Dynamic Computing in Memory (DCIM) in Resistive Crossbar Arrays

Seyedhamidreza Motaman and Swaroop Ghosh
Computer Science and Engineering, Pennsylvania State University
sxm844@psu.edu, szg212@psu.edu

Abstract*— With Von-Neumann computing struggling to match the energy-efficiency of biological systems, there is pressing need to explore alternative computing models. Recent experimental studies have revealed that Resistive Random Access Memory (RRAM) is promising alternative for DRAM. Resistive crossbar arrays possess many promising features that can not only enable high-density and low-power storage but also non Von-Neumann compute models. Most recent works focus on dot product operation with RRAM crossbar arrays, and therefore are not flexible to implement various logical functions. We propose a low-power dynamic computing in memory system which can implement various functions in Sum of Product (SOP) form in RRAM crossbar array architecture. We evaluate the proposed technique by performing simulation over wide range of MCNC benchmarks. Simulation results show 1.42X and 20X latency improvement as well as 2.6X and 12.6X power saving compared to static [9] and MAGIC [10] computing in memory methods.

Keywords—Resistive RAM, Sense Margin, computing in memory, Process Variation, Crossbar Array.

I. Introduction

Von-Neumann computing separates memory and processing element resulting in performance and energy bottlenecks due to frequent data transfers. High density crossbar array which employs two terminal Resistive RAM (RRAM) the crosspoint of vertical and horizontal metal wires are proposed [4]. However, these architectures suffer from sneak-path problem which results in poor sense margin, higher power consumption, and limited array size. Crossbar array with a selector diode connected in series to RRAM device has been proposed [1-3] to solve the sneak path issue. Various computing in memory schemes have been proposed to implement dot products in RRAM crossbar array. Digital to analog converter (DAC) and analog to digital converter (ADC) are required as peripheral circuitry to implement dot product in RRAM crossbar array. These architectures are able to implement matrix multiplication [7] and various computing paradigms such as neuromorphic computing [5-6] and approximate computing [8]. Even though these techniques improve performance and power efficiency they face challenges such as limited application domain and need of power intensive analog circuits such as ADC and DAC.

A computing in memory paradigm is proposed [9] to implement random functions in RRAM crossbar array. This technique offers full programmability across storage and computation. Even though it provides the flexibility of partitioning the hardware resources between computation and storage to achieve optimal performance, the implementation details of arbitrary functions are not discussed. This technique also suffers from poor sense margin (that can limit the array size) as well as increased power consumption, making it impractical for computing in memory applications. Memristor Aided LoGIC (MAGIC) has been proposed [10] where memristors act as an input with previously stored data, and an additional memristor serves as an output to implement logic gates. In this method, the logical operation is associated with write operation leading to higher power and latency overhead. Since the inputs are programmed into memristors the gate must be reprogrammed for new input data incurring substantial power overhead.

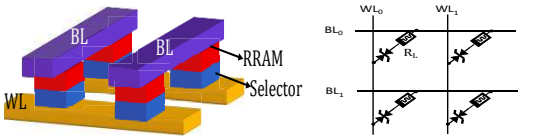


Fig. 1 Crossbar array with metal oxide RRAM and selector diode at each crosspoint; and, (b) schematic of crossbar array with selector diode.

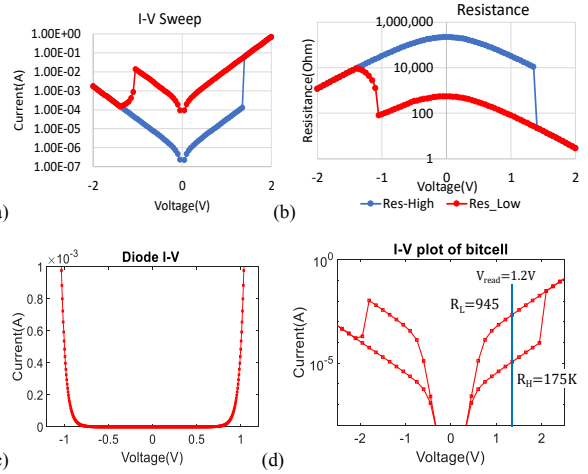


Fig. 2 (a) I-V curve RRAM model used in this study; (b) I-R characteristic of the RRAM model; (c) I-V curve of selector diode used in this study; and, (d) the I-V characteristic of bitcell composed of RRAM and selector diode.

In this paper, we propose a Dynamic Computing in Memory (DCIM) paradigm using RRAM crossbar array which benefits from nonlinear characteristic of selector diode to improve sense margin in order to implement higher fan-in gates. In addition, this technique reduces the power consumption associated with logical operation significantly by eliminating the static current compared to [9]. It also eliminates the need to write into the bitcell to perform logical operations compared to [10].

In summary we make following contributions in this paper:

- We study computing in memory systems proposed in [9-

2576-6996/18/$31.00 ©2018 IEEE
DOI 10.1109/ICCD.2018.00036

10] thoroughly and explain their bottlenecks.

- We develop a dynamic computing in memory technique to overcome sense margin limitation to implement higher fan-in AND/OR gates using RRAM crossbar array while reducing power consumption.
- We perform process, voltage and temperature variation analysis to determine optimum reference voltage to maximize read yield.
- We present comparative analysis of proposed technique with respect to other techniques for MCNC benchmarks in terms of power and latency.

The paper is organized as follows: In Section II, we describe the basics of crossbar array architecture as well as the state-of-art in-memory computing architectures using RRAM crossbar array. The proposed dynamic computing in memory technique is introduced in Section III. In Section IV, we investigate the effect of process and temperature variation on robustness of the proposed DCIM. In Section V, we discuss carry select adder implementation using proposed DCIM. In Section VI, DCIM is evaluated and compared to state-of-art CIM methods in terms of power and latency. The conclusions are drawn in Section VII.

II. Background

In this section we explain the basics of crossbar array architecture and read and write operations. We also discuss the state-of-art computing in memory systems using RRAM crossbar and describe its challenges.

