Design and Implementation of a Low-Power Memristor-Based Piccolo-80 Lightweight Encryption Algorithm Using VTM Logic Gates

Journal:	*IEEE Canadian Journal of Electrical and Computer Engineering*
Manuscript ID	CJECE-OA-2025-Jul-160.R1
Manuscript Type:	Original Article
Date Submitted by the Author:	03-Oct-2025
Complete List of Authors:	Mozafari, Farzad; University of Windsor Department of Electrical and Computer Engineering Ahmadi, Majid; University of Windsor Department of Electrical and Computer Engineering
Area of Research (select one area from the list below):	Computers - Comp
Keywords:	Cryptography, Digital circuits, Hardware design languages
Abstract:	**Lightweight cryptography has become increasingly critical for ensuring secure communication in energy-constrained Internet of Things (IoT) systems. Memristor-based architecture provides a promising approach for secure communication in energy-sensitive and hardware-constrained applications. Piccolo is a lightweight encryption algorithm that offers high security while enabling compact hardware implementation. Additionally, Piccolo is specifically designed to operate efficiently in resource-limited environments, making it a strong candidate for low-energy applications such as IoT devices. However, earlier implementations of the Piccolo algorithm on FPGA platforms, CMOS, and hybrid MeMOS (Memristor-CMOS) technology have faced challenges with high power consumption, hardware overhead, and limited scalability. This paper presents a novel architecture for implementing the Piccolo-80 encryption algorithm using the VTM (Voltage-to-Memristance) approach, in which the design maps Piccolo's primary operations onto VTM stateful logic gates. This enhances performance, reduces switching activity, and leverages the non-volatile properties of memristors. The proposed design introduces VTM-based memristor logic gates that significantly reduce hardware complexity and power consumption compared to previous implementations. The results from comparing CMOS and hybrid MeMOS implementations in terms of area and energy consumption demonstrate that hardware implementation of Piccolo's lightweight algorithm using the VTM approach not only improves energy efficiency but also enables the design of optimized, low-power circuits. The design achieves a power consumption of 17.4 mW at 1.8 V and 133 MHz, with only 1214**

	Gate Equivalents (GEs), reducing power by up to 32% and area by nearly 20% compared to state-of-the-art hybrid MeMOS designs.

CJECE-October 2025

Design and Implementation of a Low-Power Memristor-Based Piccolo-80 Lightweight Encryption Algorithm Using VTM Logic Gates

Farzad Mozafari, *Member, IEEE*, Majid Ahmadi, *Life Fellow, IEEE*

***Abstract*—Lightweight cryptography has become increasingly critical for ensuring secure communication in energy-constrained Internet of Things (IoT) systems. Memristor-based architecture provides a promising approach for secure communication in energy-sensitive and hardware-constrained applications. Piccolo is a lightweight encryption algorithm that offers high security while enabling compact hardware implementation. Additionally, Piccolo is specifically designed to operate efficiently in resource-limited environments, making it a strong candidate for low-energy applications such as IoT devices. However, earlier implementations of the Piccolo algorithm on FPGA platforms, CMOS, and hybrid MeMOS (Memristor-CMOS) technology have faced challenges with high power consumption, hardware overhead, and limited scalability. This paper presents a novel architecture for implementing the Piccolo-80 encryption algorithm using the VTM (Voltage-to-Memristance) approach, in which the design maps Piccolo's primary operations onto VTM stateful logic gates. This enhances performance, reduces switching activity, and leverages the non-volatile properties of memristors. The proposed design introduces VTM-based memristor logic gates that significantly reduce hardware complexity and power consumption compared to previous implementations. The results from comparing CMOS and hybrid MeMOS implementations in terms of area and energy consumption demonstrate that hardware implementation of Piccolo's lightweight algorithm using the VTM approach not only improves energy efficiency but also enables the design of optimized, low-power circuits. The design achieves a power consumption of 17.4 mW at 1.8 V and 133 MHz, with only 1214 Gate Equivalents (GEs), reducing power by up to 32% and area by nearly 20% compared to state-of-the-art hybrid MeMOS designs.**

***Index Terms* — Memristor-based Hardware Implementation of Piccolo-80 Algorithm, Internet of Things (IoT), Lightweight Encryption, Voltage to Memristance (VTM).**

I. Introduction

In recent years, rapid technological advances and the widespread use of modern electronic devices, embedded systems, and the IoT [1] have increased the need for lightweight and high-performance ciphers. In 2023, NIST selected the Ascon family as the winner of the Lightweight Cryptography (LWC) competition, and it was later formalized in NIST SP 800-232 (2025). The standard includes authenticated encryption, hashing, and extendable-output functions (XOFs), emphasizing simplicity and efficiency for constrained platforms [2]. Following this, hardware implementations of Ascon have been widely explored. Khan et al. developed ASIC accelerators for Ascon-128 and Ascon-128a at a 32-nm technology node, comparing loop-folded, loop-unrolled, and fully unrolled architectures. They showed a clear trade-off: loop-folded designs save area, while fully unrolled ones maximize throughput at a higher cost [3, 4]. Alharbi et al. extended this work with FPGA implementations on Xilinx 7-series devices, where an iterative FSM-based design with buffer-driven datapaths improved frequency while maintaining modest power and balanced area–performance, confirming Ascon's practicality for IoT systems [5].

Beyond CMOS, emerging memory technologies have been proposed to reduce energy consumption and minimize data movement. Siddiqi et al. introduced a memristor-based GIFT-128 using RRAM crossbars, enabling each round with a single read, cutting energy use, and supporting reconfigurable S-boxes for side-channel protection [6]. Ajmi et al. proposed similar in-memory AES designs, executing operations directly within memory arrays to improve efficiency [7]. Other efforts target substitution boxes (S-boxes), the nonlinear core of lightweight ciphers. Penumalli et al. designed an 8T-SRAM compute-in-memory S-box that reduces area and improves DPA resistance. At the same time, Dutra e Silva Jr. et al. developed chaos-based S-boxes that replace static tables with chaotic mappings, reducing storage and adding strong

Date on submitted paper: Oct 3, 2025. Farzad Mozafari and Majid Ahmadi are with the Electrical and Computer Engineering (ECE) Department, University of Windsor (e-mail: mozafarf@uwindsor.ca, ahmadi@uwindsor.ca). "This work was supported in part by the ECE Department, University of Windsor, Windsor, ON N9B 3P4, Canada."

Corresponding author: Farzad Mozafari, Second Author: *Majid Ahmadi. Farzad Mozafari is a PhD student at the Department of ECE, University of Windsor, Windsor, Ontario, Canada. Majid Ahmadi is a Professor of ECE, University of Windsor, Ontario, Canada, ahmadi@uwindsor.ca.*

nonlinearity [8, 9]. In 2023, potential weaknesses of the Piccolo cipher against Differential Power Analysis (DPA) were identified, leading to the development of a new protection scheme to enhance the algorithm's resistance to DPA [10].

Rashidi has contributed extensively to lightweight block cipher hardware for FPGA, CMOS, and ASIC platforms. In 2019, FPGA implementations of HIGHT and PRESENT were optimized through gate-level improvements, enhancing performance and side-channel resistance on Virtex-5 and Spartan-3 devices [11]. That year, high-throughput ASIC architectures for SIMON and SPECK in 180 nm CMOS applied Sklansky adders and tree-structured XOR logic to reduce delay, raise frequency, and scale across block and key sizes [12]. In 2020, this work expanded to CLEFIA and PRINCE, where logic simplification and composite-field methods improved CLEFIA's area and efficiency. PRINCE utilized resource-shared S-boxes, a two-cycle architecture, and optimized linear operations to reduce delay and area while enhancing throughput for RFID and latency-sensitive applications [13, 14]. Later designs introduced flexible CMOS hardware for PRESENT, SIMON, and LED, utilizing optimized S-boxes, efficient MixColumns, and resource-sharing techniques to support multiple block and key sizes. The Camellia design in 180 nm CMOS applied composite-field inversion and resource sharing to reduce cost while maintaining strong throughput-to-area ratios [15, 16]. More recently, glitches in CMOS implementations of PRESENT, HIGHT, and SPECK were addressed through a filter circuit that stored intermediate data, eliminating unwanted switching activity with minimal area overhead and improving reliability and power stability without altering the algorithms [17].

Although many lightweight ciphers have been developed, there remains a need for modular algorithms that achieve efficient performance on resource-constrained devices. The Piccolo encryption algorithm [18] was specifically designed to address the needs of embedded systems, IoT devices, and other lightweight platforms with hardware and energy limitations. Its main feature is a simple, modular design, making it suitable for hardware implementation in embedded applications [19]. A key advantage of Piccolo is its use of a Generalized Feistel Network (GFN) architecture combined with a Substitution-Permutation Network (SPN) [20], which reduces design complexity while maintaining high security. Early efforts to implement Piccolo-80 in hardware focused on FPGAs [21] and their modular structure enable the efficient utilization of FPGA resources. Ramu et al. (2019) proposed a new architecture for Piccolo-80 designed for high-speed Radio Frequency Identification (RFID) security applications [22]. Their implementation demonstrated Piccolo-80's suitability for constrained environments by achieving competitive throughput and area efficiency. They showed that the algorithm can deliver high performance with minimal resources, utilizing simple key scheduling and parallel processing. Later efforts focused on improving the area efficiency of Piccolo-80. In 2021, Mishra et al. presented an architectural study that significantly reduced the number of logic gates required [23], making it more practical for devices with limited space. The researchers focused on techniques such as logic optimization, data path width reduction, and efficient use of combinational circuits to reach this goal. Recently, researchers have explored the potential of emerging technologies, such as memristors, to further enhance the hardware performance of Piccolo-80. In 2019, Masoumi implemented the Piccolo-80 algorithm in hardware using a hybrid MeMOS architecture, evaluating performance based on resource use and power efficiency [24]. While previous studies have mainly focused on architectural and logic-level optimizations of Piccolo-80 in conventional CMOS, FPGA, and hybrid MeMOS technologies, these implementations still face significant challenges, including high dynamic power consumption, increased circuit area, additional hardware overhead, limited scalability, and increased vulnerability to side-channel attacks. These limitations make them less suitable for energy-constrained environments, such as IoT devices, and highlight the need for novel hardware solutions. This work introduces a new memristor-based design for Piccolo-80 using the VTM method. By utilizing the non-volatile nature of memristors, the proposed architecture enables stateful single-step operations, reduces switching activity, features compact logic components, and consumes less power. The entire round function, including the S-box and MixColumn operations, is implemented entirely with VTM stateful logic gates. This architecture addresses the power, area, and scalability challenges of earlier designs, highlighting the potential of emerging memristor technologies for secure, energy-efficient IoT applications.

The paper is organized as follows: Section II provides a brief overview of the Piccolo-80 cipher's structure, Section III explains the design and implementation of the Piccolo-80 memristor-based encryption algorithm using the VTM method. Section IV presents the simulation and results, while Section V summarizes the discussion and conclusions.

II. The Piccolo Block Cipher

Piccolo is an ultra-lightweight block cipher first introduced in 2011 by Kyoji Shibutani and colleagues at Sony Corporation, Japan [18]. Using a 64-bit block size, Piccolo supports two key sizes: 80 bits and 128 bits, referred to as the Piccolo-80 and Piccolo-128 algorithms, respectively. Both versions share the same basic structure, consisting of two main parts: the data processing section, which handles encryption and decryption, and the key scheduling section, which organizes and distributes the keys for each encryption round.

The primary difference between Piccolo-80 and Piccolo-128 is the number of rounds in both the encryption and key scheduling processes.

A. Data Processing Section

In the Piccolo algorithm [18], the component responsible for data processing performs encryption and decryption using a

GFN [20], which is constructed with four 16-bit branches. This section involves multiple processing rounds, each beginning with the input passing through a nonlinear S-Box layer. The The S-Box layer consists of fixed 4-bit substitution boxes that introduce nonlinearity, thereby increasing the cipher's resistance to linear and differential attacks. The output is then combined with a diffusion matrix (M) to improve security. Additionally, a round key generated by the key scheduling section is XORed with the current data in each round. The Piccolo encryption algorithm uses the Gr function as its round function to update the internal state. The number of rounds, denoted by r, is 25 in Piccolo-80 and 31 in Piccolo-128. Algorithm I is presented below [18].

$$G_r : \begin{cases} \{0,1\}^{64} \times \{\{0,1\}^{16}\}^4 \times \{\{0,1\}^{16}\}^{2r} \rightarrow \{0,1\}^{64} \\ (X_{(64)}, wk_{0(16)}, \ldots, wk_{3(16)}, rk_{0(16)}, \ldots, rk_{2r-1(16)}) \rightarrow Y_{(64)} \end{cases}$$

Algorithm I: Piccolo encryption function

$G_r\ (X_{(64)}, wk_0, \ldots, wk_3, rk_0, \ldots, rk_{2r-1})$:
$X_{0(16)} \mid X_{1(16)} \mid X_{2(16)} \mid X_{3(16)} \leftarrow X_{(64)}$
$X_0 \leftarrow X_0 \oplus wk_0,\ X_2 \leftarrow X_2 \oplus wk_1$
For $i \leftarrow 0$ *to* $r-2$ *do*
$X_1 \leftarrow X_1 \oplus F(X_0) \oplus rk_{2i},\ X_3 \leftarrow X_3 \oplus F(X_2) \oplus rk_{2i+1}$
$X_0|X_1|X_2|X_3 \leftarrow RP\ (X_0|X_1|X_2|X_3)$
$X_1 \leftarrow X_1 \oplus F(X_0) \oplus rk_{2r-2},\ X_3 \leftarrow X_3 \oplus F(X_2) \oplus rk_{2r-1}$
$X_0 \leftarrow X_0 \oplus wk_2,\ X_2 \leftarrow X_2 \oplus wk_3$
$Y_{(64)} \leftarrow X_0|X_1|X_2|X_3,$

As shown in Figure 1, the Gr function, responsible for data processing in the Piccolo algorithm, operates over r rounds and accepts the following inputs: The function takes a 64-bit data block X ∈ {0,1}, four 16-bit subkeys *wki* ∈ {0,1}^16 (where $0 \le i < 4$), and 2r 16-bit round keys *rki* ∈ {0,1}^16 (where $0 \le i < 2r$). After applying cryptographic operations such as mixing and permutation to the data and keys, the function produces a 64-bit output Y ∈ {0, 1}^64.

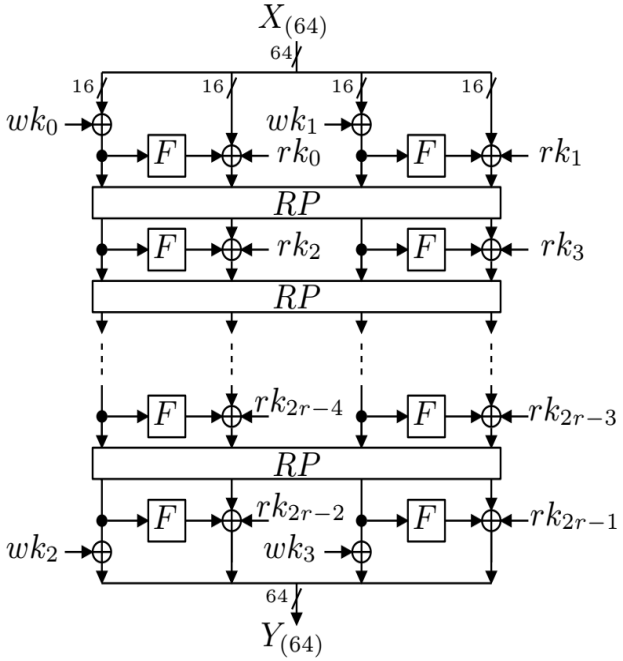


Fig. 1. Diagram of the Gr Encryption function [18].

Function F is an essential component of the Piccolo algorithm, significantly enhancing its encryption security.

As shown in Figure 2(a), the F-function: {0, 1}^16 → {0, 1}^16 is a 16-bit function consisting of two layers of S-boxes (shown in Figure 2(b)), processed through a diffusion matrix.

Each S-Box layer includes four bijective 4-bit S-Boxes, with their input-output relationships shown in hexadecimal format in Table I. From a hardware design perspective, the S-Box is efficient, utilizing only four NOR gates, three XOR gates, and one XNOR gate.

TABLE I
CONFIGURATION OF THE PICCOLO ALGORITHM'S S-BOX [18]

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
S[x]	e	4	b	2	3	8	0	9	1	a	7	f	6	c	5	d

The 16-bit data X is updated by applying the substitution

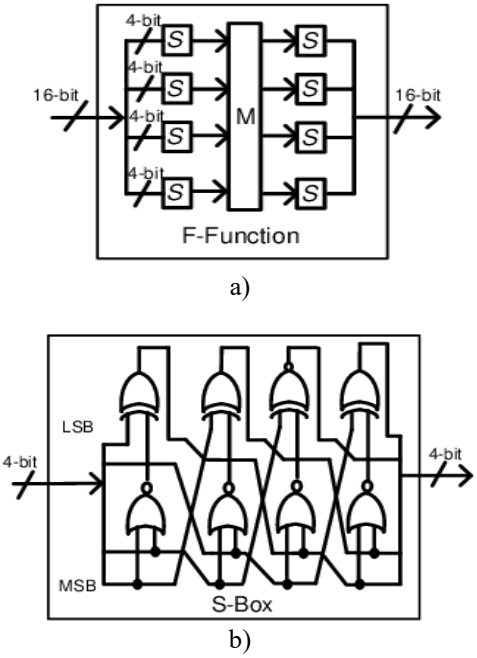


Fig. 2. Piccolo algorithm (a) Structure of the F-function, (b) Substitution box structure [18].

function S independently to its four 4-bit components [18].

$$(x_{0(4)}, x_{1(4)}, x_{2(4)}, x_{3(4)}) \leftarrow (S(x_{0(4)}), S(x_{1(4)}), S(x_{2(4)}), S(x_{3(4)})) \quad (1)$$

In the Piccolo-80 encryption algorithm, the MixColumn operation utilizes a 4×4 matrix M over GF(2^4) (Galois Field of size 2^4) to introduce diffusion. The diffusion matrix M is defined as:

$$M = \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix}$$

The 16-bit data X (16) is updated using the diffusion matrix M by multiplying the transpose of the 4-bit components with M.

$$^{t}(x_{0(4)}, x_{1(4)}, x_{2(4)}, x_{3(4)}) \leftarrow M.\ ^{t}(x_{0(4)}, x_{1(4)}, x_{2(4)}, x_{3(4)}) \quad (2)$$

The symbol $^{t}(a)$ indicates the transpose of the matrix or vector a. Each element in this matrix represents an element in GF(2^4), where multiplication follows the irreducible polynomial modulus $x^4 + x + 1$. A 64-bit to 64-bit bijective mapping is performed by the Round Permutation (RP) function. First, the 64-bit input is divided into eight equal 8-bit blocks, and then the permutation is applied. Figure 3 illustrates the configuration of RP.

$$(x_{0(8)}, x_{1(8)}, \ldots, x_{7(8)}) \leftarrow (x_{2(8)}, x_{7(8)}, x_{4(8)}, x_{1(8)}, x_{6(8)}, x_{3(8)}, x_{0(8)}, x_{5(8)})) \quad (3)$$

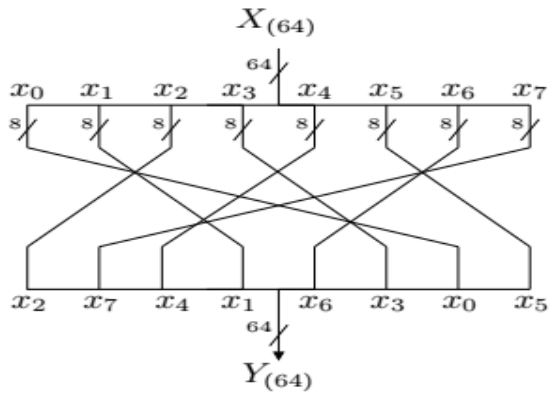


Fig. 3. Round Permutation [18].

