CHAPTER 7

ENERGY-EFFICIENT LOGIC AND ARITHMETIC CIRCUITS

7.1 Introduction

The next-generation computing systems will be designed to meet the requirement of energy-efficient computing in big data applications. The recent rise in the use of data-intensive applications such as multimedia, artificial intelligence, and pattern/voice recognition has led to the emergence of huge streaming data [1]. The processing of huge amount of data have become one of the biggest challenges for the conventional von-neumann based computing system. The von-neumann architecture-based systems have separate processing and memory units. The instruction and data are stored in the memory while computation takes place in the processor. For executing any instruction, the processor fetches the data from memory and writes the results back to memory, as shown in Figure 7.1.

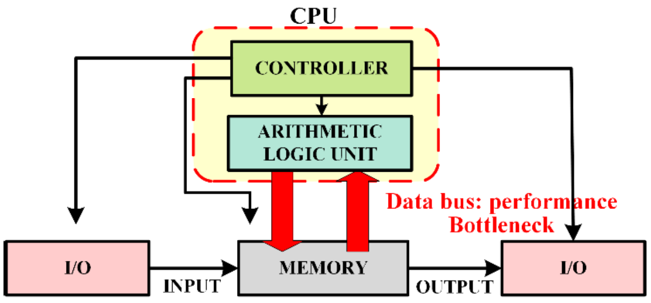


Figure 7.1: Conventional Von-Neumann Architecture

The processing of large data set in a conventional von-neumann system results in more communication time than computation time due to limited memory bandwidth [2]. At the same time, frequent data movement between the memory and core results in considerable power consumption [3]. These challenges are most commonly known as memory wall or von-neumann bottleneck [4]. Therefore, new computing paradigms are needed to handle the demands of rapidly growing big data applications. To overcome the von-neumann bottleneck, there has been significant effort in the past to move processing closer to the memory [5]–[8]. These attempts are broadly classified into two categories: Near-Memory Computing (NMC) and In-Memory Computing (IMC). In near-memory computing, the physical distance between the processor and memory is reduced by integrating both of these units on the same chip, as shown in Figure 7.2 (a). However, integrating a high-performance processor with high-density memory on a single die is challenging due to different design rules [9], [10]. Recently, researchers are working in the direction of in-memory computing, which utilizes memory block to perform the computations, as shown in Figure 7.2 (b).

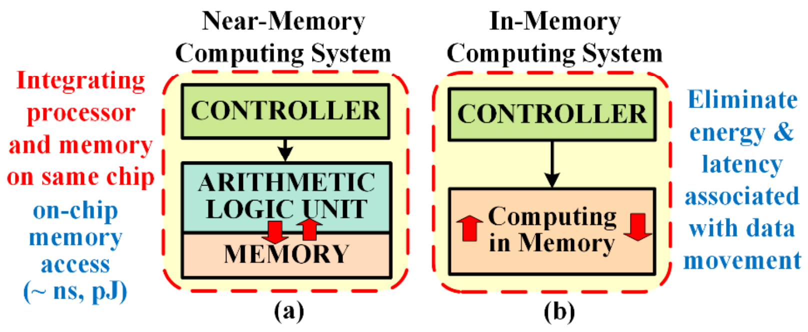

Near-Memory Computing System
In-Memory Computing System
Integrating processor and memory on same chip
on-chip memory access (~ ns, pJ)
CONTROLLER
ARITHMETIC LOGIC UNIT
MEMORY
(a)
CONTROLLER
Computing in Memory
(b)
Eliminate energy & latency associated with data movement

Figure 7.2: (a) NMC (b) IMC

The key benefit of performing computations within the memory is the elimination of data movement overhead, which results in large internal bandwidth and low-power consumption. The emerging non-volatile devices are considered as one of the most promising candidates to realize in-memory computation. In addition to energy-efficient computing, their non-volatile nature allows power-hungry blocks to be turned off during the long idle period, which results in the elimination of standby power consumption. This technique is known as normally-off computing, and it is highly beneficial for low-power applications that spend a significant amount of their time in idle state. Among the various emerging non-volatile devices, magnetic tunnel junction has attracted a lot of attention as an ideal candidate for future embedded memory. It offers high integration density, high scalability, unlimited endurance, and CMOS compatibility [14], [15]. While magnetic technology has shown promising results as an embedded non-volatile storage element, its unique properties can also be exploited for in-memory computations. The spin-transfer torque and spin-hall effect based magnetic tunnel junction devices (STT-MTJ & SHE-MTJ) are considered as most suitable for on-chip cache applications. They can achieve the integration density of DRAM and potentially match the performance of SRAM. Compared to other competing non-volatile devices, STT-MTJ and SHE-MTJ have high retention time, high endurance, and compatibility with the CMOS fabrication process [1], [2]. Therefore, in this work, we utilize STT-MTJ and SHE-MTJ to develop computational magnetic random-access memory (C-MRAM), which can implement boolean function within the memory itself. To further reduce the power consumption, we realize the concept of approximate computing in the proposed C-MRAM array based on SHE-MTJ device. In recent times, approximate computing has emerged as a potential solution to achieve low power computing [16]. It is based on the fact that many real-time applications such as multimedia, wireless sensors, data mining, and search engines can produce output of acceptable quality even though the computational accuracy is low [17], [18]. The ability to relax the requirement for computational accuracy is leveraged to improve the area, delay, and power metrics. To incorporate the aforementioned concept and demonstrate the application level benefits of the proposed C-MRAM array, we design an approximate full adder (Ax-ADD), which can be used in low power and low precision applications.

7.2 STT-MTJ based Computational Magnetic Random-Access Memory (C-MRAM)

Computational Magnetic Random-Access Memory (C-MRAM) is a concept in which memory array based on STT-MTJ device is used to perform basic computation. Figure 7.3 (a) illustrates the general structure of C-MRAM [15]. It consists of a 2T-1MTJ bitcell structure. The two access transistors allow memory to work in two modes: memory mode and logic mode.

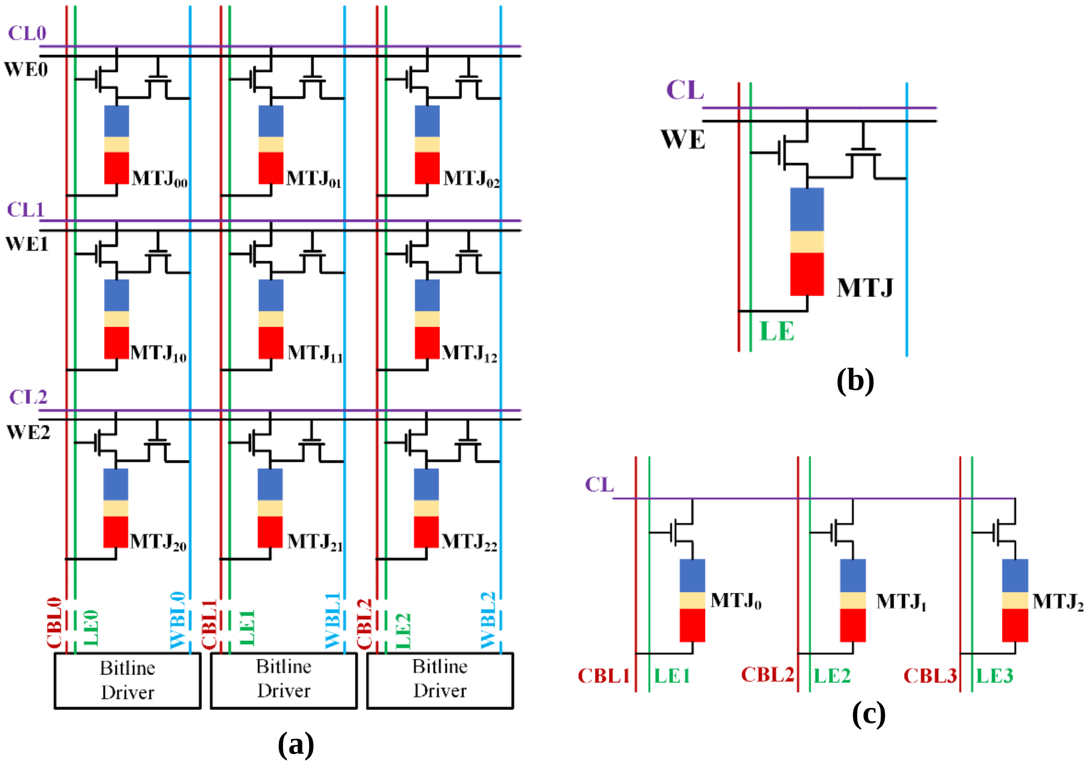


Figure 7.3: (a) STT-MTJ based C-MRAM array (b) 2T-1MTJ bitcell (c) C-MRAM row

During the memory mode, the proposed C-MRAM array work as a typical memory with the additional advantage of non-volatility. Whereas during the logic mode, the proposed array performs the logic operation on input operands within the array itself.