A. Basics of RRAM Crossbar Array

A crossbar memory array consists of wordlines (WL) and bitlines (BL) where memory cell resides at their cross point as shown in Fig. 1. In this paper, we use a bipolar RRAM model [11] in which RESET/SET is performed at different voltage polarities. The I-R and I-V characteristic of the RRAM is shown in Fig. 2(a-b). The memory cell switches from High Resistance State (HRS) to Low Resistance State (LRS) if a positive voltage greater than threshold voltage is applied across the bitcell. Similarly, the bitcell switches from low to high resistance state if negative voltage is applied. Crossbar memory architecture achieves minimal cell size however, the sneak leakage current can reduce sense margin significantly. In order to increase sense margin and eliminate sneak leakage, we employ a memory bitcell which is composed of a RRAM device connected to a symmetric selector diode in series (Fig. 1(a-b)). The I-V characteristic of the selector diode is modeled by the following function as discussed in [1]:

$$I_{SEL} = \gamma.\sinh(\alpha.V)$$

where γ is a conductance parameter, and α represents the nonlinearity of selector diode. This model fits reasonably with the experimental I-V characteristic for selector devices based on MIM diode and punch through diode [12-13]. The design parameters of RRAM and selector diode are reported in Table I. The I-V curve of selector diode is illustrated in Fig. 2(c). Fig. 2(d) depicts the I-V curve of the bitcell composed of selector diode and RRAM device. It can be observed that the difference between low and high resistance increases by adding a selector diode which in turn improves the sense margin.

Table-1 List of design parameters

Parameters	Values
RRAM high resistance state (R_H) at 1.2V	18KΩ
RRAM low resistance state (R_L) at 1.2V	440 Ω
RRAM read Latency	0.5ns
RRAM write Latency	22ns
Nonlinear factor of selector (α)[1]	18.4
On-state current of selector (I_{ON})[1]	100uA
Selector Conductance Factor (γ)[1]	$2*10^{-12}$
bitcell high resistance state (R_H) at 1.2V	175KΩ
RRAM low resistance state (R_L) at 1.2V	945Ω
Bitcell write latency at 2.5V	25nS
Bitline Capacitance	30fF

Read Operation: For reading the bitcell, the commonly used ground/ground (GND-GND) scheme is employed. To access the bitcells in the array, the selected WL is connected to V_{READ} and the selected BLs are connected to sense-amplifier (SA) while all unselected BLs and WLs are biased at GND. Although this read scheme improves the sense margin, it also increases the power consumption. Other proposed read schemes include FL-FL (floating-floating) and GND-FL [1]. The current through selected bitcell which is generated by applied voltage to the selected WL, is converted to V_{out} by a sense resistance (R_{sense}). Read operation is performed by comparing output voltage (V_{out}) with a reference voltage (V_{REF}) using a SA as shown in Fig. 3. Maximum sense margin for both reading '0'

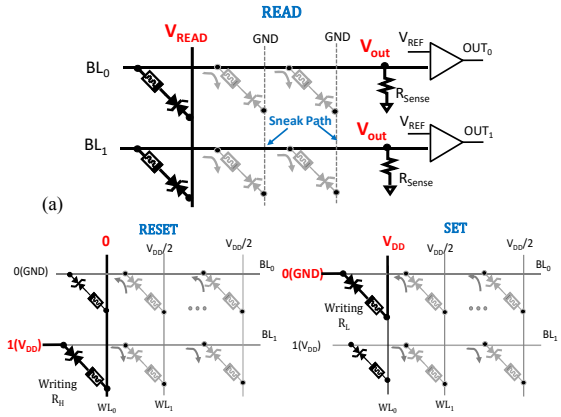


Fig. 3 RRAM crossbar array (a) GND-GND read scheme; and, (b) $V_{DD}/2$ write technique. Sneak paths are shown for read and write operations.

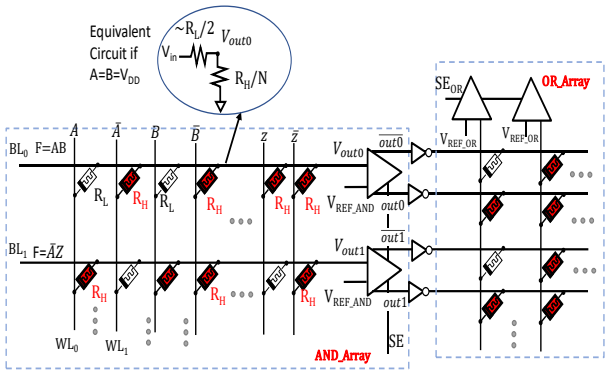


Fig. 4 Static computing in memory architecture in RRAM crossbar array.

(SM0) and reading '1' (SM1) is achieved by setting the $R_{Sense} = \sqrt{R_{OFF}/R_{ON}}$. The state of the unselected bitcells affects the sense margin (as shown in Fig. 3(a)). The worst-case sneak path also results in the worst-case SM which occurs when the unselected bitcells are in LRS since the sneak current is at maximum in this case.

Write Operation: We employ the $V_{DD}/2$ writing scheme where the selected WL is connected to V_{DD} and selected BL is connected to GND/V_{DD} (depending on input data) while other unselected BLs and WLs are biased at $V_{DD}/2$ (Fig. 3(b)). The write operation is performed in RESET and SET phases. Initially, the desired data is applied to the selected BLs. In the RESET phase the selected WL is connected to ground, hence the logical '0' is written to bitcell (programed to HRS). In the SET phase the selected WL is connected to V_{DD} and the logical '1' is written into bitcell (programed to LRS).

B. Static Computing in Memory (SCIM) Method

A configurable computing in memory system based on RRAM crossbar architecture which provides full programmability across computation and storage has been proposed in [9]. However, the detailed circuit implementation is not discussed. We extend the idea borrowed from this paper, to implement arbitrary functions in terms of sum of product within RRAM crossbar array for comparative analysis. In this method, the crossbar array is implemented using RRAM without selector diodes. A 2-input AND gate implementation using crossbar array is shown in Fig. 4. Each input and its complement are applied to a WL. In order to realize logical A.B, the cells connected to A and B are programmed to LRS and the cells connected to $\bar{A}$ and $\bar{B}$ are programmed to HRS while all other bitcells are programed to HRS (e.g., the bitcells connected to input Z and $\bar{Z}$ as illustrated in Fig. 4). The array inputs connected to WLs are applied to different gates implemented on different BLs. All the gates are evaluated concurrently by applying the data input to the array.

