IOP Conf. Series: Materials Science and Engineering **341** (2018) 012025 doi:10.1088/1757-899X/341/1/012025

Implementation of digital equality comparator circuit on memristive memory crossbar array using material implication logic

Adib Haron, Fazren Mahdzair, Anas Luqman, Nazmie Osman and Syed Abdul Mutalib Al Junid

Electronic Architecture and Application Research Group (EArA)
Faculty of Electrical Engineering Universiti Teknologi MARA
40450 Shah Alam, Selangor, Malaysia

E-mail: adib@salam.uitm.edu.my

Abstract. One of the most significant constraints of Von Neumann architecture is the limited bandwidth between memory and processor. The cost to move data back and forth between memory and processor is considerably higher than the computation in the processor itself. This architecture significantly impacts the Big Data and data-intensive application such as DNA analysis comparison which spend most of the processing time to move data. Recently, the in-memory processing concept was proposed, which is based on the capability to perform the logic operation on the physical memory structure using a crossbar topology and non-volatile resistive-switching memristor technology. This paper proposes a scheme to map digital equality comparator circuit on memristive memory crossbar array. The 2-bit, 4-bit, 8-bit, 16-bit, 32-bit, and 64-bit of equality comparator circuit are mapped on memristive memory crossbar array by using material implication logic in a sequential and parallel method. The simulation results show that, for the 64-bit word size, the parallel mapping exhibits 2.8× better performance in total execution time than sequential mapping but has a trade-off in terms of energy consumption and area utilization. Meanwhile, the total crossbar area can be reduced by 1.2× for sequential mapping and 1.5× for parallel mapping both by using the overlapping technique.

1. Introduction

The classical CMOS-based Von Neumann architectures separate memory and processor. Fundamentally, the memory is used to store data, and the processor is used to compute data. The communication between memory and processor continues to be a bottleneck for many data-intensive applications due to the massive amount, and expensive cost of moving data back and forth between memory and processor [1, 2]. In addition, the Von Neumann architectures are developed based on the memory hierarchy that is used to overcome the problem of the growing gap between memory and processor speed. The memory that closer to the processor is faster due to the data locality that speeds up the computation but smaller in size and capacity that limits the amount of data can be handled and make it difficult to process Big Data applications [3, 4]. Meanwhile, the CMOS technology gradually scales down and intrinsically reach to its physical limit by about 2024 [5, 6]. Consequently, CMOS technology faces challenges such as reduce reliability, increase leakage power consumption and saturated performance.

The concept of in-memory processing is one of the most promising solutions that use processing within the memory architecture to alleviate the communication bottleneck in Von Neumann architecture [7–11]. The processing within memory architecture is a memristive memory crossbar architecture that can perform as memory, processor and interconnect. Thus, the memristive memory crossbar array is not only used to store data but also to compute and move data along the crossbar nanowire architecture [12, 13]. The data can be programmed as resistance, meanwhile, the logic operation and data movement are driven by the voltage driver. Therefore, there is no specific load and store instructions are required for the computation. The data can be directly computed where ever they reside within the crossbar architecture. Physically, the memristive memory crossbar array can be seen as a full memory devices without any computing element surrounded or integrated. The logic operations are performed by the voltage driver. The technology enabler for this architecture is memristor devices. [14]. Memristor is one of the promising devices that has features such as non-volatility, higher scalability, high integration density, and CMOS compatible [15].

The memristive memory crossbar array provides a huge amount of parallelism that can be exploited. The vast number of the horizontal and vertical nanowire enables massive parallelism for computation and communication. However, the parallelism is limited due to the nature of architecture that shares a single horizontal line to many vertical lines or vice versa. Therefore, either horizontal or vertical lines only can be used for parallelism at one time. In terms of logic gates operation, material implication logic is a suitable method to execute the nature of logic gates within the memristive memory crossbar array. However, most studies on implication logic operations show that the implication logic executed in sequential [7, 16]. Therefore, the total steps of computation and communication by using implication logic contributes to a long execution time and high energy consumption.