Memory Mode (MM): In memory mode, Word Enable (WE) signal is asserted high whereas Logic Enable (LE) signal is asserted low, which connects MTJ device to the bitlines, as shown in Figure 6.3 (b). The read and write operations are performed through the two bitlines: Write Bit Line (WBL) and Common Bit Line (CBL).

Logic Mode (LM): In logic mode, Logic Enable (LE) is asserted high to connect the MTJ to the Connector Line (CL) in each row, as shown in Figure 6.3 (c). Several MTJ are connected to CL to realize multiple input single output logic function. The different logic functions can be performed by choosing the appropriate voltage at the CBL.

7.2.1 Implementing logic operations

The key to realize a logic function in the proposed STT-MTJ based C-MRAM array is to supply biasing voltage (Vbias) to the input MTJ devices through a common bitline (CBL). The bitline corresponding to the output is grounded. The conditional switching of the output MTJ device based on the state of the input MTJ device implements the logic function. Let us consider the case where two input logic operations generate a single output, as shown in Figure 7.4. The inputs are stored as MTJ resistances, R1 and R2 and output as MTJ resistance, Rout. The initial value of Rout is set to either (LHS) or high resistance state (HRS). All three MTJ devices (input and output) are connected through a common Connector Line (CL). To realize a logic function both the input MTJ devices are supplied with biasing voltage (Vbias) through common bitline (CBL1=CBL2=V_{bias}) while the output bitline is grounded (CBL3=GND).

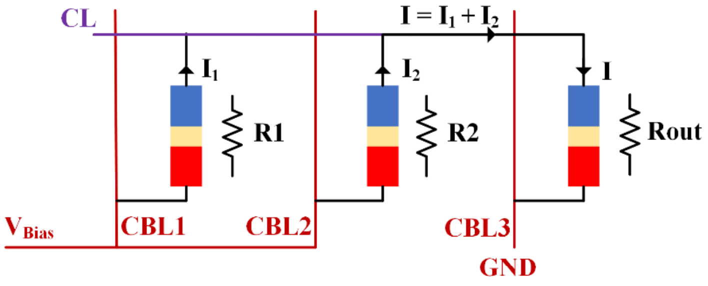


Figure 7.4: Logic function being performed by STT-MTJ based C-MRAM

As a result, a summation current ($I = I_1 + I_2$) flows through the connector line (CL) which is expressed as follows:

$$I = \frac{V_{Bias}}{\left(\frac{R1.R2}{R1+R2}\right)+Rout} \tag{7.1}$$

If the magnitude of the resultant current is greater than the critical current of the MTJ device ($I > Ic$), the output MTJ switches its state. Whereas, for the case when magnitude of the resultant current is less than the critical current of the MTJ device ($I < Ic$), the output MTJ retains its previous state. The type of logic function to be performed in the proposed STT-MTJ based C-MRAM depends on two factors: 1) Biasing Voltage (V_{Bias}) applied at the CBL control line, (2) Preset or Initial value of the output MTJ device. The corresponding bias voltage and the preset value required to implement NAND and NOR logic gates are presented in the following sub-section.

7.2.1.1 NAND logic gate

Table 7.1 presents the boolean expression and truth table for NAND logic function. The first two column A & B, represent all the possible states of input and the third column C, represent the state of output.

TABLE 7.1: TRUTH TABLE OF NAND LOGIC OPERATION

A	**B**	**Cout = A . B**
0	0	1
0	1	1
1	0	1
1	1	0

In this work, we assume that the high resistance state (HRS or R_H) of MTJ device is mapped to logic '0' while low resistance state (LRS or R_L) of the MTJ device is mapped to logic '1'. The resultant current (I) flowing through the output Cout for all the cases of A & B are given by equations 7.2, 7.3, and 7.4.

$$I_{00} = V_{Bias}/\ 0.5(R_H) +\ R_{out} \tag{7.2}$$

$$I_{01}\ /\ I_{10} =\ V_{Bias}/\left(\frac{R_H\ .\ R_L}{R_H + R_L}\right) +\ R_{out} \tag{7.3}$$

$$I_{11} =\ V_{Bias}/0.5(R_L) + R_{out}) \tag{7.4}$$

Using the fact that logic '0' is represented by the high resistance state of the MTJ device and logic '1' is represented by the low resistance state of the MTJ device, it can be implied that the current (I) is maximum when both the inputs are in low resistance state or '11'. The current monotonically decreases as given by equation 7.5.

$$I_{00} < \quad I_{10}/I_{01} < \ I_{11} \tag{7.5}$$

From the truth table of NAND, it can be observed that first three cases should result in logic ‘1’ while the last case should result in logic ‘0’. Since low resistance represent logic ‘1’, current would be maximum in case 4. Assuming the preset value for the NAND gate to be logic ‘1’. Then by choosing an appropriate bias voltage, switching of output MTJ from logic ‘1’ to ‘0’ can be obtained only for the fourth case (A and B = 1). However, if the preset value is assumed to be logic ‘0’, then the first case with lowest current value (I_{00}) should switch the output while fourth case with greater current value (I_{11}) should not. Therefore, the correct functionality of NAND is achieved only when the output is preset to logic ‘1’ and the chosen biasing voltage meet the following criterion: the current I_{11} should be greater than Ic while I_{10}/I_{01} and I_{00} should be less than Ic.

7.2.1.2 NOR logic gate

Table 7.2 presents the boolean expression and truth table for the NOR logic gate. From the truth table of NOR gate, it can be observed that the last three cases should result in logic ‘0’ while the first case should result in logic ‘1’. Since high resistance represents logic ‘0’, current would be minimum in case 1. Assuming the preset value for NOR gate to be logic ‘1’. Then by choosing an appropriate bias voltage, switching of the output MTJ from logic ‘1’ to ‘0’ can be obtained for all three cases (A/B = { 0/1, 1/0, 1/1}).

TABLE 7.2: TRUTH TABLE OF NOR LOGIC GATE

A	**B**	**Cout = A + B**
0	0	1
0	1	0
1	0	0
1	1	0

However, if the preset value is assumed to be logic ‘0’, then the first case with lowest current value (I_{00}) should switch the output while the other cases with greater current value (I_{01} /I_{10}/ I_{11}) should not. Therefore, the correct functionality of NOR gate is achieved only when the output is preset to logic ‘1’ and the chosen biasing voltage meet the following criterion: the currents (I_{01} , I_{10}, I_{11}) should be greater than Ic while I_{00} should be less than Ic.

7.2.2 Circuit-level analysis

In this section, we verify the correct functionality of the logic operation in the STT-MTJ based Computational Magnetic Random-Access Memory (C-MRAM) by performing extensive simulation using the SPICE circuit simulator. We simulate the C-MRAM in logic mode to determine the biasing voltage required for performing NAND & NOR logic operations. The NAND functionality is achieved when the output MTJ switches from LRS (logic '1') to HRS (logic '0') for the case when both the input MTJ are in LRS (logic '1'). Table 7.3 tabulates the biasing voltage (Vbias) applied at CBL and the corresponding switching time of output MTJ.

TABLE 7.3: BIASING VOLTAGE AND SWITCHING TIME FOR NAND & NOR LOGIC GATES

NAND		NOR	
Biasing Voltage (V)	**Switching Times (ns)**	**Biasing Voltage (V)**	**Switching Times (ns)**
0.287	39.8	0.326	40
0.29	35.6	0.33	35
0.3	26.4	0.35	21.9
0.31	21.1	0.38	14.3
0.32	17.7	0.41	10.3
0.326	16.1	0.413	9

