results, we observed that that our chosen 24 ns duration was

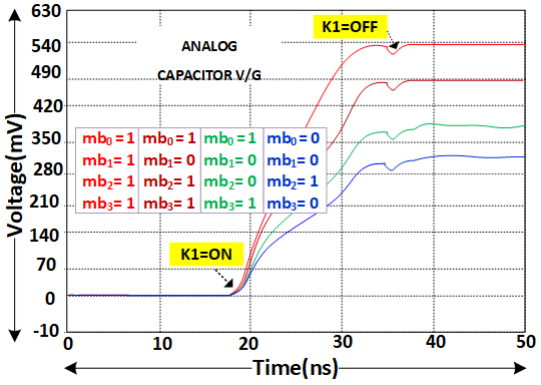

Fig. 6: Analog capacitor voltage for different 4-bit stochastic numbers.

sufficient to provide a total of *N* distinguishable voltage levels on the analog capacitor even for *N*=256. Thus, regardless of the value *N* (i.e., the length of the input stochastic operand), this S_to_A step can achieve the analog representation of an input stochastic operand with iso-latency of 24 ns.

C. A_to_U Conversion (Step 3)

As shown in Fig. 3, a flash ADC undertakes analog to digital (binary) conversion in two stages. In the first stage, an input analog voltage is converted into the equivalent transition-coded unary number using an array of voltage comparators. In the second stage, the unary number is then converted into the corresponding binary number using a priority encoder. The A_to_U conversion step of our AGNI substrate implements this second stage of a flash ADC by re-purposing the SAs as comparators. This re-purposing of SAs as comparators is realized using three phases of the A_to_U step of our AGNI substrate. For that, signals EQ, SEL, B1, and sense_n (sense_p) are employed.

In the first phase, SEL is toggled OFF at 38 ns, as shown in Fig. 5(c), to select V_{REF} values from the voltage divider circuit in the precharge units of all *N* SAs of a BLgroup (e.g., see V_{REFB0}, V_{REFB1}, V_{REFB2}, and V_{REFB3} in Fig. 4). Then, in the second phase, EQ is toggled ON at 38 ns and then toggled OFF at 42 ns, to precharge the bitlines of all *N* SAs to their respective V_{REF} values. As a result, between the 38 ns and 42 ns time-stamps, as shown in Fig. 5(d), the bitline voltages evolve to their respective V_{REF} values. Then, in the next phase, B1 is toggled ON at 43 ns so that the LANE and analog capacitor are connected to the bitlines to enable mutual charge sharing. As a result, the bitline voltages get perturbed by the 45 ns time-stamp (Fig. 5(d)). If the bitline corresponding to a SA (out of a total of *N* SAs) was precharged to a voltage level greater (less) than the voltage level accrued on the analog capacitor, the perturbation due to charge sharing would increase (decrease) the voltage of that bitline. To sense and amplify this perturbation, sense_n (sense_p) is toggled ON at 45 ns. Consequently, the voltages on the bitlines evolve to their full-swing values (V_{DD} or 0V) at 50ns, depending on the direction of the voltage perturbation. Therefore, some of the *N* SAs end up storing logic '1' and the others end up stoting logic '0', and the number of logic '1's out of *N* SAs provides the unary representation of the voltage on the analog capacitor.

Note that the positions of the '1's in the unary representation differ compared to the positions of the '1's in the input stochastic operand. To understand this, suppose the voltage accrued on the analog capacitor is $0.5V_{MAX}$ for *N*=4. This would really happen for the case where {mb0=1, mb1=0, mb2=0, mb3=1}. In this case, the perturbation would increase the voltages on only BL_0 and BL_1 in Fig. 4; the voltages on BL_2 and BL_3 would actually decrease after the perturbation. As a result, the SAs would sense and amplify logic '1's only for BL_0 and BL_1, thereby providing the positions of '1's in the unary representation to be 0011 (from the left to the right) when the positions of '1's in the stochastic input operand is 1001. This change in the positions of '1's in the unary representation favors the use of the priority encoder to convert from unary to the binary number format. Without this change, converting into the binary number format would require more complex combinational logic, such as a parallel pop counter used in [18] [9].