The concept of in-memory processing allows flexibility for the designer to store data on any best location within the crossbar, perform computation in parallel and manage the communication between task efficiently to achieve the best performance. The implementation of material implication logic on memristive memory crossbar array depends on the logic design of the circuit, i.e., digital equality comparator circuit. To exploit the massive parallelism on the crossbar array either on horizontal or vertical lines, the steps of the imply logic must be executed in a sequence that is equal to the number of switching energy for computation. The first question arises as to how can we parallelize the computation steps of the equality comparator circuit on memristive memory crossbar array so that we can reduce the number of steps for computation? Secondly, can we mapped the equality comparator circuit on memristive memory crossbar array so that we can decrease the area of the crossbar array used to compute the logic gates of the equality comparator circuit?

This paper proposes four design methodologies to implement the equality comparator circuit on memristive memory crossbar array. The logic gates of the equality comparator circuit are mapped on the memristive memory crossbar array in such a way the research question aforementioned can be solved. The contributions of this paper are:

- Reduced the number of steps for computation of equality comparator circuit on memristive memory crossbar array by exploiting gate-level parallelism.
- Reduced the number of memristors used to implement equality comparator circuit on memristive memory crossbar array by using material implication logic.

The rest of this paper is structured as follows. Section 2 describes the background and related work of digital equality comparator circuit, memristive memory crossbar array and material implication logic. Section 3 proposes the design methodology to implement equality comparator circuit on memristive memory crossbar array. Section 4 evaluates the performance of the proposed implementation. Finally, section 5 concludes the results.

2. Background and Related Work

This section presents first the digital equality comparator circuit and its conversion to NAND gate. Subsequently, the architecture of memristive memory crossbar array and lastly the material implication logic.

The equality comparator logic circuit is part of the computation in the pairwise DNA sequence alignment [17] and is used to compare two DNA sequence. The purpose is to find the regions of similarity that may indicate relationships between two biological sequences. Figure 1 shows the 2-bit digital equality comparator circuit using the classical XNOR and AND gates. The variety of the different types of logic gates can be converted into a NAND gate only as shown in figure 2. The conversion is important due to the simplicity of its implementation on memristive memory crossbar array using the material implication logic.

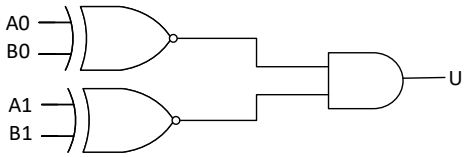

Figure 1. 2-bit equality comparator logic circuit using XNOR and AND gate.

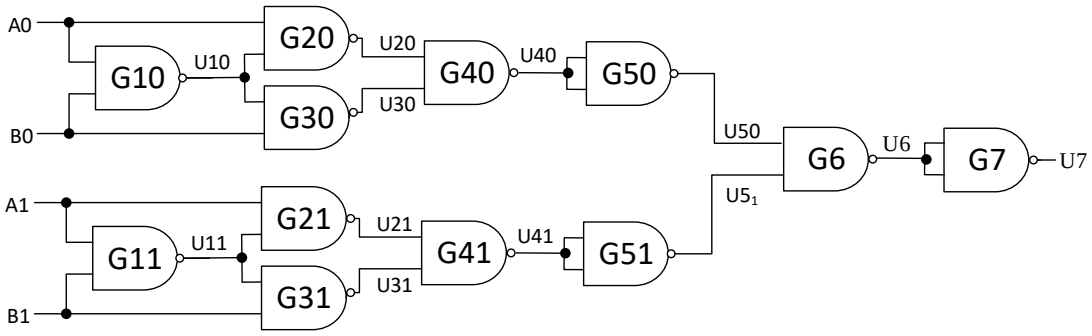

Figure 2. 2-bit equality comparator logic circuit using NAND gate.

The memristive devices are interconnected in the junction of the crossbar array architecture consists of horizontal and vertical nanowires. The crossbar array allows the resistance value of the memristive devices to be read and written at any junction in the crossbar. Therefore, each memristive device at the junction of the crossbar can perform as a memory. Section 3 illustrates several concepts of the memristive memory crossbar array architecture.

Besides, the memristive devices can behave as logic circuits in which the computation can be executed on the memory itself. One of the basic and deterministic logical element that can be used to perform computation on memristive memory crossbar array is material implication logic [9]. Figure 3 and 4 show the circuit of material implication logic. Each memristive device is used to store input, output and latch the intermediate output for the next logic operation. The memristor P is used to store input 1, and memristor Q is used to store input 2. The output is stored in the memristor S. The output position of the implication logic can be configured on the right side and the left side of the memristors following the principle of the voltage divider. The implication logic can perform only one operation at one time due to the fact that only one memristor can be switched in a horizontal or vertical line for every single voltage control applied.