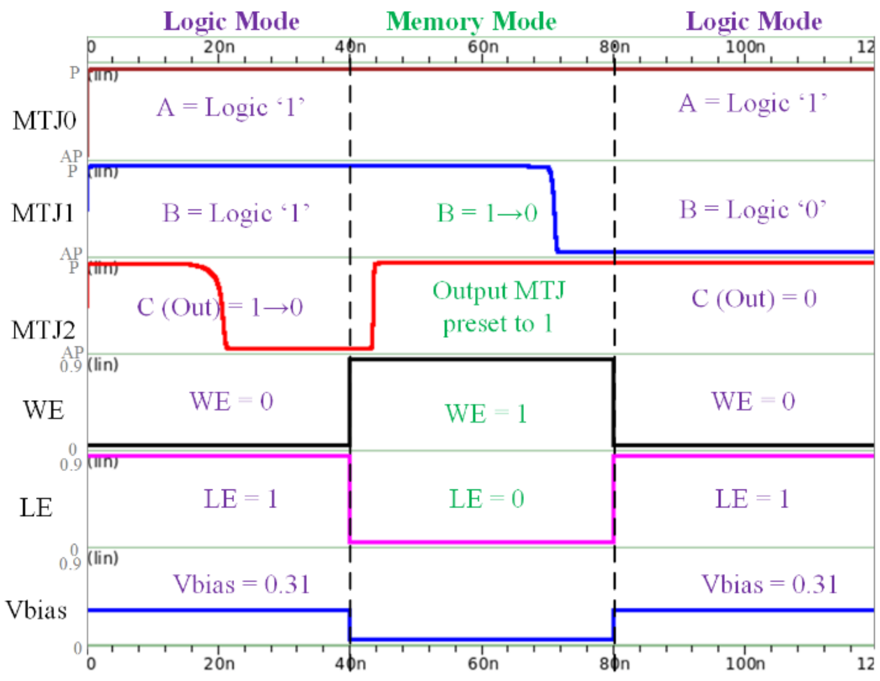


Figure 7.5: The transient simulation waveform performing NAND logic function in STT-MTJ based C-MRAM

Similarly, for NOR logic function, the output MTJ switches from LRS (logic '1') to HRS (logic '0') when either one or the input MTJ are in LRS (logic '1'). Table 7.3 tabulates the biasing voltage (Vbias) applied at CBL and the corresponding switching time of output MTJ.

Figure 7.5 and 7.6 plots the transient simulation waveform for NAND and NOR logic functions. The NAND operation on the input operand A and B stored in MTJ0 and MTJ1 is performed using a bias voltage 0.31V. For the case (A=1 & B=1), the output MTJ switches to logic '0'. Similarly, NOR operation on operand A and B is performed with a bias voltage of 0.35V.

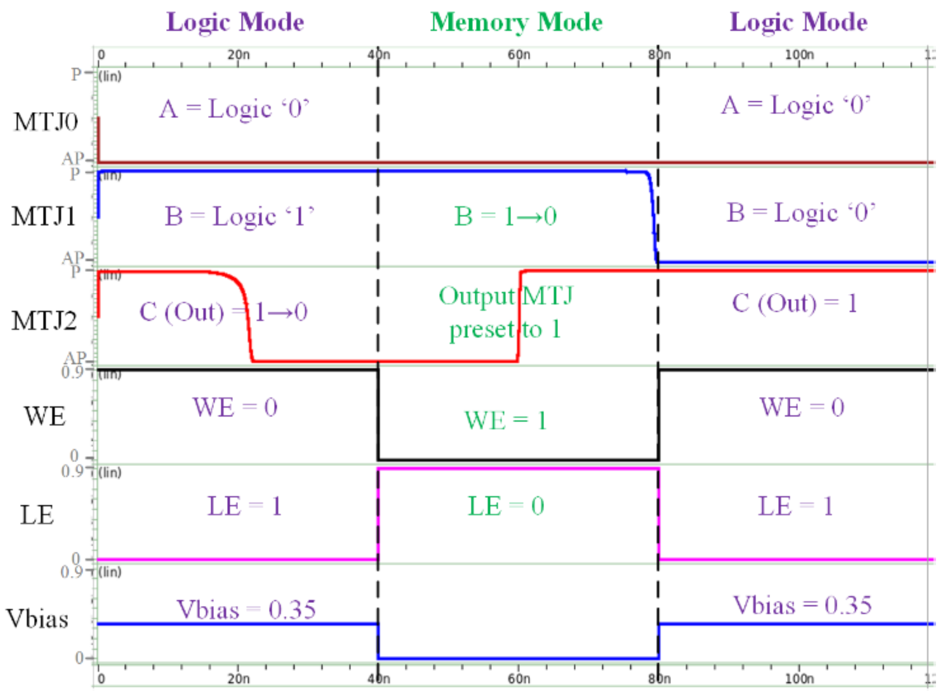


Figure 7.6: The transient simulation waveform for performing NOR logic function in STT-MTJ based C-MRAM

7.3 SHE-MTJ based Computational Magnetic Random-Access Memory (C-MRAM)

The proposed SHE-MTJ based C-MRAM array is presented in Figure 7.7. The basic storage unit of the proposed C-MRAM array is 2T-1MTJ bitcell, which offers the advantage of non-volatility over the conventional memory bitcell. The 2T-1MTJ bitcell of C-MRAM, as shown in Figure 7.7, consists of one SHE-MTJ device and two NMOS access transistors (Tx1 and Tx2) controlled by the global wordlines (WL0 and WL1). The 2T-1MTJ bitcell is connected to global bitline (BL) through terminal T1 of the MTJ device. Similarly, it is connected to the global sourceline (SL) and the SHE control line (SCL) through the access transistors Tx1 and Tx2, respectively, as illustrated in Figure 7.7.

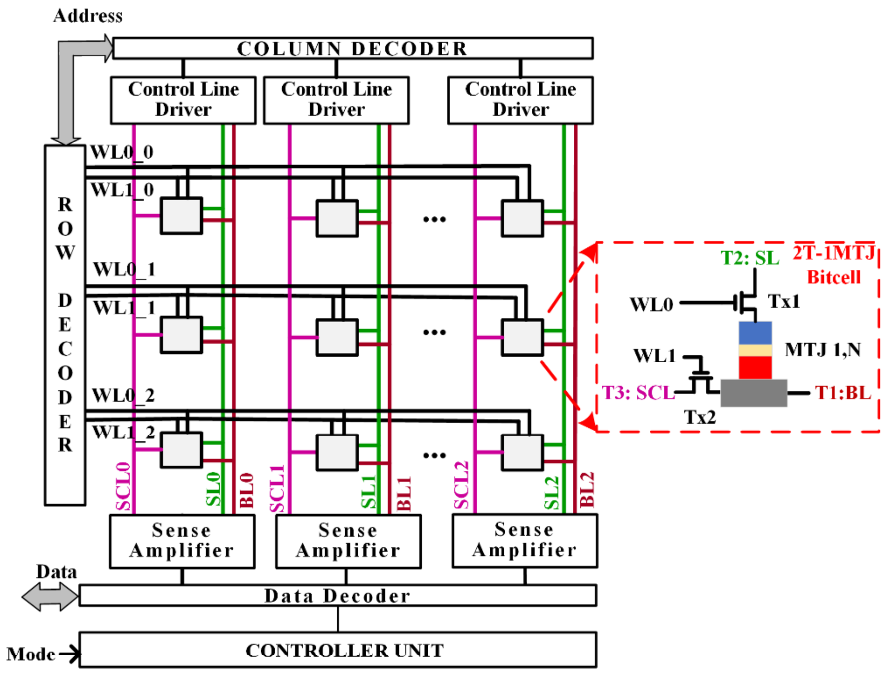


Figure 7.7: Proposed SHE-MTJ based Computational Magnetic Random-Access Memory

The proposed array can operate in two modes: memory mode and logic mode. The two modes of operation are selected using a controller unit and a control signal named, Mode. If the control signal (Mode) is low, the array operates in memory mode. Conversely, if the control signal (Mode) is high, the array operates in logic mode. The detailed description of these two modes of operation are as follows:

Memory Mode: During this mode of operation, the proposed C-MRAM array works as a standard memory with the additional advantage of non-volatility. In memory mode, the data is read or written into the 2T-1MTJ bitcell using the read and write operation. During the write operation, the data is stored into the bitcell using the global BL, SL, and SCL control lines of the proposed C-MRAM array. The voltage at BL, SL, and SCL control lines are used for generating I_{STT} and I_{SHE} currents required for the successful write operation, as shown in Figure 7.8. Let us consider the case where logic '1' is written into the bitcell. The control line BL0 is set to vdd, SL0 is set to gnd, and SCL0 is set to gnd, as illustrated in Figure 7.8 (a). The I_{STT} current from BL0 to SL0 and I_{SHE} current from BL0 to SCL0 flow through the device, which results in switching of the MTJ to HRS, representing logic '1'. In contrast, for writing logic '0' into the bitcell, BL0 is set to gnd while SL0 and SCL0 are set to vdd, as illustrated in Figure 7.8 (b). The I_{STT} current from SL0 to BL0 and I_{SHE} current from SCL0 to BL0 flow through the device. Consequently, the state of the MTJ is switched to LRS representing logic '0'. On the other hand, during the read operation, the resistive state of the MTJ is sensed through a sense amplifier to produce output voltage corresponding to a valid logic level. The read current (I_{READ}) required to determine the state of the MTJ is generated by setting the wordline (WL0) to vdd, sourceline (SL0) to vdd and bitline (BL0) to gnd, as shown in Figure 7.8 (c). The wordline (WL1) is set to gnd to disable transistor Tx2, as shown in Figure 7.8 (c), because only STT current is sufficient to perform the read operation. The sense amplifier reads logic '1' when the MTJ is in high resistance state (HRS). Alternatively, the sense amplifier reads logic '0' when the MTJ is in low resistance state (LRS).