AND operation is performed by applying input vector and sensing the BL voltage. For A=B=1, the voltage appearing on the BL_0 is approximately V_{DD} (see the equivalent circuit in the inset of Fig. 4). For A=1 and B=0 (or A=0 and B=1), the BL_0 voltage is approximately $V_{DD}/2$. Finally, the voltage generated by applying the input vector is compared against a reference voltage (V_{AND_REF}) using a decoupled SA to determine the output of the AND operation.

As fan-in of the AND gate increases, the difference between voltage representing logical '1' and '0' reduces. The worst-case occurs when only one input is '0' and all remaining inputs are '1'. The difference between bitline voltage when all AND gate inputs are '1' (V_{AND1}) and V_{REF_AND} is defined as sense '1' margin (SM1). Sense '0' margin (SM0) for the AND operation is defined as the difference between bitline voltage when only one input is '0' (V_{AND0}) and V_{AND_REF}. Poor sense margin can result in wrong interpretation of the logical AND output. The impact of array size (the number of WLs) on V_{AND1} and V_{AND0} is shown in Fig. 5(a). This plot represents the V_{AND0} and V_{AND1} in an array of 2N WLs where all WLs are utilized to implement N-input AND gate. It can be observed that V_{AND1} remains constant with increasing AND gate fan-in. However, V_{AND0} rises with increased number of inputs which in turn degrades the SM. Note that, it is not possible to implement AND gate with more than 8 inputs, since SM reduces below the sense amplifier offset voltage which can result in wrong output.

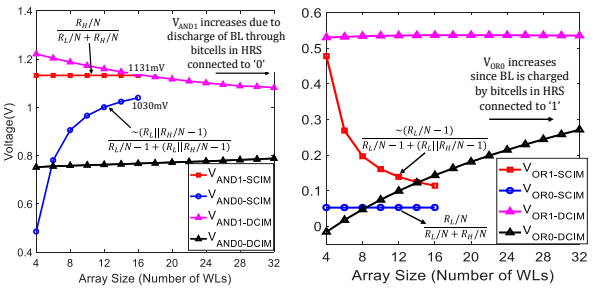


Fig. 5 (a) V_{AND1} and V_{AND0} versus AND array size; and, (b) V_{OR1} and V_{OR0} versus OR array size in an array of 2N WLs where all WLs are utilized to implement N-input gate.

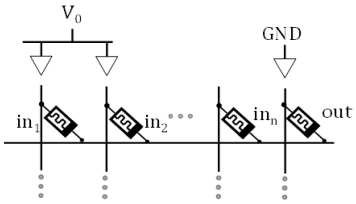
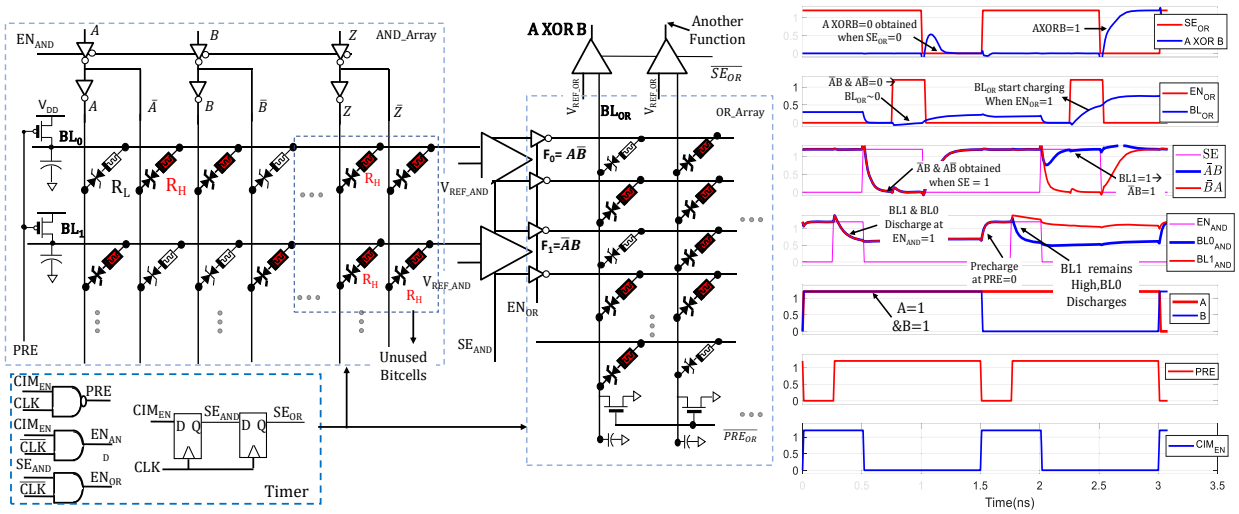


Fig. 6 MAGIC NOR gate implementation.

Any logical function can be implemented in Sum Of Product (SOP) form. Therefore, along with implementing AND function in RRAM crossbar array, we need to implement OR function as well. The OR gate implementation is similar to AND gate, except that the bitline voltage is compared against a different reference voltage (V_{REF_OR}). In order to implement the A+B (A OR B), RRAMs connected to A and B are programed to LRS, RRAMs connected to $\bar{A}$ and $\bar{B}$ are programed to HRS, and RRAMs connected to other unused WLs are programed to HRS. By applying A=B=0, the BL is pulled down to '0'. If one of the inputs is '1', a voltage near $V_{DD}/2$ appears on the bitline. The worst-case SM1 for OR array occurs when only one input value is '1' and remaining input values are '0'. The BL voltage in this case is defined as V_{OR1}. Similarly, V_{OR0} is defined as BL voltage when all inputs are '0'. As shown in Fig. 5(b), V_{OR1} reduces as the array size increases, which limits the SM.