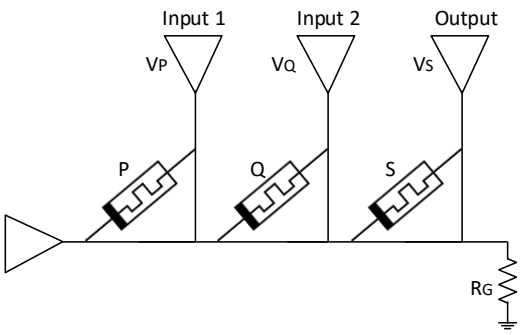

Figure 3. Right side output configuration.

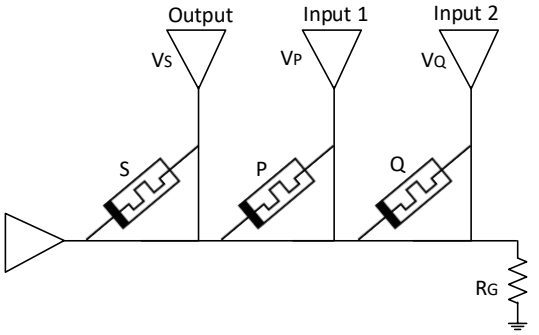

Figure 4. Left side output configuration.

3. Methodology

Currently, there is no standard way to map a set of combinational logic gates on the memristive memory crossbar array. A designer has to predetermine the location of input, intermediate output and the final output before the execution. In this section, we propose a design methodology to map the equality comparator circuit on memristive memory crossbar array architecture. The mapping scheme can be divided into four types:

- Sequential Mapping Non-Overlapping (SMNO)
- Sequential Mapping Overlapping (SMO)
- Parallel Mapping Non-Overlapping (PMNO)
- Parallel Mapping Overlapping (PMO)

The purpose of parallel over sequential technique is to reduce the number of computation steps. The parallelism exploited is gate-level parallelism. The purpose of overlapping over non-overlapping technique is to reduce the number of memristors used to compute a set of logic gates. The Non-Overlapping technique uses the right side output configuration (see figure 3) and the overlapping technique uses both the right side output configuration and left side output configuration (see figure 3 and 4) in order to reduce the number of working memristor.

Sequential Mapping Non-Overlapping (SMNO) is a naive method to execute the equality comparator circuit on the memristive memory crossbar array. Each logic gates in equality comparator are executed in sequential, and therefore each logic gates need to be labeled with a number (see figure 2). The sequential method is slow due to the fact that only one implication logic step operation can be executed at one time for a vertical or horizontal line. As a result, the number of steps for the execution time is usually high and equal to the number of memristor switching for the energy consumption. Table 1 shows the step-by-step of 2-bit sequential mapping non-overlapping with the sequence of logic gates and number of steps needed for each gate operation. The symbol $\uparrow$ and $\sim$ indicate the NAND gate and NOT gate operation respectively. Figure 5 shows the arrangement of memristors that are responsible for storing input data, output data, and the intermediate data.

Table 1. 2-bit sequential mapping non-overlapping.

Sequence	Logic Gate	Memristor Mapping	Number of steps
1	G10	$A0 \uparrow B0 \rightarrow U10$	3
2	G20	$A0 \uparrow U10 \rightarrow U20$	3
3	G30	$U10 \uparrow B0 \rightarrow U30$	3
4	G40	$U20 \uparrow U30 \rightarrow U40$	3
5	G50	$\sim U40 \rightarrow U50$	2
6	G11	$A1 \uparrow B1 \rightarrow U11$	3
7	G21	$A1 \uparrow U11 \rightarrow U21$	3
8	G31	$U11 \uparrow B1 \rightarrow U31$	3
9	G41	$U21 \uparrow U31 \rightarrow U41$	3
10	G51	$\sim U41 \rightarrow U51$	2
11	G6	$U50 \uparrow U51 \rightarrow U6$	3
12	G7	$\sim U6 \rightarrow U7$	2