B. Key scheduling section

The key scheduling in the Piccolo cipher generates two types of keys: the whitening key (*wki*), used during the initial and final rounds of encryption, and the round keys (*rki*), which are unique for each encryption round. In 80-bit mode, KS^{80} divides the primary key into five separate 16-bit subkeys, from which it generates the whitening and round keys. Each subkey is split into two 8-bit halves, and constants con^{80}_{i} and con^{128}_{i} are used in the respective key scheduling functions. Algorithm II shows the steps used in the process.

Algorithm II: key scheduling

$wk_0 \leftarrow kL_0 \mid kR_1,\ wk_1 \leftarrow kL_1 \mid kR_0$
$wk_2 \leftarrow kL_4 \mid kR_3,\ wk_3 \leftarrow kL_3 \mid kR_4$
For i $\leftarrow 0$ *to* $(r-1)$ *do*
$(rk_{2i}, rk_{2i+1}) \leftarrow (Con^{80}{}_{2i}, Con^{80}{}_{2i+1}) \oplus \begin{cases} (k_2, k_3) \\ (k_0, k_1) \\ (k_4, k_4) \end{cases}$
(k_2, k_3), *if i mod 5 = 0 or 2,*
(k_0, k_1), *if i mod 5 = 1 or 4,*
(k_4, k_4), *if i mod 5 = 3,*

Notably, the Piccolo encryption algorithm's linear layer involves a bitwise XOR operation applied to the state during each round, which effectively mixes the data bits by combining them with the round key. The linear layer is essential for achieving diffusion and is implemented in the MixColumn stage. Additionally, this layer performs a matrix multiplication in GF(2^4) between the input vectors and a fixed matrix, operating on four 16-bit half-blocks using a 4×4 matrix. This process combines parts of the input half-blocks and enhances the dependency between the input and output, which improves resistance to linear attacks.

The primary challenge of matrix multiplication in the Piccolo encryption algorithm is its computational complexity and the associated hardware implementation costs, which involve multiple bitwise addition and multiplication operations. Additionally, implementing it in hardware increases delay and resource consumption, and the algorithm can be implemented using either a serial or parallel architecture.

This work employs a parallel GF(2^4) multiplier, allowing the multiplication process to be completed in a single clock cycle. Overall, the structure can be implemented with either 16 XOR and 16 NAND gates or 18 XOR and 12 AND gates, where the critical path consists of 4 XOR gates and a single NAND/AND gate. Typically, a parallel GF(2^m) architecture includes (m + 1) mod-2 adders, each with m inputs, m × (m + 1) two-input AND gates, and m XOR gates that operate on two inputs [10]. To design a GF(2^4) multiplier based on polynomial representation that computes the product of two input vectors $P = (p3, p2, p1, p0)$ and $Q = (q3, q2, q1, q0)$, standard polynomial multiplication is first applied to the inputs. The result is then simplified using the irreducible polynomial $f(x) = x^4 + x + 1$ as the modulus. Therefore, the resulting expression is:

$$P(x).Q(x) modf(x) = (p_0q_3 + p_1q_2 + p_2q_1 + p_3q_0 + p_3q_3)\,x^3 + (p_0q_2 + p_1q_1 + p_2q_1 + p_2q_0 + p_2q_3 + p_3q_2 + p_3q_3)\,x^2 + (p_0q_1 + p_1q_0 + p_1q_3 + p_2q_2 + p_2q_3 + p_3q_1 + p_3q_2)\,x + (p_0q_0 + p_1q_3 + p_2q_2 + p_3q_1) \quad (4)$$

Equation (4) can be simplified and expressed in the following form [14]:

$$P(x).Q(x)\ modf(x) = t_3x^3 + t_2x^2 + t_1x + t_0 \quad (5)$$

Where:

$$t_0 = p_0q_0 \oplus p_1q_3 \oplus p_2q_2 \oplus p_3q_1$$

$$t_1 = p_0q_1 \oplus p_1q_0 \oplus p_1q_3 \oplus p_2q_2 \oplus p_2q_3 \oplus p_3q_1 \oplus p_3q_2$$

$$t_2 = p_0q_2 \oplus p_1q_1 \oplus p_2q_1 \oplus p_2q_0 \oplus p_2q_3 \oplus p_3q_2 \oplus p_3q_3$$

$$t_3 = p_0q_3 \oplus p_1q_2 \oplus p_2q_1 \oplus p_3q_0 \oplus p_3q_3 \quad (6)$$

A Boolean-based implementation of a GF(2^4) parallel multiplier is shown in Figure 4.

III. Memristor-Based Architecture Design For Piccolo-80

The use of memristors in cryptographic hardware design is increasing due to their unique properties, including non-volatility and low energy consumption. This enables the Piccolo-80 cryptographic algorithm to take advantage of a

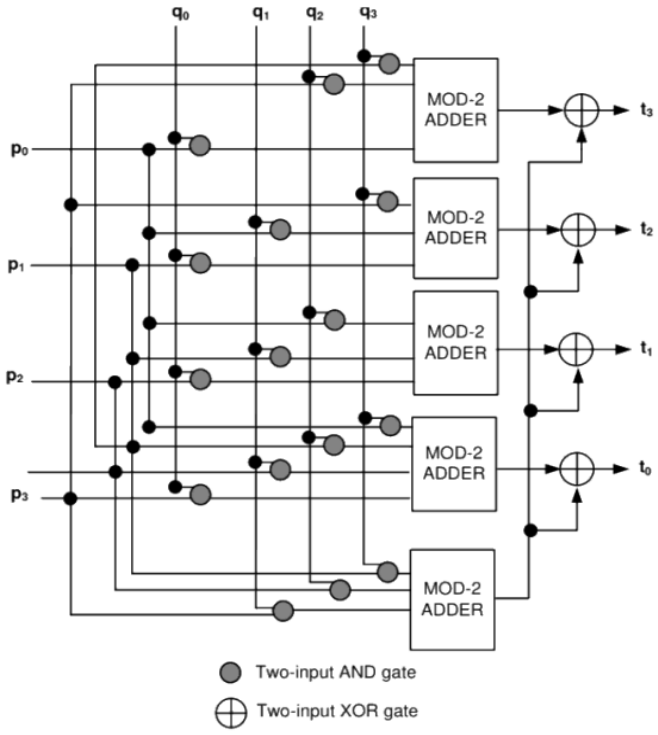


Fig. 4. Parallel GF(2^4) multiplier implemented with mod-2 adders and logic gates [10].

memristor-based architecture designed to enhance efficiency while reducing power consumption and hardware size. As a result, memristors can perform logic functions vital for cryptographic processes in encryption algorithms, such as those used in Piccolo-80. VTM-based logic gates, including NAND, NOR, AND, OR, XOR, and XNOR, provide a compact and energy-efficient alternative to traditional CMOS designs. These gates enable single-step logic operations, enhancing both speed and simplicity in hardware implementations.

A. Memristors

Memristors are two-terminal devices with unique properties, including bidirectionality, process compatibility, and non-volatility [25]. These features make memristors highly valuable for security applications. They operate with very low power consumption and provide high computing speeds, making them useful in applications such as digital memory, logic circuits, and hardware security systems. Their inherent qualities offer several advantages over CMOS-based security methods. Additionally, the non-volatile nature of memristors allows them to retain information without power, which is crucial for secure and reliable storage solutions.

One of the earliest uses of memristors in hardware security is the development of Physically Unclonable Functions (PUF) [26]. Memristor-based PUFs leverage natural process variations in memristor devices to generate unique and unpredictable responses to challenges, making them ideal for device authentication and secure key generation. Moreover, memristors are utilized in the creation of True Random Number Generators (TRNGs) [27]. These memristor-based TRNGs utilize the inherent randomness and process variation of memristors to produce random numbers, enhancing cryptographic security. The authors have also proposed a VTM architecture [28–31] to support the efficient implementation of memristive-based digital circuits.

B. Implementation of NAND/NOR logic circuit via VTM method

The stateful NAND/NOR logic gate shown in [28], as depicted in Figure 5, uses a common structure that can be configured to perform either NAND or NOR operations based on two different input voltages. The logic operation occurs in a single step. The configuration comprises two negatively biased input memristors, Input Memristor 1 and Input Memristor 2, and one output memristor. Voltages In1 and In2 are applied to the respective input memristors, causing the output memristor (Mout) to switch its resistance state to either a Low Resistance State (LRS) or a High Resistance State (HRS), corresponding to the resulting logic value. The LRS and HRS refer to the low and high resistance levels of the output memristor, respectively.

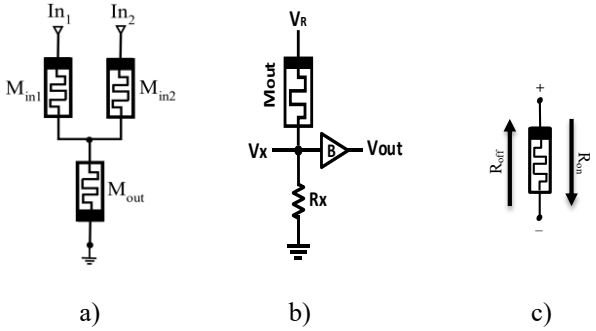


Fig. 5. a) Design of NAND/NOR, b) Circuit for reading output states, c) Memristor switching direction [28].

LRS typically indicates a binary ‘1’, and HRS represents a binary ‘0’.

Table II shows the operation of the introduced NAND/NOR gate, where +V corresponds to logic ‘1’ and 0 V to logic ‘0’.

TABLE II

LOGIC TRUTH TABLE FOR THE DESIGNED NAND/NOR [28].

In1	In2	Resistance State	NAND Output	Resistance State	NOR Output
0	0	LRS	‘1’	LRS	‘1’
0	1	LRS	‘1’	HRS	‘0’
1	0	LRS	‘1’	HRS	‘0’
1	1	HRS	‘0’	HRS	‘0’

C. Implementation of AND/OR logic circuit via VTM method

The introduced AND/OR logic gate [31], shown in Figure 6, has a configuration similar to that of the NAND/NOR gate, except that the input memristors are connected at the positive terminal.

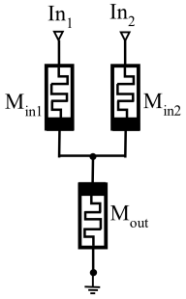


Fig. 6. Design of AND/OR [31].

Table III summarizes the logical output of the AND/OR gates presented in [31].

TABLE III
Logic Truth Table for the Designed AND/OR [31].

In1	In2	Resistance State	AND Output	Resistance State	OR Output
0	0	HRS	'0'	HRS	'0'
0	1	HRS	'0'	LRS	'1'
1	0	HRS	'0'	LRS	'1'
1	1	LRS	'1'	LRS	'1'

D. Design of XOR and XNOR logic gates via VTM approach

As shown in [31], the XOR and XNOR gates are depicted in Figures 7(a) and 7(b), respectively. The XNOR gate can be designed by placing the output memristor downward. Importantly, all gates introduced through the VTM method operate with a single-step process.

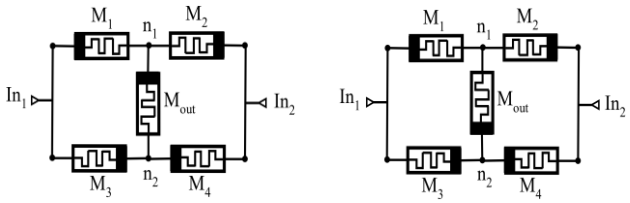


Fig. 7. Implementation of a) XOR b) XNOR gate using the VTM approach [31].

Tables IV and V present the truth table for the proposed XOR and XNOR gates, respectively.

TABLE IV
Logic Truth Table for the Designed XOR [31].

In1	In2	Resistance State	XOR Output
0	0	HRS	'0'
0	1	LRS	'1'
1	0	LRS	'1'
1	1	HRS	'0'

TABLE V
Logic Truth Table for the Designed XNOR [31].

In1	In2	Resistance State	XNOR Output
0	0	LRS	'1'
0	1	HRS	'0'
1	0	HRS	'0'
1	1	LRS	'1'

Figure 8 shows a full adder built with the gates designed using the VTM approach, with the Sum and Carry outputs stored in the relevant output memristors. Mod-2 adders, which are typically constructed using XOR gates, are employed to perform binary addition without carry propagation. In Three-input designs, the mod-2 sum (A ⊕ B ⊕ Cin) matches the sum output of a full adder.

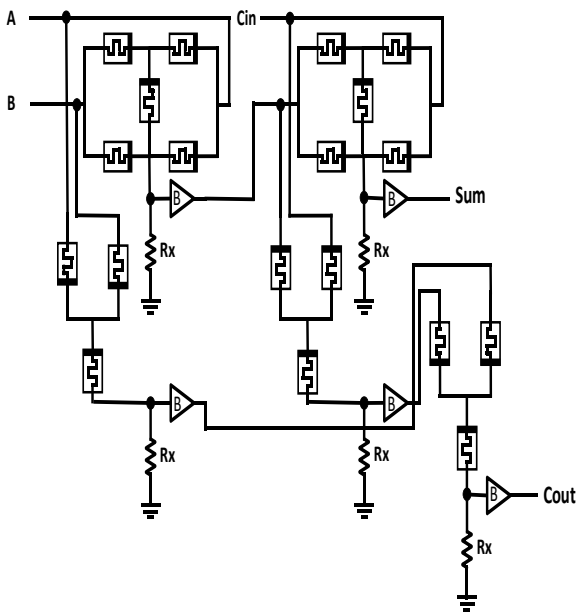


Fig. 8. Schematic diagram of the proposed full adder [31].

Table VI shows the truth table for the proposed full adder.

TABLE VI
LOGIC TABLE OF THE FULL ADDER IMPLEMENTED IN [31].

Logic inputs				Full Adder Outputs	
A	B	Cin	Logic Combination	Resistance State (Sum)	Resistance State (Carry)
0	0	0	"000"	HRS	HRS
0	0	1	"001"	LRS	HRS
0	1	0	"010"	LRS	HRS
0	1	1	"011"	HRS	LRS
1	0	0	"100"	LRS	HRS
1	0	1	"101"	HRS	LRS
1	1	0	"110"	HRS	LRS
1	1	1	"111"	LRS	LRS

E. Proposed hardware implementation of the Piccolo algorithm using memristor-based VTM stateful logic gates.

The complete Piccolo encryption algorithm can be implemented within a memristor-based architecture using the VTM method. The Piccolo cipher utilizes a GFN, where the internal state is divided into four 16-bit words, resulting in four processing branches. This layout is illustrated in Figure 9.

In each round of the Piccolo algorithm, two FN operations are performed. Each FN includes two steps: the F-function and the AddKey operation. The F-function is a non-keyed 16×16-bit transformation applied to the first branch of the FN. Additionally, each round uses two round keys, one for each FN.

Additionally, the algorithm uses pre- and post-whitening keys: *wk0* and *wk1*, which are combined with the internal state via bitwise XOR before the first round, while *wk2* and *wk3* are applied after the final round. A byte-level permutation occurs after the two FN operations in each round. It is assumed that both whitening and round keys can be accessed either through pre-configuration stored in memory or generated dynamically at runtime.

Figure 10 illustrates the complete implementation of the Piccolo-80 algorithm, constructed using VTM stateful logic gates. In this architecture, each VTM gate not only computes its output but also stores it directly in the output memristor; thus, the 64-bit state of the algorithm is maintained throughout the computation directly within the memristive cells, leading to a significant reduction in power consumption and hardware area compared to conventional CMOS designs.

In this design, a 64-bit input is initially split into four 16-bit words, each of which is divided into four 4-bit nibbles for parallel processing. At the start and end of the encryption, whitening keys (*wks*) are XORed with the half-blocks, while round keys (*rks*) are added at specific points during the rounds, especially after nonlinear layers or between core transformations. Each round includes the F-function, which first passes the nibbles through a fixed 4-bit S-box, a nonlinear layer. As shown in Figure 2, the S-box consists of only four NOR gates, three XOR gates, and one XNOR gate, all of which can be fully implemented using the proposed VTM logic gates. The output then moves to the MixColumn layer, where a diffusion matrix over GF(2^4) with the irreducible polynomial $x^4 + x + 1$, as defined in Equation (6), provides strong bit-level diffusion.

This is followed by another S-box layer applied to the nibbles. The result is XORed with the round key and then passed to the RP block, which applies a fixed permutation to the nibbles and rapidly propagates local changes throughout the entire 64-bit state.

This process repeats for 25 rounds in Piccolo-80 (and 31 rounds in Piccolo-128), with the final whitening applied to produce the 64-bit ciphertext. All these components—including the S-boxes, MixColumn, and RP—are implemented with VTM gates. Four S-boxes operate in parallel on each 16-bit word, while the MixColumn block is designed according to the hardware circuit in Figure 4 without needing CMOS multipliers or additional registers. In each round, the architecture requires 94 XOR/XNOR gates, 68 AND/OR gates, and 32 NAND/NOR gates. Simulation results at 1.8 V supply voltage and 133 MHz operating frequency show that, compared to traditional CMOS and hybrid MeMOS designs, the proposed design significantly reduces power consumption and hardware area.

In the memristor-based Piccolo-80 architecture, essential cryptographic operations, such as GF(2^m) addition and multiplication, are performed using memristor circuits. Memristors enable the development of parallel architectures, optimize the gate count, and decrease delay in the critical path with minimal power use. Furthermore, memristor-based stateful logic can be implemented in parallel configurations, offering the potential for faster computation.

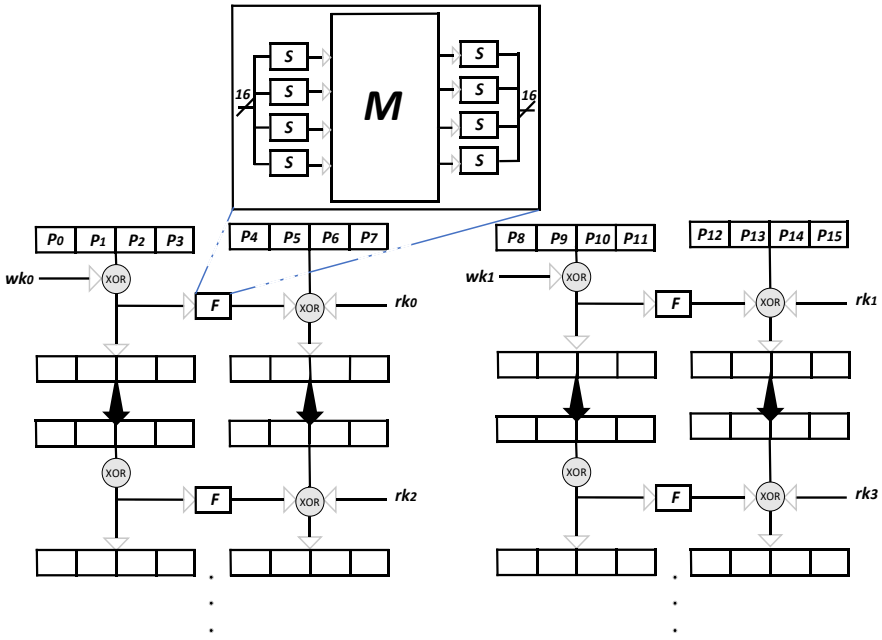


Fig. 9. Piccolo-80 structure [18].