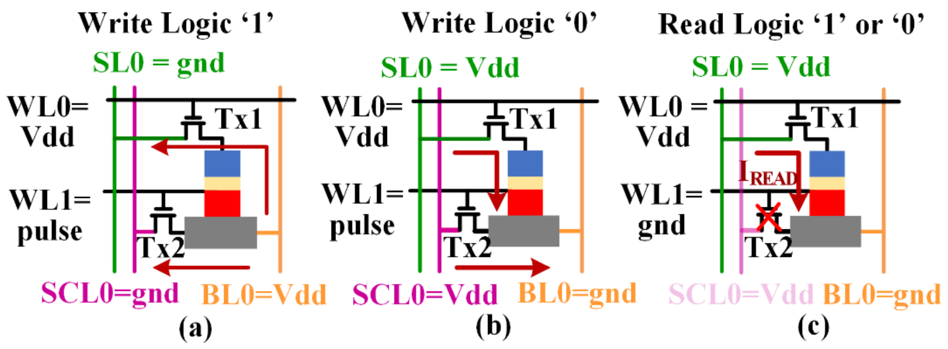


Figure 7.8: Status of control lines and direction of current flow through the 2T-1MTJ bitcell during (a) Write logic '1' operation (b) Write logic '0' Operation (c) Read Operation

7.3.1 Implementing logic operation

The key idea behind implementing the logic in memory is to apply both the input operands simultaneously to the C-MRAM array. The results of logic operations are stored directly in the memory array (as the resistive state of the MTJ device), which eliminates the need for separate write-back operations to store the result. To understand the logic mode of operation, let us consider a case where the logic operation is performed on a single bitcell, as shown in Figure 7.9. The two input operands are denoted as Op_A and Op_B, whereas the result of the logic operation is denoted as Out_C. The logic operation is performed by applying input operand Op_A to the sourceline (SL0), the compliment of Op_A (Op_A_bar) to the bitline (BL0) and operand Op_B to the SHE control line (SCL0) of the 2T-1MTJ bitcell, as shown in Figure 7.9 (a). In this implementation, the conditional switching of the MTJ device based on the applied input operands computes the logic function. The different combination of operands Op_A and Op_B applied to the MTJ device result in four different cases, as illustrated in Figure 7.9 (b) and (c). The voltage at sourceline (SL) and bitline (BL) is responsible for generating the I_{STT} current, while the voltage at SHE controls line (SCL) is responsible for generating the I_{SHE} current needed to realize the logic operation through conditional switching. Now, if both the I_{STT} and I_{SHE} currents flow through the MTJ, the state of the device is flipped as in cases 1 and 4. However, if only I_{STT} is present, the device retains its previous state as in cases 2 and 3. Let us analyze these cases individually:

Case 1: The input operands Op_A and Op_B are both logic '0'. With reference to Figure 7.9 (a), the voltage applied at SL is 0V (Op_A), the voltage applied at BL is vdd (Op_A_bar), and the voltage applied at SCL is 0V (Op_B). As a result, both the I_{SHE} and I_{STT} currents flow through the device, as shown in Figure 7.9 (b). Due to the simultaneous action of I_{SHE} and I_{STT} currents, the state of the MTJ is switched to HRS, which represents logic '1'.

Case 2: The operand Op_A is logic '0' while operand Op_B is logic '1', which implies the voltage applied at SL is 0V (Op_A), the voltage applied at BL is vdd (Op_A_bar), and the voltage applied at SCL is vdd (Op_B). As a result, only I_{STT} current flows through the device, as shown in Figure 7.9 (c), which cannot switch the MTJ within the required amount of time. Hence, the MTJ retains its previous state (PS).

Case 3: The operand Op_A is logic '1' while operand Op_B is logic '0'. Therefore, the voltage applied at SL is vdd (Op_A), the voltage applied at BL is 0V (Op_A_bar) and the voltage applied at SCL is 0V (Op_B). Similar to case 2, the MTJ device maintains its previous state since only I_{STT} current flows through the device, as shown in Figure 7.9 (c).

Case 4: The input operands Op_A and Op_B are both logic '1', and hence the voltage applied at SL is vdd (Op_A), the voltage applied at BL is 0V (Op_A_bar), and the voltage applied at SCL is vdd (Op_B). Similar to case 1, the joint action of I_{SHE} and I_{STT} currents result in the switching of MTJ to LRS, representing logic '0', as shown in Figure 7.9 (b).

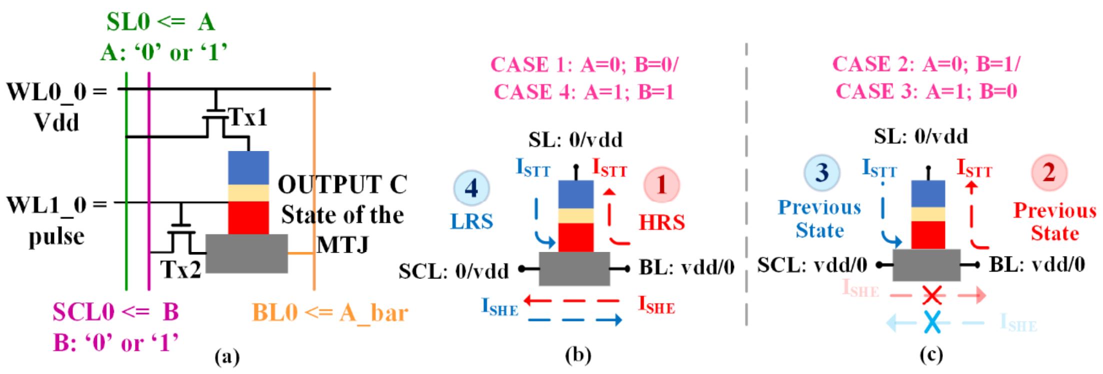


Figure 7.9: (a) Status of control lines to perform logic operation in proposed C-MRAM array (b) Direction of current flow while performing logic function for cases: Case 1: A=0; B=0 and Case 4: A=1; B=1 (c) Direction of current flow while performing logic function for cases: Case 2: A=0; B=1 and Case 3: A=1; B=0.

The results of all the four cases are summarized in a truth table given in Table 7.4.

TABLE 7.4: TRUTH TABLE FOR TWO INPUT LOGIC OPERATIONS

	Op_A	**Op_B**	**Out_C (State of the MTJ)**
Case 1	0	0	HRS (Logic ‘1’)
Case 2	0	1	Previous State
Case 3	1	0	Previous State
Case 4	1	1	LRS (Logic ‘0’)

Based on the above truth table, a general expression for implementing logic function in C-MRAM is presented as follows:

$$C_{i+1} = \bar{A}.C_i + \bar{B}.C_i + \bar{A}.\bar{B} \tag{7.6}$$

where, A and B are the input operands Op_A and Op_B applied at SL and SCL control line of the array, Ci represents the previous state (PS) of the bitcell, and Ci+1 represents the final result of the logic function (Out_C). The various logic gates in the proposed C-MRAM array are modeled by controlling two factors: a) initial value of the MTJ (PRESET) b) Voltages at control lines BL, SL, and SCL.

7.3.1.1 NAND/AND logic gate

To realize the NAND logic gate, only one of the input combinations (Op A = 1 and Op B = 1) should results in logic ‘0’ while other input combinations should result in logic ‘0’, as shown in Table 7.5. This functionality is achieved by presetting the output MTJ (Output C) to high resistance state (logic ‘1’) and applying Op A and Op B directly to the SL and SCL control lines of the bitcell. As a result, the switching of MTJ state occurs only in one case when both Op A and Op B are logic ‘1’, as illustrated in Table 7.6.

TABLE 7.5: TRUTH TABLE OF NAND & AND LOGIC GATES

Op B	**Op A**	**NAND**	**AND**
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

TABLE 7.6: STATUS OF CONTROL LINES TO PERFORM NAND & AND LOGIC OPERATIONS IN PROPOSED C-MRAM

	Op A	Op B	BL	SL	SCL	PRESET	Output
NAND	0	0	Vdd	Gnd	Gnd	HRS	HRS (1)
	0	1	Vdd	Gnd	Vdd	HRS	HRS (1)
	1	0	Gnd	Vdd	Gnd	HRS	HRS (1)
	1	1	Gnd	Vdd	Vdd	HRS	HRS→LRS (1→0)
AND	0	0	Gnd	Vdd	Vdd	LRS	LRS (0)
	0	1	Gnd	Vdd	Gnd	LRS	LRS (0)
	1	0	Vdd	Gnd	Vdd	LRS	LRS (0)
	1	1	Vdd	Gnd	Gnd	LRS	LRS→HRS (0→1)

On the other hand, to realize AND gate, the MTJ (output C), is preset to LRS (logic ‘0’) and the biasing voltage at SL and SCL control lines are applied using the complimented value of input Op A (Op A_bar) and Op B (Op B_bar). Consequently, the switching only happens for the case when Op A and Op B are both logic ‘1’.