C. Memristor Aided LoGIC (MAGIC) [10]

In this CIM architecture, memristors act as an input with previously stored data, and an additional memristor serves as an output to implement logic gates. This technique consists of two sequential stages. As shown in Fig. 6, a 2-input NOR gate composed of two RRAMs (in_1 and in_2) is connected to an output RRAM (*out*). In the initial stage, the output RRAM is programed to low resistance state and the input values are written to memristors in_1 and in_2. In the second stage, voltage V_0 is applied to memristors in_1 and in_2, and the *out* memristor is connected to GND to evaluate the NOR operation. The applied voltage results in a current that flows through RRAMs in_1 and in_2 and appears at RRAM *out*. If both input memristors are logical '0' (high resistance), the voltage appearing across the output RRAM is less than the switching threshold of the output RRAM thus it does not change and remains at logical '1'. For all other input combinations, the voltage across output



(a) (b)

Fig. 7(a) XOR implementation using proposed DCIM architecture in RRAM crossbar array; and, (b) timing diagram of logical XOR operation.

RRAM is greater than the threshold voltage. Hence, the output memristor switches to high resistance state (logical '0'). Finally, the state of output resistance is sensed using sense amplifier to determine the result of logical NOR operation. Since logical operation is associated with write operation in this method, the latency and power overhead are substantial. The proposed dynamic CIM eliminates the need of a write operation to improve latency and power overhead.

III. Proposed Dynamic Computing in Memory

In this section, we describe the operation of DCIM and study the impact of fan-in on sense margin and power. 65nm predictive technology [14] is used to perform simulation.

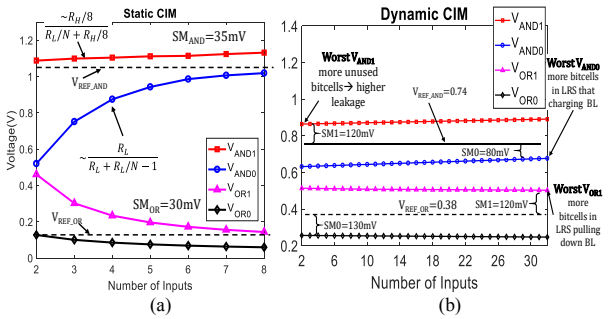
A. Basic Operation

DCIM aims to overcome sense margin limitation for higher fan-in AND/OR gates using RRAM crossbars. DCIM decreases power consumption due to two reasons: 1) sneak path leakage reduces significantly by employing a selector diode; 2) dynamic-sensing eliminates the static power consumption for performing logical operations. In this technique, each memory cell is composed of a RRAM device connected in series to a selector diode. Computing in memory is accomplished by implementing the functions in SOP form. Thus, both AND and OR operations are required to implement the logical functions. We dedicate separate arrays to perform each function and call them AND-array and OR-array.

In the proposed architecture, the wordlines serve as the inputs and the bitlines are the output of AND functions. Initially both AND and OR arrays are programmed to implement the desired function. The programing is similar to static technique. For instance, in order to implement $A\bar{B}$, the bitcells connected to A and $\bar{B}$ are programed to LRS while the bitcells connected to $\bar{A}$ and B are programed to HRS (Fig. 7(a)). All bitcells connected to other array inputs/WLs which are not part of AND gate inputs are programed to HRS (e.g., the bitcells connected to input Z and $\bar{Z}$). To perform AND operation, the BL is initially precharged to V_{DD}. Once the inputs are applied, the BL either remains precharged or discharges based on the input vector. In the previous example, if V_{DD} (logical '1') is applied to inputs A and $\bar{B}$, the BL remains precharged since these inputs are connected to bitcell in LRS. However, the leakage of HRS bitcells connected to GND discharges the BL negligibly. Any other input combination discharges the BL significantly since GND is connected to a bitcell in LRS. Finally, the BL voltage is compared against the V_{REF_AND} to determine the result of AND operation. The result of the AND function and its complement are provided as input to the OR array to obtain SOP output. Programing of OR array is similar to AND array. However, in OR array BLs are predischarged to '0'. The predischarge of OR array BLs is performed during the AND array evaluation phase, therefore the latency of predischarge phase is hidden. Finally, the voltage generated on the OR array BL is compared against V_{REF_OR} to achieve the result of OR operation.

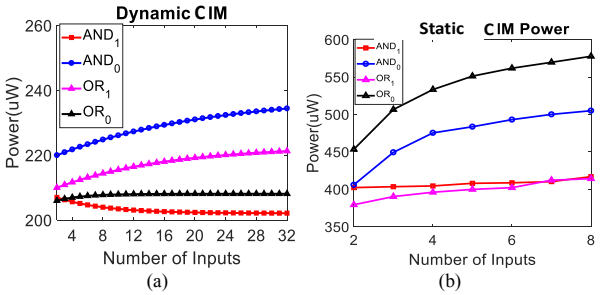
The effect of array size (number of WLs) on the SM is investigated to determine the best array size (Fig. 5). Since two WLs and two bitcells are required for implementing each input of AND gate, the number of WLs is twice the number of AND gate inputs. As depicted in Fig. 5(a-b) as array size increases SM for AND/OR operations degrades. It can be observed that proposed DCIM improves SM significantly compared to SCIM, thus larger array size (higher fan-in gates) can be realized.

Fig. 7 shows the implementation of XOR function in DCIM. The BL_0 and BL_1 are programmed to implement $A\bar{B}$ and $\bar{A}B$ functions respectively. Note that the bitcells connected to WLs which are not contributing in XOR implementation (called the unused bitcells) are programmed to HRS. Initially, the PRE signal is activated to precharge AND array BLs to V_{DD}. Next, inputs (A and B) are applied by asserting EN_{AND}. As shown in Fig. 7(b), when A, B=1 both BL_0 and BL_1 fall to 0.65V. Since this voltage is less than V_{REF_AND}=0.74V, outputs of sense amplifiers which determine the results of $A\bar{B}$ and $\bar{A}B$ functions are pulled down to '0' at the edge of SE_{AND}. Since inputs of OR array (F_0=$A\bar{B}$ and F_1=$\bar{A}B$) are '0', the OR array BL (BL_{OR}) remains discharged with voltage of approximately '0' (i.e. A



(a) (b)

Fig. 8 (a) $V_{AND,1}$,$V_{AND,0}$, V_{OR1} and V_{OR0} versus gate fan-in for, (a) conventional CIM in array of 16 WLs, (b) DCIM in array of 64 WLs.



(a) (b)

Fig. 9 Power consumption versus number of inputs; (a) Dynamic CIM and, (b) static CIM.

