materials

Article

Comparative Analysis of Reconfigurable Platforms for Memristor Emulation

Margarita Mayacela [1,*], Leonardo Rentería [2], Luis Contreras [1] and Santiago Medina [1]

1 Faculty of Civil and Mechanical Engineering, Research and Development Directorate, Technical University of Ambato, Ambato 180207, Ecuador; lf.contreras@uta.edu.ec (L.C.); wsmedina@uta.edu.ec (S.M.)
2 Faculty of Engineering, National University of Chimborazo, Av. Antonio José de Sucre, Riobamba 060108, Ecuador; leonardo.renteria@unach.edu.ec
* Correspondence: cm.mayacela@uta.edu.ec; Tel.: +593-960596700

Abstract: The memristor is the fourth fundamental element in the electronic circuit field, whose memory and resistance properties make it unique. Although there are no electronic solutions based on the memristor, interest in application development has increased significantly. Nevertheless, there are only numerical Matlab or Spice models that can be used for simulating memristor systems, and designing is limited to using memristor emulators only. A memristor emulator is an electronic circuit that mimics a memristor. In this way, a research approach is to build discrete-component emulators of memristors for its study without using the actual models. In this work, two reconfigurable hardware architectures have been proposed for use in the prototyping of a non-linearity memristor emulator: the FPAA (Field Programing Analog Arrays) and the FPGA (Field Programming Gate Array). The easy programming and reprogramming of the first architecture and the performance, high area density, and parallelism of the second one allow the implementation of this type of system. In addition, a detailed comparison is shown to underline the main differences between the two approaches. These platforms could be used in more complex analog and/or digital systems, such as neural networks, CNN, digital circuits, etc.

Keywords: memristor emulator; electronic circuit; FPAA; FPGA

check for updates

Citation: Mayacela, M.; Rentería, L.; Contreras, L.; Medina, S. Comparative Analysis of Reconfigurable Platforms for Memristor Emulation. *Materials* **2022**, *15*, 4487. https://doi.org/10.3390/ma15134487

Received: 3 May 2022
Accepted: 7 June 2022
Published: 25 June 2022

1. Introduction

To date, three fundamental passive elements have been used to design electronic circuits: resistors, capacitors, and inductors. In 1971, Leon Chua from the University of California at Berkeley reasoned from symmetry arguments that there should be a fourth fundamental element, which he named memristor [1]. This element is named a memristor, as it combines the behavior of a memory and a resistor. A Memristor is a two-terminal element whose resistance depends on the magnitude, direction, and duration of the applied voltage [2].

In theory, its memory capacity should be infinity, but even if it is not capable of storing energy, it can be used to store information indirectly.

The most important feature is that this device lets us store information in an analog way, leaving us with the possibility to develop applications that we could not even imagine some years ago. Since the fabrication of the Hewlett Packard (HP) memristor [3], interest in memristor research has increased [4]. Several people have started to take advantage of the memristor capacities in different fields, such as analog applications [5–10], digital circuits and memories [11–13], and the neuromorphic field because of its analog storing capability [14–26].

Although there is a commercial memristor now available [27], research based on this device in the market has been rare [28], probably due to cost and technical issues, such as voltage and current limitations, measurement, and others [29]. In this sense, the research continues, overall, on circuit modeling [30–32] or numerical modeling [33] only to simulate memristor behavior. The modeling and simulation phases are very important for the design of a system or a device [34], but they also need to verify and validate the model in the physical world.

Therefore, there is a need for a physical module that works like a real memristor, that is, a memristor emulator. This memristor emulator is an electronic circuit that mimics a memristor. Hardware implementations of the memristor emulator have been developed, and different models have been proposed. Some of them use digital and analog mixed circuits [35], where a microcontroller acquires the voltage applied to the memristor; it calculates and updates the resistance value by setting the digital potentiometer to obtain the required resistance value. This model is relatively simple to implement, but the time response and resolution of the memristance are limited by the A/D and D/A converters and the digital potentiometer. It has also been implemented as a pure analog model [5], which follows pretty fine the memristor behavior, the connection with other circuit elements is difficult, and the memristance is not guaranteed for a long time. Another emulating circuit [14] shows the features of the memristor; it keeps the memristance constant and stable over a long period of time, and it is compatible with other circuit devices. It has a small variation range of the memristance and a limited ratio between the maximum and minimum values of memristance. Additionally, electrical devices, such as light-dependent diodes [36], light-dependent resistors [37], and junction field effect transistors [38], have been used for this type of implementation despite their small memristance variation range.