7.3.1.2 NOR/OR logic gate

The truth table of NOR logic gate as tabulated in Table 7.7, it can be observed that only the combination of Op B=0 and Op A=0 should results in logic ‘1’ while all other combination of input Op A and Op B should result in logic ‘0’.

TABLE 7.7: TRUTH TABLE OF NOR & OR GATES

Op B	Op A	NOR	OR
0	0	1	0
0	1	0	1
1	0	0	1
1	1	0	1

The NOR functionality is achieved by presetting the output MTJ (Output C) to LRS (logic ‘0’) and applying Op A and Op B directly to the SL and SCL of the MTJ device. As a result, the switching of MTJ state takes place only in one case when both Op A and Op B are logic ‘0’, as illustrated in Table 7.8.

TABLE 7.8: STATUS OF CONTROL LINES TO PERFORM NOR & OR LOGIC OPERATIONS IN PROPOSED C-MRAM

	Op A	Op B	BL	SL	SCL	PRESET	Output
NOR	0	0	Vdd	Gnd	Gnd	LRS	LRS→HRS (0→1)
	0	1	Vdd	Gnd	Vdd	LRS	LRS (0)
	1	0	Gnd	Vdd	Gnd	LRS	LRS (0)
	1	1	Gnd	Vdd	Vdd	LRS	LRS (0)
OR	0	0	Gnd	Vdd	Vdd	HRS	HRS→LRS (1→0)
	0	1	Gnd	Vdd	Gnd	HRS	HRS (1)
	1	0	Vdd	Gnd	Vdd	HRS	HRS (1)
	1	1	Vdd	Gnd	Gnd	HRS	HRS (1)

On the other hand, for OR operation, the output MTJ (output C), is preset to HRS (logic '1') and the biasing voltage at SL, BL, and SCL control lines are applied based on complimented value of input operand Op A (Op A_bar) and Op B (Op B_bar). Consequently, the switching only happens for the case when Op A and Op B are both logic '0'.

7.3.2 Circuit-level analysis

To demonstrate and verify the operation of logic gates within the C-MRAM array, circuit-level simulations are performed for all input combinations using SPICE simulator, as shown in Figure 7.10.

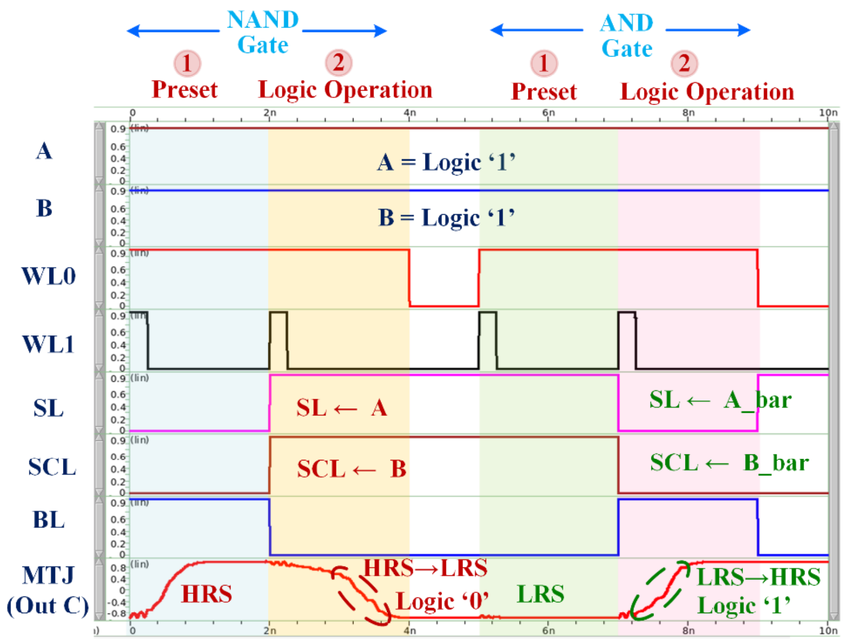


Figure 7.10: Simulation waveform for the NAND & AND logic operation in proposed SHE-MTJ based C-MRAM

The simulation waveform depicted in Figure 7.10 plots the result for NAND & AND logic operations for the case when both the input operands are logic high. For NAND logic operation, the state of the MTJ device is first preset to HRS, and then operands are applied, resulting in logic ‘1’ at the output, which verifies the correct NAND operation. Similarly, for AND operation, the state of the MTJ is first preset to LRS, and then operands are applied, resulting in logic ‘1’ at the output (Out C), which validates the correct AND operation.

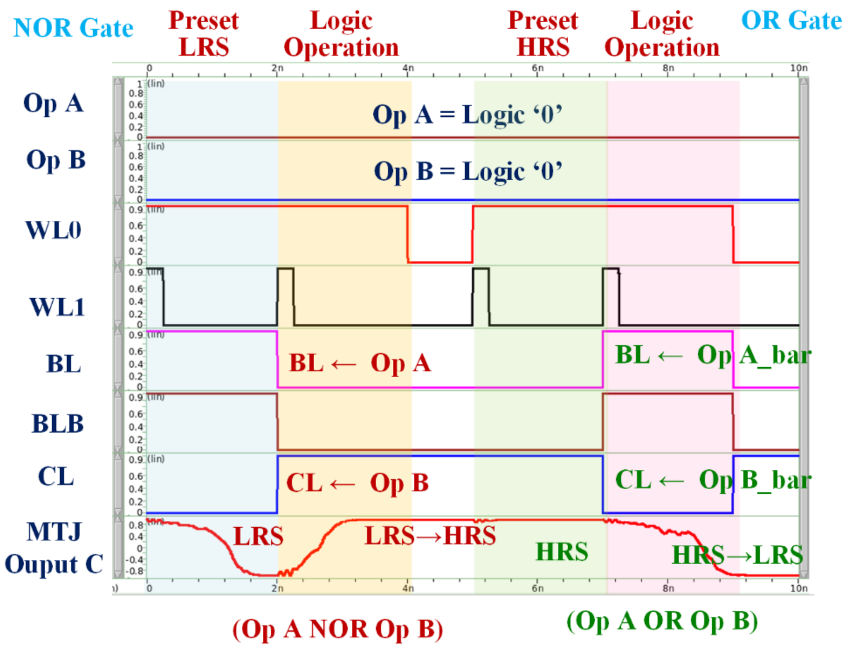


Figure 7.11: Simulation waveform for the NOR & OR logic operation in proposed SHE-MTJ based C-MRAM

Figure 7.11 plots simulation waveform for the NOR & OR gate operation for the case when both the operands Op A and Op B are logic ‘0’. For NOR logic operation, the MTJ (output C) is preset to LRS (logic ‘0’), and control lines SL and SCL are applied with Op A (logic ‘0’) and Op B (logic ‘0’), resulting in switching of the MTJ (output C) from LRS to HRS (logic ‘1’). For OR logic operation, Op A (logic ‘0’) and Op B (logic ‘0’) applied on control lines SL and SCL results in switching of the MTJ (output C) from HRS to LRS (logic ‘0’).

