Skyrmion Logic-In-Memory Architecture for Maximum/Minimum Search

*electronics*

MDPI

*Article*

# Skyrmion Logic-In-Memory Architecture for Maximum/Minimum Search

**Luca Gnoli *, Fabrizio Riente, Marco Vacca, Massimo Ruo Roch and Mariagrazia Graziano**

Department of Electronics and Telecommunications, Politecnico di Torino, 10129 Torino, Italy; fabrizio.riente@polito.it (F.R.); marco.vacca@polito.it (M.V.); massimo.ruoroch@polito.it (M.R.R.); mariagrazia.graziano@polito.it (M.G.)

* Correspondence: luca.gnoli@polito.it

**Abstract:** In modern computing systems there is the need to utilize a large amount of data in maintaining high efficiency. Limited memory bandwidth, coupled with the performance gap between memory and logic, impacts heavily on algorithms performance, increasing the overall time and energy required for computation. A possible approach to overcome such limitations is Logic-In-Memory (LIM). In this paper, we propose a LIM architecture based on a non-volatile skyrmion-based recetrack memory. The architecture can be used as a memory or can perform advanced logic functions on the stored data, for example searching for the maximum/minimum number. The circuit has been designed and validated using physical simulations for the memory array together with digital design tools for the control logic. The results highlight the small area of the proposed architecture and its good energy efficiency compared with a reference CMOS implementation.

**Keywords:** logic in memory; maximum search; skyrmions; spintronics



**Citation:** Gnoli, L.; Riente, F.; Vacca, M.; Ruo Roch, M.; Graziano, M. Skyrmion Logic-In-Memory Architecture for Maximum/Minimum Search. *Electronics* **2021**, *10*, 155. https://doi.org/10.3390/electronics10020155

Received: 19 October 2020
Accepted: 8 January 2021
Published: 12 January 2021



## 1. Introduction

In modern computing systems, the requirement to utilize large amounts of data to maintain a high efficiency is becoming a major concern for design. Data that is stored in the memory is read, elaborated in the processing units, and then written back. The continuous scaling and improvement of the CMOS technology made processing units every year more powerful. At the same time, memory technologies have not improved at the same rate, creating a performance gap between storage and processing units. Moreover, the limited bandwidth available for data transfer makes this limitation even more critical for the overall performance of computing systems. In addition, the data movement from and to the memory requires a non-negligible amount of energy that has a great impact on power consumption [1]. To mitigate these issues, a solution proposed in the literature is to move the computation inside the memory unit [2]. In this paradigm, called Logic-in-Memory (LIM), the data stored is elaborated without the need of moving the data outside of memory. The required logic is distributed inside the memory or in the peripheral circuitry.

Spintronic devices are particularly suitable for implementing this approach. These devices have both memory and logic capabilities, taking advantage of phenomena linked to magnetization manipulation. In particular, skyrmions have been proposed as a replacement of plain domains in racetrack memories as proposed by Parkin in 2008 [3]. In this device, information encoded in domains is written, shifted, and read serially on a magnetic nanostrip. Along with the memory application, many solutions for boolean and non-boolean logic operations have been proposed in the literature. These solutions take advantage of the skyrmion mutual repulsion and rich dynamic to obtain different logic functions. In [4], a set of logic gates based on skyrmion-domain wall pair conversion was proposed. The solution implementing OR and AND gates with patterned ferromagnetic structures also provides a robust mechanism to duplicate the input information. Another solution proposed in [5] takes advantage of natural deviation and mutual repulsion of skyrmions to

obtain logic functions, without the need of any conversion. The skyrmions, encoding the input information, are pushed through a patterned structure to obtain the logic function. Synchronization is also possible using patterned constrictions [6] or voltage-controlled magnetic anisotropy barriers [7]. Finally, in [8,9] synaptic devices have been proposed, taking advantage of interactions of skyrmion with potential barriers and mutual repulsion between skymions. In such devices, the inner state, represented by a set of skyrmions is modified by means of currents. The skyrmions pushed in and out from a reading zone mimic the promotion and inhibition neural process.

Despite the proposal of many logic and memory devices based on skyrmions, the evaluation of digital architectures in which boolean information can be stored and processed is still lacking. In the literature, examples of elaboration architectures based on skyrmion are presented in [10,11]. In [10], the authors propose an accelerator for convolutional neural networks based on skyrmions. The paper shows a device able to classify images stored in the form of skyrmions. In [11], the authors propose a logic in memory implementation for binary neural networks, in which skyrmion racetrack memory is used to accelerate the computations. Finally, in [12] the authors propose a cache memory based on skyrmion and evaluate its performance with respect to SRAM and other spintronic technologies. The main contributions of this paper are the following:

- We designed a logic in memory architecture based on skyrmions that can find the minimum or the maximum value stored within the memory;
- We designed a memory cell based on skyrmions, capable of operating not only as memory but also as a computing device. From the storage point of view it can be used as a classical RAM memory, but it integrates logic capabilities implementing AND, OR Boolean functions without the need of and electric conversion for the processing phase;
- The entire entire memory cell was studied through micromagnetic simulations. The cell includes a processing zone, where the elaboration is non-destructive making possible to maintain the information even after computation;
- We evaluated the entire system performance, with an increasing number of words in the array starting from 2048 up to 65,536. The evaluation takes into account not only the skyrmions-based memory array, but also the contribution coming from the peripheral CMOS circuitry to control the array;
- We compared the array performance with an existing CMOS implementation in term of dissipated power and the energy per bit.

The proposed design shows a small area occupation along with good energy and timing performance especially for big memory sizes, compared with a reference Logic-in-Memory implementations based on CMOS. Furthermore, since skyrmions are used to store the information the system does not have static power consumption when operations are not performed on data.

The paper is structured as follows: In Section 2, a background about skyrmions is presented. In Section 3, the basic cell is described and a typical function is presented and in Sections 4 and 5 the complete system is described. Section 7 contains the results of performance evaluation, which are presented and discussed. Finally in Section 8, final considerations and future prospects about the system are provided.

## 2. Background

Magnetic skyrmions are chiral magnetic configurations with unitary topological charge. They can be found stable in materials with high spin orbit coupling lacking inversion symmetry, in particular, bulk materials like B-20 ferromagnets [13] and thin films with ferromagnets in contact with high spin orbit materials like Pt, Ir, W, and Ta [14,15]. The skyrmion state is stabilized by the Dzyaloshinskii–Moriya interaction (DMI). The interaction between neighboring atoms can be expressed as $-\mathbf{D}_{i,j} \cdot (\mathbf{S}_i \times \mathbf{S}_j)$ where $\mathbf{S}_i$ and $\mathbf{S}_j$ are spins in sites i and j respectively and $\mathbf{D}_{i,j}$ is the DMI vector [16].

The most interesting characteristics of skyrmions are the small size, the low depinning currents, and the stability at room temperature. In the literature, many studies are focused on memory applications. In particular, skyrmions have been proposed as a replacement of the plain domain walls in racetrack memories [17] proposed by Parkin in 2008 [3]. In racetracks, bits, encoded as physically connected magnetic domains, are written serially by a write head to the memory. The domains are then shifted to be stored and later read if needed.