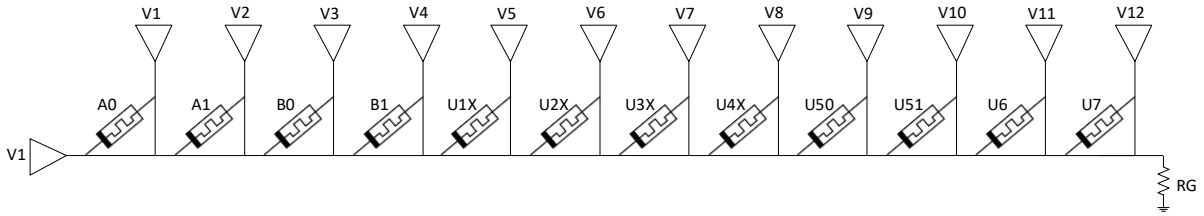

Figure 5. 2-bit sequential mapping non-overlapping on memristor-based crossbar array.

Sequential Mapping Overlapping (SMO) is a method to improve the Sequential Mapping Non-Overlapping in terms of the utilization of the number of memristors. This method is capable of reducing the number of memristors whereby the same physical memristor can be reused to store the intermediate output for the next operation so that the intermediate output of the previous operation is overwritten. The process requires the relocation of the output position as shown in figure 3 and 4. Table 2 shows the step-by-step of 2-bit Sequential Mapping Overlapping and the arrangement of memristors are illustrated in figure 6. The arrangement and location of data input and output of SMO are similar to SMNO except that the memristor U6 and U4X can be eliminated. Therefore, the overall number of memristors of SMO can be reduced when compared with SMNO.

Table 2. 2-bit sequential mapping overlapping.

Sequence	Logic Gate	Memristor Mapping	Number of steps
1	G10	$A0 \uparrow B0 \rightarrow U10$	3
2	G20	$A0 \uparrow U10 \rightarrow U20$	3
3	G30	$U10 \uparrow B0 \rightarrow U30$	3
4	G40	$U20 \uparrow U30 \rightarrow U10$	3
5	G50	$\sim U10 \rightarrow U50$	2
6	G11	$A1 \uparrow B1 \rightarrow U11$	3
7	G21	$A1 \uparrow U1 \rightarrow U21$	3
8	G31	$U11 \uparrow B1 \rightarrow U31$	3
9	G41	$U21 \uparrow U31 \rightarrow U11$	3
10	G51	$\sim U11 \rightarrow U51$	2
11	G6	$U50 \uparrow U51 \rightarrow U11$	3
12	G7	$\sim U11 \rightarrow U7$	2

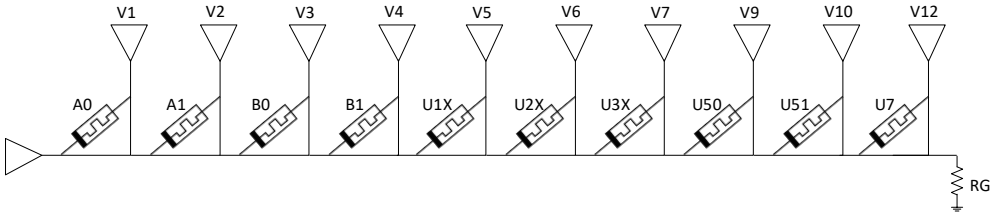

Figure 6. 2-bit sequential mapping overlapping on memristor-based crossbar array.

Parallel Mapping Non-Overlapping (PMNO) is a mapping method that developed to improve the execution time of Sequential Mapping Non-Overlapping. The parallelism is achieved due to the ability to switch multiple memristors by a single voltage control in horizontal or vertical line. Table 3 shows the step-by-step of 2-bit parallel mapping non-overlapping. The total number of steps of sequences for PMNO are eight which is less than SMNO that required twelve steps of sequences. In sequence 6, we need to execute the communication operation or moving the particular data downward, and the rest are computation operations. As shown in figure 7, the horizontal line is used for computation, and the vertical line is used for communication. We arrange the input data in the vertical line instead of the horizontal line, i.e., A0, A1 and B0, B1. The voltage driver V1 and V2 are able to perform the logic operation for both A0, A1 and B0, B1 in parallel. The final result from the first vertical line, i.e., U50 is transferred to the memristor U50'. Then, the second vertical line performs the computation until the final result in U7.