Moreover, recent trends in hardware design have seen a strong increase in the use of programmable devices [39], such as CPLDs and FPGAs [40,41], and more recently, field programmable analog arrays (FPAA) [42]. Programmable devices can provide flexible and efficient platforms. These devices satisfy the performance, cost, and power requirements of most hardware prototyping architectures [43]. Reconfigurable architectures combine some of the flexibility of software with the high performance and speed of hardware.

In this paper, we propose a completely different method to emulate the memristor, an analog reconfigurable hardware architecture based on FPAA, and a digital reconfigurable hardware architecture based on FPGA. These include some important features of any memristor emulator [14]. The easy programming and reprogramming of the first hardware architecture and the performance, high area density, and parallelism of the second one allow the implementation of this type of system [44,45].

The principal difference when compared to ordinary hardware emulator implementation is the ability to make substantial changes. It is possible to adapt the hardware during runtime by "loading" a new circuit on the reconfigurable architecture, and there is a significant increase in the number of emulated devices.

Additionally, knowing that there are different models of memristors with an analog pure architecture, completely different circuits for each model must be built to test them. Moreover, with microcontroller-based architecture, it is easier to obtain by changing the software; however, the speed, accuracy, and number of emulated devices are always limited by the rest of the electronic components.

Then, the use of this technology will allow an increase in design complexity while allowing a decrease in the amount of time spent debugging, wiring, and other hardware implementation problems.

2. Materials and Methods

2.1. Memristor Overview

The conception of memristor as the fourth fundamental component in circuit theory creates a new approach to nonlinear circuit design. As is known, circuit elements (Figure 1) reflect relationships between pairs of the four electromagnetic quantities of charge, current, voltage, and magnetic flux [46].

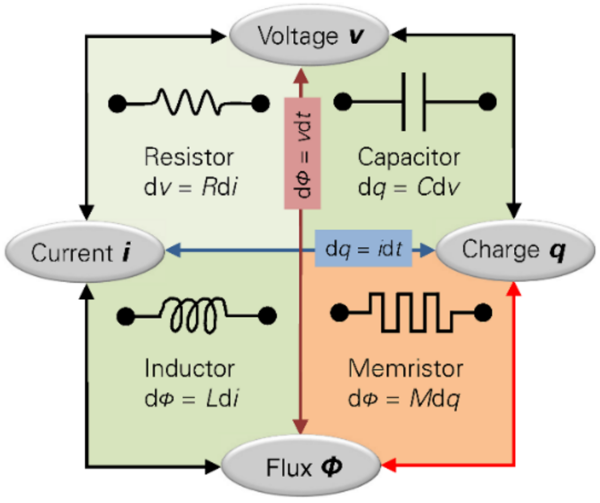


Figure 1. Electronic components relationship.

Resistance relates voltage and current ($dv = Rdi$), capacitance relates charge and voltage ($dq = Cdv$), and inductance relates flux and current ($d\varphi = Ldi$), respectively [47].

The memristor keeps the last resistance value when it is turned off and retrieves the value when it is turned back on. Memristor has several interesting properties, including pinched hysteresis and dynamic negative resistance, which can have a significant impact on nanoelectronics [2].

A memristor is characterized by its memristance (M) and memductance (W). These are described by the charge-dependent and flux-dependent equations, respectively, as follows [10]:

$$M(Q) = \frac{d\varnothing(q)}{dq} \tag{1}$$

$$W(\varnothing) = \frac{dq(\varnothing)}{d\varnothing} \tag{2}$$

This property is similar to the fundamental element resistor, which is characterized by its resistance R [48]. It may be noted that memristance is similar to variable resistance. A nonlinear version of Ohm's law can be expressed as a current-controlled memristive system [49]:

$$V_m(t) = M(x, I_m, t)I_m(t) \tag{3}$$

$$\frac{dx}{dt} = f(x, I_m, t) \tag{4}$$

where x is a vector representing n internal state variables, $Vm(t)$ and $Im(t)$ denote the voltage and current across the device, and M the memristance. Similarly, a voltage-controlled memristive system [49]:

$$I_m(t) = G(x, V_m, t)V_m(t) \tag{5}$$

$$\frac{dx}{dt} = f(x, V_m, t) \tag{6}$$

where G is the memductance.