Skyrmions can be moved by currents flowing either in the ferromagnet or in the heavy metal. When the current flows in the ferromagnets, the texture moves due to the Spin Transfer Torque (STT) effect. When the current is injected in the heavy metal a spin-dependent scattering effect, called Spin Hall Effect (SHE), generates vertical spin currents in the ferromagnet in contact with the metal. The current generated exert a torque on the magnetic texture and consequently a movement. SHE is generally preferred as it is proved to be more efficient for domain movement [16].

Skyrmions when pushed by currents show a significant longitudinal component with respect to the current direction as highlighted in Figure 1 [16]. The deviation, called Skyrmion Hall Effect (SkHE), limits the maximum speed achievable by skyrmions in a track. The natural repulsion of skyrmions from track edges compensates the longitudinal movement only up to a threshold current density over which the skyrmion is annihilated on the track edges and the information is lost [6].

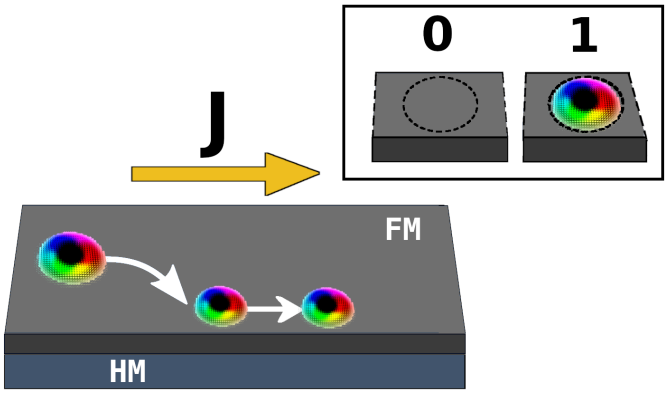



**Figure 1.** Skyrmion movement inside a plain nanostrip in response to a current flow. In the inset, the encoding of binary information with skyrmions.

To reduce the magnitude of deviations possible solutions are: Synthetic Antiferromagnets (SAFs) [18], Potential gradients [19], and Curbed tracks [20]. SAFs require two antiferromagnetically coupled ferromagnetic layers, in which information is stored as a couple of skyrmions with opposite core directions coupled. The opposite core directions of the two skyrmions and the consequent opposite deviations result in a zero net movement in transverse direction.

Potential gradient allows the creation of a preferred path for skyrmion movement [19] allowing to counteract the SkHE far from the edges, reducing greatly the chance of annihilation and consequently rising the maximum current density. In addition, potential gradients can be created and controlled by means of localized electric fields. Indeed, a voltage applied across the material allows to rise or lower locally the magnetic anisotropy. This effect is called Voltage Controlled Magnetic Anisotropy (VMCA) [21,22].

Finally, a curbed track constricts the movement of the skyrmions allowing the definition of a preferred movement path in which lateral deviation has a strong compensation. Moreover in the last two solutions, the strength of demagnetizing and anisotropy related energies are different with respect to a plain track, resulting in different skyrmion dimensions and increased velocities under the same current densities [19,20,23].

## 3. Memory Cell

The basic element of the proposed architecture is the memory cell. In Figure 2, a portion of the track hosting the skyrmion is showed. The curb in the ferromagnet constricts the movement of the skyrmion in a direction longitudinal to the current mitigating the SkHE, allowing the skyrmion to reach higher speeds compared to a plain nanotrack [20]. The stack composition is W\$Co_{20}Fe_{60}B_{20}$\MgO. The ferromagnet thickness is 1.5 nm at the edges and 1.0 nm in the curb. The track is 56 nm wide, with a curb of 30 nm. The material parameters are reported in Table 1.

**Table 1.** Parameters used for micromagnetic simulations and current distribution computation.

| Simulation Parameters | | |
|---|---|---|
| Saturation Magnetization [16] | $1 \times 10^6$ | $A\,m^{-1}$ |
| Uniaxial Anisotropy Constant [16] | $8 \times 10^5$ | $J\,m^{-2}$ |
| Exchange Stiffness [16] | $2 \times 10^{-11}$ | |
| Damping constant [16] | 0.015 | |
| Spin Hall Angle [24] | 0.4 | |
| Film resistivity [24] | 165 | $\mu\Omega\,cm$ |

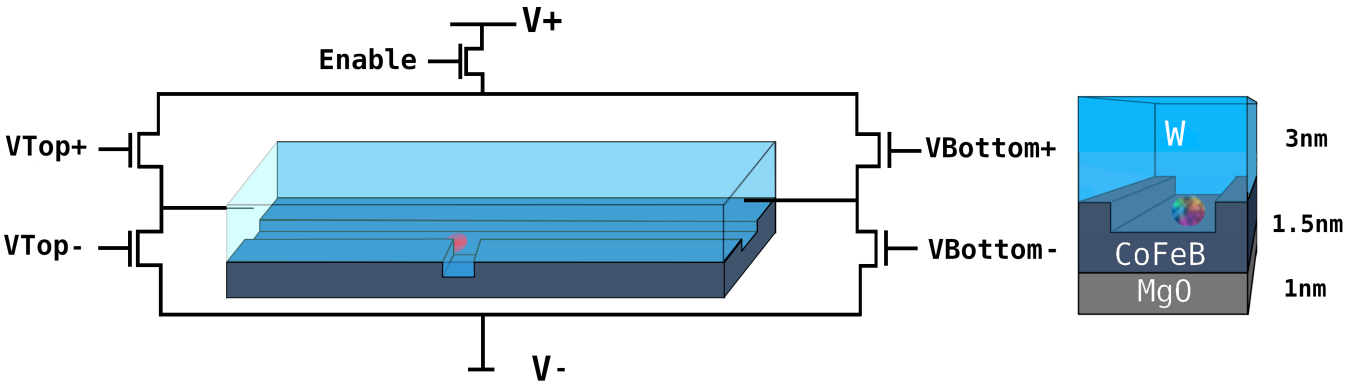
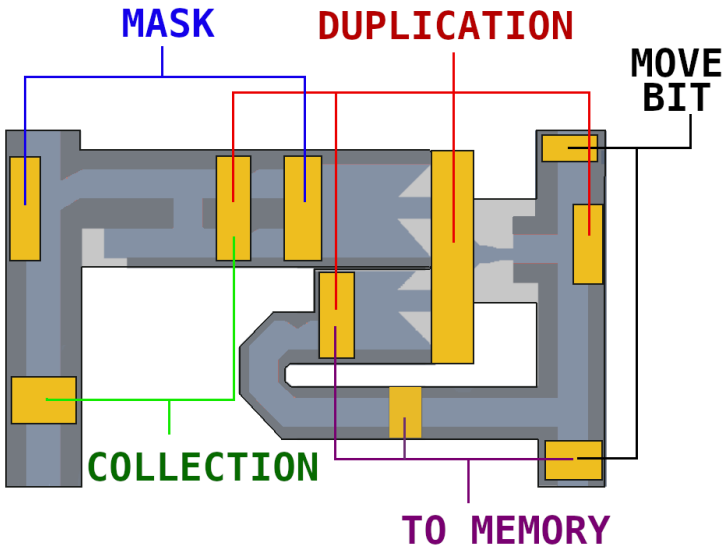



**Figure 2.** Basic memory racetrack structure and stack composition. The track is connected to V+ and V− in order to move the skyrmion in both directions.