XOR B=0). If A=0 and B=1, BL_0 discharges to 0.65V while BL_1 remains precharged which results in $F_0 = A\bar{B} = 0$ and $F_1 = \bar{A}B = 1$. Since F_1 is '1' and is connected to a bitcell in LRS, it charges the BL_{OR} to 0.52V while EN_{OR} is asserted. Finally, the voltage of BL_{OR} is compared against V_{REF_OR}=0.38V at the edge of SE_{OR} which produces '1' at the output of SA. Note that OR array sense enable ($\overline{SE_{OR}}$) is an active low signal. Since the voltage generated on bitline of OR array is less than 0.52V, a PMOS based SA with active low sense enable is employed (Section IV. B).

The PRE, EN and SE signals are generated in the timer (located at the middle of subarray). The duty cycle of EN depends on BL capacitance and the bitcell resistance. In addition, SM depends on the EN pulse width. The EN pulse with is chosen in such a way that V_{OR1} rises to 90% of its steady state voltage. By applying EN, V_{OR0} also rises due to leakage of unused bitcells. Therefore, the EN pulse with must be chosen in such a way to maximize V_{OR1} and minimize the increase of V_{OR0}. The same argument holds true for V_{AND1} and V_{AND0}. Moreover, increasing the EN pulse width results in higher power consumption since both V_{OR0} and V_{OR1} will increase. Thus, there is a tradeoff between power and sense margin. We have swept the EN width from to 0.1nS to 0.5nS in order to optimize both SM and power. The EN pulse width of 0.25ns achieves sufficient sense margin while preserving power consumption. The PRE pulse width depends on the BL capacitance and the width of precharge transistor. Based on simulation result, a PRE pulse width of 0.25nS is sufficient to precharge/predischarge the BL before logical AND/OR operation. The CIM operation starts at the edge CIM_{EM} which is provided as input to the timer (inputs are provided to AND array simultaneously). The timer receives CIM_{EN} and produces PRE, EN and SE signals (clock frequency is 2GHz). The power and area overheard of timer is negligible.

B. *Impact of Gate Fan-in on Sense Margin*

In the previous section, we investigated the effect of array size on the SM. The purpose of this study is to determine the array size that achieves maximum sense margin while preserving the area efficiency. In other words, it represents the sense margin of AND/OR operation in an array of 2N WLs where all WLs are utilized to implement N inputs AND gate. In this section we study the sense margin with respect to AND gate fan-in. Let us assume that a 4-input AND gate is implemented in an array of 64 WLs. Since 8 WLs are required to implement 4-input AND gate, 8 bitcells are programmed to implement the AND gate while bitcells connected to the rest of WLs are programmed to HRS. The loading effect of unused array inputs connected bitcells in HRS reduces sense margin. Unused array input and its complements are connected to two bitcells in HRS. In case of static CIM, applying inputs to the unused WLs degrade the sense margin. This can be understood by comparing Fig. 5 with Fig. 8(a). For instance, 2-input OR gate SM is significantly higher when the array consists of 4 WLs (see Fig. 5(b)) versus 16 WLs (see Fig. 8(a)).

The impact of unused WLs on sense margin is more severe in DCIM. Suppose input Z value (as depicted in Fig. 7) which does not belong to 2-input AND gate implemented on BL_0 is '0'. Since BL_0 is precharged to V_{DD} initially, the voltage across selector diode is VDD, and it is ON initially. As BL voltage discharges through bitcell connected to Z the voltage across selector diode reduces, and it becomes strongly ON to weakly ON. The selector diode is OFF/weakly OFF in the bitcell which is connected to $\bar{Z}$. Therefore, input Z=0 discharges the BL, while input $\bar{Z} = 1$ cannot compensate the effect of Z by charging the BL (since bitcell connected to $\bar{Z}$ is OFF). This result in lower V_{AND1}, leading to SM degradation. As gate fan-in decreases the number of unused bitcells increases. Thus, V_{AND1} reduction increases as fan-in decrease since the leakage through unused bitcells increases. As shown in Fig. 8(b), 2-input AND gate achieves worst-case V_{AND1} (higher number of unused bitcells result in higher leakage and lower V_{AND1}).

As mentioned earlier, V_{AND0} is the voltage appears on the BL when only one input is '0'. For 32-input AND gate, V_{AND0} is the BL voltage where 31 inputs connected to bitcells in LRS is pulling up the BL weakly (since selector diode is OFF) while only one input is pulling it down strongly. Thus, as the number of input increases (e.g., from 2 to 32), the number of bitcells in LRS which weakly pulls the BL up increases (e.g. 1 versus 31). Therefore, as depicted in Fig. 8(b), 32-inputs AND gate results in the worst-case V_{AND0} (higher V_{AND0}) while 2-inputs AND gate result in the best V_{AND0}. The same argument holds true for V_{OR1} and V_{OR0} . V_{OR1} and V_{OR0} in an array of 64 WLs is also

TABLE II: Parameters used for process variation study.

Device	Parameter	Mean	Std. Dev.
PMOS	V_{TH}	423mV	$A_{V_T}/\sqrt{WL}$ [1]
NMOS	V_{TH}	365mV	$A_{V_T}/\sqrt{WL}$ [1]
RRAM	Inial Gap	R_L =0.2nm R_h =1.7nm	7%
RRAM	Oxide Thickness	12nm	5%

[1] A_{V_T} *is* Pelgroom coefficient which is ~4.5mV/μm for 65nm technology

shown in Fig. 8(b). 32-input OR gate results in worst-case V_{OR1} since more bitcells in LRS pulls the BL down.

C. Impact of Gate Fan-in on Power

The power consumption of proposed DCIM for AND and OR operations are shown in Fig. 9(a). In case of AND operation we assume the BL is precharged to V_{DD} and the power consumption is summation of the power drawn from supply after applying inputs, the power consumed by the sense amplifier and the power required to precharge the BL back to V_{DD}. For the OR operation the power consumption is the power drawn from the supply to charge the bitline, and the power consumed by the sense amplifier. It can be noted that as the number of input increases, the power consumption of AND1 operation reduces. As shown in Fig. 8(b), V_{AND1} increases with the number of inputs. Hence, less power is consumed to precharge the bitline back to V_{DD}. AND0 operation results in higher power consumption since the bitline discharges to a lower voltage when the result of AND operation is '0'. Therefore, more power is consumed to precharge the BL back to V_{DD}. Fig. 9(b) depicts the power consumption of static CIM. It can be noted that static CIM power consumption is significantly higher (almost 3X on average) due to static current which flows through the bitcells during logical AND/OR evaluation.