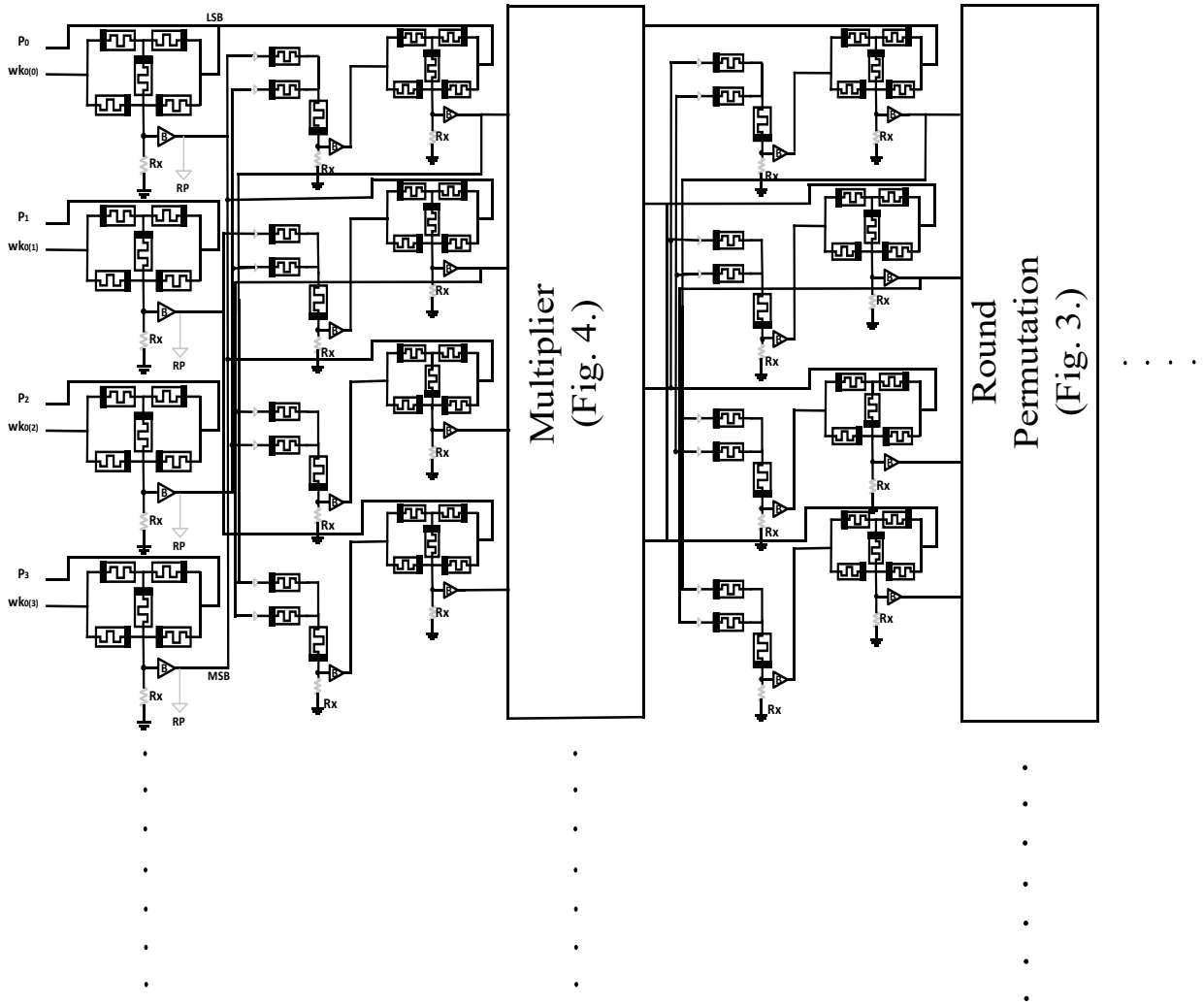


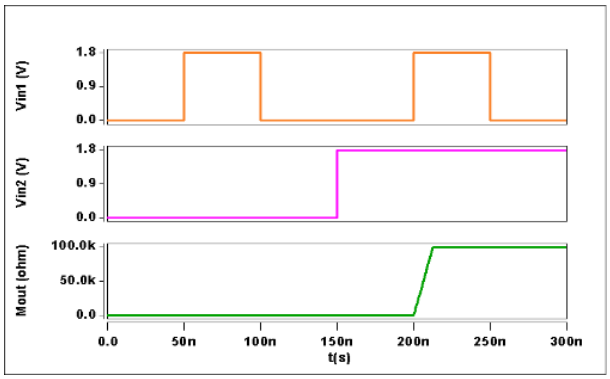
Fig. 10. Proposed architecture of the Piccolo algorithm using VTM stateful logic gates.

IV. Simulation and Results

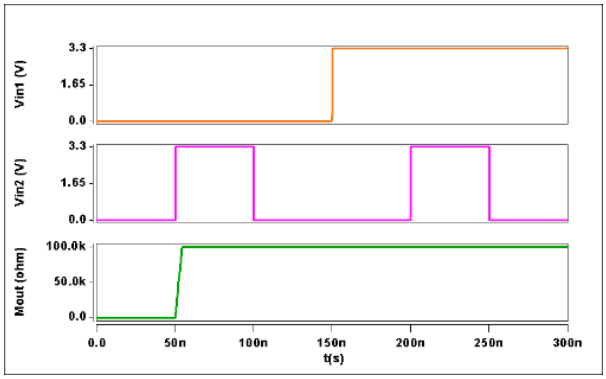
This section demonstrates how to implement the Piccolo-80 algorithm using both FPGA hardware and a memristor-based approach modeled through SPICE netlists in Cadence Virtuoso. Design simulation, validation, and power analysis are performed with Cadence Spectre. The simulations use the JART VCM v1b var memristor model [32].

A. Implementation of NAND/ NOR write logic

The NAND/NOR circuit design was simulated, and the outputs for all input combinations are shown in Figure 11.



a)



b)

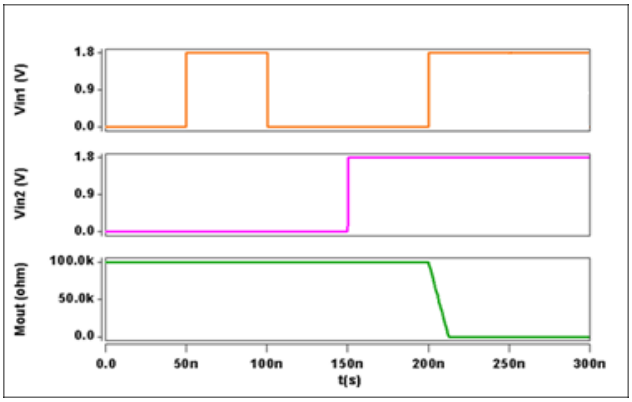
Fig. 11. Simulation results for the write logic operations: (a) NAND operation, (b) NOR operation [28].

B. Implementation of AND/OR Write Logic

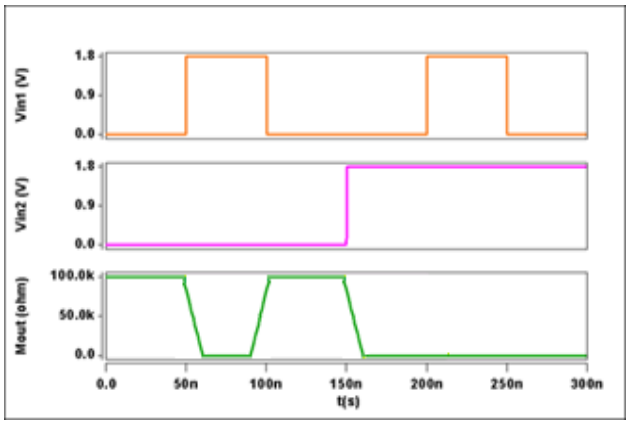
Figure 12 shows the simulation results of the designed AND/OR circuit. The gate's output is indicated by the resistance of Mout, with changes in memristance reflecting the corresponding logic states.

C. Simulation Results for the Read Circuit

By setting Mout=HRS, as shown in Figure 13(a), the simulation results demonstrate the read process corresponding to logic '0'.



a)



b)

Fig. 12. Simulation results of the write operation for a) AND operation, b) OR operation [31].

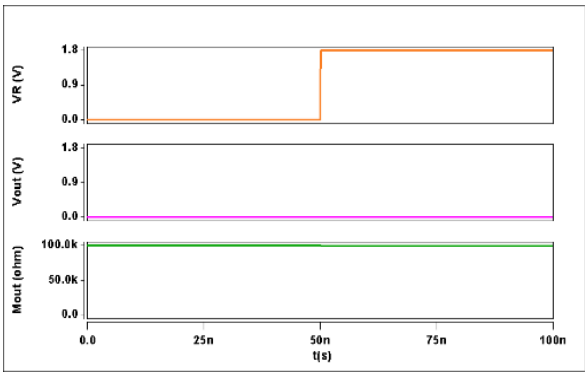
As shown in Figure 13(b), when Mout is set to the LRS, the current does not reach the output memristor during the 0–50 ns interval, resulting in Vout being 0 V.

Between 50 and 100 ns, the output memristor is read, and the resulting voltage, Vout, equals V_R, indicating a logic high state. The simulation data corresponds to the read operation for a logic '1' [28].

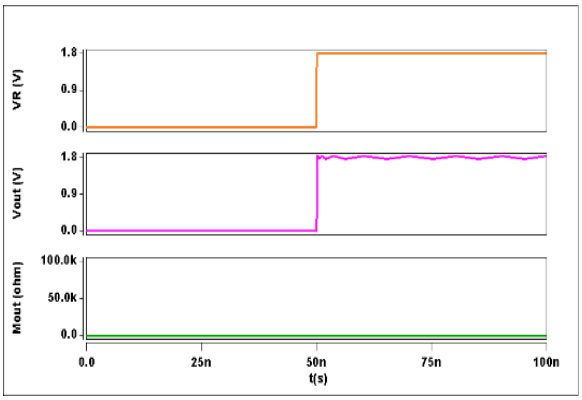
D. Write Logic Using XOR Gate

Figure 14(a) shows the simulated output of the designed XOR circuit. The gate's output depends on the resistance of the output memristor, Mout, where changes in memristance indicate different logic states.

Similarly, Figure 14(b) displays the simulated output of the proposed XNOR gate, confirming that the resistance state of the output memristor varies as expected for each input condition.

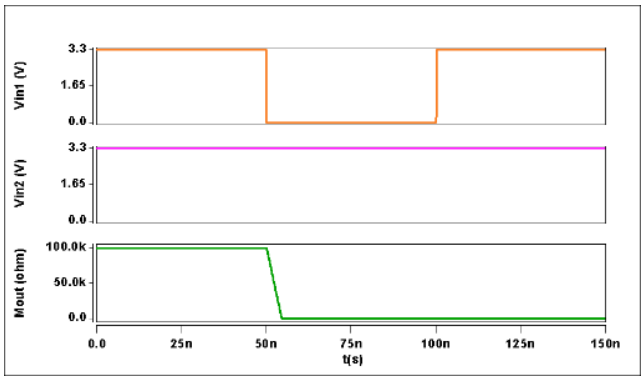


a)

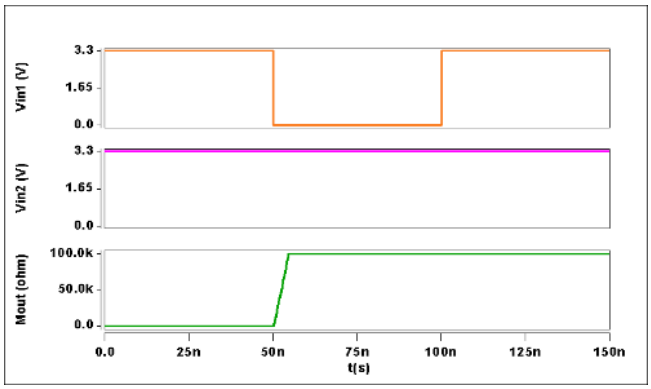


b)

Fig. 13. Simulated output of the designed read circuit for a) Mout= HRS, b) Mout= LRS [28].



a)



b)

Fig. 14. Simulated outputs of the write logic operations: (a) XOR operation, (b) XNOR operation [31].

E. Hardware Implementation of the Piccolo Encryption Algorithm Using FPGA.

The Piccolo-80 lightweight block cipher (64-bit block, 80-bit key, 25 rounds) was implemented in VHDL and simulated in ModelSim. Table VII presents the results, which are verified against standard test vectors to confirm functional correctness. Functional correctness was validated using the standard Piccolo-80 test vectors, proving that the 80-bit key and 64-bit plaintext in Table VII generate the expected ciphertext.

TABLE VII

PICCOLO ALGORITHM 80-BIT KEY TEST VECTOR [18].

Key length	80-bit
Key	$(00112233\ 44556677\ 8899)_{16}$
Plaintext	$(01234567\ 89ABCDEF)_{16}$
Ciphertext	$(8D2BFF99\ 35F84056)_{16}$

According to the architecture of the Piccolo encryption algorithm, the synthesis process is divided into three main parts: the S-Box layer, the F-function stage, and the data processing section. The results from the VHDL-based simulation are displayed in Figure 15.

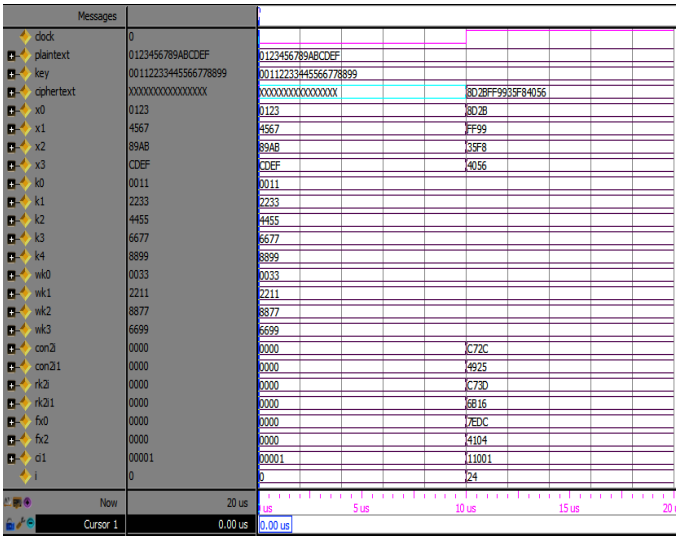


Fig.15. Simulation results for the VHDL implementation of the Piccolo algorithm.

The Piccolo-80 algorithm was then developed and synthesized using Xilinx ISE tools, targeting the Virtex-5 XC5VFX200T FPGA, which is fabricated in 65 nm CMOS technology. In the testbench, a 10 ns clock period (100 MHz) was defined, while post-synthesis timing analysis reported a maximum operating

frequency of approximately 427.716 MHz. Each encryption requires about 100 clock cycles, coordinated by four ALUs, two comparators, and three shift registers, resulting in a throughput of 640 Mbps at 100 MHz. Hardware synthesis was primarily performed to validate functionality and confirm that the VHDL design operates correctly on the FPGA hardware. The resource utilization results are summarized in Table VIII.

TABLE VIII

HARDWARE RESOURCE USAGE FOR THE PICCOLO ALGORITHM IMPLEMENTATION

Resource	Available	Utilization	(%)
Register	122880	289	1%
LUT	122880	291	1%
Slice	30720	104	1%
IO	960	64	6%
BUFG	32	6	18%

Table IX compares the Piccolo-80 algorithm implementation across different FPGA platforms, showing variations in efficiency and power consumption among the proposed schemes.

F. Using proposed stateful logic gates implemented through the VTM approach.

Since the Piccolo algorithm can be implemented using basic Boolean gates, the proposed circuit can fully perform all algorithmic steps, including bit permutations (P-Box), logic gate operations, linear transformations (Diffusion Layer), and Nonlinear Substitution Layer (S-Box). The main advantage of this design is that function F is implemented with VTM memristor-based logic gates. Because memristors can perform logical operations and store data simultaneously, they enable more efficient execution of this function, reduce hardware complexity, and increase processing speed when implementing the Piccolo algorithm.

Using the architecture shown in Figure 10, the Piccolo-80 lightweight block cipher was implemented in a round-based mode. The encryption process employs a 64-bit block and an 80-bit key, with 25 rounds. As previously mentioned, the round keys are assumed to be predefined and stored in memory before the encryption process starts.

The proposed design uses 94 XOR and XNOR gates, 68 AND/OR gates, and 32 NAND/NOR gates per encryption round. Additionally, the circuit operates at a supply voltage of 1.8V, which helps reduce power consumption, and a frequency of 133 MHz.

Also, integrating memristors into the design significantly reduces power consumption and simplifies hardware complexity. The non-volatile nature of memristors also ensures high stability and reliability.

To evaluate the energy efficiency of the proposed implementation, both dynamic and static power components were examined separately using the Cadence Spectre simulator. The results show that static (leakage) power was minimal due to the non-volatile nature of memristors, with over 90% of the total power consumed by dynamic switching activity. Additionally, gate-level simulations reveal that the substitution layer (S-Box), because of its frequent XOR/XNOR operations, accounts for approximately 43% of the total dynamic power. The diffusion matrix and permutation steps consume 29% and 18%, respectively, while key mixing operations contribute the remaining 10%. Furthermore, a frequency sweep analysis indicates that the design scales effectively, with only a moderate linear increase in power as frequency rises, thereby maintaining energy efficiency even at higher operational speeds.

These findings confirm the architectural benefit of the VTM-based approach in lowering active power, particularly in high-speed or energy-sensitive applications. The memristor-based VTM method reduces power consumption, and the small size of memristors results in a significant decrease in the area required compared to traditional hybrid MeMOS implementations.

Table IX

SUMMARY OF PICCOLO-80 HARDWARE IMPLEMENTATION RESULTS ON FPGA PLATFORMS.