One important aspect to keep in mind about the memristor is its dependence on the "state" variable x. The state variable describes how the system "looks" inside [13].

A memristor, which is a two-terminal circuit element, will provide hysteresis loops in an i–v plot when subject to an alternating voltage signal [48]. The hysteresis loops are very valuable when memristive systems are to be identified, and the loops normally run through the origin in an I–V plot (Figure 2).

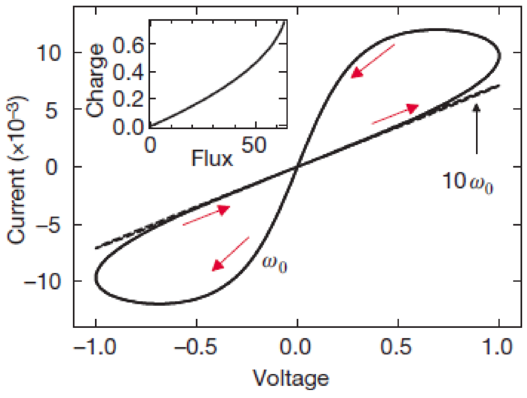


Figure 2. Typical behavior of memristors.

In the case of linear elements in which M is a constant, memristance is identical to resistance, and thus, it is not of special interest. However, the most valuable functions of circuits are attributable to their non-linear characteristics. If M is itself a function of q, yielding a nonlinear circuit element, then the situation becomes more interesting [3]. The compatibility of memristors with integrated circuits could provide new circuit functions at extremely high two-terminal device densities [2].

In Different published, mathematical memristor models are used as a baseline to research device features [32].

Memristors exist in various types depending on how they are built; moreover, there are systems that exhibit properties of memristors and so are called “memristive systems” [2].

If we consider a flux-controlled memristor described by Equation (2), where $q(\varnothing)$ is a smooth continuous cubic function of the form [10]:

$$q(\varnothing) = -\alpha \cdot \varnothing + \beta \cdot \varnothing^3 \tag{7}$$

With α, $\beta > 0$. As a result, in this case, the memductance $W(\varnothing)$ is provided by the following expression:

$$W(\varnothing) = \frac{dq(\varnothing)}{d\varnothing} = -\alpha + 3 \cdot \beta \cdot \varnothing^2 \tag{8}$$

Since $\varnothing(t) \triangleq \int v_m(t)dt$, an expression of the current through the memristor with cubic nonlinearity is [49]:

$$i_m(t) = \left(-\alpha + 3 \cdot \beta \cdot \left(\int v_m(t)dt\right)^2\right) \cdot v_m(t) \tag{9}$$

However, a certain methodology for how to experiment with the calculation of the memristance function and obtain the memristor flux-charge characteristic has not been defined [50,51]; therefore, there is a need to calculate its memristance. One method might be applying a DC or AC voltage, measuring its current and voltage, and then, by taking the integration of its current, its charge or memristance, as a function of the current, can be calculated [52,53].

2.2. Circuit Implementation

Numerical simulation plays an important role in analyzing systems and predetermining design parameters prior to their physical realization [54]. Before implementation, a memristor model was constructed in a Matlab environment. The backward Euler method [55] was used for numerical integration in which the incoming sample was multiplied by the sampling time and added to the last result of the integrator as follows:

$$\frac{dy}{dt} = f(t, y) \quad ; \quad y(t_0) = y_0 \tag{10}$$

$$y_{n+1} = y_n + hf(t_{n+1}, y_{n+1}) \tag{11}$$

As, $\varnothing \triangleq v_m(t)$ we can rewrite Equation (10) as follow:

$$\varnothing_{n+1} = \varnothing_n + hf(t_{n+1}) \tag{12}$$

the term y_{n+1} disappears because the equation only depends on the function in time; therefore, the problem is limited to solving the integral of the input voltage, squaring the result, and calculating the current.

After testing and tuning, the numerical simulation results (Figure 3) can be used as a reference for real implementation. Notice the unrealistic values of the current; for that reason, the set of equations requires rescaling in the physical implementation.