IV. Process and Temperature Variation Analysis

A. Impact of Process and Temperature Variation on Sense Margin

The impact of process and temperature variation on V_{AND1} and V_{AND0} are investigated to determine the best V_{REF_AND} to achieve robustness. Process variation analysis is carried out using detailed Monte Carlo simulation in 65nm technology [14]. For RRAM we have assumed oxide thickness and initial filament gap variations. The variations in CMOS circuitry is lumped in threshold voltage fluctuation. The mean and standard deviation of these parameters are provided in Table II. As mentioned earlier, 2-input AND gate results in the worst-case V_{AND1}, and 32-input AND gate results in the worst-case V_{AND0}. Furthermore, higher temperature results in higher bitcell resistance, leading to higher V_{AND0} which in turn degrades the SM0. Whereas, lower temperature leads to lower bitcell resistance and lower V_{AND1} degrading SM1. In order to obtain the worst-case V_{AND0} under process and temperature variation, we run 1000 points Monte-Carlo simulation at 90°C. Similarly, 1000 points Monte-Carlo simulation is performed at -10°C to achieve the worst-case V_{AND1}. The simulation result is shown in Fig. 10 (a). Since standard deviation of V_{AND1} ($\sigma_{V_{AND_1}}$) is greater than V_{AND0}, a voltage slightly less than ($\mu_{V_{AND_0}} + \mu_{V_{AND_1}}$)/2 is chosen as V_{REF_AND} to maximize the AND operation read yield. We have performed the same analysis to obtain the V_{REF_OR}. The worst-case V_{OR1} occurs at higher temperature (90°C), since higher resistance increase the RC delay, thereby the BL is charged to a lower voltage reducing V_{OR1}. Similarly, the worst V_{OR0} occurs at lower temperature. Monte-Carlo simulation is carried out at different temperatures to determine the optimum V_{REF_OR}. The results are shown in Fig. 10(b). Since the $\sigma_{V_{OR0}}$ is greater than $\sigma_{V_{OR1}}$ we pick a

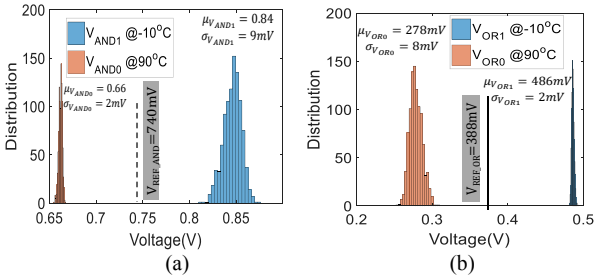


Fig. 10 (a) V_{AND1} and V_{AND0} distribution for 1000 Monte-Carlo points @ -10°C and 90°C; and, (b) V_{OR0} and V_{OR1} distribution.

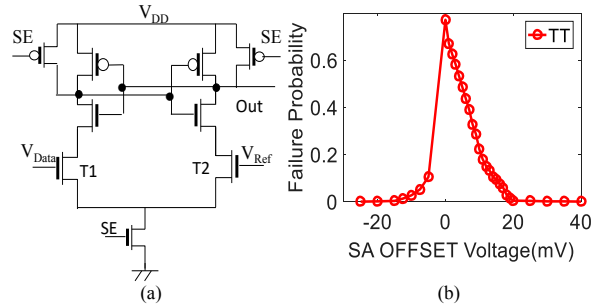


Fig. 11(a) Sense amplifier circuit; and, (b) SA offset voltage distribution for 1000 points Monte-Carlo simulations.

voltage greater than ($\mu_{V_{OR0}} + \mu_{V_{OR1}}$)/2 as V_{REF_OR} to maximize the OR operation read yield.

B. Sense Amplifier OFFSET Voltage Analysis

The sense-amplifier offset voltage (V_{SA_OFFSET}) depends on the sense time and transistor size since increasing the transistor size decreases the transistor threshold voltage variation. We design the sense amplifier to reduce the offset while meeting the area and delay requirements. We considered sense time of 0.5nS. In order to achieve V_{SA_OFFSET}, we fix reference voltage (V_{REF_AND}) at 740 mV and sweep V_{Data} (Fig. 11(a)). For each sweep 1000 points Monte-Carlo simulation is carried out and sense amplifier failure distribution is shown in Fig. 11(b). This distribution can be modeled by a Gaussian distribution with $\mu_{V_{SA\_OFFSET}} = 8mV$ and $\sigma_{V_{SA\_OFFSET}} = 16mV$.

C. Read Yield

The statistical distribution of sense margin and V_{SA_OFFSET} caused by process variation can be modeled by Gaussian distribution. Since read access pass occurs when SM > V_{SA_OFFSET}, read access pass yield for a bitcell with state 0 or 1 ($RAPY_0$ or $RAPY_1$) can be achieved by combining distribution of V_{SA_OS} and $SM_{0,1}$ [15]:

$$RAPY_{0,1} = \frac{\mu_{SM_{0,1}} - \mu_{V_{SA\_OFFSET}}}{\sqrt{\sigma^2_{SM_{0,1}} - \sigma^2_{V_{SA\_OFFSET}}}} \quad (1)$$

Where $\mu_{SM_{0,1}}(\mu_{V_{SA_{OFFSET}}})$ is mean sense margin and $\sigma_{SM_{0,1}}(\sigma_{V_{SA_{OFFSET}}})$ is the standard deviation of sense margin. RAPY for a bitcell is defined as the smaller of $RAPY_0$ and $RAPY_1$. To obtain RAPY we assume that V_{REF} is produced by a voltage regulator with negligible variation (5mV). Based on the Monte-Carlo simulation, the RAPY of AND and OR operations are found to be 4.2σ and 4.9σ respectively. The