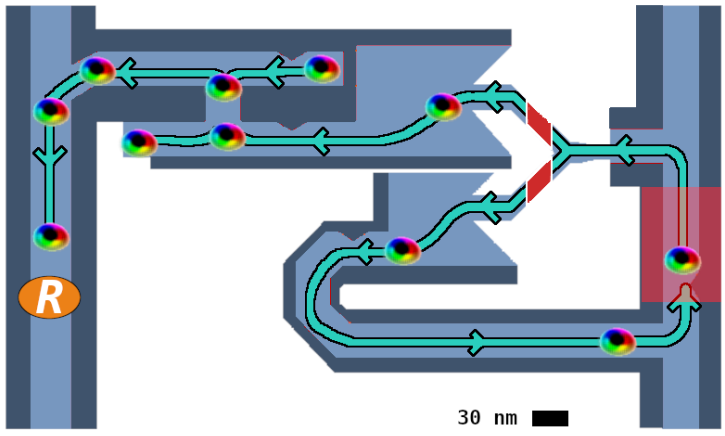
The information, encoded in the presence or absence of the skyrmion, is stored in the racetrack in a defined portion of the strip, occupied by the information only when all the bits are correctly stored. From now on we will consider this section as the basic cell of the memory. With respect to a common racetrack, the proposed design expands the capabilities of the basic cell connecting a patterned structure on one side of the track in correspondence of the zone storing the bit. The resulting structure is shown in Figure 3b.

The track shown on the left in the image is the racetrack memory. The skyrmion is stored and moves here as long as memory operations are executed. The patterned structure to process the information stored is connected on the right. The skyrmion is expected to enter in the processing zone from the top track, in correspondence to the duplication structure. Here the skyrmion, if present, is converted in a domain wall (DW)-pair to be able to duplicate the information as proposed in [4]. The domain wall reaching the fork splits in two. The two copies of the information produced, then, are converted back into skyrmions. The one on the top of the racetrack will be the input for the masking operation. The other input for the masking operation is nucleated by a write head. The skyrmion produced by duplication on the bottom is put back in place inside the racetrack memory through the track connected to the memory. Finally, the vertical track on the left in the image collects the skyrmions coming from the operation guiding it in the direction of a read head for further elaborations in the surrounding logic. The read head in the Read-Out zone is present every two cells to reduce the collection latency. An example of the complete

movement of skyrmion during the elaboration phase is reported in Figure 4. To guide the skyrmion, a set of contacts is needed for local current generation. Their position is shown in Figure 3a. The contacts in figure are activated in groups during the different phases of the elaboration. When a set of contacts is activated, the others are left floating to reduce at the minimum sneaky currents and keep the control as simple as possible. The contacts are connected to the W layer. The position of the contacts is also important for power consumption as it will be discussed in Section 7. A current density of $1 \times 10^{10}$ A m$^{-2}$ is employed to move the skyrmion inside the racetrack. To unlock the skyrmion from a notch, a current pulse of $30 \times 10^{10}$ A m$^{-2}$ and duration 0.3 ns is needed. Finally a pulse of intensity $55 \times 10^{10}$ A m$^{-2}$ is needed to convert the skyrmion from and to a domain wall pair. The duration for the two operations is 0.5 ns for the conversion from skyrmion to domain wall pair and 0.6 ns for the conversion from domain wall pair to skyrmion.




(**a**)




(**b**)

**Figure 3.** In (**a**), the contacts for current generation in the memory cell. The grey area indicates the tungsten layer. For every operation, a different set of contacts is involved to generate the required local currents to move the skyrmion. In (**b**), the path followed by a skyrmion entering in the processing zone in case a 1 is stored in the cell and the mask bit is set to 1.

### *3.1. Cell Operation*

In this section, the complete operation of a memory cell will be presented. In Figure 4, the cell operation is shown in the case a bit equal to 1 is stored inside the cell.

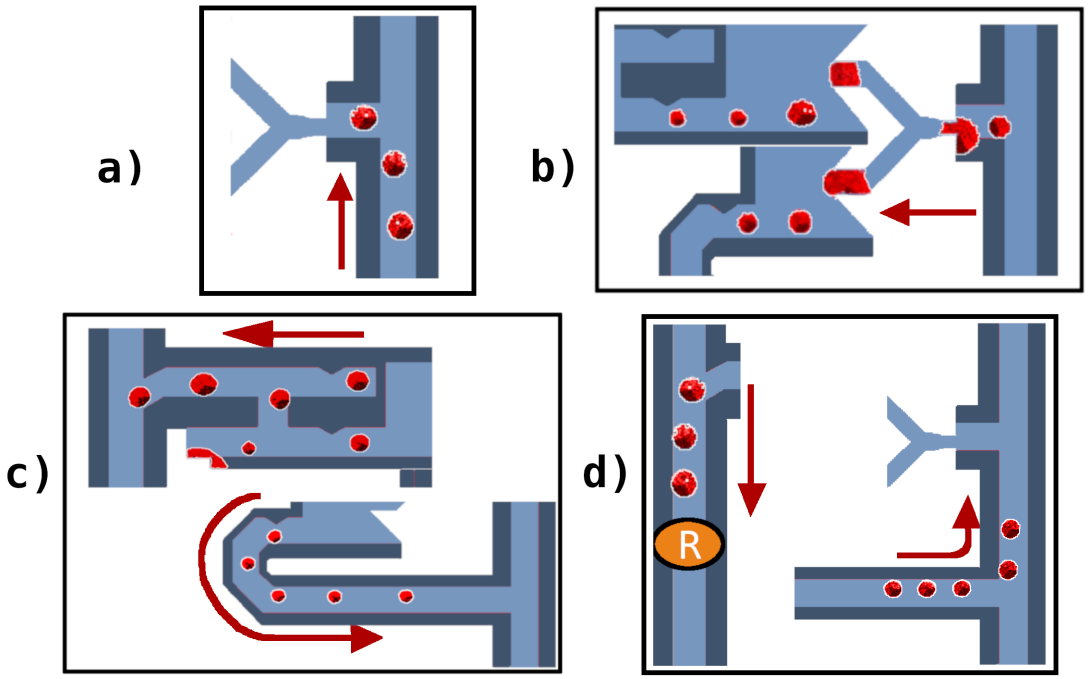



**Figure 4.** In figure the four phases of elaboration of the stored skyrmions. In (**a**) the information is moved from the memory track into the duplication gate. In (**b**) the information is duplicated by means of skyrmion-domain wall conversion. In (**c**) the information is processed on top and moved back to the memory cell on bottom. In (**d**) the result of the operation is read on the left, the information is restored into the cell on the right.

The skyrmion processing is divided in four steps:

1. To start the circuit operation, the current in the racetrack is reversed and the skyrmion can enter inside the processing zone (Figure 4a);
2. The skyrmion is guided inside the duplication element. The magnetic bubble, here, converted in a domain wall pair is duplicated. The two domain wall pairs are pushed to the edge of the constriction and then converted back into two skyrmions by means of a second current pulse (Figure 4b). Both the skyrmions are then pushed to reach the correspondent notches for synchronization. During this step another skyrmion is nucleated by the write head inside the processing zone to perform the masking operation;
3. The skyrmion in the top track is pushed over the notch to reach the AND gate to execute the masking with the input from the mask operation. The skyrmion in the bottom track enters in the return path to reach the memory cell (Figure 4c);
4. The result of the masking operation is guided through the racetrack to reach the reading head in order to be collected. The skyrmion on the bottom is put back inside the memory track (Figure 4d).