Table 3. 2-bit parallel mapping non-overlapping.

Sequence	Logic Gate	Memristor Maping	Logic Gate	Memristor Mapping	Steps
1	G10	$A0 \uparrow B0 \rightarrow U10$	G11	$A1 \uparrow B1 \rightarrow U11$	3
2	G20	$A0 \uparrow U10 \rightarrow U20$	G21	$A1 \uparrow U11 \rightarrow U21$	3
3	G30	$U10 \uparrow B0 \rightarrow U30$	G31	$U11 \uparrow B1 \rightarrow U31$	3
4	G40	$U20 \uparrow U30 \rightarrow U40$	G41	$U21 \uparrow U31 \rightarrow U41$	3
5	G50	$\sim U40 \rightarrow U50$	G51	$\sim U41 \rightarrow U51$	2
6	——Communication Steps : U50 move downward inline with U51——				3
7	G6	$U50 \uparrow U51 \rightarrow U6$			3
8	G7	$\sim U6 \rightarrow U7$			2

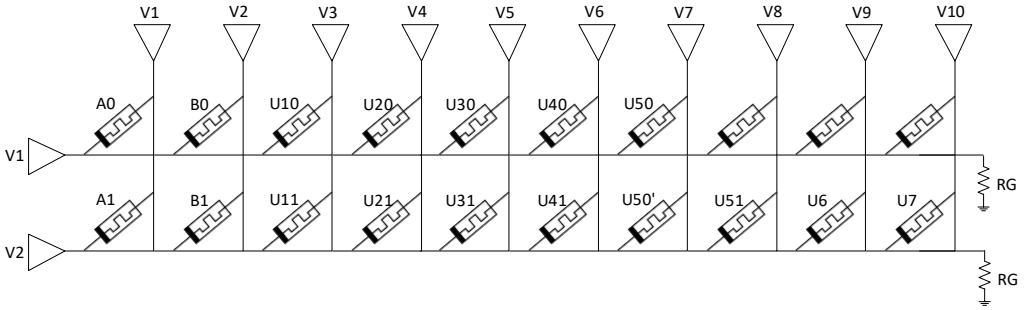

Figure 7. 2-bit parallel mapping non-overlapping on memristor-based crossbar array.

Parallel Mapping Overlapping (PMO) is a method that used to achieve both; reduce the number of steps computation and reduce the number of memristors by using the parallel and overlapping techniques. The PMO method is the most efficient method than all other methods previously. Table 4 shows the step-by-step of sequences of 2-bit parallel mapping overlapping and the memristors arrangement and data locality are shown in figure 8. All of the mapping techniques applied to 2-bit can be extended to 4-bit, 8-bit, 16-bit, 32-bit and 64-bit word size.

Table 4. 2-bit parallel mapping overlapping.

Sequence	Logic Gate	Memristor Maping	Logic Gate	Memristor Mapping	Steps
1	G10	$A0 \uparrow B0 \rightarrow U10$	G11	$A1 \uparrow B1 \rightarrow U11$	3
2	G20	$A0 \uparrow U10 \rightarrow U20$	G21	$A1 \uparrow U11 \rightarrow U21$	3
3	G30	$U10 \uparrow B0 \rightarrow U30$	G31	$U11 \uparrow B1 \rightarrow U31$	3
4	G40	$U20 \uparrow U30 \rightarrow U10$	G41	$U21 \uparrow U31 \rightarrow U11$	3
5	G50	$\sim U10 \rightarrow U50$	G51	$\sim U11 \rightarrow U51$	2
6	——Communication Steps : U50 move downward inline with U51——				3
7	G6	$U50 \uparrow U51 \rightarrow U11$			3
8	G7	$\sim U11 \rightarrow U7$			2

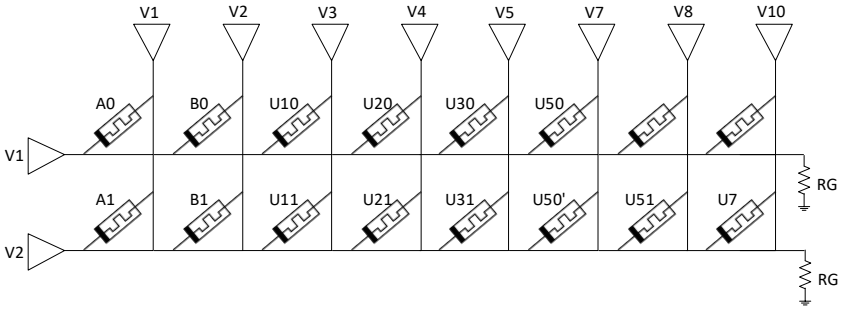

Figure 8. 2-bit parallel mapping overlapping on memristor-based crossbar array.