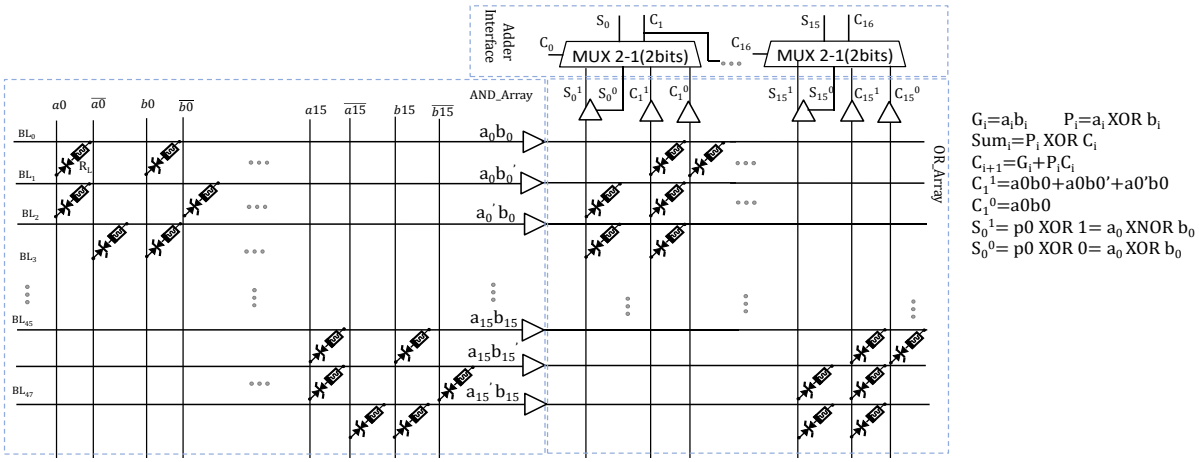


Fig. 12 Implementation of 16-bit carry select adder using DCIM scheme. For sake of brevity only low resistance connections are shown.

static CIM results in significantly lower yield. The RAPY of AND and OR operations are found to be 1.7σ and 1σ respectively.

V. Implementation of Carry Select Adder using DCIM

In order to perform addition, carry select adder is implemented. Fig. 12 demonstrate the implementation of 16-bit carry select adder using DCIM. For sake of brevity only low resistance connections are shown. In the carry select addition approach two sets of sum and outgoing carry are computed considering incoming carry is either '0' or '1'. Once the incoming carry is known, we only need to select the correct set of outputs (out of the two sets using multiplexer) without waiting for the carry to propagate further. In Fig. 12, S_0^0 and C_1^0 indicate the sum and carry output when incoming carry is '0'. Similarly, S_0^1 and C_1^1 indicate the sum and carry output when incoming carry is '1'. As demonstrated in Fig. 12, the carry selection takes place at the adder interface. Based on the C_0 value, $S_0(C_1)$ is selected from the previously computed S_0^0 and S_0^1 (C_1^0 and C_1^1). Next, C_1 is propagated to the input select of next multiplexer to determine the value of S_1 and C_2 and so forth. This technique is of great interest since it enables implementing adder in two-level format (in form of SOP) without need of carry propagation. However, it requires multiplexers to perform output selection, which can be done using CMOS MUX in the peripheral. Pass gates are used to implement the MUXs in order to minimize the CMOS area overhead. Larger adders can be implemented by propagating output carry (C_{16}) to the input carry of other arrays that implements another set of 16-bit adder. Table III depicts latency and power of 16-bit adder implemented in three CIM techniques. The SCIM latency and power are obtained from simulation. Since SCIM cannot accommodate more than 8 inputs, we employ two CIM arrays to implement 16-bit adder where the output carry of first CIM array is provided as input to input carry of the second array. Therefore, 16-bit addition latency is identical for both static and dynamic CIM. The MAGIC latency and power are estimated from Table VI in [10] by employing the RRAM model that we used in this paper. Even though DCIM requires more number of cells (since larger array result in more unused bitcells) to implement 16-bit adder, it achieves 12X power saving in 16-bit addition and achieves significantly lower latency compared to MAGIC.

Table III. Comparison of 16-bits adder implementation using different CIM schemes

16-bits Adder	Latency	# of RRAM	Power	# Logical Operations
DCIM (This paper)	2 cycles+carry selection delay =2nS	2*64*48	48mW	64 AND2 32 OR3 32 OR2
SCIM	2nS	64*48	64mW	Same as above
MAGIC	12N+1 (Cycles)=4246ns	177	579mW	193 NOR

VI. Evaluation and Comparison of Different Computing in Memory Techniques

In this section we compare the proposed DCIM with SCIM and MAGIC in terms of power and latency.

A. Power

In order to perform comparison, two-level benchmarks of MCNC benchmark suite [16] are used. A script is written in order to extract number of AND/OR gates and their fan-in for each SOP function. Unlike CMOS gates, where power is only consumed during '0' → '1' transition, the power is consumed during both '0' → '1' and '1'→'0' transitions in the CIM techniques. Initially, we assume the probability of each input being '1' as 0.5. In order to obtain power dissipation, the probability of logical AND/OR when output is '0'/'1' is calculated at each stage. Thus, the power consumption of each gate can be expressed as follows:

$$Pr_{AND1}(N) = 1/2^N \quad (2)$$

$$Pr_{OR0}(N) = Pr_0(in_1) * Pr_0(in_2) * \ldots\ldots * Pr_0(in_N) \quad (3)$$

$$P_{AND}(N) = Pr_{AND1}(N) * P_{AND1}(N) + (1-Pr_1(N)) * P_{AND0}(N) \quad (4)$$

$$P_{OR}(N) = Pr_{OR0}(N) * P_{OR0}(N) + Pr_1(N) * P_{OR1}(N) \quad (5)$$

Where $P_{OR0}(N)$ and $Pr_{OR0}(N)$ are the power and probability of N-input logical OR gate when the output is '0'. Fig. 13(a) shows the power comparison of DCIM with respect to other techniques. Dynamic CIM provides 12.6X and 2.6X power saving compared to static CIM and MAGIC respectively.