Cryptographic Method	FPGA Model	Slices	Look-Up Tables (LUTs)	Freq. (MHz)	Efficiency (Mbps/Slice)	Power Consumption (mW)	Latency (ns)
Piccolo-80 [10]	Xilinx Spartan-6 XC6SLX25	135	419	189	3.44	174	137.81
Piccolo-80 [10]	Xilinx Virtex-4 XC4VLX25	273	525	273	2.45	383	95.69
Piccolo-80 [10]	Xilinx Virtex-5 XC5VLX50T	47	150	315	7.81	500	174.35
Piccolo-80 [10]	Xilinx Spartan-6 XC6SLX16	35	104	183	6.11	70	299.28
Unprotected CMOS [24]	Xilinx Spartan-6 XC6SLX9	64	106	206	2.06	98	485.44
Protected CMOS [24]	Xilinx Spartan-6 XC6SLX9	73	138	188	2.3	105	381.18
Piccolo-80 (This work)	Virtex-5 XC5VFX200T	104	291	428	6.15	334	100.06

TABLE X
COMPARISON OF THE PROPOSED IMPLEMENTATIONS AND OTHER RELATED WORKS ON THE PICCOLO CIPHER.

Implementation	Method	Gate Equivalents (GEs)	Frequency (MHz)	Throughput (Mbps)	Power Dissipation (mW)	Thr./Area (Mbps/GEs)	Latency (ns)
Piccolo-80 [24]	Protected Hybrid MeMOS	1512	112	286.7	25.6	0.189	223.29
Piccolo-80 [24]	Unprotected Hybrid MeMOS	1352	120	307.2	22.5	0.227	208.33
Piccolo-80 (This work)	VTM	1214	133	340	17.4	0.28	188.24

As shown in Table X, the proposed VTM-based design demonstrates clear improvements over prior Hybrid MeMOS implementations. In terms of hardware cost, the VTM implementation requires only 1214 GEs, representing a 19.7% reduction compared to the protected MeMOS design with 1512 GEs and a 10.2% decrease compared to the unprotected configuration with 1352 GEs. Operating at 133 MHz, the VTM design achieves the highest frequency among all platforms. It provides a throughput of 340 Mbps, which is higher than both protected (286.7 Mbps) and unprotected MeMOS (307.2 Mbps) designs. More importantly, power dissipation drops significantly to 17.4 mW, resulting in 32% and 22.6% power savings compared to the protected and unprotected hybrid MeMOS architectures, respectively. Moreover, the throughput-to-area efficiency reaches 0.280 Mbps/GE, which is higher than the 0.189 Mbps/GE and 0.227 Mbps/GE reported for the protected and unprotected MeMOS designs. Additionally, the latency is reduced to 188.24 ns, which is lower than the protected MeMOS at 223.29 ns and the unprotected MeMOS at 208.33 ns, showing reductions of 15.7% and 9.6%, respectively, further emphasizing the speed advantage of the VTM design.

These results confirm that the VTM architecture not only reduces power and area but also improves efficiency, making it a strong option for lightweight and energy-constrained IoT applications. According to Cadence, the power measurements encompass both static and dynamic power. The circuit's power consumption is significantly influenced by the memristor's LRS and HRS. The proposed architecture further highlights the advantages of memristor-based designs for efficient cryptographic operations. It is essential to note that switches are utilized to address the current sneak path problem, allowing each gate in separate rows to operate independently. Specifically, the input memristors are connected in series with NMOS transistors that function as switches.

Figure 16 shows the power dissipation of various Piccolo-80 implementations, including MeMOS-protected, MeMOS-unprotected, and the proposed VTM-based design. As illustrated, the VTM implementation has the lowest power consumption (17.4 mW), representing a clear improvement over both MeMOS designs (25.6 mW and 22.5 mW).

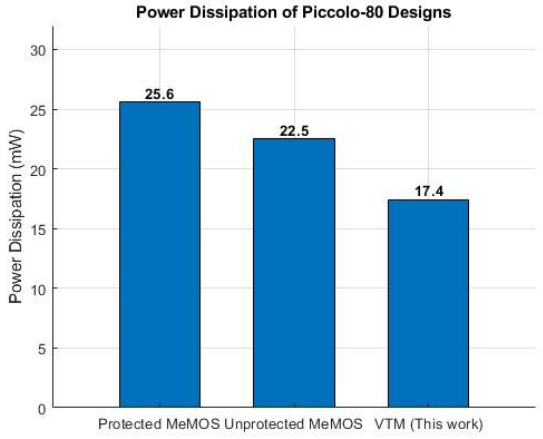


Fig 16. Comparison of Power Consumption in Piccolo-80 Algorithm Implementations.

G. Comparison with Other Lightweight Block Ciphers

To further highlight the importance of the proposed design, this work compares its VTM-based Piccolo-80 implementation with reported hardware results of other well-known lightweight block ciphers. Table XI shows the comparative results of several lightweight encryption algorithms alongside the proposed architecture. As demonstrated, the VTM-based Piccolo-80 achieves one of the smallest hardware areas with only 1214 GEs, significantly less than many widely studied lightweight ciphers. It also features competitive power consumption of 17.4 mW at 133 MHz, maintaining energy efficiency even at a higher operating frequency than most comparable designs, such as PRESENT and LED, and ASCON (evaluated in CMOS and CMOS/MRAM), which are typically evaluated at 100 MHz.

Although its throughput (340 Mbps) and CPD (7.52 ns) are lower than those of high-speed implementations like PRINCE, SIMON, and ASCON, the throughput-to-area ratio remains efficient compared to more complex algorithms, such as CLEFIA and Camellia. Overall, these results suggest that the proposed design achieves a strong balance between low area and power dissipation, while maintaining reliable performance, making it a practical and efficient choice for secure IoT applications.

TABLE XI
COMPARISON OF DIFFERENT LIGHTWEIGHT BLOCK CIPHER IMPLEMENTATIONS.

Cipher Implementation	Technology	Area (GEs)	Throughput (Mbps)	Thr./Area (Mbps/GE)	Power Consumption (mW)	CPD (ns)
ASCON CMOS [4]	--	10152.9	--	--	745 µW (at 100 MHz)	--
ASCON CMOS/MRAM [4]	Hybrid CMOS/MRAM	10715.5	--	--	714.8 µW (at 100 MHz)	--
CLEFIA [12] (flexible)	180 nm CMOS	14,951	2109.6 (128-B K) 1506.86 (192-B K) 1375.823(256-B K)	0.141 (128-B K) 0.101 (192-B K) 0.092 (256-B K)	--	4.045
PRINCE [13] (Low-cost)	180 nm CMOS	7046	2857.653	0.406	--	2.036
PRESENT [14] (flexible)	180 nm CMOS	4214	1492.54 (64/128-B K)	0.354	10.397 (at 100 MHz)	1.34
LED [14] (flexible)	180 nm CMOS	3556	680.3 (64-B K) 1010.1 (128-B K)	0.191 (64-B K) 0.284 (128-B K)	8.751 (at 100 MHz)	1.92
Camellia [15] (128/192/256)	180 nm CMOS	19,142	1271.73 (128/192-B K) 1059.78 (256- B K)	0.066 (128/192-BK) 0.055 (256- B K)	29.943 µW (at 100 KHz)	6.71
SIMON [16] (flexible)	180 nm CMOS	5647.2	2760.76	0.489	14.196 (at 100 MHz)	0.776
SPECK [16] (flexible)	180 nm CMOS	6170.65	1254.83	0.203	15.543 (at 100 MHz)	2.282
Piccolo (This work)	Memrisor-Based	1214	340 (80-B K)	0.28	17.4 (at 133 MHz)	7.52

Abbreviation: GE: Gate Equivalent, Thr.: Throughput, CPD = Critical Path Delay, B=bit, K=keys

H. Security Analysis

Nonlinear operations in the Piccolo cipher, particularly the S-Box layer results in significant power leakage, making it a prime target for Differential Power Analysis (DPA) attacks [14]. Power consumption in CMOS circuits is primarily determined by dynamic switching activity. Consequently, side-channel leakage arises from correlations between this switching activity and the processed key bits. An attacker can measure the power at specific times and detect even slight variations in switching activity, which may help infer the key bits. S-Box operations are a significant source of side-channel leakage, allowing attackers to exploit power analysis to recover whitening and round keys. In the proposed hardware architecture, using VTM-based memristor logic—which is non-volatile and stores its state as resistance—helps reduce power leakage. These gates retain data through resistance rather than electric charge, allowing many logic operations to occur with minimal changes in the circuit. This feature prevents large fluctuations in voltage or current, thereby lowering switching activity, stabilizing power consumption, and making it more difficult for attackers to observe and analyze. As a result, the design achieves lower switching activity, fewer power fluctuations, and more stable power behavior at the circuit level, which enhances resistance against DPA. However, this finding is based on circuit-level architecture and simulation. A thorough assessment of information leakage involves statistical methods, such as the Test Vector Leakage Assessment (TVLA) t-test or Correlation Power Analysis (CPA). Although these experiments are beyond this study's scope, they are important directions for future research.

V. CONCLUSION

This paper introduces an efficient hardware implementation of the Piccolo-80 encryption algorithm using memristor-based logic gates with the VTM method. Memristor-based stateful logic gates can change resistance under voltage control and retain their value even after the voltage is removed, which significantly reduces overall power consumption. The proposed architecture for each round utilizes only 194 memristor-based gates, including NAND/NOR, AND/OR, XOR, and XNOR, which can switch functions via voltage control. It performs all the primary operations of the Piccolo algorithm, including P-Box permutations, S-Box substitutions, linear propagation, and nonlinear transformations, while maintaining the round-based encryption structure of the algorithm.

This preserves full compatibility with the original Piccolo algorithm structure while utilizing the computational advantages of memristor-based algorithms. Additionally, this

implementation offers notable improvements in power and area efficiency compared to earlier related designs. As a result, static power is almost eliminated, dynamic power is significantly reduced, and overall area is minimized. The memristor-based VTM approach achieves power savings of 32% and 22.6% over protected and unprotected MeMOS hybrid designs, respectively, while reducing hardware area by 19.7% and 10.2%. Simulation results from Cadence Spectre software confirm the design's power and hardware efficiency. The implementation consumes only 17.4 mW of power at 1.8 V and 133 MHz, utilizing less space than comparable designs. Furthermore, the throughput-to-area efficiency achieves 0.280 Mbps/GE, representing an improvement over the 0.189 Mbps/GE and 0.227 Mbps/GE reported for the protected and unprotected MeMOS designs. Additionally, gate-level power analysis showed that the substitution layer consumes approximately 43% of the dynamic power, followed by diffusion (29%), permutation (18%), and key mixing (10%). This breakdown highlights the main power hotspots and verifies the effectiveness of VTM-based optimization.

These features make it ideal for resource-constrained environments, such as IoT devices, where optimizing power and area is crucial. Furthermore, utilizing the VTM method for logic gates enhances security by reducing power leakage and increasing resistance to DPA attacks, thereby making the design more robust for secure IoT applications.

REFERENCES

[1] Abhijan Bhattacharyya, “Internet of Things,” - Technology, Applications and Standardization, Published 2018, doi: 10.5772/Intech open 70907

[2] M. S. Turan, K. A. McKay, J. Kang, J. Kelsey, and D. Chang, “Ascon-Based Lightweight Cryptography Standards for Constrained Devices: Authenticated Encryption, Hash, and Extendable Output Functions,” *NIST Special Publication 800-232*, Aug. 2025.

[3] M. A. Khan, M. Ahmad, M. Aslam, M. W. Aslam, and S. A. Parah, “Securing the IoT ecosystem: ASIC-based hardware realization of Ascon lightweight cipher,” *International Journal of Information Security*, Springer, 2024.

[4] Roussel, N., Potin, O., Di Pendina, G., Dutertre, J. M., Rigaud, J.B.: CMOS/STT-MRAM based Ascon LWC: A power efficient hardware implementation. In: *2022 29th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pp. 1–4. IEEE (2022)

[5] A. R. Alharbi, A. Aljaedi, A. Aljuhni, M. K. Alghuson, H. Aldawood, and S. S. Jamal, “Evaluating Ascon hardware on 7-series FPGA devices,” *IEEE Access*, vol. 12, pp. 149076–149089, 2024, doi: 10.1109/ACCESS.2024.3471694

[6] M. A. Siddiqi, J. A. G. Hernández, A. Gebregiorgis, R. Bishnoi, C. Strydis, S. Hamdioui, and M. Taouil, “Memristor-Based Lightweight Encryption,” *arXiv preprint* arXiv:2404.00125, 2024.

[7] H. Ajmi, F. Zayer, and H. Belgacem, “In-Memory Computing Architecture for Efficient Hardware Security,” *arXiv preprint* arXiv:2408.11570, 2024.

[8] K. R. Penumalli, T. R. Kadiyam, V. Birudu, V. Gonuguntla, and R. Vaddi, “Design of an SLIM Cipher S-Box with 8T-SRAM CiM for Energy-Efficient, Lightweight, and DPA-Resistant Edge AI,” *Engineering, Technology & Applied Science Research*, vol. 15, no. 4, pp. 25902–25914, Aug. 2025. doi: 10.48084/etasr.11870.

[9] É. C. Dutra e Silva Junior, C. A. de M. Cruz, I. A. L. Saraiva, F. G. Santos, C. R. P. dos Santos Junior, L. S. Indrusiak, W. A. Finamore, and M. Glesner, “Chaos-Based S-Boxes as a Source of Confusion in Cryptographic Primitives,” *Electronics*, vol. 14, no. 11, p. 2198, 2025. doi: 10.3390/electronics14112198.

[10] M. Masoumi, “Design and Evaluation of a Power Analysis Resilient Implementation of Piccolo-80 Lightweight Encryption Algorithm,” *Journal of Hardware and Systems Security*, vol. 7, pp. 101–109, 2023, doi: 10.1007/s41635-023-00191-1.

[11] B. Rashidi, “High-throughput and lightweight hardware structures of HIGHT and PRESENT block ciphers,” *Microelectronics Journal*, vol. 90, pp. 232–252, 2019, doi: 10.1016/j.mejo.2019.06.012.

[12] B. Rashidi, “High-throughput and flexible ASIC implementations of SIMON and SPECK lightweight block ciphers,” *International Journal of Circuit Theory and Applications*, vol. 47, no. 8, pp. 1254–1268, 2019, doi: 10.1002/cta. 2645.

[13] B. Rashidi, “Efficient and flexible hardware structures of the 128-bit CLEFIA block cipher,” *IET Computers & Digital Techniques*, vol. 14, no. 2, pp. 69–79, 2020, doi: 10.1049/iet-cdt.2019.0157.

[14] B. Rashidi, “Low-cost and two-cycle hardware structures of the PRINCE lightweight block cipher,” *International Journal of Circuit Theory and Applications*, vol. 48, no. 8, pp. 1227–1243, 2020, doi: 10.1002/cta. 2832.

[15] B. Rashidi, “Flexible structures of lightweight block ciphers PRESENT, SIMON and LED,” *IET Circuits, Devices & Systems*, vol. 14, no. 3, pp. 369–380, 2020, doi: 10.1049/iet-cds.2019.0363.

[16] B. Rashidi, “Flexible and high-throughput structures of Camellia block cipher for security of the Internet of Things,” *IET Computers & Digital Techniques*, vol. 15, no. 2, pp. 171–184, 2021, doi: 10.1049/cdt2.12025.

[17] B. Rashidi, “Glitch-less hardware implementation of block ciphers based on an efficient glitch filter,” *Integration, the VLSI Journal*, vol. 85, pp. 20–26, 2022, doi: 10.1016/j.vlsi.2022.02.007.

[18] Shibutani K, Isobe T, Hiwatari H, Mitsuda A, Akishita T, Shirai T. “Piccolo: an ultra-lightweight blockcipher, *International Workshop on Cryptographic Hardware and Embedded Systems*,” 2011, Sep 28, pp. 342-357, Springer, Berlin, Heidelberg.

[19] Ramu, G., Mishra, Z., Singh, P., & Acharya, B. (2020). “Performance optimized architectures of Piccolo block cipher for low resource IoT applications,” *International Journal of High-Performance Systems Architecture, 9*(1), 49–57.

[20] Serhii Naumenko; Inna Rozlomii; Andrii Yarmilko, “The Built on Feistel Network Architecture Block Ciphers Modification,” *14th International Conference on Advanced Computer Information Technologies (ACIT).* September 2024, doi:10.1109/ACIT62333.2024.10712597.

[21] J. Feng, Y. Wei, B. Wei, “Novel optimized implementations for the Piccolo cipher based on field-programmable gate arrays,” *International Journal of Circuit Theory and Applications*, vol. 53, no. 2, pp. 771–789, 2025, doi:10.1002/cta. 4160.

[22] G. Ramu; Z. Mishra; P. Singh; B. Acharya, “Hardware implementation of Piccolo Encryption Algorithm for constrained RFID application,” in *Proc. 9th Annu. IEMECON – Inf. Technol., Electromech. Eng. Microelectron. Conf.*, 2019, pp. 85–89. doi: 10.1109/IEMECONX.2019.8877071.

[23] Mishra, S., Mishra, Z., & Acharya, B. “Area Optimized Hardware Architecture of Piccolo-80 Lightweight Block Cipher,” *Proceedings of Fifth International Conference on Microelectronics, Computing and Communication Systems*, 2021, pp 341–350.

[24] Masoumi, M. (2019). “Design and Evaluation of Memristor-Based Piccolo-80 Lightweight Encryption Algorithm for Future IoT,” *J. Inf. Secur. Appl.,* vol. 48, 102371, 2019. doi: 10.1016/j.jisa.2019.102371.

[25] L. Chua, “Memristor-the missing circuit element,” *IEEE Transactions on Circuit Theory*, vol. 18, pp. 507-519, 1971.

[26] Chandranshu Gupta, Gaurav Varshney. “A Lightweight and Secure PUF-Based Authentication and Key-exchange Protocol for IoT Devices,” *arXiv:2311.04078v1 [cs.CR]*, 7 Nov 2023.

[27] Xu, H., Qu, W., & Xu, Q. (2020). "Memristor-based true random number generator," *IEEE Transactions on Circuits and Systems II*: Express Briefs, 67(10), 1860-1864.

[28]. F. Mozafari, M. J. Sharifi, M. Ahmadi, A. Ahmadi, "A novel memristive architecture for memristor-based logic," *Proc. IEEE Int. Midwest Symp. Circuits Syst. (MWSCAS)*, pp. 812–815, 2021.

[29]. F. Mozafari, M. Ahmadi, A. Ahmadi, "Design of a new memristive-based architecture using VTM method," *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, pp. 980–984, 2022.

[30]. F. Mozafari, M. Ahmadi, A. Ahmadi, "A programmable circuit based on the combination of VTM cellular crossbars," *Proc. 19th Int. Conf. Synthesis, Modeling, Analysis, Simulation Methods Appl. Circuit Design (SMACD)*, pp. 1–4, 2023.

[31]. F. Mozafari, M. Ahmadi, and A. Ahmadi, "Design and implementation of full adder circuit based on VTM-logic gates," *Proc. IEEE 66th Int. Midwest Symp. Circuits Syst. (MWSCAS)*, pp. 389–393, 2023.

[32]. EMRL. (2020) JART – J¨ulich Aachen Resistive Switching Tools. *[Online]*. Available: http://www.emrl.de/JART#Artikel 1.

Farzad Mozafari (Member, IEEE) is currently pursuing a Ph.D. degree in Electrical Engineering at the University of Windsor, Windsor, ON, Canada. His research explores Memristor-Based Lightweight Encryption. His broader research interests include hardware security, emerging non-volatile memory technologies, VLSI design, cryptographic architecture, and energy-efficient computing systems. He is particularly interested in the intersection of nanoscale devices and secure hardware design, focusing on developing scalable, low-power solutions for next-generation embedded and IoT systems.