## 4. Memory Array

The modified memory array is based on the model of a racetrack memory. In the memory array, every word is memorized in a different nanotrack. As shown in Figure 5, a write head is present at the beginning of the nanotrack to nucleate the skyrmion. At least one read head is present along the track. Words are written and read sequentially in every racetrack.

To write the information inside the memory every new bit, nucleated at the write head, is shifted along the racetrack. After a complete write operation, the bits of the word are stored in precise portions of the track. To read the information stored in the track, the skyrmions need to reach a read head. The skyrmions are pushed with a short current

pulse and then shifted in the direction of read head. Here, every skyrmion is sensed as a variation in the resistance of a magnetic tunnel junction in which the ferromagnet, hosting the skyrmion, corresponds to the free layer of the reading device [25]. As shown in Figure 5, the value is compared with a reference resistance. With a single read head on the track, the required latency for a read operation is equal to $Nbit * Tcycle$. In order to reduce the latency for a complete read operation more read heads are placed on the track, as shown in Figure 5, allowing the read of multiple bits at every memory cycle. After the read operation the word needs to be shifted back in the original position in order to be ready for computation.

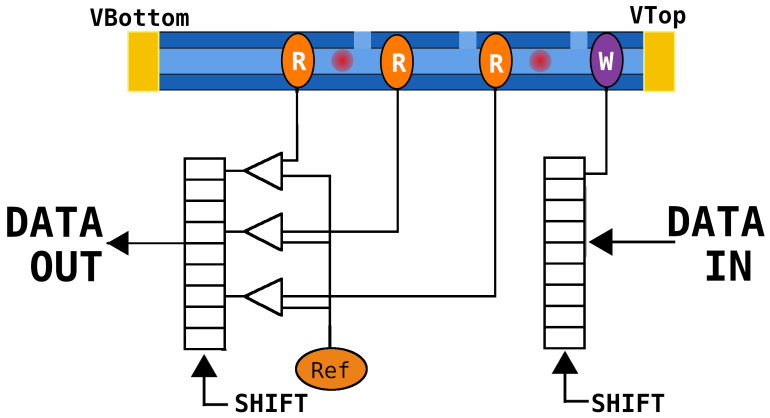



**Figure 5.** Read and write mechanism. The track has more than one read head to reduce the read operation latency. In order to sense the skyrmion presence under the read-head, the value produced is compared with a reference.

## 5. Logic in Memory

### *5.1. Maximum/Minimum Search Algorithm*

The algorithm implemented allows the identification of the maximum or the minimum value inside a set of words [26,27]. It exploits parallel operations in order to speedup the computation and eventually make the latency independent from the number of words processed. The latter, only if the resources allows such level of parallelization as the logic in memory case. The algorithm, given a set of words of length N, searches for the maximum (minimum) in the set.

Its pseudocode is reported in Algorithm 1. The first loop of the algorithm enables all the words for the computation setting the *enable* signal to 1 for all the words in the array. This matrix will signal step by step the words included in the comparison for the maximum (minimum) search. Afterwards, the main loop starts, processing sequentially all the bits of the stored words starting from the most significant bit (MSB). At line 4, the $x$ variable is evaluated to signal if the current iteration of the algorithm will be valid for maximum (minimum) evaluation or if the algorithm will continue to the next step without modifying the set of words in the comparison. In case of a maximum (minimum) search all the selected bits are equal to 0 (1), the iteration will not alter the set of words left in the comparison. Then, the enable signal will be recomputed for every word in parallel. The enable signal will continue to be 1 if the bit under examination is 1 (0) or if the $x$ variable is set to 1 meaning all 0 s (1 s) were found in computation. In every other case, the enable signal will change to 0. The algorithm will continue until the LSB is reached. The values at 1 in the enable signal indicates the maximum position. The bits at every iteration of the algorithm are selected by means of a mask applied to all words under examination. The minimum search algorithm can be derived from the one above negating the bits of the input.

**Algorithm 1:** Maximum algorithm.

**Data:** The input of the algorithm is the *bit* matrix containing all the bits stored in memory.

MAXIMUM():

**for** $i = 1$ *downto* $N_{words}$ **do**
  $\text{enable}_{N_{bit},i}=1$
**end**
**for** $j = N_{bit}$ *downto* 1 **do**
  $x_j = (bit_{j,1} \cdot enable_{j,1}) + (bit_{j,2} \cdot enable_{j,2}) + ... + (bit_{j,N_{word}} \cdot enable_{j,N_{word}})$
  **Parallel for** $k = 1$ *to* $N_{words}$ **do**
    $enable_{k,j} = (enable_{k-1,j} \cdot bit_{k,j}) + (enable_{k-1,j} \cdot (\overline{x_j}))$
  **end**
**end**
**return** $enable_{\,0}$

### 5.2. Control Logic

A complete representation of the logic in memory architecture is shown in Figure 6.

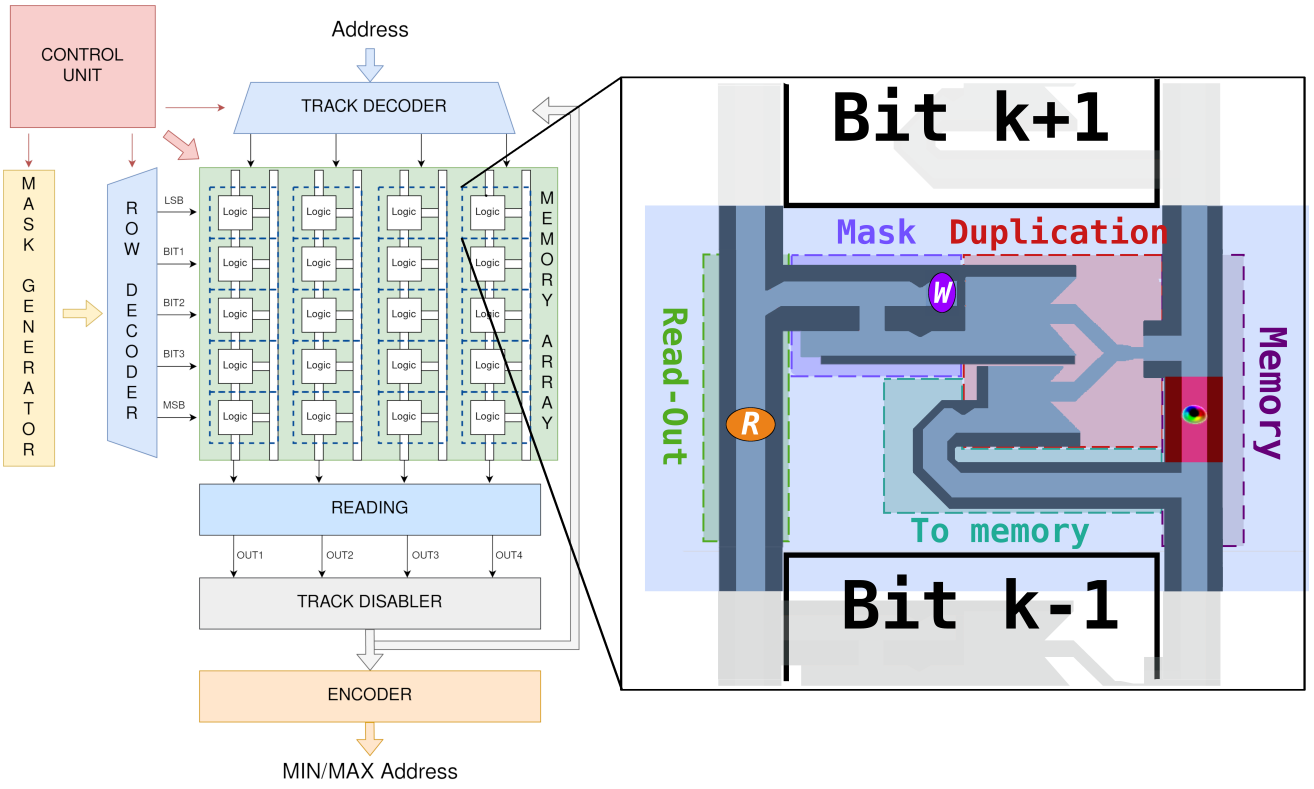



**Figure 6.** Logic in memory architecture schematic. On the left the block diagram of the complete circuit. On the right the basic cell of the array. The skyrmion is stored inside the memory in correspondence of the red dot.