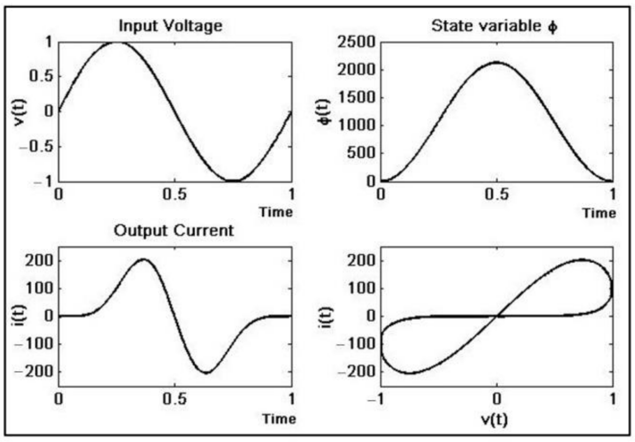


Figure 3. Memristor time response simulation. $v(t)$ = 1sin($2\pi ft$), f = 60 Hz, h = 0.05, α = 0.677 × 10^{-3}, β = 0.029 × 10^{-3}.

2.2.1. FPAA Emulator

The Field-Programmable Analog Array is an integrated circuit that can be configured to implement various analog functions [39]. This is the analog equivalent of FPGA [56]. This electronic device has a feature that can be used to programmatically change component values and interconnections; in other words, it can be dynamically reconfigured. Additionally, FPAA provides more efficient and economical solutions for analog dynamic system designs [45].

The most important elements in an FPAA are the Configurable Analogue Blocks (CAB), which manipulate the signals and the interconnecting routing network. Each element contains configurable modules (CAMs) [39].

The analogue blocks have parameters that can be programmed to accommodate according to the application [39].

The device used is the anadigm AN221E04 (Figure 4), which is composed of four programmable blocks. The dynamic range of the signals in the FPAA device is bounded by physical constraints [56]. In effect, the FPAA device has ±3 V saturation level, so the signal magnitude has to be scaled according to a previous numerical simulation.

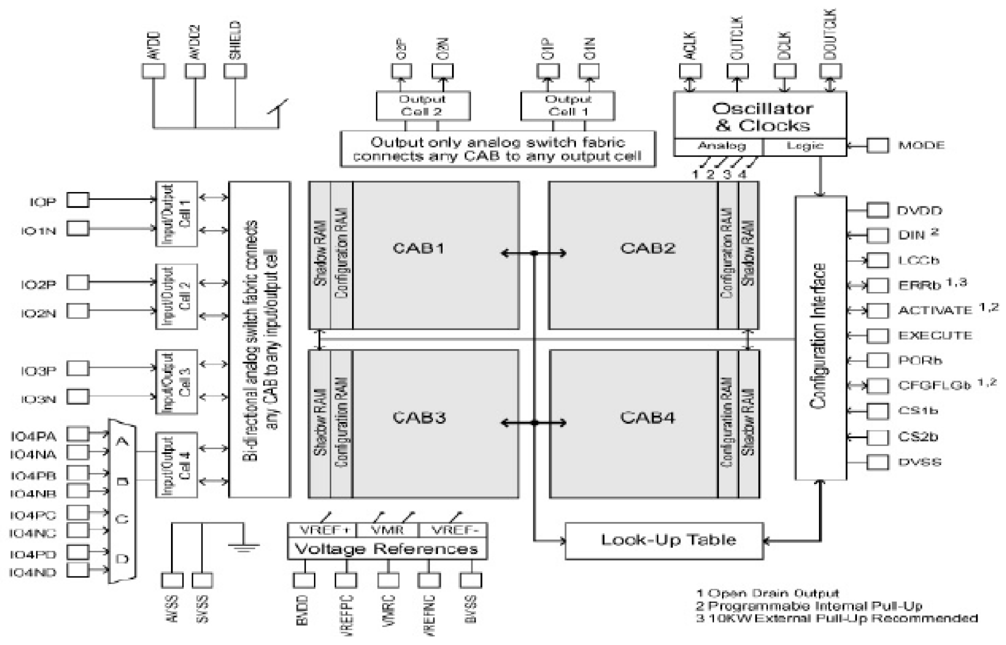


Figure 4. Anadigm AN221E04 Architecture.