Majid Ahmadi (Life Fellow, IEEE) received the B.Sc. degree in Electrical Engineering from Sharif University of Technology (formerly Arya Mehr University), Tehran, Iran, in 1971, and the Ph.D. degree in Electrical Engineering from Imperial College London, U.K., in 1977. He joined the Department of Electrical and Computer Engineering at the University of Windsor, Canada, in 1980, where he is currently a University Professor and Director of the Research Center for Integrated Microsystems. His research interests include digital signal processing, machine vision, pattern recognition, neural networks, VLSI design, and computer arithmetic. He has coauthored a book and published over 650 papers. He has served as Regional Editor for the Journal of Circuits, Systems, and Computers and Associate Editor for the Pattern Recognition journal. His honors include the Honorable Mention Award from Pattern Recognition (1992) and the Distinctive Contributed Paper Award from the Multiple-Valued Logic Conference (1999).

CJECE-October 2025

Design and Implementation of a Low-Power Memristor-Based Piccolo-80 Lightweight Encryption Algorithm Using VTM Logic Gates

Farzad Mozafari, *Member, IEEE*, Majid Ahmadi, *Life Fellow, IEEE*

***Abstract*—Lightweight cryptography has become increasingly critical for ensuring secure communication in energy-constrained Internet of Things (IoT) systems. Memristor-based architecture provides a promising approach for secure communication in energy-sensitive and hardware-constrained applications. Piccolo is a lightweight encryption algorithm that offers high security while enabling compact hardware implementation. Additionally, Piccolo is specifically designed to operate efficiently in resource-limited environments, making it a strong candidate for low-energy applications such as IoT devices. However, earlier implementations of the Piccolo algorithm on FPGA platforms, CMOS, and hybrid MeMOS (Memristor-CMOS) technology have faced challenges with high power consumption, hardware overhead, and limited scalability. This paper presents a novel architecture for implementing the Piccolo-80 encryption algorithm using the VTM (Voltage-to-Memristance) approach, in which the design maps Piccolo's primary operations onto VTM stateful logic gates. This enhances performance, reduces switching activity, and leverages the non-volatile properties of memristors. The proposed design introduces VTM-based memristor logic gates that significantly reduce hardware complexity and power consumption compared to previous implementations. The results from comparing CMOS and hybrid MeMOS implementations in terms of area and energy consumption demonstrate that hardware implementation of Piccolo's lightweight algorithm using the VTM approach not only improves energy efficiency but also enables the design of optimized, low-power circuits. The design achieves a power consumption of 17.4 mW at 1.8 V and 133 MHz, with only 1214 Gate Equivalents (GEs), reducing power by up to 32% and area by nearly 20% compared to state-of-the-art hybrid MeMOS designs.**

***Index Terms* — Memristor-based Hardware Implementation of Piccolo-80 Algorithm, Internet of Things (IoT), Lightweight Encryption, Voltage to Memristance (VTM).**

I. Introduction

In recent years, rapid technological advances and the widespread use of modern electronic devices, embedded systems, and the IoT [1] have increased the need for lightweight and high-performance ciphers. In 2023, NIST selected the Ascon family as the winner of the Lightweight Cryptography (LWC) competition, and it was later formalized in NIST SP 800-232 (2025). The standard includes authenticated encryption, hashing, and extendable-output functions (XOFs), emphasizing simplicity and efficiency for constrained platforms [2]. Following this, hardware implementations of Ascon have been widely explored. Khan et al. developed ASIC accelerators for Ascon-128 and Ascon-128a at a 32-nm technology node, comparing loop-folded, loop-unrolled, and fully unrolled architectures. They showed a clear trade-off: loop-folded designs save area, while fully unrolled ones maximize throughput at a higher cost [3, 4]. Alharbi et al. extended this work with FPGA implementations on Xilinx 7-series devices, where an iterative FSM-based design with buffer-driven datapaths improved frequency while maintaining modest power and balanced area–performance, confirming Ascon's practicality for IoT systems [5].

Beyond CMOS, emerging memory technologies have been proposed to reduce energy consumption and minimize data movement. Siddiqi et al. introduced a memristor-based GIFT-128 using RRAM crossbars, enabling each round with a single read, cutting energy use, and supporting reconfigurable S-boxes for side-channel protection [6]. Ajmi et al. proposed similar in-memory AES designs, executing operations directly within memory arrays to improve efficiency [7]. Other efforts target substitution boxes (S-boxes), the nonlinear core of lightweight ciphers. Penumalli et al. designed an 8T-SRAM compute-in-memory S-box that reduces area and improves DPA resistance. At the same time, Dutra e Silva Jr. et al. developed chaos-based S-boxes that replace static tables with chaotic mappings, reducing storage and adding strong

Date on submitted paper: Oct 3, 2025. Farzad Mozafari and Majid Ahmadi are with the Electrical and Computer Engineering (ECE) Department, University of Windsor (e-mail: mozafarf@uwindsor.ca, ahmadi@uwindsor.ca). "This work was supported in part by the ECE Department, University of Windsor, Windsor, ON N9B 3P4, Canada."

Corresponding author: Farzad Mozafari, Second Author: *Majid Ahmadi. Farzad Mozafari is a PhD student at the Department of ECE, University of Windsor, Windsor, Ontario, Canada. Majid Ahmadi is a Professor of ECE, University of Windsor, Ontario, Canada, ahmadi@uwindsor.ca.*

nonlinearity [8, 9]. In 2023, potential weaknesses of the Piccolo cipher against Differential Power Analysis (DPA) were identified, leading to the development of a new protection scheme to enhance the algorithm's resistance to DPA [10].

Rashidi has contributed extensively to lightweight block cipher hardware for FPGA, CMOS, and ASIC platforms. In 2019, FPGA implementations of HIGHT and PRESENT were optimized through gate-level improvements, enhancing performance and side-channel resistance on Virtex-5 and Spartan-3 devices [11]. That year, high-throughput ASIC architectures for SIMON and SPECK in 180 nm CMOS applied Sklansky adders and tree-structured XOR logic to reduce delay, raise frequency, and scale across block and key sizes [12]. In 2020, this work expanded to CLEFIA and PRINCE, where logic simplification and composite-field methods improved CLEFIA's area and efficiency. PRINCE utilized resource-shared S-boxes, a two-cycle architecture, and optimized linear operations to reduce delay and area while enhancing throughput for RFID and latency-sensitive applications [13, 14]. Later designs introduced flexible CMOS hardware for PRESENT, SIMON, and LED, utilizing optimized S-boxes, efficient MixColumns, and resource-sharing techniques to support multiple block and key sizes. The Camellia design in 180 nm CMOS applied composite-field inversion and resource sharing to reduce cost while maintaining strong throughput-to-area ratios [15, 16]. More recently, glitches in CMOS implementations of PRESENT, HIGHT, and SPECK were addressed through a filter circuit that stored intermediate data, eliminating unwanted switching activity with minimal area overhead and improving reliability and power stability without altering the algorithms [17].

Although many lightweight ciphers have been developed, there remains a need for modular algorithms that achieve efficient performance on resource-constrained devices. The Piccolo encryption algorithm [18] was specifically designed to address the needs of embedded systems, IoT devices, and other lightweight platforms with hardware and energy limitations. Its main feature is a simple, modular design, making it suitable for hardware implementation in embedded applications [19]. A key advantage of Piccolo is its use of a Generalized Feistel Network (GFN) architecture combined with a Substitution-Permutation Network (SPN) [20], which reduces design complexity while maintaining high security. Early efforts to implement Piccolo-80 in hardware focused on FPGAs [21] and their modular structure enable the efficient utilization of FPGA resources. Ramu et al. (2019) proposed a new architecture for Piccolo-80 designed for high-speed Radio Frequency Identification (RFID) security applications [22]. Their implementation demonstrated Piccolo-80's suitability for constrained environments by achieving competitive throughput and area efficiency. They showed that the algorithm can deliver high performance with minimal resources, utilizing simple key scheduling and parallel processing. Later efforts focused on improving the area efficiency of Piccolo-80. In 2021, Mishra et al. presented an architectural study that significantly reduced the number of logic gates required [23], making it more practical for devices with limited space. The researchers focused on techniques such as logic optimization, data path width reduction, and efficient use of combinational circuits to reach this goal. Recently, researchers have explored the potential of emerging technologies, such as memristors, to further enhance the hardware performance of Piccolo-80. In 2019, Masoumi implemented the Piccolo-80 algorithm in hardware using a hybrid MeMOS architecture, evaluating performance based on resource use and power efficiency [24]. While previous studies have mainly focused on architectural and logic-level optimizations of Piccolo-80 in conventional CMOS, FPGA, and hybrid MeMOS technologies, these implementations still face significant challenges, including high dynamic power consumption, increased circuit area, additional hardware overhead, limited scalability, and increased vulnerability to side-channel attacks. These limitations make them less suitable for energy-constrained environments, such as IoT devices, and highlight the need for novel hardware solutions. This work introduces a new memristor-based design for Piccolo-80 using the VTM method. By utilizing the non-volatile nature of memristors, the proposed architecture enables stateful single-step operations, reduces switching activity, features compact logic components, and consumes less power. The entire round function, including the S-box and MixColumn operations, is implemented entirely with VTM stateful logic gates. This architecture addresses the power, area, and scalability challenges of earlier designs, highlighting the potential of emerging memristor technologies for secure, energy-efficient IoT applications.

The paper is organized as follows: Section II provides a brief overview of the Piccolo-80 cipher's structure, Section III explains the design and implementation of the Piccolo-80 memristor-based encryption algorithm using the VTM method. Section IV presents the simulation and results, while Section V summarizes the discussion and conclusions.

II. The Piccolo Block Cipher

Piccolo is an ultra-lightweight block cipher first introduced in 2011 by Kyoji Shibutani and colleagues at Sony Corporation, Japan [18]. Using a 64-bit block size, Piccolo supports two key sizes: 80 bits and 128 bits, referred to as the Piccolo-80 and Piccolo-128 algorithms, respectively. Both versions share the same basic structure, consisting of two main parts: the data processing section, which handles encryption and decryption, and the key scheduling section, which organizes and distributes the keys for each encryption round.

The primary difference between Piccolo-80 and Piccolo-128 is the number of rounds in both the encryption and key scheduling processes.

A. Data Processing Section

In the Piccolo algorithm [18], the component responsible for data processing performs encryption and decryption using a

GFN [20], which is constructed with four 16-bit branches. This section involves multiple processing rounds, each beginning with the input passing through a nonlinear S-Box layer. The The S-Box layer consists of fixed 4-bit substitution boxes that introduce nonlinearity, thereby increasing the cipher's resistance to linear and differential attacks. The output is then combined with a diffusion matrix (M) to improve security. Additionally, a round key generated by the key scheduling section is XORed with the current data in each round. The Piccolo encryption algorithm uses the Gr function as its round function to update the internal state. The number of rounds, denoted by r, is 25 in Piccolo-80 and 31 in Piccolo-128. Algorithm I is presented below [18].

$$G_r: \begin{cases} \{0,1\}^{64} \times \{\{0,1\}^{16}\}^4 \times \{\{0,1\}^{16}\}^{2r} \rightarrow \{0,1\}^{64} \\ (X_{(64)}, wk_{0(16)}, \ldots, wk_{3(16)}, rk_{0(16)}, \ldots, rk_{2r-1(16)}) \rightarrow Y_{(64)} \end{cases}$$

Algorithm I: Piccolo encryption function

$G_r\ (X_{(64)}, wk_0, \ldots, wk_3, rk_0, \ldots, rk_{2r-1})$:
$X_{0(16)} \mid X_{1(16)} \mid X_{2(16)} \mid X_{3(16)} \leftarrow X_{(64)}$
$X_0 \leftarrow X_0 \oplus wk_0,\ X_2 \leftarrow X_2 \oplus wk_1$
For $i \leftarrow 0$ *to* $r-2$ *do*
$X_1 \leftarrow X_1 \oplus F(X_0) \oplus rk_{2i},\ X_3 \leftarrow X_3 \oplus F(X_2) \oplus rk_{2i+1}$
$X_0|X_1|X_2|X_3 \leftarrow RP\ (X_0|X_1|X_2|X_3)$
$X_1 \leftarrow X_1 \oplus F(X_0) \oplus rk_{2r-2},\ X_3 \leftarrow X_3 \oplus F(X_2) \oplus rk_{2r-1}$
$X_0 \leftarrow X_0 \oplus wk_2,\ X_2 \leftarrow X_2 \oplus wk_3$
$Y_{(64)} \leftarrow X_0|X_1|X_2|X_3$,

As shown in Figure 1, the Gr function, responsible for data processing in the Piccolo algorithm, operates over r rounds and accepts the following inputs: The function takes a 64-bit data block X ∈ {0,1}, four 16-bit subkeys *wki* ∈ {0,1}^16 (where $0 \le i < 4$), and 2r 16-bit round keys *rki* ∈ {0,1}^16 (where $0 \le i < 2r$). After applying cryptographic operations such as mixing and permutation to the data and keys, the function produces a 64-bit output Y ∈ {0, 1}^64.

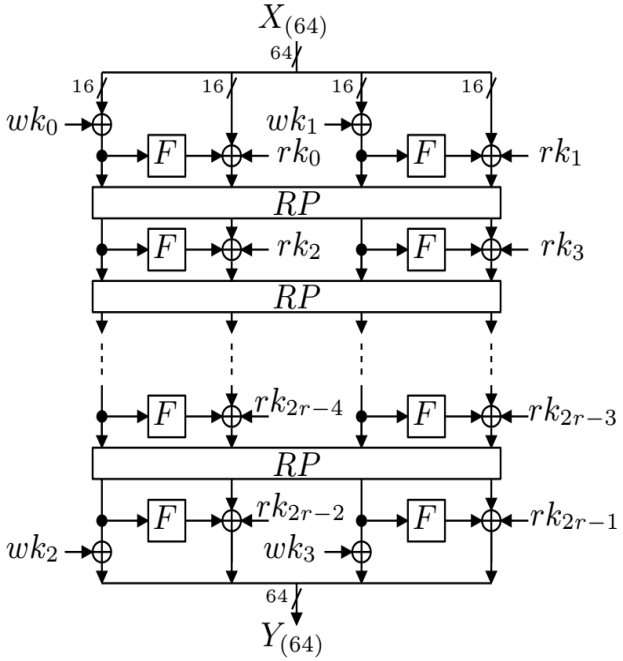


Fig. 1. Diagram of the Gr Encryption function [18].

Function F is an essential component of the Piccolo algorithm, significantly enhancing its encryption security.

As shown in Figure 2(a), the F-function: {0, 1}^16 → {0, 1}^16 is a 16-bit function consisting of two layers of S-boxes (shown in Figure 2(b)), processed through a diffusion matrix.

Each S-Box layer includes four bijective 4-bit S-Boxes, with their input-output relationships shown in hexadecimal format in Table I. From a hardware design perspective, the S-Box is efficient, utilizing only four NOR gates, three XOR gates, and one XNOR gate.

TABLE I
CONFIGURATION OF THE PICCOLO ALGORITHM'S S-BOX [18]

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
S[x]	e	4	b	2	3	8	0	9	1	a	7	f	6	c	5	d

The 16-bit data X is updated by applying the substitution

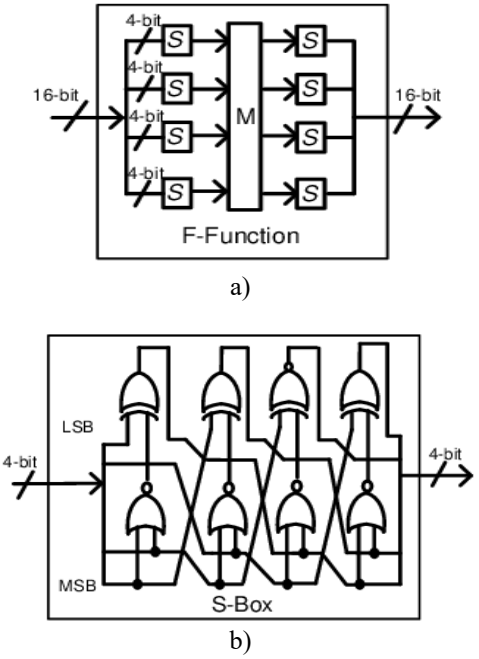


Fig. 2. Piccolo algorithm (a) Structure of the F-function, (b) Substitution box structure [18].

function S independently to its four 4-bit components [18].

$$(x_{0(4)}, x_{1(4)}, x_{2(4)}, x_{3(4)}) \leftarrow (S(x_{0(4)}), S(x_{1(4)}), S(x_{2(4)}), S(x_{3(4)})) \quad (1)$$

In the Piccolo-80 encryption algorithm, the MixColumn operation utilizes a 4×4 matrix M over GF(2^4) (Galois Field of size 2^4) to introduce diffusion. The diffusion matrix M is defined as:

$$M = \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix}$$

The 16-bit data X (16) is updated using the diffusion matrix M by multiplying the transpose of the 4-bit components with M.

$$^{t}(x_{0(4)}, x_{1(4)}, x_{2(4)}, x_{3(4)}) \leftarrow M.\ ^{t}(x_{0(4)}, x_{1(4)}, x_{2(4)}, x_{3(4)}) \quad (2)$$

The symbol $^{t}(a)$ indicates the transpose of the matrix or vector a. Each element in this matrix represents an element in GF(2^4), where multiplication follows the irreducible polynomial modulus $x^4 + x + 1$. A 64-bit to 64-bit bijective mapping is performed by the Round Permutation (RP) function. First, the 64-bit input is divided into eight equal 8-bit blocks, and then the permutation is applied. Figure 3 illustrates the configuration of RP.

$$(x_{0(8)}, x_{1(8)}, \ldots, x_{7(8)}) \leftarrow (x_{2(8)}, x_{7(8)}, x_{4(8)}, x_{1(8)}, x_{6(8)}, x_{3(8)}, x_{0(8)}, x_{5(8)})) \quad (3)$$

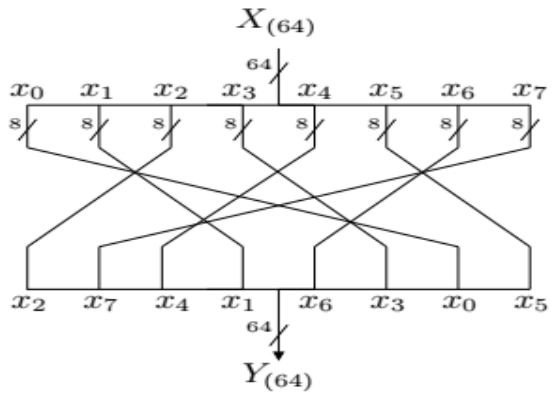


Fig. 3. Round Permutation [18].

B. Key scheduling section

The key scheduling in the Piccolo cipher generates two types of keys: the whitening key (*wki*), used during the initial and final rounds of encryption, and the round keys (*rki*), which are unique for each encryption round. In 80-bit mode, KS^{80} divides the primary key into five separate 16-bit subkeys, from which it generates the whitening and round keys. Each subkey is split into two 8-bit halves, and constants con^{80}_{i} and con^{128}_{i} are used in the respective key scheduling functions. Algorithm II shows the steps used in the process.