TABLE III: MAE, MAPE, RMSE, and V_{MAX} for AGNI substrate for different BLgroup sizes (different values of *N*).

N	MAE	MAPE%	RMSE	$V_{MAX}(mV)$
16	0.28	3.58	0.41	630
32	0.41	3.93	0.50	715
64	0.37	1.58	1.03	735
128	0.29	0.97	0.43	755
256	0.20	0.59	0.35	785

D. U_to_B Conversion (Step 4)

This step employs ISO, L1 and B1 signals as well as a priority encoder (similar to the one discussed Section II-B), to convert the unary number stored in the SAs at the end of *Step 3* into its corresponding binary number. For that, ISO is toggled ON at 45 ns (Fig. 5(f)), so that the ISO transistors turn ON to connect the bitlines to the priority encoder (Fig. 4). Therefore, when the SAs complete evolving the bitline voltages according to their stored unary number at 50 ns (as discussed in *Step 3*), the stored unary number reaches the priority encoder at 50 ns through the bitlines via the ISO transistors. Immediately after that, at around 51 ns, the priority encoder starts providing the converted binary number at its output. Therefore, L1 is toggled ON at 51 ns to enable latching of the priority encoder output (i.e., the binary number result). Then, L1 is toggled OFF at 52 ns, B1 and ISO are toggled OFF at 55 ns, to mark the end of the full operation cycle of AGNI.

Thus, AGNI can convert input stochastic number into the binary format in 55 ns (from *Step 1* to *Step 4*), irrespective of the size of the input stochastic operand (i.e., the value of *N*). Finally, at the end of 55 ns, each BLgroup of AGNI becomes available to convert a new stochastic operand.

During this step, the bitline voltages experience another glitch at 55 ns time-stamp (labeled as *glitch 3* in Fig. 5(d)), due to the toggling OFF of B1 that disconnects the bitlines from the LANE and analog capacitor.

V. Evaluation

A. Overheads of AGNI Substrate

To evaluate the area overheads of AGNI's peripheral units, we modeled our AGNI substrate on 2D DDR4_512 DRAM organization at 45 nm technology node using CACTI [23]. Each DRAM cell consumes $6F^2$ area, while the bitline pitch is 3F. Further, the stripes of SAs, precharge units, and write drivers have the heights of *117F, 90F, and 27F* respectively [24] [25]. Additionally, the heights of the peripheral units of AGNI, such as S_to_A, A_to_U and U_to_B are *27F*, *27F*, and *110F*, respectively. Therefore, the effective height of the AGNI substrate per 2D DDR4 DRAM tile comes out to be *164F*. Therefore, AGNI's total area overhead is $492F^2$. Moreover, we also evaluated the area and power overheads of the charge pump circuits, which we utilized to realize

TABLE IV: Charge pump (CP) area and power dissipation.

N	Area of CP (Acp)(μm^2)	Dynamic power per CP (W)	total wasted power per CP (W)
16	0.0087	1.30E-09	3.91E-09
32	0.0186	2.74E-09	8.22E-09
64	0.038	5.55E-09	1.67E-08
128	0.077	1.12E-08	3.37E-08
256	0.158	2.28E-08	6.85E-08

the voltage divider circuit (depicted in Fig. 4) that provides V_{REF} values to the precharge units. We did this evaluation for different values of *N*, using the methods from [26]. The results of our evaluation are reported in Table IV.

B. Setup for Performance Evaluation

We modeled our AGNI substrate in LTSPICE for $45\eta m$ gpdk technology node for five BLgroup sizes, i.e., with *N* = 16, 32, 64, 128, and 256. We considered the number of bitlines (*L*) per DRAM tile to be 512. *We will make our LTSPICE models publicly available*. For each *N* value, we simulated all possible stochastic numbers as input operands and simulated their conversion into binary numbers using our AGNI substrate's model in LTSPICE. Based on this exercise, we evaluated the mean absolute error (MAE) (Eq. 1), mean absolute percentage error (MAPE) (Eq. 2), and Root mean square error (RMSE) (see Eq. 3) for our simulated stochastic to binary conversions. In these equations (Eqs. 1 to 3), y_i is the predicted value, x_i is the actual value, and n is the total number of data points. Errors in AGNI substrate mainly emanate from the noise fluctuations during the charge-sharing phases, whenever such fluctuations are larger than the tolerable margins. The resultant error numbers are provided in Table III. In addition, as discussed earlier, the table also lists our evaluated V_{MAX} values.