The logic that surrounds the array provides the electrical interface and control for the memory array to execute the memory operations and the additional logic. The TRACK DECODER controls the word activation based on the input address or the tracks involved into the operation. In case of reading and writing, the decoder activates only one track based on the provided address, while for the maximum/minimum search, all the rows are active for operation at the same time. To also control the memory array a ROW DECODER is present. The block filters the control signals to activate only the elaboration of the required bits, avoiding useless operations on the portion of the memory not involved in

the current step of the algorithm. A MASK GENERATOR block produces the mask bits and controls the ROW DECODER. The mask generator is composed for the implemented algorithm by a shift register with the same length as the width of the stored words. A READING block is attached to the memory to read out the stored information and the masking results. The TRACK DISABLER block controls the set of rows that will continue in the next steps of the algorithms during the search operation based on the values provided by the READING block. This information is shared with the TRACK DECODER block during the search and the ENCODER for the final result. Finally all the operations are coordinated by an finite state machine (FSM) control unit. The control unit synchronizes the different blocks of the circuit and controls the generation of currents in the array to correctly guide the stored skyrmions. To keep the FSM as simple as possible, the signals generated are the same for every cell in the array. Only the cells activated by TRACK and ROW DECODER blocks will be affected by the control signals. To trigger a search operation an external signal, FIND, is connected as an input to the circuit.

The surrounding CMOS logic was described using VHDL language and synthesized using Synopsys Design Compiler software. The software was used for area, power, and timing estimations. The circuit was synthesized using the Nangate 15 nm library.

### *5.3. Maximum/Minimum Search Operation*

The search of a maximum/minimum starts when the signal FIND is activated. The mask is loaded by the MASK GENERATOR block and the processing of the word stored starts from the MSB selected by the ROW DECODER. The FSM now goes through five steps to process the information stored in memory as shown in Figure 7. The first four are executed inside the memory array in which the information stored in the cell is processed as explained in Section 3.1. When the selected bit has been processed, the results are collected by the reading unit. In the fifth step, the result of evaluation is used to select the words that will continue in the evaluation. The TRACK DISABLE based on the reading results generates the signals to the TRACK decoder which tracks what will be disabled in the next cycles. At every new repetition of the four cycle, the mask is shifted by one position. The elaboration in the memory array and the evaluation of the results is repeated for *Nbit* times. After the last evaluation, the output of the TRACK DISABLE block indicates the position of the maximum. The FSM moves into the ENCODER state. The signal of track disable is translated by the ENCODER block in the output address. Finally the FSM returns in IDLE state.

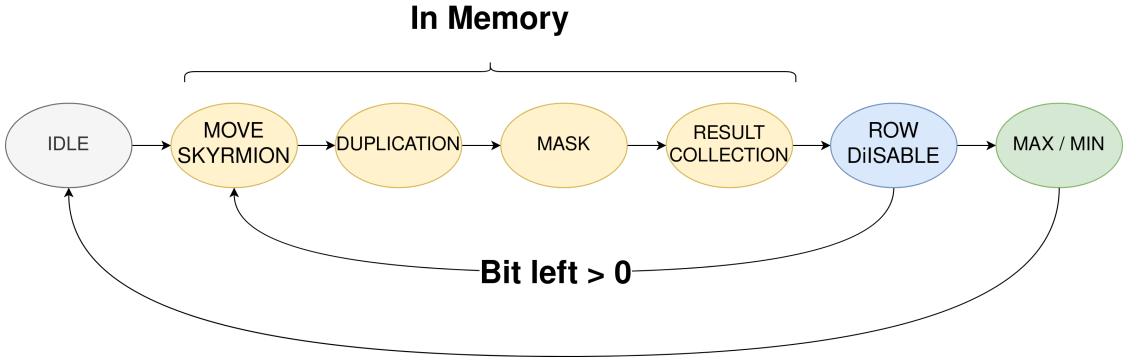


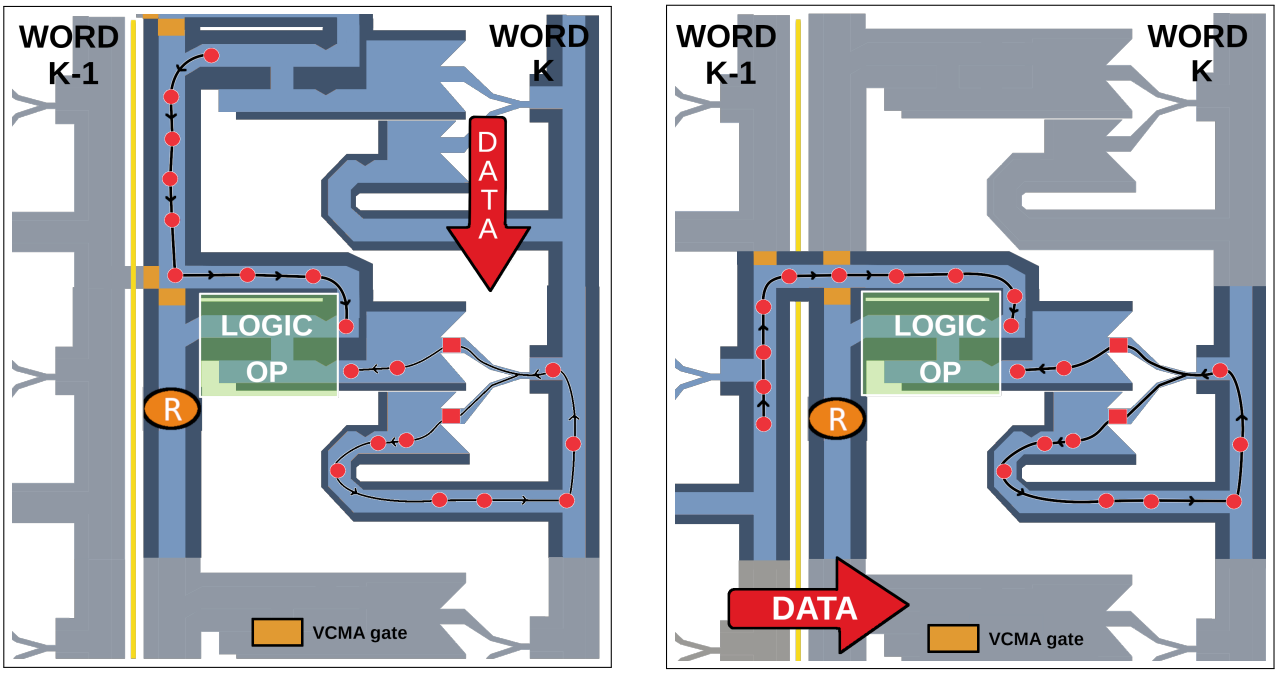

**Figure 7.** FSM state diagram for maximum/minimum search.

### *5.4. Bitwise Operations between Rows and Columns*

The cells composing the array, as shown in Figure 8, are connected in order to execute bit-wise operations between columns and rows to allow further elaboration of the stored information. In particular, the elaboration zone of each cell can be fed with the output of the cell above, Figure 8a, allowing the concatenation of Boolean operations along the column. Alternatively, the operation executed in the elaboration zone can involve the

data coming from neighbouring cells, Figure 8b, belonging to different columns allowing row-wise operations. These operations allow further elaboration of the stored information and the concatenation of logic operations for the execution of more complex algorithms.