4. Simulation Result and Evaluation

In this section, we evaluate all the mapping schemes in section 3. We quantify the number of steps and number of switching for both computation and communication operations, as well as the number of memristors. The number of steps is used to evaluate the execution time (second), the number of switching is used to evaluate the energy consumption (joule), and lastly, the number of memristors is used to evaluate the area utilization (meter square). The number of steps and switching for both computation and communication are shown in equation 1 and 2 for sequential and parallel mapping respectively. Equations 3,4,5,6 shows the number of memristor used in SMNO, SMO, PMNO, and PMO respectively. The n in all equation is referred to 2-bit, 4-bit, 8-bit, 16-bit, 32-bit and 64-bit and tabulated in table 5, 6, 7 and 8. The cost of the delay of a single switching memristor is 200 *picosecond*, the energy of a single switching memristor is 0.25 *femtojoule* and finally, the area of one memristor is $1{\times}10^{-4}$ $u\text{m}^2$. Subsequently, the performances are plotted as a graph in figure 9, 10, and 11.

$$sm = 14 * n + 5 * (n - 1) \tag{1}$$

$$pm = 14 + 5 * (n - 1) + 3 * n/2 \tag{2}$$

$$smno = 4 + 3 * n + 2 * (n - 1) \tag{3}$$

$$smo = 3 + 3 * n + (n - 1) \tag{4}$$

$$pmno = n * (4 + n + 2 * (n - 1)) \tag{5}$$

$$pmo = n * (3 + n + (n - 1)) \tag{6}$$

Table 5 shows the sequential mapping non-overlapping performance metric from 2-bit to 64-bit. The number of steps and switching for computation and the number of memristors increase in parallel with the number of bits or word size. The number of steps and switching for computation are equal due to the sequential steps executed by the implication logic. Meanwhile, the number of steps and switching for communication is zero because the data movement operations in sequential mapping are not required.

Table 5. Sequential mapping non-overlapping.

Number of bits	2-bit	4-bit	8-bit	16-bit	32-bit	64-bit
Number of steps computation	33	71	147	299	603	1211
Number of steps communication	0	0	0	0	0	0
Number of switching computation	33	71	147	299	603	1211
Number of switching communication	0	0	0	0	0	0
Number of memristors	12	22	42	82	162	322

Table 6 tabulates the sequential mapping overlapping performance metrics from 2-bit to 64-bit. The number of steps and switching both for computation and communication SMO are equal to SMNO. However, the number of memristors used in SMO are less than SMNO due to the overlapping technique.

Table 6. Sequential mapping overlapping.

Number of bits	2-bit	4-bit	8-bit	16-bit	32-bit	64-bit
Number of steps computation	33	71	147	299	603	1211
Number of steps communication	0	0	0	0	0	0
Number of switching computation	33	71	147	299	603	1211
Number of switching communication	0	0	0	0	0	0
Number of memristors	10	18	34	66	130	258

Table 7 shows the parallel mapping non-overlapping performance metric from 2-bit to 64-bit. The number of steps for computation in PMNO is less than the number of steps for computation in SMNO due to the exploitation of parallelism at gate-level. However, the number of switching for computation in PMNO is equal to the SMNO due to the similar design of logic gate in the equal comparator. Moreover, in PMNO, the data movement in vertical line contribute to the number of steps and switching for communication.

Table 7. Parallel mapping non-overlapping.