7.4 Proposed Low-Power Arithmetic Function

In this section, we implement an approximate adder (Ax-ADD) using the SHE-MTJ based C-MRAM. The approximate adder is utilized in low-power image processing applications to implement energy-efficient computing by combining the benefits of both in-memory computing and approximate computing. In addition to eliminating data transfer overhead, the proposed approximate adder offers a further reduction in power consumption by relaxing the demand for computational accuracy. A conventional full adder is an arithmetic block which performs addition of three input operands A, B, and C to produce two outputs, denoted as SUM and Carry_out. The expressions for the SUM and Carry_out are presented in equations 7.7 and 7.8:

$$SUM = A \oplus B \oplus C \text{ or } (A \oplus B).\bar{C} + \overline{(A \oplus B)}.C \tag{7.7}$$

$$Carry\_out = AB + BC + AC \tag{7.8}$$

Different from the conventional adder, proposed approximate adder (Ax-ADD) performs the addition operation on input operands Op A, Op B and Op C to generate numerically approximate outputs, SUM$_{approx}$ and Carry_out$_{approx}$ which deviate from the conventional adder outputs, as shown in Table 7.9. The SUM$_{approx}$ and Carry_out$_{approx}$ are expressed as follows:

$$\begin{aligned} SUM_{approx} &= \bar{C}B\bar{A} + C\bar{A}\bar{B} + CB\bar{A} + ABC \\ &= \bar{A}\,(\bar{C}B + C\bar{B}) + BC(\bar{A} + A) \\ &= \bar{A}\,(B \oplus C) + BC \end{aligned} \tag{7.9}$$

$$\begin{aligned} Carry\_out_{approx} &= AB\bar{C} + A\bar{B}C + \bar{A}BC + ABC \\ &= AB(\bar{C} + C) + AC(\bar{B} + B) + BC(\bar{A} + A) \\ &= AB + BC + AC \end{aligned} \tag{7.10}$$

TABLE 7.9: TRUTH TABLE OF PROPOSED APPROXIMATE ADDER (AX-ADD)

Input Operands			**Exact Adder**		**Proposed Approximate Adder Ax-ADD**	
C	**B**	**A**	**Carry**	**SUM**	**Carry_out$_{approx}$**	**SUM$_{approx}$**
0	0	0	0	0	0 √	0 √
0	0	1	0	1	0 √	0 ×
0	1	0	0	1	0 √	1 √
0	1	1	1	0	1 √	0 √
1	0	0	0	1	0 √	1 √
1	0	1	1	0	1 √	0 √
1	1	0	1	0	1 √	1 ×
1	1	1	1	1	1 √	1 √

The truth table for the proposed approximate adder and conventional adder is presented in Table 7.9. It can be observed that there are only two cases (C=0; B=0; A=1 & C=1; B=1; A=0) in which SUM$_{approx}$ deviates from the conventional adder output SUM. However, there is no deviation in Carry_out$_{approx}$ which prevents the propagation of erroneous results to higher stages when a multi-bit adder is designed.

The proposed approximate adder is implemented in C- MRAM array using the logic operations realized in Section 7.3. The adder requires three computational steps to generate the final outputs, SUM$_{approx}$ and Carry_out$_{approx}$. The steps for generating, SUM$_{approx}$ output of the proposed approximate adder depicted in Figure 7.12 are given as follows:

Step 1: The initial value of the bitcell is set to HRS (PRESET = logic '1').

Step 2: The C-MRAM array is applied with input operands at the control lines to perform the logic operation and generate an intermediate result, denoted as SUM$_{intermediate}$. Specifically, in a bitcell, the input operand Op_A is applied at both the sourceline (SL0) and the SHE control line (SCL0), whereas the compliment of operand Op_A (Op_A_bar) is applied at the bitline (BL0), as shown in Figure 7.12. The following expression for SUM$_{intermediate}$ is derived from equation 7.6, by replacing A and B with input operand Op_A, and C_i with logic '1':

$$SUM_{intermediate} = \bar{A} + \bar{A} + \bar{A}.\bar{A}$$

$$SUM_{intermediate} = \bar{A} \tag{7.11}$$

Step 3: The C-MRAM array is applied with rest of input operands at the control lines to generate the final result, SUM$_{approx}$. At the bitcell, the operand Op_B is applied at bitline

(BL0), compliment of operand Op_B (Op_B_bar) is applied at the sourceline (SL0) and the compliment of operand Op_C (Op_C_bar) is applied at the SHE control line (SCL0). The output expression for the final result SUM$_{approx}$ is given in equation 7.12. It is obtained from equation 7.6, by replacing A with Op_B_bar, B with Op_C_bar and C_i with Op_A_bar.

$$SUM_{approx} = B\,.\,C + B\,.\,\bar{A} + \bar{A}\,.C$$

$$SUM_{approx} = (B \oplus C)\,.\,\bar{A} + B.C \tag{7.12}$$

Similarly, steps for generating Carry_out$_{approx}$ output of the proposed approximate adder depicted in Figure 7.13 are given as follows:

Step 1: Initialize bitcell to LRS (PRESET = logic ‘0’).

Step 2: Apply operand Op_A to bitline (BL0) and compliment of operand Op_A (Op_A_bar) to SL0 and SCL0 control lines of the bitcell to compute Carry_out$_{intermediate}$. The expression for Carry_out$_{intermediate}$ is given as follows:

$$Carry\_out_{intermediate} = A \tag{7.13}$$

Step 3: Finally, apply operand Op_B to bitline (BL0), compliment of operand Op_B (Op_B_bar) to sourceline (SL0) and compliment of operand Op_C (Op_C_bar) to SHE control line (SCL0) of the bitcell to compute Carry_out$_{approx}$. The final output Carry_out$_{approx}$ is expressed as follows:

$$Carry\_out_{approx} = AB + BC + AC \tag{7.14}$$

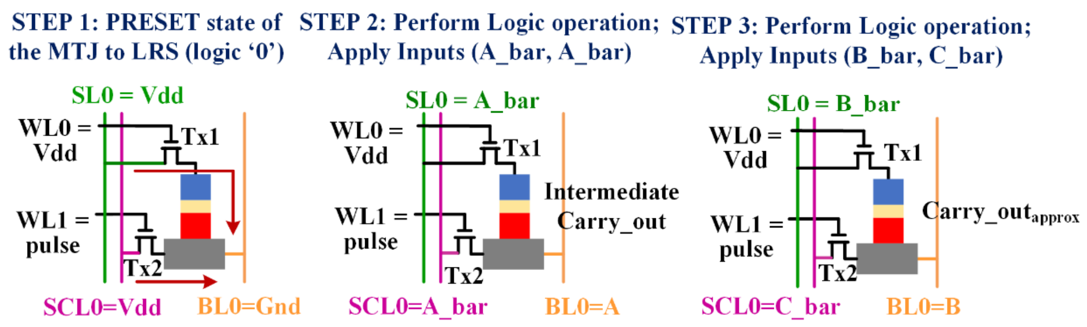

STEP 1: PRESET state of the MTJ to LRS (logic ‘0’)
STEP 2: Perform Logic operation; Apply Inputs (A_bar, A_bar)
STEP 3: Perform Logic operation; Apply Inputs (B_bar, C_bar)
SL0 = Vdd
WL0 = Vdd
Tx1
WL1 = pulse
Tx2
SCL0=Vdd
BL0=Gnd
SL0 = A_bar
Intermediate Carry_out
SCL0=A_bar
BL0=A
SL0 = B_bar
Carry_out$_{approx}$
SCL0=C_bar
BL0=B

Fig   nate

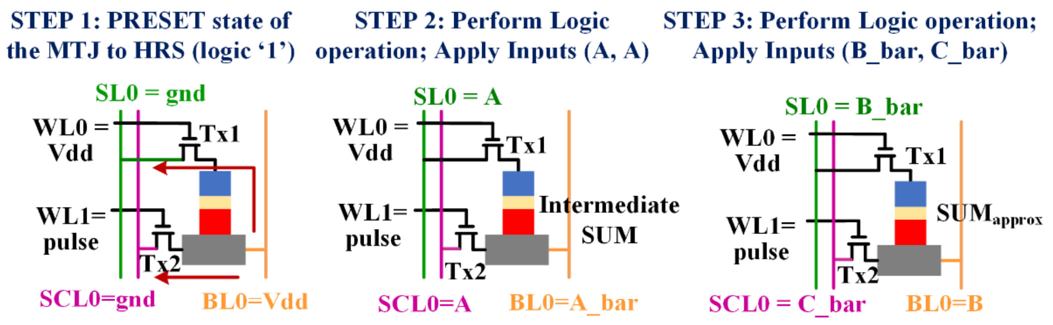
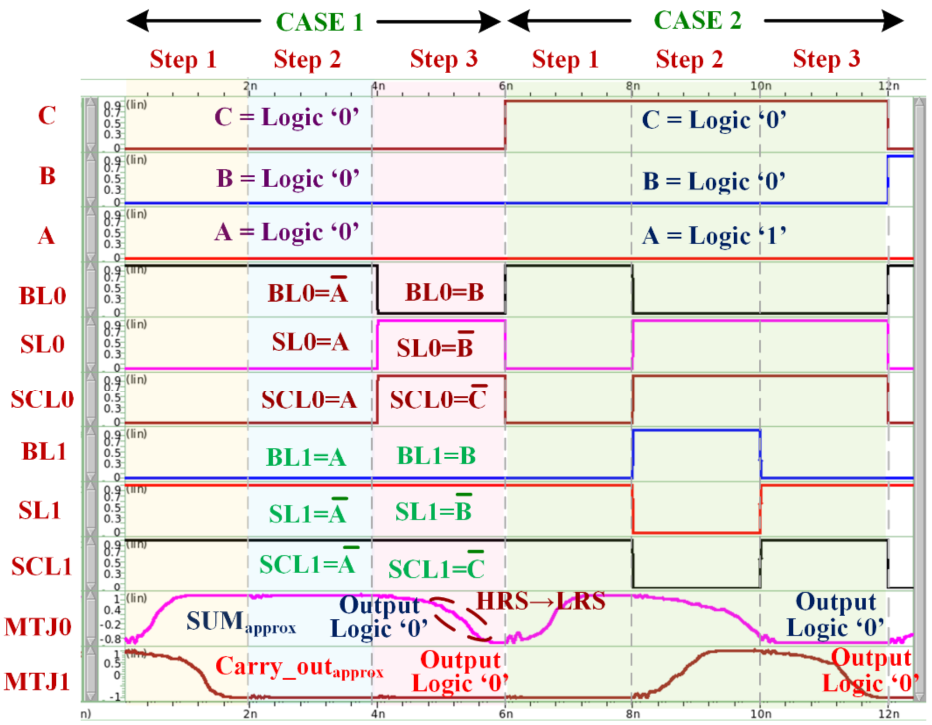