Algorithm II: key scheduling

$wk_0 \leftarrow kL_0 \mid kR_1,\ wk_1 \leftarrow kL_1 \mid kR_0$
$wk_2 \leftarrow kL_4 \mid kR_3,\ wk_3 \leftarrow kL_3 \mid kR_4$
For i $\leftarrow 0$ *to* $(r-1)$ *do*
$(rk_{2i}, rk_{2i+1}) \leftarrow (Con^{80}{}_{2i}, Con^{80}{}_{2i+1}) \oplus \begin{cases} (k_2, k_3) \\ (k_0, k_1) \\ (k_4, k_4) \end{cases}$
(k_2, k_3), *if* $i \bmod 5 = 0$ *or* 2,
(k_0, k_1), *if* $i \bmod 5 = 1$ *or* 4,
(k_4, k_4), *if* $i \bmod 5 = 3$,

Notably, the Piccolo encryption algorithm's linear layer involves a bitwise XOR operation applied to the state during each round, which effectively mixes the data bits by combining them with the round key. The linear layer is essential for achieving diffusion and is implemented in the MixColumn stage. Additionally, this layer performs a matrix multiplication in GF(2^4) between the input vectors and a fixed matrix, operating on four 16-bit half-blocks using a 4×4 matrix. This process combines parts of the input half-blocks and enhances the dependency between the input and output, which improves resistance to linear attacks.

The primary challenge of matrix multiplication in the Piccolo encryption algorithm is its computational complexity and the associated hardware implementation costs, which involve multiple bitwise addition and multiplication operations. Additionally, implementing it in hardware increases delay and resource consumption, and the algorithm can be implemented using either a serial or parallel architecture.

This work employs a parallel GF(2^4) multiplier, allowing the multiplication process to be completed in a single clock cycle. Overall, the structure can be implemented with either 16 XOR and 16 NAND gates or 18 XOR and 12 AND gates, where the critical path consists of 4 XOR gates and a single NAND/AND gate. Typically, a parallel GF(2^m) architecture includes (m + 1) mod-2 adders, each with m inputs, m × (m + 1) two-input AND gates, and m XOR gates that operate on two inputs [10]. To design a GF(2^4) multiplier based on polynomial representation that computes the product of two input vectors $P = (p3, p2, p1, p0)$ and $Q = (q3, q2, q1, q0)$, standard polynomial multiplication is first applied to the inputs. The result is then simplified using the irreducible polynomial $f(x) = x^4 + x + 1$ as the modulus. Therefore, the resulting expression is:

$$P(x).Q(x) modf(x) = (p_0q_3 + p_1q_2 + p_2q_1 + p_3q_0 + p_3q_3)\,x^3 + (p_0q_2 + p_1q_1 + p_2q_1 + p_2q_0 + p_2q_3 + p_3q_2 + p_3q_3)\,x^2 + (p_0q_1 + p_1q_0 + p_1q_3 + p_2q_2 + p_2q_3 + p_3q_1 + p_3q_2)\,x + (p_0q_0 + p_1q_3 + p_2q_2 + p_3q_1) \quad (4)$$

Equation (4) can be simplified and expressed in the following form [14]:

$$P(x).Q(x)\ modf(x) = t_3x^3 + t_2x^2 + t_1x + t_0 \quad (5)$$

Where:

$$t_0 = p_0q_0 \oplus p_1q_3 \oplus p_2q_2 \oplus p_3q_1$$

$$t_1 = p_0q_1 \oplus p_1q_0 \oplus p_1q_3 \oplus p_2q_2 \oplus p_2q_3 \oplus p_3q_1 \oplus p_3q_2$$

$$t_2 = p_0q_2 \oplus p_1q_1 \oplus p_2q_1 \oplus p_2q_0 \oplus p_2q_3 \oplus p_3q_2 \oplus p_3q_3$$

$$t_3 = p_0q_3 \oplus p_1q_2 \oplus p_2q_1 \oplus p_3q_0 \oplus p_3q_3 \quad (6)$$

A Boolean-based implementation of a GF(2^4) parallel multiplier is shown in Figure 4.

III. Memristor-Based Architecture Design For Piccolo-80

The use of memristors in cryptographic hardware design is increasing due to their unique properties, including non-volatility and low energy consumption. This enables the Piccolo-80 cryptographic algorithm to take advantage of a

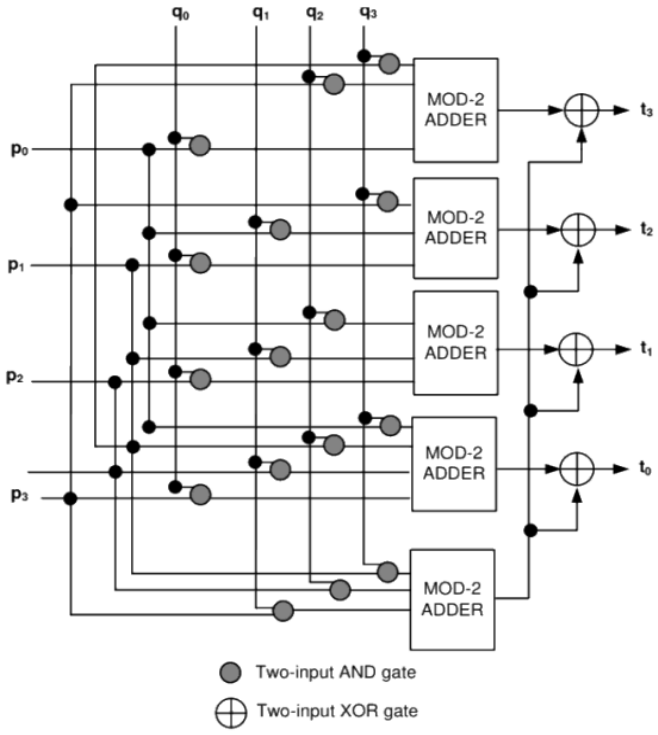


Fig. 4. Parallel GF(2^4) multiplier implemented with mod-2 adders and logic gates [10].

memristor-based architecture designed to enhance efficiency while reducing power consumption and hardware size. As a result, memristors can perform logic functions vital for cryptographic processes in encryption algorithms, such as those used in Piccolo-80. VTM-based logic gates, including NAND, NOR, AND, OR, XOR, and XNOR, provide a compact and energy-efficient alternative to traditional CMOS designs. These gates enable single-step logic operations, enhancing both speed and simplicity in hardware implementations.

A. Memristors

Memristors are two-terminal devices with unique properties, including bidirectionality, process compatibility, and non-volatility [25]. These features make memristors highly valuable for security applications. They operate with very low power consumption and provide high computing speeds, making them useful in applications such as digital memory, logic circuits, and hardware security systems. Their inherent qualities offer several advantages over CMOS-based security methods. Additionally, the non-volatile nature of memristors allows them to retain information without power, which is crucial for secure and reliable storage solutions.

One of the earliest uses of memristors in hardware security is the development of Physically Unclonable Functions (PUF) [26]. Memristor-based PUFs leverage natural process variations in memristor devices to generate unique and unpredictable responses to challenges, making them ideal for device authentication and secure key generation. Moreover, memristors are utilized in the creation of True Random Number Generators (TRNGs) [27]. These memristor-based TRNGs utilize the inherent randomness and process variation of memristors to produce random numbers, enhancing cryptographic security. The authors have also proposed a VTM architecture [28–31] to support the efficient implementation of memristive-based digital circuits.

B. Implementation of NAND/NOR logic circuit via VTM method

The stateful NAND/NOR logic gate shown in [28], as depicted in Figure 5, uses a common structure that can be configured to perform either NAND or NOR operations based on two different input voltages. The logic operation occurs in a single step. The configuration comprises two negatively biased input memristors, Input Memristor 1 and Input Memristor 2, and one output memristor. Voltages In1 and In2 are applied to the respective input memristors, causing the output memristor (Mout) to switch its resistance state to either a Low Resistance State (LRS) or a High Resistance State (HRS), corresponding to the resulting logic value. The LRS and HRS refer to the low and high resistance levels of the output memristor, respectively.

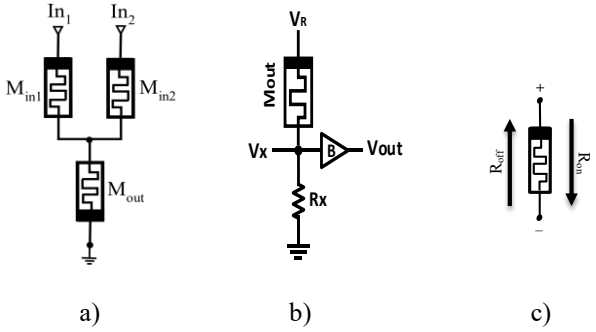


Fig. 5. a) Design of NAND/NOR, b) Circuit for reading output states, c) Memristor switching direction [28].

LRS typically indicates a binary ‘1’, and HRS represents a binary ‘0’.

Table II shows the operation of the introduced NAND/NOR gate, where +V corresponds to logic ‘1’ and 0 V to logic ‘0’.

TABLE II

LOGIC TRUTH TABLE FOR THE DESIGNED NAND/NOR [28].

In1	In2	Resistance State	NAND Output	Resistance State	NOR Output
0	0	LRS	‘1’	LRS	‘1’
0	1	LRS	‘1’	HRS	‘0’
1	0	LRS	‘1’	HRS	‘0’
1	1	HRS	‘0’	HRS	‘0’

C. Implementation of AND/OR logic circuit via VTM method

The introduced AND/OR logic gate [31], shown in Figure 6, has a configuration similar to that of the NAND/NOR gate, except that the input memristors are connected at the positive terminal.

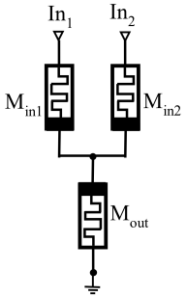


Fig. 6. Design of AND/OR [31].

Table III summarizes the logical output of the AND/OR gates presented in [31].

TABLE III
Logic Truth Table for the Designed AND/OR [31].

In1	In2	Resistance State	AND Output	Resistance State	OR Output
0	0	HRS	'0'	HRS	'0'
0	1	HRS	'0'	LRS	'1'
1	0	HRS	'0'	LRS	'1'
1	1	LRS	'1'	LRS	'1'

D. Design of XOR and XNOR logic gates via VTM approach

As shown in [31], the XOR and XNOR gates are depicted in Figures 7(a) and 7(b), respectively. The XNOR gate can be designed by placing the output memristor downward. Importantly, all gates introduced through the VTM method operate with a single-step process.

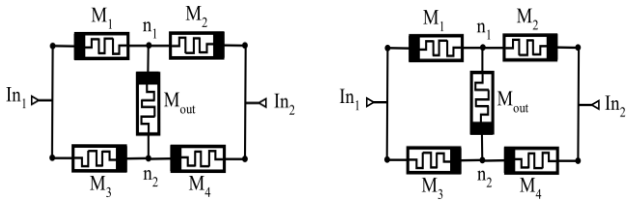


Fig. 7. Implementation of a) XOR b) XNOR gate using the VTM approach [31].

Tables IV and V present the truth table for the proposed XOR and XNOR gates, respectively.

TABLE IV
Logic Truth Table for the Designed XOR [31].

In1	In2	Resistance State	XOR Output
0	0	HRS	'0'
0	1	LRS	'1'
1	0	LRS	'1'
1	1	HRS	'0'

TABLE V
Logic Truth Table for the Designed XNOR [31].

In1	In2	Resistance State	XNOR Output
0	0	LRS	'1'
0	1	HRS	'0'
1	0	HRS	'0'
1	1	LRS	'1'

Figure 8 shows a full adder built with the gates designed using the VTM approach, with the Sum and Carry outputs stored in the relevant output memristors. Mod-2 adders, which are typically constructed using XOR gates, are employed to perform binary addition without carry propagation. In Three-input designs, the mod-2 sum (A ⊕ B ⊕ Cin) matches the sum output of a full adder.

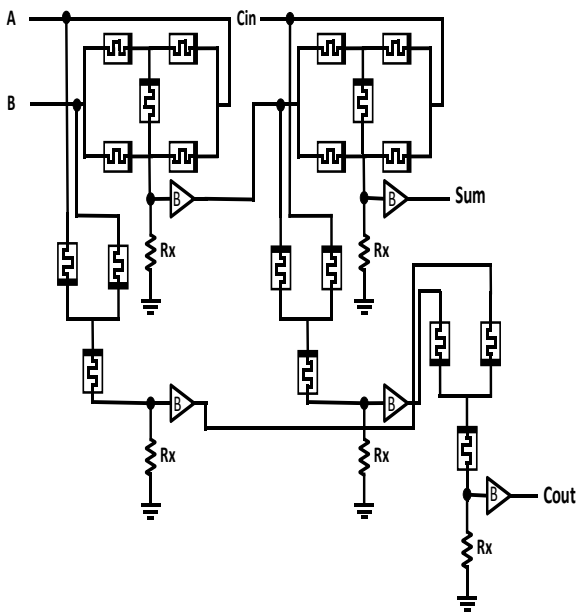


Fig. 8. Schematic diagram of the proposed full adder [31].

Table VI shows the truth table for the proposed full adder.

TABLE VI
LOGIC TABLE OF THE FULL ADDER IMPLEMENTED IN [31].

Logic inputs				Full Adder Outputs	
A	B	Cin	Logic Combination	Resistance State (Sum)	Resistance State (Carry)
0	0	0	“000”	HRS	HRS
0	0	1	“001”	LRS	HRS
0	1	0	“010”	LRS	HRS
0	1	1	“011”	HRS	LRS
1	0	0	“100”	LRS	HRS
1	0	1	“101”	HRS	LRS
1	1	0	“110”	HRS	LRS
1	1	1	“111”	LRS	LRS

E. Proposed hardware implementation of the Piccolo algorithm using memristor-based VTM stateful logic gates.

The complete Piccolo encryption algorithm can be implemented within a memristor-based architecture using the VTM method. The Piccolo cipher utilizes a GFN, where the internal state is divided into four 16-bit words, resulting in four processing branches. This layout is illustrated in Figure 9.

In each round of the Piccolo algorithm, two FN operations are performed. Each FN includes two steps: the F-function and the AddKey operation. The F-function is a non-keyed 16×16-bit transformation applied to the first branch of the FN. Additionally, each round uses two round keys, one for each FN.

Additionally, the algorithm uses pre- and post-whitening keys: *wk0* and *wk1*, which are combined with the internal state via bitwise XOR before the first round, while *wk2* and *wk3* are applied after the final round. A byte-level permutation occurs after the two FN operations in each round. It is assumed that both whitening and round keys can be accessed either through pre-configuration stored in memory or generated dynamically at runtime.

Figure 10 illustrates the complete implementation of the Piccolo-80 algorithm, constructed using VTM stateful logic gates. In this architecture, each VTM gate not only computes its output but also stores it directly in the output memristor; thus, the 64-bit state of the algorithm is maintained throughout the computation directly within the memristive cells, leading to a significant reduction in power consumption and hardware area compared to conventional CMOS designs.

In this design, a 64-bit input is initially split into four 16-bit words, each of which is divided into four 4-bit nibbles for parallel processing. At the start and end of the encryption, whitening keys (*wks*) are XORed with the half-blocks, while round keys (*rks*) are added at specific points during the rounds, especially after nonlinear layers or between core transformations. Each round includes the F-function, which first passes the nibbles through a fixed 4-bit S-box, a nonlinear layer. As shown in Figure 2, the S-box consists of only four NOR gates, three XOR gates, and one XNOR gate, all of which can be fully implemented using the proposed VTM logic gates. The output then moves to the MixColumn layer, where a diffusion matrix over GF(2^4) with the irreducible polynomial $x^4 + x + 1$, as defined in Equation (6), provides strong bit-level diffusion.

This is followed by another S-box layer applied to the nibbles. The result is XORed with the round key and then passed to the RP block, which applies a fixed permutation to the nibbles and rapidly propagates local changes throughout the entire 64-bit state.

This process repeats for 25 rounds in Piccolo-80 (and 31 rounds in Piccolo-128), with the final whitening applied to produce the 64-bit ciphertext. All these components—including the S-boxes, MixColumn, and RP—are implemented with VTM gates. Four S-boxes operate in parallel on each 16-bit word, while the MixColumn block is designed according to the hardware circuit in Figure 4 without needing CMOS multipliers or additional registers. In each round, the architecture requires 94 XOR/XNOR gates, 68 AND/OR gates, and 32 NAND/NOR gates. Simulation results at 1.8 V supply voltage and 133 MHz operating frequency show that, compared to traditional CMOS and hybrid MeMOS designs, the proposed design significantly reduces power consumption and hardware area.

In the memristor-based Piccolo-80 architecture, essential cryptographic operations, such as GF(2^m) addition and multiplication, are performed using memristor circuits. Memristors enable the development of parallel architectures, optimize the gate count, and decrease delay in the critical path with minimal power use. Furthermore, memristor-based stateful logic can be implemented in parallel configurations, offering the potential for faster computation.

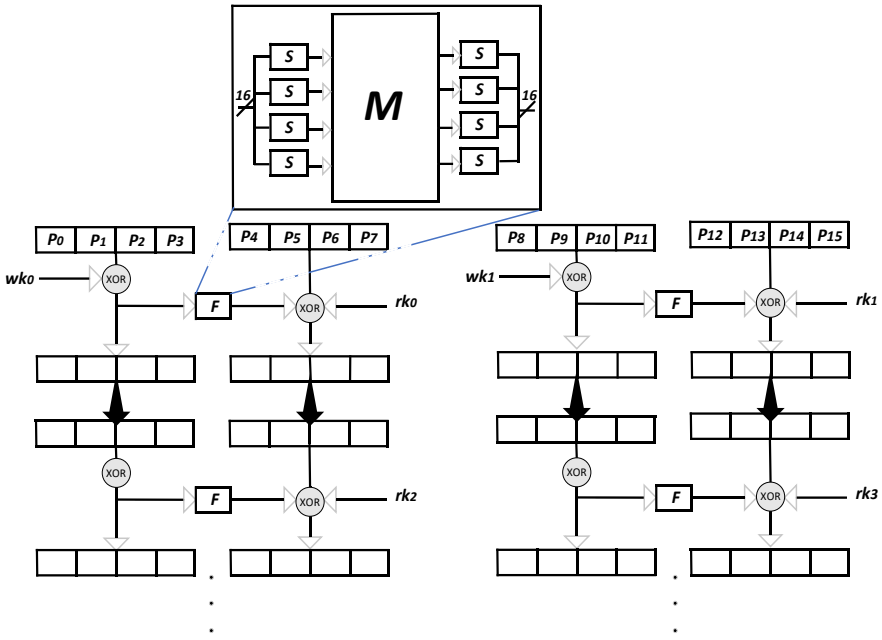


Fig. 9. Piccolo-80 structure [18].