To allow a more efficient movement between cells, VCMA gates are placed along the tracks. These gates shown in Figure 8 with orange squares, are controlled electrically and when active impose a potential barrier to the skyrmion allowing a more efficient movement under current .




(**a**) (**b**)

**Figure 8.** Logical operation between rows and columns. (**a**) Column-wise operation and (**b**) row-wise operation.

To further expand the capabilities of the presented Logic-in-Memory device, the logic section of each cell can be specialized to execute different logic operations with different gates as shown in [5] in order to allow the mapping of even more complex algorithms in memory.

## 6. Methods

The basic cells of memory array were simulated using a GPU accelerated micromagnetic simulation software, Mumax$^3$ [28,29]. The simulation discretization is set to $1 \times 1 \times 0.5\,\mathrm{nm}^3$. The material parameters used in the micromagnetic simulation are reported in Table 1. Micromagnetic simulations, as the one showed in Figure 4, were used to verify the correct behavior of the designed circuit and to estimate the current density needed to accomplish every operation.

The memory cells were then simulated with Elmer FEM software [30] with the static current module. The simulations were performed in order to analyze the current distribution in the memory cell and evaluate the power consumption during circuit operation. The basic cell was simulated with the currents required for the correct information movement and elaboration. To generate the currents, a set of contacts was placed on the cell in the positions showed in Figure 3a. The power of every operation was derived. The value of resistivity used for the evaluation is reported in Table 1. An example of the simulation results is showed in Figure 9. The figure shows the movement of the information inside the processing zone.

The surrounding logic was synthesized with Synopsys Design Compiler with different array dimensions. To extract the power consumption the array, was simulated during a maximum search.

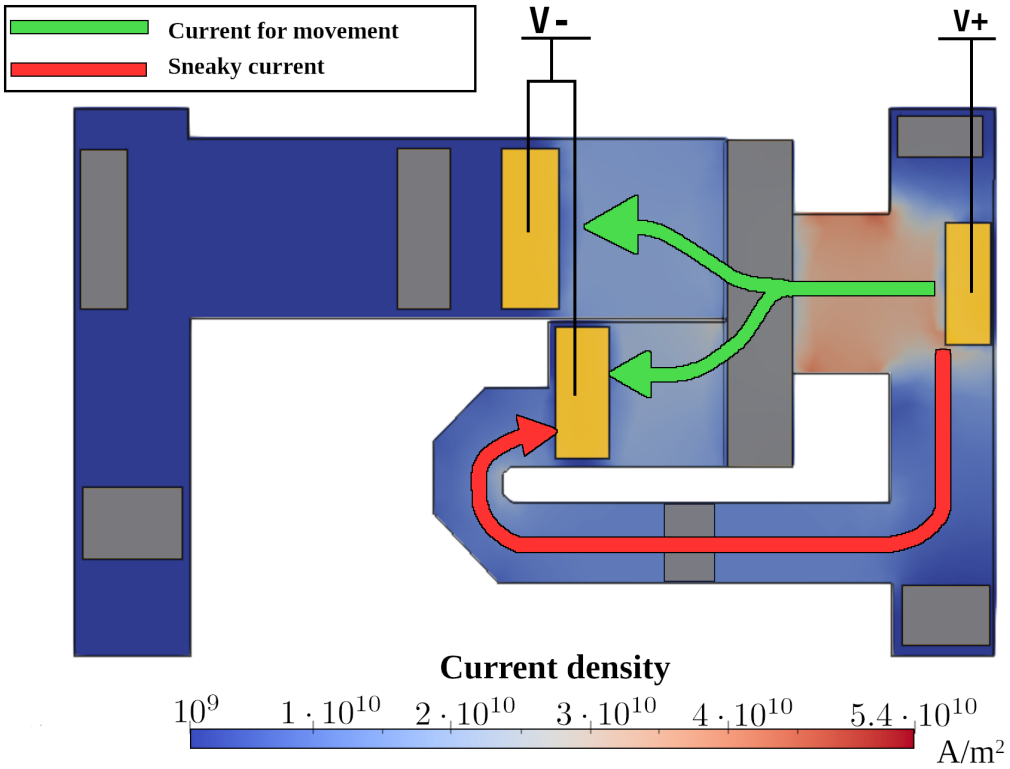



**Figure 9.** Current paths during duplication operation. The color gradient indicates the current density inside the cell. In green the current path needed to move the skyrmion. In red the additional path generated. The current flowing in this path is not needed for current operation and increase the power consumption in the operation.

## 7. Results

In this section, the performances's section of the array will be shown and compared with a CMOS in memory implementation based on a modified CAM cell presented in [27]. The performance of the designed architecture is shown in Table 2 and in Figure 10. The maximum frequency achievable by the architecture is restricted by the memory array to 285 MHz. The longest path the skyrmion has to travel limits the control logic in frequency in the explored configurations. In particular, the critical paths inside the array are represented by the path the skyrmion has to travel to return into the memory array, and the distance between consecutive cells inside the array. In the surrounding logic, the most critical component is the encoder that sets the maximum working frequency of the circuit. Compared to the reference CMOS implementation on 64 bits, it is possible to notice that the achievable frequency is higher starting from sizes of memory bigger than 32Kword. To execute the algorithm in the proposed implementation, the cycles required to process the information are equal to 5**Nbit* + 1. As shown in Table 2, the latency of a complete search operation for skyrmion implementation is lower when the memory size increases. For the logic in memory CMOS implementation, the latency reported was computed starting from the values of frequency and latency reported in the article. In comparison, the skyrmion implementation is able to achieve lower computational times for bigger memory sizes, due to the higher working frequency despite the additional two cycles required at every iteration of the basic steps.

**Table 2.** Area and timing, word width = 64 bit.

| Words | Area ($\mu m^2$) | | | Frequency (MHz) | | Latency (ns) | | |
|---|---|---|---|---|---|---|---|---|
| | Proposed | [27] | Ratio | Proposed | [27] | Proposed | [27] | Ratio |
| 2048 | 42,307 | 89,665 | 0.47 | 285 * | 1785 | 1120 | 108 | 10.30 |
| 4096 | 95,036 | 181,694 | 0.52 | 285 * | 1041 | 1120 | 183 | 6.20 |
| 8192 | 190,056 | 368,489 | 0.51 | 285 * | 571 | 1120 | 373 | 3.00 |
| 16,384 | 381,625 | 747,729 | 0.51 | 285 * | 298 | 1120 | 647 | 1.74 |
| 32,768 | 748,076 | 1,516,103 | 0.49 | 285 * | 149 | 1120 | 1293 | 0.86 |
| 65,536 | 1,427,936 | 3,045,940 | 0.46 | 285 * | 75 | 1120 | 2560 | 0.44 |

* limited by maximum array frequency.