Number of bits	2-bit	4-bit	8-bit	16-bit	32-bit	64-bit
Number of steps computation	19	29	49	89	169	329
Number of steps communication	3	6	12	24	48	96
Number of switching computation	33	71	147	299	603	1211
Number of switching communication	3	6	12	24	48	96
Number of memristors	16	56	208	800	3136	12416

Table 8 shows the parallel mapping overlapping performance metric from 2-bit to 64-bit.The number of steps for computation and communication in PMO are equal to PMNO. Similarly,

the number of switching for computation and communication has the same value for both for PMO and PMNO due to the sequential steps in communication. Note that the communication or data movement for PMNO and PMO are executed in sequential. However, the number of memristors in PMO is less than PMNO due to the overlapping technique applied.

Table 8. Parallel mapping overlapping.

Number of bits	2-bit	4-bit	8-bit	16-bit	32-bit	64-bit
Number of steps computation	19	29	49	89	169	329
Number of steps communication	3	6	12	24	48	96
Number of switching computation	33	71	147	299	603	1211
Number of switching communication	3	6	12	24	48	96
Number of memristors	12	40	144	544	2112	8320

Figure 9 illustrates the performance of execution time for all mapping schemes. The sequential mapping, i.e., SMNO and SMO both has the same execution time due to the sequential mapping technique. The parallel mapping, i.e., PMNO and PMO are equal in execution time due to the parallel mapping technique. On average, from 2-bit to 64-bit, the PMNO exhibits 2.4× better execution time than SMNO. The result is also similar for PMO over SMO.

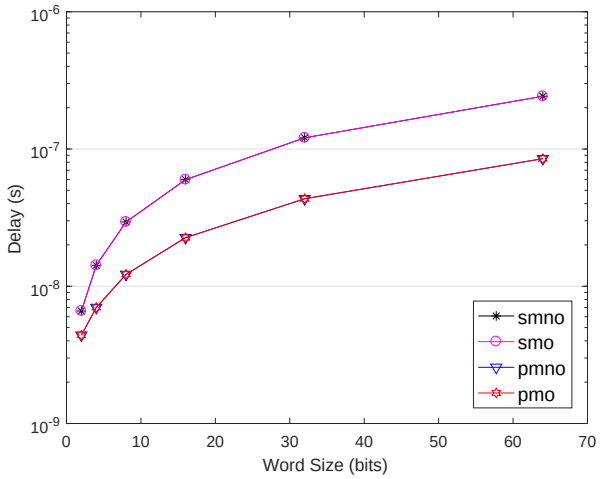

Figure 9. Execution time.

Figure 10 illustrates the energy consumption for all mapping schemes. The parallel mapping exhibits slightly higher energy consumption than the sequential mapping due to the communication or data movement cost. On average, parallel mapping consumes 1.08× higher energy than sequential mapping.

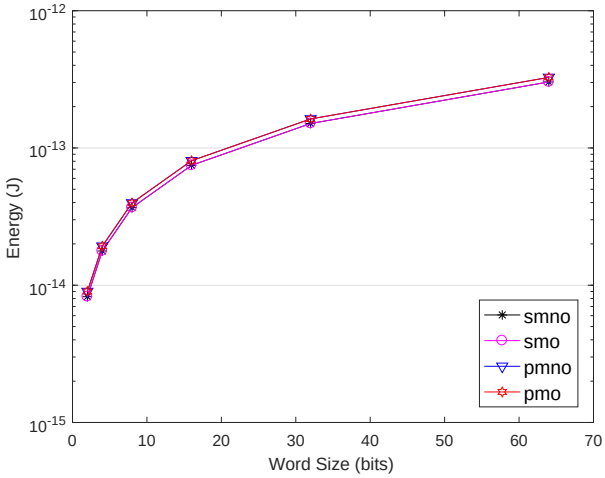

Figure 10. Energy consumption.

Figure 11 illustrates the area utilization for all mapping schemes. The parallel mapping schemes perform better in execution time than sequential mapping schemes. As a trade-off, the parallel mapping non-overlapping consumes 12.8× larger area than the sequential mapping non-overlapping in average. Meanwhile, the parallel mapping overlapping consumes 10.7× larger area than sequential mapping overlapping in average.