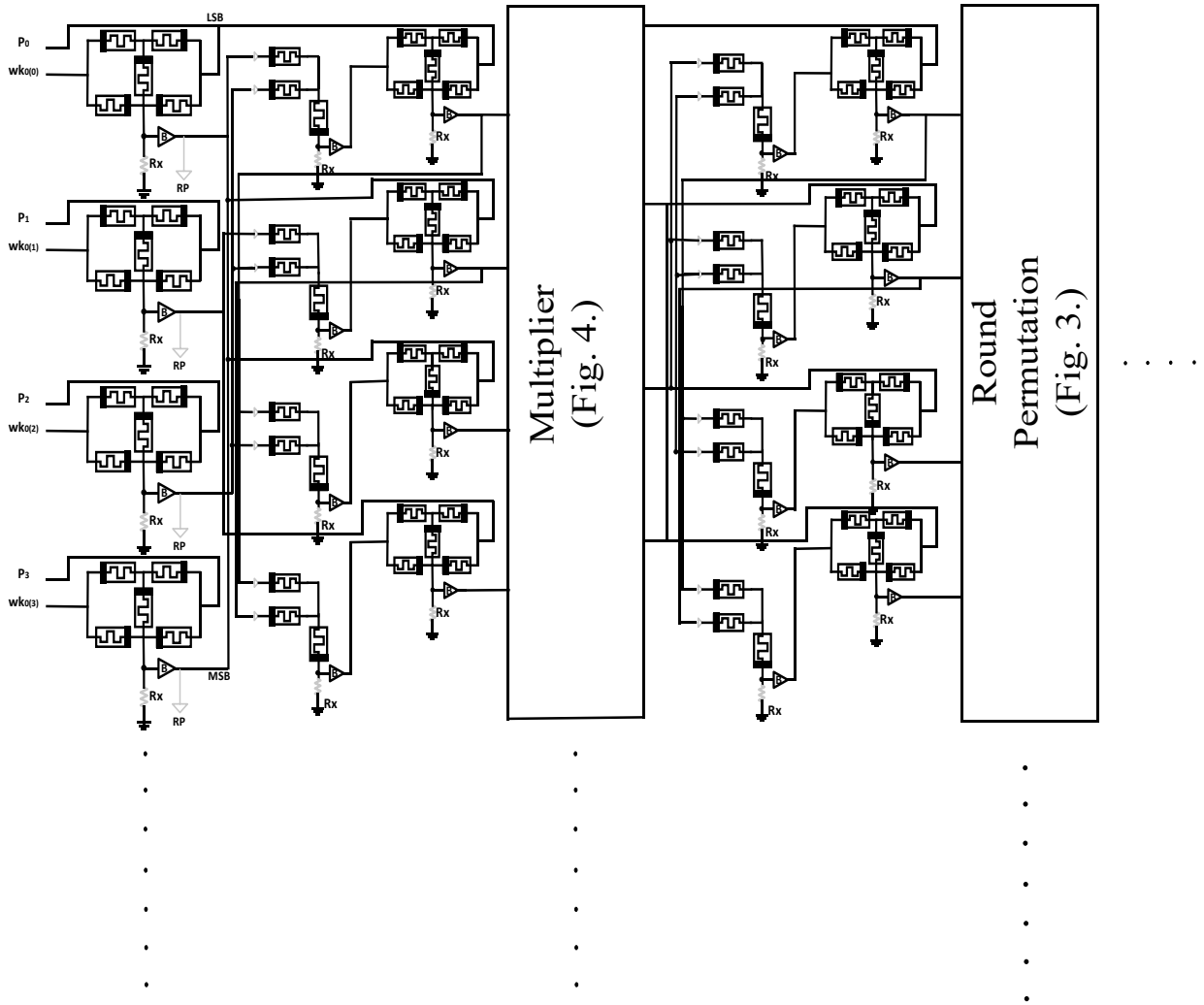


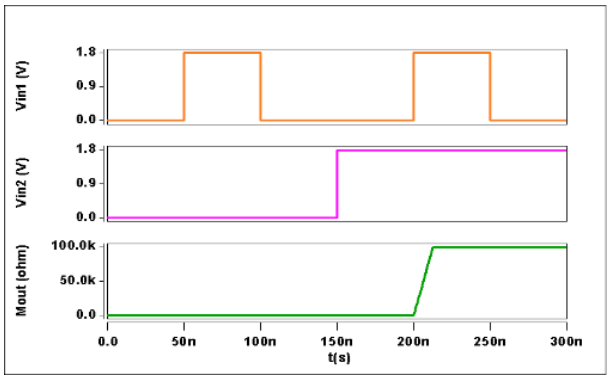
Fig. 10. Proposed architecture of the Piccolo algorithm using VTM stateful logic gates.

IV. Simulation and Results

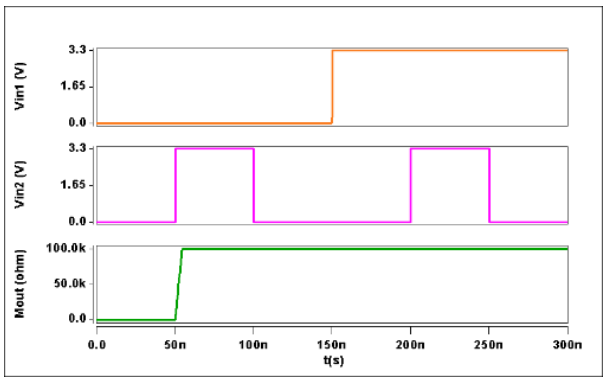
This section demonstrates how to implement the Piccolo-80 algorithm using both FPGA hardware and a memristor-based approach modeled through SPICE netlists in Cadence Virtuoso. Design simulation, validation, and power analysis are performed with Cadence Spectre. The simulations use the JART VCM v1b var memristor model [32].

A. Implementation of NAND/ NOR write logic

The NAND/NOR circuit design was simulated, and the outputs for all input combinations are shown in Figure 11.



a)



b)

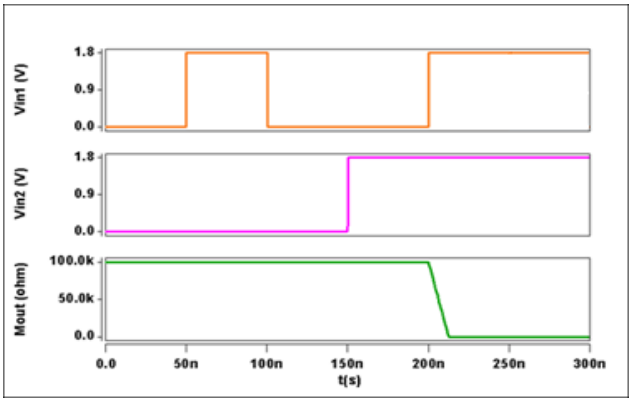
Fig. 11. Simulation results for the write logic operations: (a) NAND operation, (b) NOR operation [28].

B. Implementation of AND/OR Write Logic

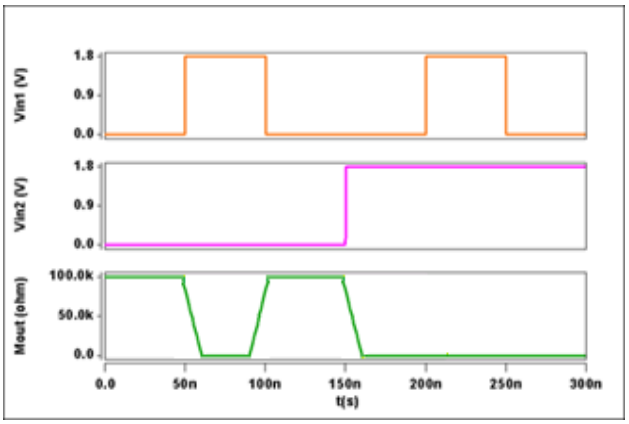
Figure 12 shows the simulation results of the designed AND/OR circuit. The gate's output is indicated by the resistance of Mout, with changes in memristance reflecting the corresponding logic states.

C. Simulation Results for the Read Circuit

By setting Mout=HRS, as shown in Figure 13(a), the simulation results demonstrate the read process corresponding to logic '0'.



a)



b)

Fig. 12. Simulation results of the write operation for a) AND operation, b) OR operation [31].

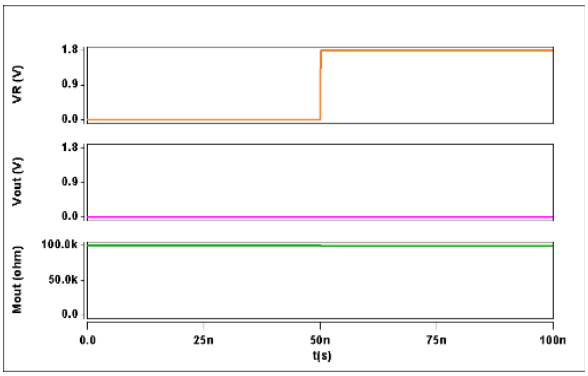
As shown in Figure 13(b), when Mout is set to the LRS, the current does not reach the output memristor during the 0–50 ns interval, resulting in Vout being 0 V.

Between 50 and 100 ns, the output memristor is read, and the resulting voltage, Vout, equals V_R, indicating a logic high state. The simulation data corresponds to the read operation for a logic '1' [28].

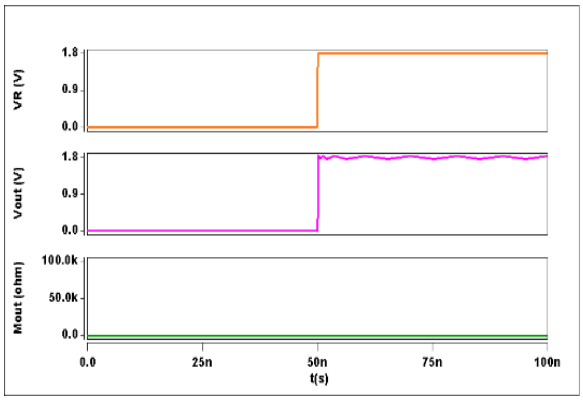
D. Write Logic Using XOR Gate

Figure 14(a) shows the simulated output of the designed XOR circuit. The gate's output depends on the resistance of the output memristor, Mout, where changes in memristance indicate different logic states.

Similarly, Figure 14(b) displays the simulated output of the proposed XNOR gate, confirming that the resistance state of the output memristor varies as expected for each input condition.

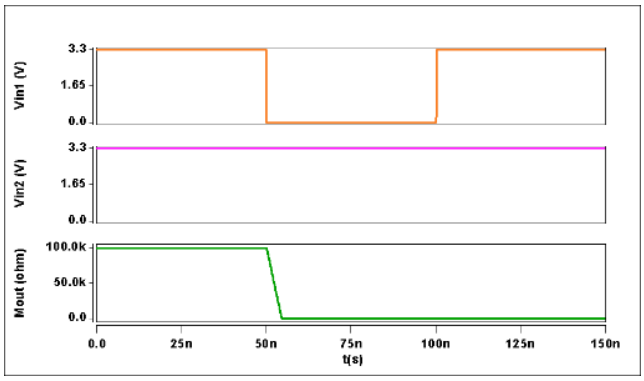


a)

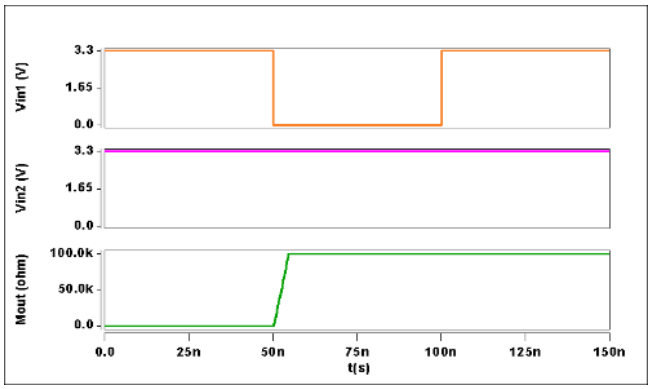


b)

Fig. 13. Simulated output of the designed read circuit for a) Mout= HRS, b) Mout= LRS [28].



a)



b)

Fig. 14. Simulated outputs of the write logic operations: (a) XOR operation, (b) XNOR operation [31].

E. Hardware Implementation of the Piccolo Encryption Algorithm Using FPGA.

The Piccolo-80 lightweight block cipher (64-bit block, 80-bit key, 25 rounds) was implemented in VHDL and simulated in ModelSim. Table VII presents the results, which are verified against standard test vectors to confirm functional correctness. Functional correctness was validated using the standard Piccolo-80 test vectors, proving that the 80-bit key and 64-bit plaintext in Table VII generate the expected ciphertext.

TABLE VII

PICCOLO ALGORITHM 80-BIT KEY TEST VECTOR [18].

Key length	80-bit
Key	$(00112233\ 44556677\ 8899)_{16}$
Plaintext	$(01234567\ 89ABCDEF)_{16}$
Ciphertext	$(8D2BFF99\ 35F84056)_{16}$

According to the architecture of the Piccolo encryption algorithm, the synthesis process is divided into three main parts: the S-Box layer, the F-function stage, and the data processing section. The results from the VHDL-based simulation are displayed in Figure 15.

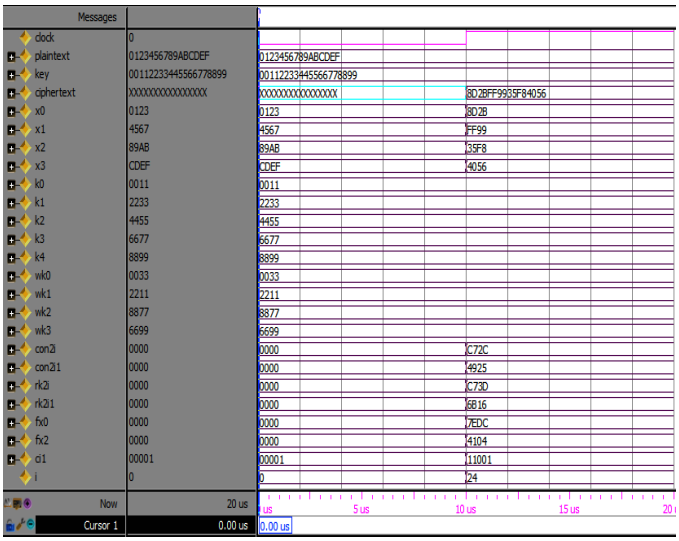


Fig.15. Simulation results for the VHDL implementation of the Piccolo algorithm.

The Piccolo-80 algorithm was then developed and synthesized using Xilinx ISE tools, targeting the Virtex-5 XC5VFX200T FPGA, which is fabricated in 65 nm CMOS technology. In the testbench, a 10 ns clock period (100 MHz) was defined, while post-synthesis timing analysis reported a maximum operating

frequency of approximately 427.716 MHz. Each encryption requires about 100 clock cycles, coordinated by four ALUs, two comparators, and three shift registers, resulting in a throughput of 640 Mbps at 100 MHz. Hardware synthesis was primarily performed to validate functionality and confirm that the VHDL design operates correctly on the FPGA hardware. The resource utilization results are summarized in Table VIII.

TABLE VIII

HARDWARE RESOURCE USAGE FOR THE PICCOLO ALGORITHM IMPLEMENTATION

Resource	Available	Utilization	(%)
Register	122880	289	1%
LUT	122880	291	1%
Slice	30720	104	1%
IO	960	64	6%
BUFG	32	6	18%

Table IX compares the Piccolo-80 algorithm implementation across different FPGA platforms, showing variations in efficiency and power consumption among the proposed schemes.

F. Using proposed stateful logic gates implemented through the VTM approach.

Since the Piccolo algorithm can be implemented using basic Boolean gates, the proposed circuit can fully perform all algorithmic steps, including bit permutations (P-Box), logic gate operations, linear transformations (Diffusion Layer), and Nonlinear Substitution Layer (S-Box). The main advantage of this design is that function F is implemented with VTM memristor-based logic gates. Because memristors can perform logical operations and store data simultaneously, they enable more efficient execution of this function, reduce hardware complexity, and increase processing speed when implementing the Piccolo algorithm.

Using the architecture shown in Figure 10, the Piccolo-80 lightweight block cipher was implemented in a round-based mode. The encryption process employs a 64-bit block and an 80-bit key, with 25 rounds. As previously mentioned, the round keys are assumed to be predefined and stored in memory before the encryption process starts.

The proposed design uses 94 XOR and XNOR gates, 68 AND/OR gates, and 32 NAND/NOR gates per encryption round. Additionally, the circuit operates at a supply voltage of 1.8V, which helps reduce power consumption, and a frequency of 133 MHz.

Also, integrating memristors into the design significantly reduces power consumption and simplifies hardware complexity. The non-volatile nature of memristors also ensures high stability and reliability.

To evaluate the energy efficiency of the proposed implementation, both dynamic and static power components were examined separately using the Cadence Spectre simulator. The results show that static (leakage) power was minimal due to the non-volatile nature of memristors, with over 90% of the total power consumed by dynamic switching activity. Additionally, gate-level simulations reveal that the substitution layer (S-Box), because of its frequent XOR/XNOR operations, accounts for approximately 43% of the total dynamic power. The diffusion matrix and permutation steps consume 29% and 18%, respectively, while key mixing operations contribute the remaining 10%. Furthermore, a frequency sweep analysis indicates that the design scales effectively, with only a moderate linear increase in power as frequency rises, thereby maintaining energy efficiency even at higher operational speeds.

These findings confirm the architectural benefit of the VTM-based approach in lowering active power, particularly in high-speed or energy-sensitive applications. The memristor-based VTM method reduces power consumption, and the small size of memristors results in a significant decrease in the area required compared to traditional hybrid MeMOS implementations.

Table IX

SUMMARY OF PICCOLO-80 HARDWARE IMPLEMENTATION RESULTS ON FPGA PLATFORMS.

Cryptographic Method	FPGA Model	Slices	Look-Up Tables (LUTs)	Freq. (MHz)	Efficiency (Mbps/Slice)	Power Consumption (mW)	Latency (ns)
Piccolo-80 [10]	Xilinx Spartan-6 XC6SLX25	135	419	189	3.44	174	137.81
Piccolo-80 [10]	Xilinx Virtex-4 XC4VLX25	273	525	273	2.45	383	95.69
Piccolo-80 [10]	Xilinx Virtex-5 XC5VLX50T	47	150	315	7.81	500	174.35
Piccolo-80 [10]	Xilinx Spartan-6 XC6SLX16	35	104	183	6.11	70	299.28
Unprotected CMOS [24]	Xilinx Spartan-6 XC6SLX9	64	106	206	2.06	98	485.44
Protected CMOS [24]	Xilinx Spartan-6 XC6SLX9	73	138	188	2.3	105	381.18
Piccolo-80 (This work)	Virtex-5 XC5VFX200T	104	291	428	6.15	334	100.06

TABLE X
COMPARISON OF THE PROPOSED IMPLEMENTATIONS AND OTHER RELATED WORKS ON THE PICCOLO CIPHER.