$$MAE = (\sum_{i=1}^{n} |y_i - x_i|/n) \tag{1}$$

$$MAPE = \left(\frac{1}{n}\right) \sum_{i=1}^{n} \left| \left(\frac{x_i - y_i}{x_i}\right) \right| \tag{2}$$

$$RMSE = \sqrt{\left(\frac{\sum_{i=1}^{n} (y_i - x_i)^2}{n}\right)} \tag{3}$$

We also analyze the performance of our AGNI substrate in terms of area (per BLgroup), energy-delay product (EDP) (per conversion), and $area \times latency$ (per conversion). We compare the results with two stochastic to binary conversion designs from prior work: (1) the parallel pop counter circuit from [18] (referred to as Parallel PC) which is employed by the in-DRAM computing accelerator SCOPE [9], (2) the bit-serial pop counter circuit from [18] (referred to as Serial PC) which was employed by the in-DRAM computing accelerator ATRIA [17]. The results are given in Fig. 7 (discussed in the next subsection). **System-level Evaluation:** We also leverage our in-house system-level simulator to evaluate the latency (Fig. 8(a)) and EDP (Fig. 8(b)) for the inference of four CNN benchmarks (i.e., Shufflenet_V2, MobileNet_V2, DenseNet121, Inception_V3) [27] for the Imagenet dataset. Parallel PC and Serial PC were used to simulate the inference on SCOPE [9] and ATRIA [17] respectively. We only evaluate the StoB phases of these CNNs.

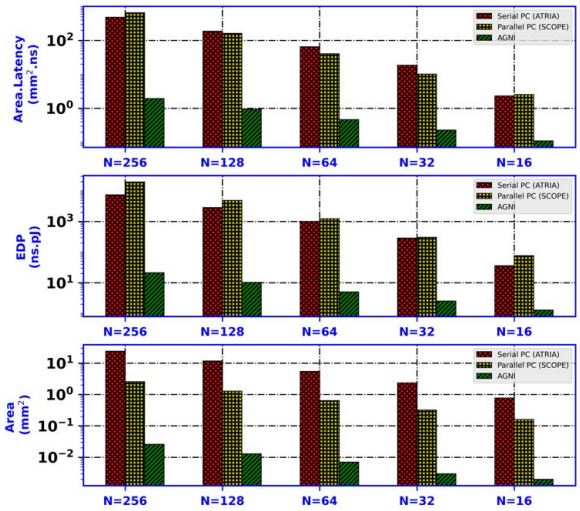

Fig. 7: $Area \times latency$ (top), energy-delay product (EDP) (middle), and area consumption (bottom) results of prior works (red and yellow columns) and our AGNI substrate (green columns).

C. Results and Discussion

The results from Table III show that AGNI achieves MAE=0.28 for *N*=16 and MAE=0.2 for *N*=256. Similarly, AGNI achieves MAPE=3.58% for *N*=16 and MAPE=0.8% for *N*=256. For a given *N*, a total of 2^N different stochastic number values can be represented. Therefore, for a larger *N*, the value n in Eqs. 1 and 2 increases exponentially, which in turn decreases the error magnitudes despite the fact that the decreased tolerance margin at a larger N increases the magnitude of the numerators in Eqs. 1 and 2.

From Fig. 7, AGNI achieves 390× less area, 21× less $area \times latency$, and 28× less EDP compared to Parallel PC for *N*=16. For higher values of *N*, AGNI showed significantly greater savings in area, $area \times latency$, and EDP. For example, for *N*=256, AGNI has 923× less area, 247× less $area \times latency$, and 350× less EDP compared to Parallel PC. Parallel PC consumes substantially higher area, $area \times latency$, and energy because it needs to employ full adder circuits [18], which increases its area and energy consumption. Note that Parallel PC has a slight edge in the latency over AGNI, but this drawback of AGNI can be tolerated for its excellent savings in area, $area \times latency$, and EDP.