The values of area, reported in table, were calculated by computing separately the area of the surrounding logic and area of the array and then summing the two values together. The results show that the area occupied by the circuit is half the one required for the CMOS implementation in all the simulated combinations.

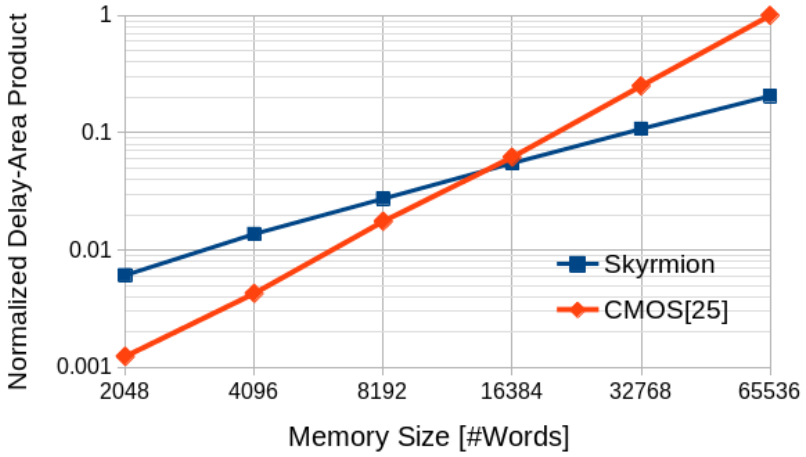



**Figure 10.** Delay area product with respect to the memory size of the proposed skyrmion implementation and CMOS [27].

Finally, the power and energy performance of the arrays were computed. In Table 3, the consumption for the different stages of the memory operations are reported. The reported values are referred to maximum search. The worst case among all simulated datasets is reported. The worst case scenario is represented by the one in which all the words remain valid during the complete algorithm. This case corresponds to the memory filled with all equal numbers. In a common case, many of the words would be discarded during the iteration of the algorithm reducing proportionally the power and energy required for operation. The main contribution to power consumption is given by the duplication phase. In this phase, the circuit, to correctly process the skyrmion, requires two current pulses. The first, generated at the beginning of the phase unlocks the movement of the skyrmion from its rest position and pushes the skyrmion toward the skyrmion domain-wall pair conversion. The second, running at half cycle, is generated to convert duplicated domain-wall pairs into skyrmions . This conversion requires a high current density equal to $55 \times 10^{10}$ A m$^{-2}$ for 0.6 ns. Sneaky currents are also an important contribution to power consumption. In the duplication phase shown in Figure 9, the V+ and V− contacts are connected not only through the fork structure used for the duplication but also by the return path that will be used later to reinsert the skyrmion into the memory. The same reasoning is valid for every other step executed in memory. The mean power consumption and the energy per bit have been calculated and are reported in Table 4.

**Table 3.** Power consumption of a single memory cell for each phase.

| Phase | Power (uW) |
|---|---|
| Move Skyrmion | 0.07 |
| Duplication | 21.67 |
| AND | 6.44 |
| Collection | 0.14 |

**Table 4.** Power and energy—Word width = 64 bit.

| Words | Power (mW) | | Energy per bit (pJ/bit) | | |
|---|---|---|---|---|---|
| | Proposed | [27] | Proposed | [27] | Ratio |
| 2048 | 34 | 25 | 327 | 20 | 16.35 |
| 4096 | 66 | 36 | 320 | 25 | 12.80 |
| 8192 | 74 | 54 | 193 | 38 | 5.00 |
| 16,384 | 133 | 85 | 177 | 52 | 1.92 |
| 32,768 | 227 | 148 | 157 | 90 | 1.74 |
| 65,536 | 574 | 270 | 153 | 164 | 0.93 |

Both the memory array and the surrounding logic contributions are taken into account. The reference case is for a 64 bit word. Comparing these values with the reference CMOS implementation, it is possible to see that CMOS implementation performs better for all the tested combinations. The different trends in energy efficiency with respect to memory dimensions, showed in Table 4, are mainly due to the fact that the frequency of operation stays stable for skyrmions while for CMOS implementation, it decreases linearly with increasing dimensions. For this reason, with dimensions greater than 32,568 words, the skyrmion implementation becomes more efficient in terms of computation. The decreasing trend in energy efficiency of the skyrmion implementation will continue until the memory array critical path sets the maximum frequency for the complete system like in the presented combinations.

## 8. Conclusions

In this paper we proposed a Logic-in-Memory architecture based on magnetic skyrmions for minimum/maximum search. The proposed circuit based on a new skyrmion memory cell is able to store information and elaborate it, maintaining at the same time the original information inside the memory. The memory can be used as a RAM memory or can perform LIM operations. The functionality of the memory array and the control logic was verified through micromagnetic simulations and hardware description language (HDL) simulation tools. The circuit proved to be small in area and to have a good energy efficiency and latency for big array dimensions with respect to a CMOS logic-in-memory reference implementation. The proposed skyrmion LIM approach could be extended and enriched to be adapted to other algorithms. The presented architecture could serve in future works as a starting framework to develop more complex LIM architectures. Future works will focus on further optimization of the structure in two aspects: (i) To explore new materials able to host skyrmions in order to optimize the performance regarding movement, dimension, and efficiency, and (ii) to further optimize the duplication mechanism section that in the presented architecture impact heavily on the overall efficiency.

**Author Contributions:** Conceptualization, L.G.; methodology, L.G. and F.R. validation, L.G.; investigation, L.G.; data analysis, L.G.; writing—original draft preparation, L.G.; writing—review and editing, L.G., F.R., M.V., M.R.R. and M.G.; visualization, L.G.; supervision, M.V., M.R.R. and M.G. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Data Availability Statement:** The data presented in this study are available on request from the corresponding author.