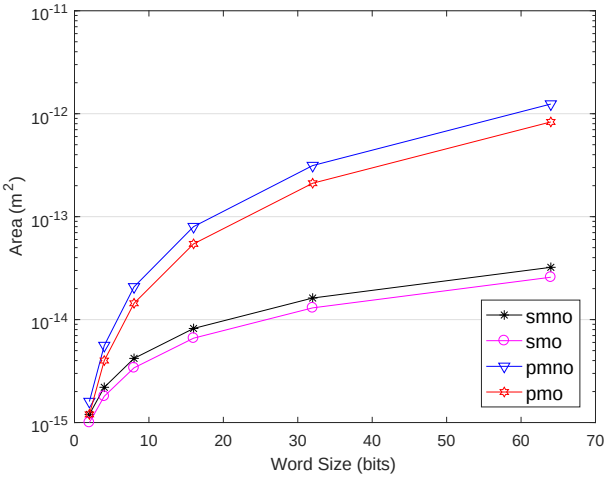

Figure 11. Area Utilization.

5. Conclusion

We show that we can reduce the amount of execution time by using the gate-level parallelism technique and the amount of area utilization by using the overlapping technique for all 2-bit, 4-bit, 6-bit, 8-bit, 16-bit, 32-bit, 64-bit of digital equality comparator circuit on memristive memory crossbar arrray architecture.

Acknowledgments
This work was funded by a research grant from the LESTARI 2017 under the file NO 600-IRMI/DANA KCM 5/3/LESTARI (128/2017). The authors would like to thank to the Institute of Research Management & Innovation (IRMI), Universiti Teknologi MARA for the funding of this research.

References

[1] Al Junid S, Tahir N, Haron M, Abd Majid Z, Idros M and Osman F 2010 *International Journal of Simulation–Systems, Science & Technology* **11**
[2] Saliman N F A, Sabri N D A, Al Junid S A M, Tahir N M and Majid Z A 2013 *Systems, Process & Control (ICSPC), 2013 IEEE Conference on* (IEEE) pp 268–274
[3] Idros M, Ali S and Islam M S 2012 *Intelligent Systems, Modelling and Simulation (ISMS), 2012 Third International Conference on* (IEEE) pp 278–282
[4] Idros M, Ali S and Islam M S 2011 *Research and Development (SCOReD), 2011 IEEE Student Conference on* (IEEE) pp 225–228
[5] Borkar S 2009 *Proceedings of the 46th Annual Design Automation Conference* (ACM) pp 93–94
[6] Author et al 2016 MORE MOORE Tech. rep. International Roadmap for Devices and Systems
[7] Kvatinsky S, Satat G, Wald N, Friedman E G, Kolodny A and Weiser U C 2014 *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* **22** 2054–2066
[8] Levy Y, Bruck J, Cassuto Y, Friedman E G, Kolodny A, Yaakobi E and Kvatinsky S 2014 *Microelectronics Journal* **45** 1429–1437
[9] Borghetti J, Snider G S, Kuekes P J, Yang J J, Stewart D R and Williams R S 2010 *Nature* **464** 873–876
[10] Kvatinsky S, Belousov D, Liman S, Satat G, Wald N, Friedman E G, Kolodny A and Weiser U C 2014 *IEEE Transactions on Circuits and Systems II: Express Briefs* **61** 895–899
[11] Linn E, Rosezin R, Tappertzhofen S, Böttger U and Waser R 2012 *Nanotechnology* **23** 305205
[12] Xie L, Du Nguyen H A, Taouil M, Hamdioui S and Bertels K 2015 *Nanoscale Architectures (NANOARCH), 2015 IEEE/ACM International Symposium on* (IEEE) pp 124–129
[13] Du Nguyen H, Xie L, Yu J, Taouil M and Hamdioui S 2017 *Design & Technology of Integrated Systems In Nanoscale Era (DTIS), 2017 12th International Conference on* (IEEE) pp 1–6
[14] Hamdioui S, Taouil M, Du Nguyen H A, Haron A, Xie L and Bertels K 2015 *Memristive Systems (MEMRISYS) 2015 International Conference on* (IEEE) pp 1–3
[15] ITRS Emerging Research Devices (ERD) report URL `http://www.itrs.net`
[16] Kvatinsky S, Kolodny A, Weiser U C and Friedman E G 2011 *Computer Design (ICCD), 2011 IEEE 29th International Conference on* (IEEE) pp 142–147
[17] Al Junid S A M, Tahir N M, Majid Z A and Idros M F M 2012 *Intelligent Systems, Modelling and Simulation (ISMS), 2012 Third International Conference on* (IEEE) pp 187–190