Similarly, from Fig. 7, AGNI achieves 8× less area, 23× less $area \times latency$, and 59× less EDP compared to Serial PC for *N*=16. For higher values of *N*, AGNI showed significantly greater savings in area, $area \times latency$, and EDP.

For example, for N=256, AGNI has 96× less area, 333× less $area \times latency$, and 930× less EDP compared to Serial PC. Serial PC performs bit-by-bit counting at a clock rate, which significantly increases its latency and energy consumption compared to AGNI. Moreover, any implementation of a counter logic in-DRAM cannot be optimized for performance or area, due to the constraints of DRAM processes [19], which increases the area overhead of Serial PC counters compared to the peripherals of our AGNI substrate. Therefore, overall, we observe AGNI to significantly gain in area, $area \times latency$, and EDP, compared to Serial PC.

System-level Results: Fig. 8 shows the normalized inference latency and EDP results for our considered CNNs. From the figure, AGNI achieves 3.9× less latency than Serial PC on Gmean. Further, AGNI achieves 397× and 1048× better EDP than Parallel PC and Serial PC, respectively, on average across all considered CNNs. The better EDP results for AGNI confirms its advantages over Parallel PC and Serial PC.

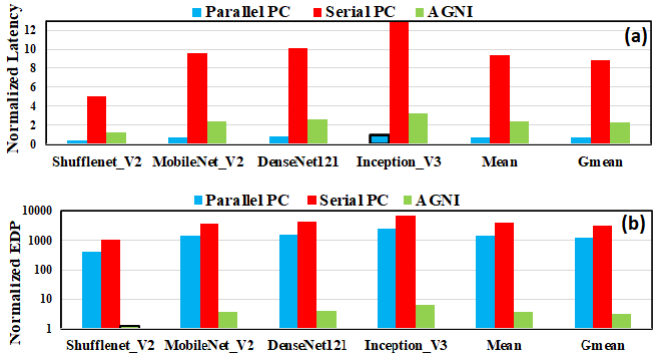

Fig. 8: System-level results for four CNNs. (a) inference latency normalized to the column for Parallel PC Inception_V3, (b) inference EDP normalized to the column for AGNI ShuffleNet_V2. Parallel PC = SCOPE [9], Serial PC = ATRIA [17].

VI. Conclusion and Future Work

In this paper, we presented a novel DRAM-based substrate called AGNI for in-situ StoB number conversion for Deep learning applications. We discussed the structure and operation of our AGNI substrate in this paper, using the results of our conducted SPICE simulations. We also presented detailed performance analysis results and overheads for our AGNI substrate. Our evaluations show that AGNI can achieve savings of at least 8× in area, at least 28× energy-delay product (EDP), and at least 21× in $area \times latency$, compared to two in-DRAM stochastic-to-binary conversion circuits from prior works. Future Work: There is a room for further reducing the latency of AGNI substrate by tightly packing various timing signals of AGNI substrate in a narrower window of time. The capacitance value of the analog capacitor and the physical implementation of the analog LANE also provide avenues for future exploration, to maximize the analog voltage range and noise margin.

Acknowledgment

We thank the anonymous reviewers for their valuable feedback. This research is supported by a grant from NSF (CNS-2139167).