STEP 1: PRESET state of the MTJ to HRS (logic ‘1’)
STEP 2: Perform Logic operation; Apply Inputs (A, A)
STEP 3: Perform Logic operation; Apply Inputs (B_bar, C_bar)
SL0 = gnd
WL0 = Vdd
Tx1
WL1= pulse
Tx2
SCL0=gnd
BL0=Vdd
SL0 = A
Intermediate SUM
SCL0=A
BL0=A_bar
SL0 = B_bar
SUM$_{approx}$
SCL0 = C_bar
BL0=B

Figure 7.13: Computation steps to generate Carry_out$_{approx}$ output of the proposed approximate full adder

7.4.1 Circuit-level analysis

To validate the operation of the proposed approximate adder (Ax-ADD), SPICE simulations are performed for all the input combinations. The simulation waveform shown in Figure 7.14 depicts the details of approximate arithmetic operation for two input cases: case 1 (C = 0; B = 0; A = 0) and case 2 (C = 0; B = 0; A = 1). Both the cases take three sequential steps to generate and store the SUM$_{approx}$ and Carry_out$_{approx}$ in two different bitcell (MTJ0 and MTJ1)



simultaneously. In case 1, the state of the MTJ0 and MTJ1 are first initialized to HRS and LRS, respectively. Then, input operands are applied in second and third sequential step, resulting in logic '0' at the outputs (MTJ0 and MTJ1), which verifies the correct approximate arithmetic operation. Similarly, for case 2, the valid result of the addition operation (SUM$_{approx}$ = Logic '0' and Carry_out$_{approx}$= Logic '0') is available after the third computational step.

Further, to quantitatively analyze all the operational modes of the proposed SHE-MTJ based C-MRAM array, we perform SPICE simulation in two phases, as illustrated by the simulation waveform in Figure 7.15. In the first phase, the proposed array is configured in memory mode. During this mode, operands Op_A, Op_B, and Op_C (110) are first written into the bitcell MTJ00, MTJ01, and MTJ02, respectively. Next, to demonstrate the advantage of non-volatility and data retention in the proposed array, the supply voltage to the array is cut-off. During the power-off period, the array bitcells retain their previous state. At the end of the first phase, supply voltage is restored for normal operation and read operation is performed to verify the value of operands Op_A, Op_B, and Op_C stored in bitcell MTJ00, MTJ01, and

Figure 7.14: Simulation waveform to validate the SUM$_{approx}$ and Carry_out$_{approx}$ outputs of proposed adder

MTJ02 before power-off. In the next phase, the C-MRAM array is configured in logic mode to demonstrate the feasibility of performing logic and arithmetic operation within the proposed array. During this mode, C-MRAM array is first configured to implement NAND logic operation on operand Op_A and Op_B, which results in valid logic '0' at the output as depicted by LRS state of bitcell MTJ10 in Figure 7.15. Finally, we implement arithmetic operation where we perform addition operation on operands Op_A, Op_B, and Op_C, which produces logic ‘1’ at the SUMapprox output, which is verified by the HRS of the bitcell MTJ11, as shown in Figure 7.15. Table 7.10 summarizes the average energy consumption and latency for the various operation performed in the 2T-1MTJ bitcell. The logic operation is performed in two computational steps, whereas arithmetic operation is performed in three computational steps, as described in Section 7.3 and 7.4. Therefore, the total energy consumption (E) and total latency (τ) for logic operation and arithmetic operation are evaluated as follows:

$$\tau_{logic} = \tau_{preset} + \tau_{step2}$$
$$E_{logic} = E_{preset} + E_{step2} \tag{7.15}$$

$$\tau_{arithmetic} = \tau_{preset} + \tau_{step2} + \tau_{step3}$$
$$E_{arithmetic} = E_{preset} + E_{step2} + E_{step3} \tag{7.16}$$

where, E_{preset}, E_{step2}, and E_{step3} are the total energy consumption during the present, second and third computational steps, respectively. Similarly, τ_{preset}, τ_{step2}, and τ_{step3} are the total latency of the preset, second and third computational steps, respectively.

	Operation	**Energy Consumption (E)**	**Latency (τ)**
Memory Operation	Write Operation	276.57 fJ	2 ns
	Read Operation	1.7 fJ	2 ns
Logic Operation	NAND Operation	522.78 fJ	4 ns
	NOR Operation	421.25 fJ	4 ns
	AND Operation	428.75 fJ	4 ns
	OR Operation	525.5 fJ	4 ns
Arithmetic	SUM	770 fJ	6 ns

Operation	Carry_out	668 fJ	6 ns

TABLE 7.10: AVERAGE ENERGY CONSUMPTION AND LATENCY FOR VARIOUS OPERATIONS PERFORMED IN C-MRAM

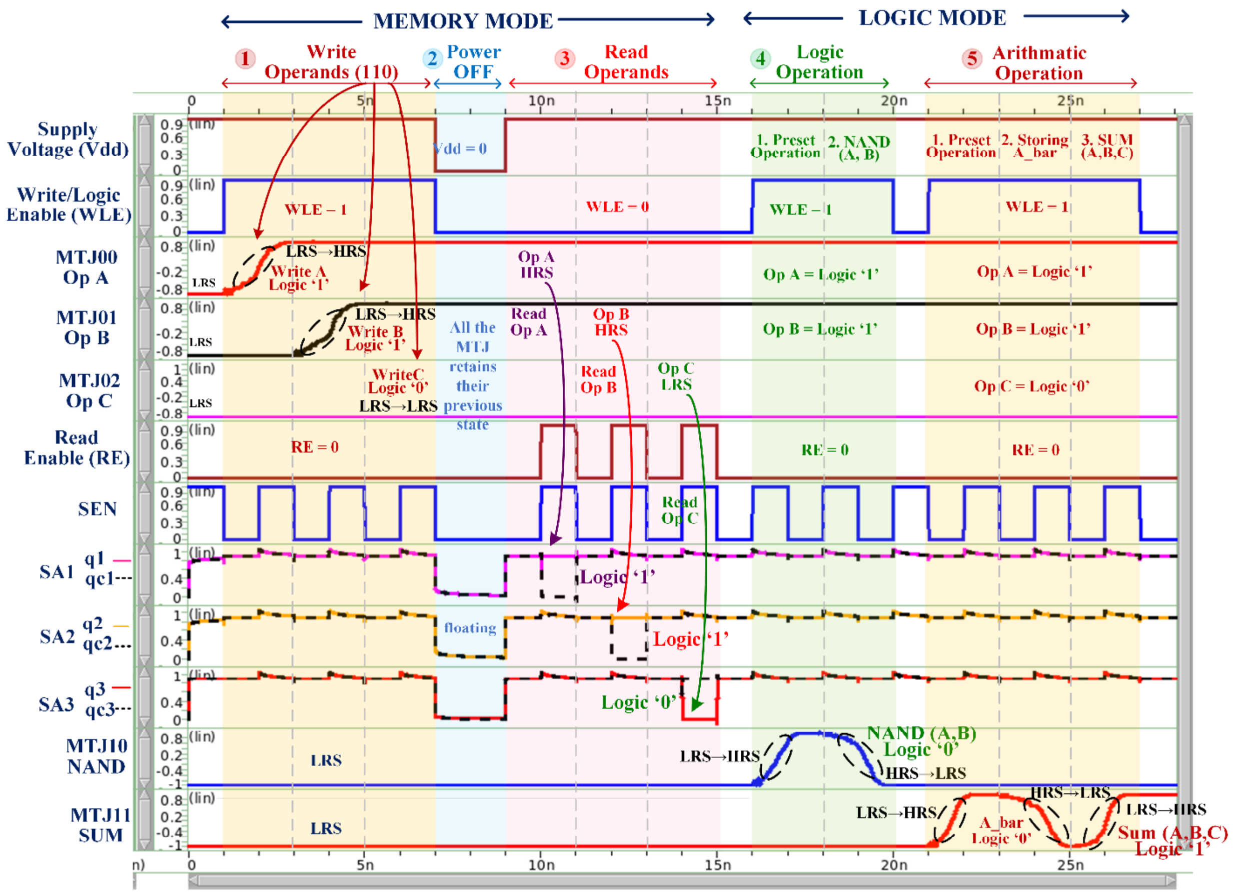


Figure 7.15: Simulation waveform to demonstrate all the operational modes of proposed SHE-MTJ based C-MRAM array