Implementation	Method	Gate Equivalents (GEs)	Frequency (MHz)	Throughput (Mbps)	Power Dissipation (mW)	Thr./Area (Mbps/GEs)	Latency (ns)
Piccolo-80 [24]	Protected Hybrid MeMOS	1512	112	286.7	25.6	0.189	223.29
Piccolo-80 [24]	Unprotected Hybrid MeMOS	1352	120	307.2	22.5	0.227	208.33
Piccolo-80 (This work)	VTM	1214	133	340	17.4	0.28	188.24

As shown in Table X, the proposed VTM-based design demonstrates clear improvements over prior Hybrid MeMOS implementations. In terms of hardware cost, the VTM implementation requires only 1214 GEs, representing a 19.7% reduction compared to the protected MeMOS design with 1512 GEs and a 10.2% decrease compared to the unprotected configuration with 1352 GEs. Operating at 133 MHz, the VTM design achieves the highest frequency among all platforms. It provides a throughput of 340 Mbps, which is higher than both protected (286.7 Mbps) and unprotected MeMOS (307.2 Mbps) designs. More importantly, power dissipation drops significantly to 17.4 mW, resulting in 32% and 22.6% power savings compared to the protected and unprotected hybrid MeMOS architectures, respectively. Moreover, the throughput-to-area efficiency reaches 0.280 Mbps/GE, which is higher than the 0.189 Mbps/GE and 0.227 Mbps/GE reported for the protected and unprotected MeMOS designs. Additionally, the latency is reduced to 188.24 ns, which is lower than the protected MeMOS at 223.29 ns and the unprotected MeMOS at 208.33 ns, showing reductions of 15.7% and 9.6%, respectively, further emphasizing the speed advantage of the VTM design.

These results confirm that the VTM architecture not only reduces power and area but also improves efficiency, making it a strong option for lightweight and energy-constrained IoT applications. According to Cadence, the power measurements encompass both static and dynamic power. The circuit's power consumption is significantly influenced by the memristor's LRS and HRS. The proposed architecture further highlights the advantages of memristor-based designs for efficient cryptographic operations. It is essential to note that switches are utilized to address the current sneak path problem, allowing each gate in separate rows to operate independently. Specifically, the input memristors are connected in series with NMOS transistors that function as switches.

Figure 16 shows the power dissipation of various Piccolo-80 implementations, including MeMOS-protected, MeMOS-unprotected, and the proposed VTM-based design. As illustrated, the VTM implementation has the lowest power consumption (17.4 mW), representing a clear improvement over both MeMOS designs (25.6 mW and 22.5 mW).

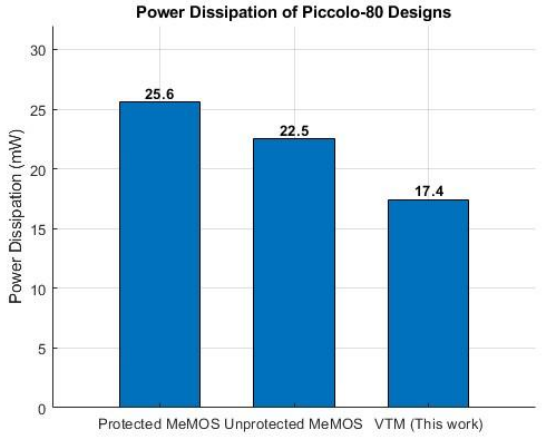


Fig 16. Comparison of Power Consumption in Piccolo-80 Algorithm Implementations.

G. Comparison with Other Lightweight Block Ciphers

To further highlight the importance of the proposed design, this work compares its VTM-based Piccolo-80 implementation with reported hardware results of other well-known lightweight block ciphers. Table XI shows the comparative results of several lightweight encryption algorithms alongside the proposed architecture. As demonstrated, the VTM-based Piccolo-80 achieves one of the smallest hardware areas with only 1214 GEs, significantly less than many widely studied lightweight ciphers. It also features competitive power consumption of 17.4 mW at 133 MHz, maintaining energy efficiency even at a higher operating frequency than most comparable designs, such as PRESENT and LED, and ASCON (evaluated in CMOS and CMOS/MRAM), which are typically evaluated at 100 MHz.

Although its throughput (340 Mbps) and CPD (7.52 ns) are lower than those of high-speed implementations like PRINCE, SIMON, and ASCON, the throughput-to-area ratio remains efficient compared to more complex algorithms, such as CLEFIA and Camellia. Overall, these results suggest that the proposed design achieves a strong balance between low area and power dissipation, while maintaining reliable performance, making it a practical and efficient choice for secure IoT applications.

TABLE XI
COMPARISON OF DIFFERENT LIGHTWEIGHT BLOCK CIPHER IMPLEMENTATIONS.

Cipher Implementation	Technology	Area (GEs)	Throughput (Mbps)	Thr./Area (Mbps/GE)	Power Consumption (mW)	CPD (ns)
ASCON CMOS [4]	--	10152.9	--	--	745 µW (at 100 MHz)	--
ASCON CMOS/MRAM [4]	Hybrid CMOS/MRAM	10715.5	--	--	714.8 µW (at 100 MHz)	--
CLEFIA [12] (flexible)	180 nm CMOS	14,951	2109.6 (128-B K) 1506.86 (192-B K) 1375.823(256-B K)	0.141 (128-B K) 0.101 (192-B K) 0.092 (256-B K)	--	4.045
PRINCE [13] (Low-cost)	180 nm CMOS	7046	2857.653	0.406	--	2.036
PRESENT [14] (flexible)	180 nm CMOS	4214	1492.54 (64/128-B K)	0.354	10.397 (at 100 MHz)	1.34
LED [14] (flexible)	180 nm CMOS	3556	680.3 (64-B K) 1010.1 (128-B K)	0.191 (64-B K) 0.284 (128-B K)	8.751 (at 100 MHz)	1.92
Camellia [15] (128/192/256)	180 nm CMOS	19,142	1271.73 (128/192-B K) 1059.78 (256- B K)	0.066 (128/192-BK) 0.055 (256- B K)	29.943 µW (at 100 KHz)	6.71
SIMON [16] (flexible)	180 nm CMOS	5647.2	2760.76	0.489	14.196 (at 100 MHz)	0.776
SPECK [16] (flexible)	180 nm CMOS	6170.65	1254.83	0.203	15.543 (at 100 MHz)	2.282
Piccolo (This work)	Memrisor-Based	1214	340 (80-B K)	0.28	17.4 (at 133 MHz)	7.52

Abbreviation: GE: Gate Equivalent, Thr.: Throughput, CPD = Critical Path Delay, B=bit, K=keys

H. Security Analysis

Nonlinear operations in the Piccolo cipher, particularly the S-Box layer results in significant power leakage, making it a prime target for Differential Power Analysis (DPA) attacks [14]. Power consumption in CMOS circuits is primarily determined by dynamic switching activity. Consequently, side-channel leakage arises from correlations between this switching activity and the processed key bits. An attacker can measure the power at specific times and detect even slight variations in switching activity, which may help infer the key bits. S-Box operations are a significant source of side-channel leakage, allowing attackers to exploit power analysis to recover whitening and round keys. In the proposed hardware architecture, using VTM-based memristor logic—which is non-volatile and stores its state as resistance—helps reduce power leakage. These gates retain data through resistance rather than electric charge, allowing many logic operations to occur with minimal changes in the circuit. This feature prevents large fluctuations in voltage or current, thereby lowering switching activity, stabilizing power consumption, and making it more difficult for attackers to observe and analyze. As a result, the design achieves lower switching activity, fewer power fluctuations, and more stable power behavior at the circuit level, which enhances resistance against DPA. However, this finding is based on circuit-level architecture and simulation. A thorough assessment of information leakage involves statistical methods, such as the Test Vector Leakage Assessment (TVLA) t-test or Correlation Power Analysis (CPA). Although these experiments are beyond this study's scope, they are important directions for future research.

V. CONCLUSION

This paper introduces an efficient hardware implementation of the Piccolo-80 encryption algorithm using memristor-based logic gates with the VTM method. Memristor-based stateful logic gates can change resistance under voltage control and retain their value even after the voltage is removed, which significantly reduces overall power consumption. The proposed architecture for each round utilizes only 194 memristor-based gates, including NAND/NOR, AND/OR, XOR, and XNOR, which can switch functions via voltage control. It performs all the primary operations of the Piccolo algorithm, including P-Box permutations, S-Box substitutions, linear propagation, and nonlinear transformations, while maintaining the round-based encryption structure of the algorithm.

This preserves full compatibility with the original Piccolo algorithm structure while utilizing the computational advantages of memristor-based algorithms. Additionally, this

implementation offers notable improvements in power and area efficiency compared to earlier related designs. As a result, static power is almost eliminated, dynamic power is significantly reduced, and overall area is minimized. The memristor-based VTM approach achieves power savings of 32% and 22.6% over protected and unprotected MeMOS hybrid designs, respectively, while reducing hardware area by 19.7% and 10.2%. Simulation results from Cadence Spectre software confirm the design's power and hardware efficiency. The implementation consumes only 17.4 mW of power at 1.8 V and 133 MHz, utilizing less space than comparable designs. Furthermore, the throughput-to-area efficiency achieves 0.280 Mbps/GE, representing an improvement over the 0.189 Mbps/GE and 0.227 Mbps/GE reported for the protected and unprotected MeMOS designs. Additionally, gate-level power analysis showed that the substitution layer consumes approximately 43% of the dynamic power, followed by diffusion (29%), permutation (18%), and key mixing (10%). This breakdown highlights the main power hotspots and verifies the effectiveness of VTM-based optimization.

These features make it ideal for resource-constrained environments, such as IoT devices, where optimizing power and area is crucial. Furthermore, utilizing the VTM method for logic gates enhances security by reducing power leakage and increasing resistance to DPA attacks, thereby making the design more robust for secure IoT applications.

REFERENCES

[1] Abhijan Bhattacharyya, "Internet of Things," - Technology, Applications and Standardization, Published 2018, doi: 10.5772/Intech open 70907

[2] M. S. Turan, K. A. McKay, J. Kang, J. Kelsey, and D. Chang, "Ascon-Based Lightweight Cryptography Standards for Constrained Devices: Authenticated Encryption, Hash, and Extendable Output Functions," *NIST Special Publication 800-232*, Aug. 2025.

[3] M. A. Khan, M. Ahmad, M. Aslam, M. W. Aslam, and S. A. Parah, "Securing the IoT ecosystem: ASIC-based hardware realization of Ascon lightweight cipher," *International Journal of Information Security*, Springer, 2024.

[4] Roussel, N., Potin, O., Di Pendina, G., Dutertre, J. M., Rigaud, J.B.: CMOS/STT-MRAM based Ascon LWC: A power efficient hardware implementation. In: *2022 29th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pp. 1–4. IEEE (2022)

[5] A. R. Alharbi, A. Aljaedi, A. Aljuhni, M. K. Alghuson, H. Aldawood, and S. S. Jamal, "Evaluating Ascon hardware on 7-series FPGA devices," *IEEE Access*, vol. 12, pp. 149076–149089, 2024, doi: 10.1109/ACCESS.2024.3471694

[6] M. A. Siddiqi, J. A. G. Hernández, A. Gebregiorgis, R. Bishnoi, C. Strydis, S. Hamdioui, and M. Taouil, "Memristor-Based Lightweight Encryption," *arXiv preprint* arXiv:2404.00125, 2024.

[7] H. Ajmi, F. Zayer, and H. Belgacem, "In-Memory Computing Architecture for Efficient Hardware Security," *arXiv preprint* arXiv:2408.11570, 2024.

[8] K. R. Penumalli, T. R. Kadiyam, V. Birudu, V. Gonuguntla, and R. Vaddi, "Design of an SLIM Cipher S-Box with 8T-SRAM CiM for Energy-Efficient, Lightweight, and DPA-Resistant Edge AI," *Engineering, Technology & Applied Science Research*, vol. 15, no. 4, pp. 25902–25914, Aug. 2025. doi: 10.48084/etasr.11870.

[9] É. C. Dutra e Silva Junior, C. A. de M. Cruz, I. A. L. Saraiva, F. G. Santos, C. R. P. dos Santos Junior, L. S. Indrusiak, W. A. Finamore, and M. Glesner, "Chaos-Based S-Boxes as a Source of Confusion in Cryptographic Primitives," *Electronics*, vol. 14, no. 11, p. 2198, 2025. doi: 10.3390/electronics14112198.

[10] M. Masoumi, "Design and Evaluation of a Power Analysis Resilient Implementation of Piccolo-80 Lightweight Encryption Algorithm," *Journal of Hardware and Systems Security*, vol. 7, pp. 101–109, 2023, doi: 10.1007/s41635-023-00191-1.

[11] B. Rashidi, "High-throughput and lightweight hardware structures of HIGHT and PRESENT block ciphers," *Microelectronics Journal*, vol. 90, pp. 232–252, 2019, doi: 10.1016/j.mejo.2019.06.012.

[12] B. Rashidi, "High-throughput and flexible ASIC implementations of SIMON and SPECK lightweight block ciphers," *International Journal of Circuit Theory and Applications*, vol. 47, no. 8, pp. 1254–1268, 2019, doi: 10.1002/cta. 2645.

[13] B. Rashidi, "Efficient and flexible hardware structures of the 128-bit CLEFIA block cipher," *IET Computers & Digital Techniques*, vol. 14, no. 2, pp. 69–79, 2020, doi: 10.1049/iet-cdt.2019.0157.

[14] B. Rashidi, "Low-cost and two-cycle hardware structures of the PRINCE lightweight block cipher," *International Journal of Circuit Theory and Applications*, vol. 48, no. 8, pp. 1227–1243, 2020, doi: 10.1002/cta. 2832.

[15] B. Rashidi, "Flexible structures of lightweight block ciphers PRESENT, SIMON and LED," *IET Circuits, Devices & Systems*, vol. 14, no. 3, pp. 369–380, 2020, doi: 10.1049/iet-cds.2019.0363.

[16] B. Rashidi, "Flexible and high-throughput structures of Camellia block cipher for security of the Internet of Things," *IET Computers & Digital Techniques*, vol. 15, no. 2, pp. 171–184, 2021, doi: 10.1049/cdt2.12025.

[17] B. Rashidi, "Glitch-less hardware implementation of block ciphers based on an efficient glitch filter," *Integration, the VLSI Journal*, vol. 85, pp. 20–26, 2022, doi: 10.1016/j.vlsi.2022.02.007.

[18] Shibutani K, Isobe T, Hiwatari H, Mitsuda A, Akishita T, Shirai T. "Piccolo: an ultra-lightweight blockcipher, *International Workshop on Cryptographic Hardware and Embedded Systems*," 2011, Sep 28, pp. 342-357, Springer, Berlin, Heidelberg.

[19] Ramu, G., Mishra, Z., Singh, P., & Acharya, B. (2020). "Performance optimized architectures of Piccolo block cipher for low resource IoT applications," *International Journal of High-Performance Systems Architecture, 9*(1), 49–57.

[20] Serhii Naumenko; Inna Rozlomii; Andrii Yarmilko, "The Built on Feistel Network Architecture Block Ciphers Modification," *14th International Conference on Advanced Computer Information Technologies (ACIT).* September 2024, doi:10.1109/ACIT62333.2024.10712597.

[21] J. Feng, Y. Wei, B. Wei, "Novel optimized implementations for the Piccolo cipher based on field-programmable gate arrays," *International Journal of Circuit Theory and Applications*, vol. 53, no. 2, pp. 771–789, 2025, doi:10.1002/cta. 4160.

[22] G. Ramu; Z. Mishra; P. Singh; B. Acharya, "Hardware implementation of Piccolo Encryption Algorithm for constrained RFID application," in *Proc. 9th Annu. IEMECON – Inf. Technol., Electromech. Eng. Microelectron. Conf.*, 2019, pp. 85–89. doi: 10.1109/IEMECONX.2019.8877071.

[23] Mishra, S., Mishra, Z., & Acharya, B. "Area Optimized Hardware Architecture of Piccolo-80 Lightweight Block Cipher," *Proceedings of Fifth International Conference on Microelectronics, Computing and Communication Systems*, 2021, pp 341–350.

[24] Masoumi, M. (2019). "Design and Evaluation of Memristor-Based Piccolo-80 Lightweight Encryption Algorithm for Future IoT," *J. Inf. Secur. Appl.,* vol. 48, 102371, 2019. doi: 10.1016/j.jisa.2019.102371.

[25] L. Chua, "Memristor-the missing circuit element," *IEEE Transactions on Circuit Theory*, vol. 18, pp. 507-519, 1971.

[26] Chandranshu Gupta, Gaurav Varshney. "A Lightweight and Secure PUF-Based Authentication and Key-exchange Protocol for IoT Devices," *arXiv:2311.04078v1 [cs.CR]*, 7 Nov 2023.

[27] Xu, H., Qu, W., & Xu, Q. (2020). "Memristor-based true random number generator," *IEEE Transactions on Circuits and Systems II*: Express Briefs, 67(10), 1860-1864.

[28]. F. Mozafari, M. J. Sharifi, M. Ahmadi, A. Ahmadi, "A novel memristive architecture for memristor-based logic," *Proc. IEEE Int. Midwest Symp. Circuits Syst. (MWSCAS)*, pp. 812–815, 2021.

[29]. F. Mozafari, M. Ahmadi, A. Ahmadi, "Design of a new memristive-based architecture using VTM method," *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, pp. 980–984, 2022.

[30]. F. Mozafari, M. Ahmadi, A. Ahmadi, "A programmable circuit based on the combination of VTM cellular crossbars," *Proc. 19th Int. Conf. Synthesis, Modeling, Analysis, Simulation Methods Appl. Circuit Design (SMACD)*, pp. 1–4, 2023.

[31]. F. Mozafari, M. Ahmadi, and A. Ahmadi, "Design and implementation of full adder circuit based on VTM-logic gates," *Proc. IEEE 66th Int. Midwest Symp. Circuits Syst. (MWSCAS)*, pp. 389–393, 2023.

[32]. EMRL. (2020) JART – J¨ulich Aachen Resistive Switching Tools. *[Online]*. Available: http://www.emrl.de/JART#Artikel 1.

Farzad Mozafari (Member, IEEE) is currently pursuing a Ph.D. degree in Electrical Engineering at the University of Windsor, Windsor, ON, Canada. His research explores Memristor-Based Lightweight Encryption. His broader research interests include hardware security, emerging non-volatile memory technologies, VLSI design, cryptographic architecture, and energy-efficient computing systems. He is particularly interested in the intersection of nanoscale devices and secure hardware design, focusing on developing scalable, low-power solutions for next-generation embedded and IoT systems.

Majid Ahmadi (Life Fellow, IEEE) received the B.Sc. degree in Electrical Engineering from Sharif University of Technology (formerly Arya Mehr University), Tehran, Iran, in 1971, and the Ph.D. degree in Electrical Engineering from Imperial College London, U.K., in 1977. He joined the Department of Electrical and Computer Engineering at the University of Windsor, Canada, in 1980, where he is currently a University Professor and Director of the Research Center for Integrated Microsystems. His research interests include digital signal processing, machine vision, pattern recognition, neural networks, VLSI design, and computer arithmetic. He has coauthored a book and published over 650 papers. He has served as Regional Editor for the Journal of Circuits, Systems, and Computers and Associate Editor for the Pattern Recognition journal. His honors include the Honorable Mention Award from Pattern Recognition (1992) and the Distinctive Contributed Paper Award from the Multiple-Valued Logic Conference (1999).