References

[1] Chen *et al.*, "Dadiannao: A machine-learning supercomputer," in *IEEE MICRO*, 2014, pp. 609–622.
[2] Shafiee *et al.*, "Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," *in IEEE ISCA*, pp. 14–26, 2016.
[3] Han *et al.*, "Eie: Efficient inference engine on compressed deep neural network," *in IEEE ISCA*, vol. 44, no. 3, pp. 243–254, 2016.
[4] v. Krizhe *et al.*, "Imagenet classification with deep convolutional networks," vol. 1097, 2010.
[5] C. Szegedy *et al.*, "Going deeper with convolutions," *in CoRR*, vol. abs/1409.4842, 2014. [Online]. Available: http://arxiv.org/abs/1409.4842
[6] Chi *et al.*, "Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory," *in IEEE ISCA*, vol. 44, no. 3, pp. 27–39, 2016.
[7] X. Sun *et al.*, "Xnor-rram: A scalable and parallel resistive synaptic architecture for binary neural networks," in *IEEE DATE*, 2018, pp. 1423–1428.
[8] S. Li *et al.*, "Drisa: A dram-based reconfigurable in-situ accelerator," in *IEEE MICRO*, 2017, pp. 288–301.
[9] S. Li, "Scope: A stochastic computing engine for dram-based in-situ accelerator," in *IEEE MICRO*, 2018, pp. 696–709.
[10] Q. Deng *et al.*, "Dracc: A dram based accelerator for accurate cnn inference," in *IEEE DAC*, 2018, pp. 1–6.
[11] Q. Den *et al.*, "Lacc: Exploiting lookup table-based fast and accurate vector multiplication in dram-based cnn accelerator," in *IEEE DAC*, 2019, pp. 1–6.
[12] A. Biswas *et al.*, "Conv-sram: An energy-efficient sram with in-memory dot-product computation for low-power convolutional neural networks," *in IEEE JSSC*, vol. 54, no. 1, pp. 217–230, 2018.
[13] J. Wang *et al.*, "14.2 a compute sram with bit-serial integer/floating-point operations for programmable in-memory vector acceleration," in *IEEE ISSCC*, 2019, pp. 224–226.
[14] S. Yin, Z. Jiang, J.-S. Seo, and M. Seok, "Xnor-sram: In-memory computing sram macro for binary/ternary deep neural networks," *in IEEE JSSC*, vol. 55, no. 6, pp. 1733–1743, 2020.
[15] S. Lee *et al.*, "Hardware architecture and software stack for pim based on commercial dram technology: Industrial product," in *IEEE ISCA*, 2021, pp. 43–56.
[16] L. Ke *et al.*, "Near-memory processing in action: Accelerating personalized recommendation with axdimm," *IEEE MICRO*, vol. 42, no. 1, pp. 116–127, 2021.
[17] S. M. Shivanandamurthy *et al.*, "Atria: A bit-parallel stochastic arithmetic based accelerator for in-dram cnn processing," in *IEEE ISVLSI*, 2021, pp. 200–205.
[18] K. Kim *et al.*, "Approximate de-randomizer for stochastic circuits," in *IEEE ISOCC*, 2015, pp. 123–124.
[19] M. Lenjani *et al.*, "Fulcrum: a simplified control and access mechanism toward flexible and practical in-situ accelerators," in *IEEE HPCA*, 2020, pp. 556–569.
[20] D. Wu *et al.*, "Ugemm: Unary computing architecture for gemm applications," in *IEEE ISCA*, 2020, pp. 377–390.
[21] D. Lee *et al.*, "Tiered-latency dram: A low latency and low cost dram architecture," in *IEEE HPCA*, 2013, pp. 615–626.
[22] L. Orosa *et al.*, "Codic: A low-cost substrate for enabling custom in-dram functionalities and optimizations," in *IEEE ISCA*, 2021, pp. 484–497.
[23] Ravipati *et al.*, "Fn-cacti: Advanced cacti for finfet and nc-finfet technologies," *in IEEE VLSI*, vol. 30, no. 3, pp. 339–352, 2021.
[24] Chang *et al.*, "Understanding reduced-voltage operation in modern dram devices: Experimental characterization, analysis, and mechanisms," *in IEEE ACM*, vol. 1, no. 1, pp. 1–42, 2017.
[25] C.-H. Huang *et al.*, "Improving the latency-area tradeoffs for dram design with coarse-grained monolithic 3d (m3d) integration," in *IEEE ICCD*, 2020, pp. 417–420.
[26] L. Jiang *et al.*, "A low power and reliable charge pump design for phase change memories," *in IEEE ISCA*, pp. 397–408, 2014.
[27] F. Chollet *et al.*, "Keras," 2015. [Online]. Available: https://keras.io/api/applications/