7.5 Conclusion

In this work, we implement basic logic gates and arithmetic function using a dual-mode memory/computing unit based on emerging non-volatile magnetic device, which helps in mitigating the von-neumann bottleneck of the conventional computing systems. We call the proposed unit Computational Magnetic Random-Access Memory (C-MRAM). The proposed array works in two modes: 1) Memory mode, where it acts as standard memory with the additional advantage of non-volatility, and 2) Logic mode, where it performs the computation inside the memory array. The unique properties of the magnetic tunnel junction device are exploited to realize logic computation within the C-MRAM array itself. The in-memory computation offers the advantage of reduce power consumption due to elimination of energy overhead associated with data movement. The proposed computational magnetic random-access memory (C-MRAM) arrays utilize emerging spin-transfer torque (STT) and spin-hall effect (SHE) magnetic tunnel junction (MTJ) to implement basic logic gates such as NAND, AND, NOR & OR. Further, we realize an approximate adder (Ax-ADD), which reduces the power consumption of arithmetic function by lowering the demand for computational accuracy. The approximate adder is implemented using SHE-MTJ device to achieve faster and energy-efficient computing. The simulation results show that both proposed C-MRAM array can efficiently perform memory and computation operations. The proposed approach of logic computation combines the benefits of in-memory computing and approximate computing in addition to normally-OFF computing. Hence, significant power saving is achieved to effectively meet the requirement of low power computing in emerging IoT applications.

REFERENCES

[1] X. Yang, Y. Hou, and H. He, "A processing-in-memory architecture programming paradigm for wireless internet-of-things applications," *Sensors (Switzerland),* vol. 19, no. 1, pp. 1–23, 2019.

[2] M. Zabihi, Z. Chowdhury, Z. Zhao, U. R. Karpuzcu, J. P. Wang, and S. Sapatnekar, "In-Memory Processing on the Spintronic CRAM: From Hardware Design to Application Mapping," *IEEE Trans. Comput.*, vol. 9340, no. c, 2018.

[3] S. Li, C. Xu, Q. Zou, J. Zhao, Y. Lu, and Y. Xie, "Pinatubo: A processing-in-memory architecture for bulk bitwise operations in emerging non-volatile memories," *Proc. - Des. Autom. Conf.,* vol. 05-09-June, 2016.

[4] W.-H. Chen et al., "Circuit design for beyond von Neumann applications using emerging memory: From nonvolatile logics to neuromorphic computing," *2017 18th International Symposium on Quality Electronic Design (ISQED),* pp. 23–28.

[5] G. Santoro, G. Turvani, and M. Graziano, "New logic-in-memory paradigms: An architectural and technological perspective," *Micromachines,* vol. 10, no. 6, 2019.

[6] W. Kang, H. Wang, Z. Wang, Y. Zhang, and W. Zhao, "In-Memory Processing Paradigm for Bitwise Logic Operations in STT–MRAM," *IEEE Trans. Magn.*, vol. 53, no. 11, pp. 1–4, Nov. 2017.

[7] S. Khoram, Y. Zha, J. Zhang, and J. Li, "Challenges and Opportunities," *Proceedings of the 2017 ACM on International Symposium on Physical Design, 2017,* vol. Part F1271, pp. 43–46.

[8] S. Bhatti, R. Sbiaa, A. Hirohata, H. Ohno, S. Fukami, and S. N. Piramanayagam, “Spintronics based random access memory: a review,” *Mater. Today*, vol. 20, no. 9, pp. 530–548, 2017.

[9] G. Singh et al., “A Review of Near-Memory Computing Architectures: Opportunities and Challenges,” in *2018 21st Euromicro Conference on Digital System Design (DSD),* 2018, vol. 1, pp. 608–617.

[10] M. Imani, Y. Kim, and T. Rosing, “MPIM: Multi-purpose in-memory processing using configurable resistive memory,” *Proc. Asia South Pacific Des. Autom. Conf. ASP-DAC,* pp. 757–763, 2017.

[11] H. Cai, Y. Wang, L. A. De Barros Naviner, J. Yang, and W. Zhao, “Exploring Hybrid STT-MTJ/CMOS Energy Solution in Near-/Sub-Threshold Regime for IoT Applications,” *IEEE Trans. Magn.*, vol. 54, no. 2, pp. 1–9, 2018.

[12] Kanika, R. S. Prasad, N. Chaturvedi, and S. Gurunarayanan, “A low power high speed MTJ based non-volatile SRAM cell for energy harvesting based IoT applications,” *Integr. VLSI J.*, no. October, pp. 2–9, 2018.

[13] M. Gao, Q. Wang, M. T. Arafin, Y. Lyu, and G. Qu, “Approximate computing for low power and security in the internet of things,” *Computer (Long. Beach. Calif).,* vol. 50, no. 6, pp. 27–34, 2017.

[14] K. Ma et al., “IAA: Incidental Approximate Architectures for Extremely Energy-Constrained Energy Harvesting Scenarios using IoT Nonvolatile Processors,” *IEEE Micro,* vol. 38, no. 4, pp. 11–19, 2018.

[15] Z. Wen, D. Le Quoc, P. Bhatotia, R. Chen, and M. Lee, “ApproxIoT: Approximate Analytics for Edge Computing,” *in 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), 2018,* vol. 2018, pp. 411–421.

[16] İ. Taştan, M. Karaca, and A. Yurdakul, “Approximate CPU Design for IoT End-Devices with Learning Capabilities,” *Electronics*, vol. 9, no. 1, p. 125, Jan. 2020.

[17] S. Venkataramani, V. Chippa, S. Chakradhar, K. Roy and A. Raghunathan, "Quality programmable vector processors for approximate computing," in *2013 46th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO*), Davis, CA, USA, 2013 pp. 1-12.

[18] A. Roohi and R. F. DeMara, “NV-Clustering: Normally-Off Computing Using Non-Volatile Datapaths,” *IEEE Trans. Comput*., vol. 67, no. 7, pp. 949–959, Jul. 2018.

[19] A. Chen, “A review of emerging non-volatile memory (NVM) technologies and applications,” *Solid. State. Electron*., vol. 125, pp. 25–38, 2016.

[20] A. Amirany and R. Rajaei, “Nonvolatile, Spin-Based, and Low-Power Inexact Full Adder Circuits for Computing-in-Memory Image Processing,” *SPIN*, vol. 9, no. 3, pp. 1–11, 2019.

[21] P. Barla, V. K. Joshi, and S. Bhat, “A novel low power and reduced transistor count magnetic arithmetic logic unit using hybrid STT-MTJ/CMOS Circuit,” *IEEE Access*, vol. 8, pp. 6876–6889, 2020.

[22] E. Eken et al., “Spin-Hall Assisted STT-RAM Design and Discussion,” *in Proceedings of the 18th System Level Interconnect Prediction Workshop on ZZZ - SLIP ’16*, 2016, pp. 1–4.

[23] Liang Chang, Z. Wang, Yuqian Gao, W. Kang, Y. Zhang and W. Zhao, "Evaluation of spin-Hall-assisted STT-MRAM for cache replacement," *2016 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH),* Beijing, 2016, pp. 73-78, doi: 10.1145/2950067.2950107.

[24] I. Ahmed, Z. Zhao, M. G. Mankalale, S. S. Sapatnekar, J.-P. Wang, and C. H. Kim, "A Comparative Study Between Spin-Transfer-Torque and Spin-Hall-Effect Switching Mechanisms in PMTJ Using SPICE," *IEEE J. Explor. Solid-State Comput. Devices Circuits*, vol. 3, no. September, pp. 74–82, 2017.

[25] A. Van Den Brink et al., "Spin-Hall-assisted magnetic random access memory," *Appl. Phys. Lett.,* vol. 104, no. 1, p. 012403, 2014.

[26] Z. Wang, W. Zhao, E. Deng, J. O. Klein, and C. Chappert, "Perpendicular-anisotropy magnetic tunnel junction switched by spin-Hall-assisted spin-transfer torque," *J. Phys. D. Appl. Phys.,* vol. 48, no. 6, 2015.