**Acknowledgments:** We gratefully acknowledge the support of the NVIDIA Corporation with the donation of the Titan XP GPU, which was used for this research. Moreover, we would like to acknowledge Chiara Cannavò for her preliminary contribution to the presented study.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Saulsbury, A.; Pong, F.; Nowatzyk, A. Missing the Memory Wall: The Case for Processor/Memory Integration. *SIGARCH Comput. Archit. News* **1996**, *24*, 90–101. [CrossRef]
2. Santoro, G.; Turvani, G.; Graziano, M. New logic-in-memory paradigms: An architectural and technological perspective. *Micromachines* **2019**, *10*, 368. [CrossRef] [PubMed]
3. Parkin, S.S.P.; Hayashi, M.; Thomas, L. Magnetic Domain-Wall Racetrack Memory. *Science* **2008**, *320*, 190–194. [CrossRef]
4. Zhang, X.; Ezawa, M.; Zhou, Y. Magnetic skyrmion logic gates: Conversion, duplication and merging of skyrmions. *Sci. Rep.* **2015**, *5*, 1–8.
5. Chauwin, M.; Hu, X.; Garcia-Sanchez, F.; Betrabet, N.; Paler, A.; Moutafis, C.; Friedman, J.S. Skyrmion Logic System for Large-Scale Reversible Computation. *Phys. Rev. Appl.* **2019**, *12*, 064053. [CrossRef]
6. Sampaio, J.; Cros, V.; Rohart, S.; Thiaville, A.; Fert, A. Nucleation, stability and current-induced motion of isolated magnetic skyrmions in nanostructures. *Nat. Nanotechnol.* **2013**, *8*, 839–844. [CrossRef] [PubMed]
7. Kang, W.; Zheng, C.; Huang, Y.; Zhang, X.; Zhou, Y.; Lv, W.; Zhao, W. Complementary Skyrmion Racetrack Memory With Voltage Manipulation. *IEEE Electron Device Lett.* **2016**, *37*, 924–927. [CrossRef]
8. Huang, Y.; Kang, W.; Zhang, X.; Zhou, Y.; Zhao, W. Magnetic skyrmion-based synaptic devices. *Nanotechnology* **2017**, *28*, 08LT02. [CrossRef]
9. Song, K.M.; Jeong, J.S.; Pan, B.; Zhang, X.; Xia, J.; Cha, S.; Park, T.E.; Kim, K.; Finizio, S.; Raabe, J.; et al. Skyrmion-based artificial synapses for neuromorphic computing. *Nat. Electron.* **2020**, *3*, 148–155. [CrossRef]
10. Liu, B.; Gu, S.; Chen, M.; Kang, W.; Hu, J.; Zhuge, Q.; Sha, E.H.M. An efficient racetrack memory-based Processing-in-memory architecture for convolutional neural networks. In Proceedings of the 2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC), Guangzhou, China, 12–15 December 2017; pp. 383–390.
11. Pan, Y.; Ouyang, P.; Zhao, Y.; Yin, S.; Zhang, Y.; Wei, S.; Zhao, W. A Skyrmion Racetrack Memory Based Computing In-Memory Architecture for Binary Neural Convolutional Network. In Proceedings of the 2019 on Great Lakes Symposium on VLSI, GLSVLSI'19, Washington, DC, USA, 9–11 May 2019; Association for Computing Machinery: New York, NY, USA, 2019; pp. 271–274. [CrossRef]
12. Chen, M.; Ranjan, A.; Raghunathan, A.; Roy, K. Cache Memory Design with Magnetic Skyrmions in a Long Nanotrack. *IEEE Trans. Magn.* **2019**, *55*, 1–9. [CrossRef]
13. Huang, S.X.; Chien, C.L. Extended Skyrmion Phase in Epitaxial FeGe(111) Thin Films. *Phys. Rev. Lett.* **2012**, *108*, 267201. [CrossRef] [PubMed]
14. Jiang, W.; Zhang, X.; Yu, G.; Zhang, W.; Wang, X.; Jungfleisch, M.B.; Pearson, J.E.; Cheng, X.; Heinonen, O.; Wang, K.L.; et al. Direct observation of the skyrmion Hall effect. *Nat. Phys.* **2017**, *13*, 162–169. [CrossRef]
15. Woo, S.; Litzius, K.; Krüger, B.; Im, M.Y.; Caretta, L.; Richter, K.; Mann, M.; Krone, A.; Reeve, R.M.; Weigand, M.; et al. Observation of room-temperature magnetic skyrmions and their current-driven dynamics in ultrathin metallic ferromagnets. *Nat. Mater.* **2016**, *15*, 501–506. [CrossRef] [PubMed]
16. Tomasello, R.; Martinez, E.; Zivieri, R.; Torres, L.; Carpentieri, M.; Finocchio, G. A strategy for the design of skyrmion racetrack memories. *Sci. Rep.* **2014**, *4*, 6784. [CrossRef] [PubMed]
17. Fert, A.; Cros, V.; Sampaio, J. Skyrmions on the track. *Nat. Nanotechnol.* **2013**, *8*, 152–156. [CrossRef] [PubMed]
18. Tomasello, R.; Puliafito, V.; Martinez, E.; Manchon, A.; Ricci, M.; Carpentieri, M.; Finocchio, G. Performance of synthetic antiferromagnetic racetrack memory: Domain wall versus skyrmion. *J. Phys. D Appl. Phys.* **2017**, *50*, 325302. [CrossRef]
19. Fook, H.T.; Gan, W.L.; Lew, W.S. Gateable skyrmion transport via field-induced potential barrier modulation. *Sci. Rep.* **2016**, *6*, 21099. [CrossRef]
20. Fook, H.T.; Gan, W.L.; Purnama, I.; Lew, W.S. Mitigation of magnus force in current-induced skyrmion dynamics. *IEEE Trans. Magn.* **2015**, *51*, 1–4. [CrossRef]
21. Amiri, P.K.; Wang, K.L. Voltage-controlled magnetic anisotropy in spintronic devices. In *Spin*; World Scientific: Singapore, 2012; Volume 2, p. 1240002.
22. Li, X.; Fitzell, K.; Wu, D.; Karaba, C.T.; Buditama, A.; Yu, G.; Wong, K.L.; Altieri, N.; Grezes, C.; Kioussis, N.; et al. Enhancement of voltage-controlled magnetic anisotropy through precise control of Mg insertion thickness at CoFeB | MgO interface. *Appl. Phys. Lett.* **2017**, *110*, 052401. [CrossRef]
23. Martinez, J.; Lew, W.; Gan, W.; Jalil, M. Theory of current-induced skyrmion dynamics close to a boundary. *J. Magn. Magn. Mater.* **2018**, *465*, 685–691. [CrossRef]

24. Zhang, C.; Fukami, S.; Watanabe, K.; Ohkawara, A.; DuttaGupta, S.; Sato, H.; Matsukura, F.; Ohno, H. Critical role of W deposition condition on spin-orbit torque induced magnetization switching in nanoscale W/CoFeB/MgO. *Appl. Phys. Lett.* **2016**, *109*, 192405. [CrossRef]
25. Penthorn, N.E.; Hao, X.; Wang, Z.; Huai, Y.; Jiang, H.W. Experimental Observation of Single Skyrmion Signatures in a Magnetic Tunnel Junction. *Phys. Rev. Lett.* **2019**, *122*, 257201. [CrossRef] [PubMed]
26. Vai, M.; Moy, M. Real-time maximum value determination on an easily testable VLSI architecture. *IEEE Trans. Circuits Syst. I Fundam. Theory Appl.* **1993**, *40*, 283–285. [CrossRef]
27. Vacca, M.; Tavva, Y.; Chattopadhyay, A.; Calimera, A. Logic-In-Memory Architecture For Min/Max Search. In Proceedings of the 2018 25th IEEE International Conference on Electronics, Circuits and Systems (ICECS), Bordeaux, France, 9–12 December 2018; pp. 853–856.
28. Vansteenkiste, A.; Leliaert, J.; Dvornik, M.; Helsen, M.; Garcia-Sanchez, F.; Van Waeyenberge, B. The design and verification of Mumax3. *AIP Adv.* **2014**, *4*, 107133. [CrossRef]
29. Mulkers, J.; Van Waeyenberge, B.; Milošević, M.V. Effects of spatially-engineered Dzyaloshinskii-Moriya interaction in ferromagnetic films. *Phys. Rev. B* **2017**, *95*, 144401. [CrossRef]
30. ElmerFEM. Open Source Multiphysical Simulation Software. Available online: http://www.elmerfem.org/ (accessed on 15 September 2020).