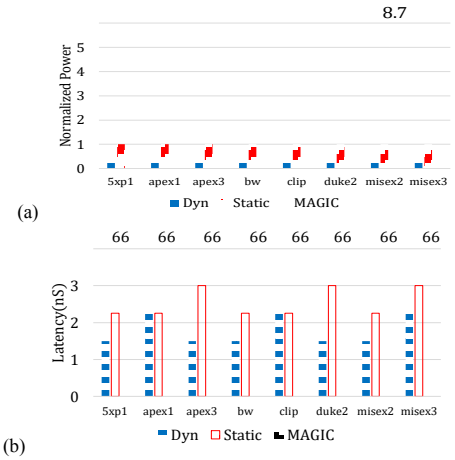


Fig. 13 (a) Power, and (b) latency comparison of various CIM schemes.

B. Latency

The latency of logical AND/OR operation for static and dynamic CIM is 0.75nS. Since DCIM support up to 32 input AND/OR gates, the gates with fan-in of more than 32 must be partitioned into lower fain-in gates which is associated with latency and power overhead. For example, a 64-input OR gate is implemented using eight 8-input OR gates. As a result, all outputs of 8-input OR gates must be ORed using another OR array. Hence, increasing the latency by another 0.75nS. The latency results for several benchmarks are shown in Fig. 13(b). DCIM achieves 1.42X improvement in latency compared to SCIM since it offers higher fan-in gate implementation. In the SCIM method, the gates with more than 8 inputs must be partitioned into lower fan-in gates. Since many functions in two-level (SOP) form are implemented using high fan-in gates, the SCIM latency is typically one or two sensing cycle longer than DCIM.

In order to obtain the MAGIC power and latency, we implemented each benchmark in two-level NOR-NOR format. In addition, fain-in and number of NOR gates to implement each function is obtained. In order to achieve consistent result, the RRAM model [11] is used where latency of writing '0'/'1' into RRAM is 22nS (Table-1). MAGIC NOR operation associated with two write operations is described in Section II. C. Since MAGIC does not suffer from limited sense margin, it can implement high fan-in NOR gates. We assume that the array is large enough to accommodate all high fan-in NOR gates required for implementing two-level benchmarks. Therefore, 22nS is needed to program inputs into RRAM array, 22ns to perform first-level NOR operation by writing into output RRAM, and 22nS to NOR the output of first-level NORs is required to achieve the SOP output. Hence, the total latency of MAGIC scheme is 66nS.

VII. CONCLUSIONS

In this paper we proposed dynamic computing in memory paradigm to overcome sense margin limitation associated with static CIM method in realizing higher fan-in AND/OR gates using RRAM crossbar array. In addition, this technique decreases power consumption significantly by eliminating the static current flow for performing logical operation compared to static CIM and, eliminates the need of writing into the bitcell to perform logical operations compared to MAGIC [10]. DCIM improves read yield of logical operations ~4X compared to SCIM. Simulation results show 1.42X and 20X latency improvement as well as 2.6X and 12.6X power saving compared to static [9] and MAGIC [10] computing in memory methods over a wide range of MCNC benchmarks.

Acknowledgements: This paper is based on work supported by Semiconductor Research Corp. (#2018-TS-2847), NSF CNS-1722557, CNS-1814710, CCF-1718474, DGE-1723687, DGE-1821766 and DARPA Young Faculty Award [#D15AP00089].

REFERENCES

[1] Zhou, Jiantao, et al. "Crossbar RRAM arrays: Selector device requirements during read operation." *IEEE Transactions on Electron Devices* 61.5 (2014): 1369-1376.

[2] Huang, Jiun-Jia, et al. "One selector-one resistor (1S1R) crossbar array for high-density flexible memory applications." *Electron Devices Meeting (IEDM), 2011 IEEE International*. IEEE, 2011.

[3] Deng, Yexin, et al. "RRAM crossbar array with cell selection device: A device and circuit interaction study." *IEEE Transactions on Electron Devices* 60.2 (2013): 719-726.

[4] Liang, Jiale, and H-S. Philip Wong. "Cross-point memory array without cell selectors—Device characteristics and data storage pattern dependencies." IEEE Transactions on Electron Devices 57.10 (2010): 2531-2538.

[5] G. W. Burr, et al. "Experimental demonstration and tolerancing of a large-scale neural network (165 000 synapses) using phase-change memory as the synaptic weight element." TED, 2015.

[6] S. Yu, et al. "A neuromorphic visual system using RRAM synaptic devices with Sub-pJ energy and tolerance to variability: Experimental characterization and large-scale modeling." IEDM, 2012.

[7] Ni, Leibin, et al. "An energy-efficient matrix multiplication accelerator by distributed in-memory computing on binary RRAM crossbar." Design Automation Conference (ASP-DAC), 2016 21st Asia and South Pacific. IEEE, 2016.

[8] B. Li, Y. Shan, et al. Memristor-based approximated computation. In ISLPED, pages 242{247, Sept 2013.

[9] Zha, Yue, and Jing Li. "Reconfigurable in-memory computing with resistive memory crossbar." *Proceedings of the 35th International Conference on Computer-Aided Design*. ACM, 2016.

[10] Talati, Nishil, et al. "Logic design within memristive memories using memristor-aided loGIC (MAGIC)." *IEEE Transactions on Nanotechnology* 15.4 (2016): 635-650.

[11] Jiang, Z., Wong, H. P. (2014). Stanford University Resistive-Switching Random Access Memory (RRAM) Verilog-A Model. nanoHUB. doi:10.4231/D37H1DN48

[12] Govoreanu, Bogdan, et al. "High-performance metal-insulator-metal tunnel diode selectors." *IEEE Electron Device Letters* 35.1 (2014): 63-65.

[13] Srinivasan, V. S. S., et al. "Punchthrough-diode-based bipolar RRAM selector by Si epitaxy." *IEEE Electron Device Letters* 33.10 (2012): 1396-1398.

[14] Predictive technology model, ASU, http://www.asu.edu/~ptm.

[15] Nho, Hyunwoo, et al. "Numerical estimation of yield in sub-100-nm SRAM design using Monte Carlo simulation." *TCAS II,* 2008.

[16] Yang, Saeyang. *Logic synthesis and optimization benchmarks user guide: version 3.0*. Microelectronics Center of North Carolina (MCNC), 1991.