The complete system (Figure 5) is composed of a gain inverter, two analog multipliers, and two sum-filter modules (Table 1).

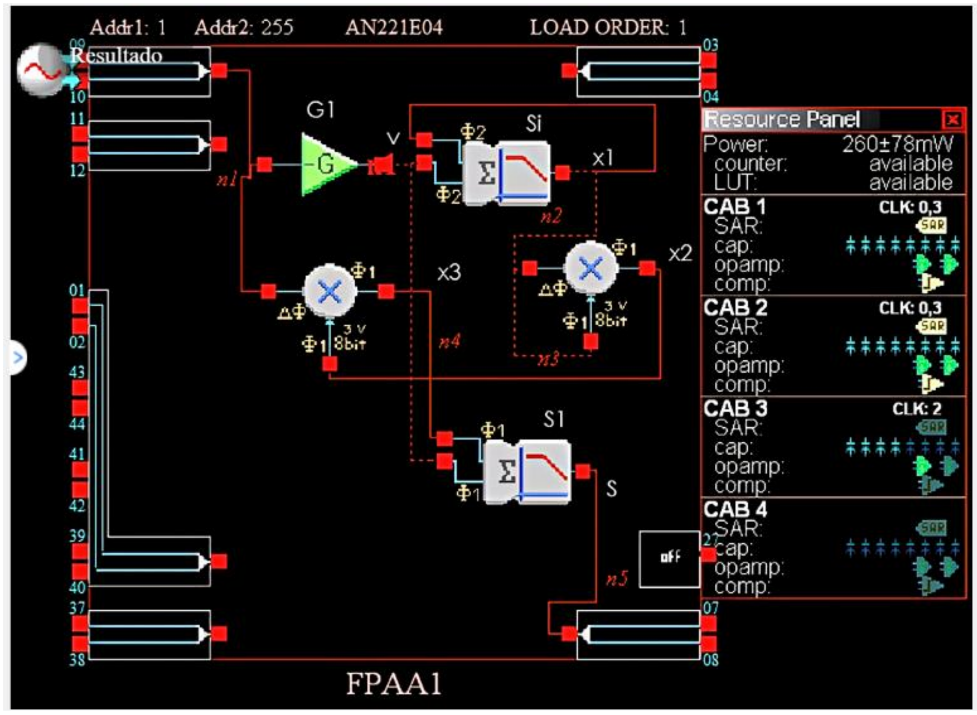
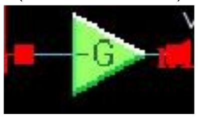


Figure 5. Schematic of FPAA memristor emulator.

Table 1. FPAA emulator parameter settings.

Name	Options	Parameters	Clocks
Multiplier 1 (Multiplier v1.2.2) Anadigm (Approved)	**Sample and Hold** Off **Y Input Full Scale** 3 Volts	**Multiplication Factor** 1.00	**Clock A** 125 kHz (Chip Clock 3) **Clock B** 2 MHz (Chip Clock 0)
Multiplier 2 (Multiplier v1.2.2) Anadigm (Approved)	**Sample and Hold** Off **Y Input Full Scale** 3 Volts	**Multiplication Factor** 1.00	**Clock A** 125 kHz (Chip Clock 3) **Clock B** 2 MHz (Chip Clock 0)
SumFilter 2 (SumFilter v1.1.3) Anadigm (Approved)	**Output Changes On** Phase 2 **Input 1** Non-inverting **Input 2** Non-inverting **Input 3** Off	**Corner Frequency [kHz]** 12.5 **Gain 1 (Upper Input)** 0.870 **Gain 2 (Lower Input)** 0.0676	**Clock A** 125 kHz (Chip Clock 3)
G1 (GainInv v1.1.4)  Anadigm (Approved)		**Gain** 0.0100	**Clock A** 2 MHz (Chip Clock 0)
SumFilter 2 (SumFilter v1.1.3) Anadigm (Approved)	**Output Changes On** Phase 1 **Input 1** Non-inverting **Input 2** Non-inverting **Input 3** Off	**Corner Frequency [kHz]** 100.0 **Gain 1 (Upper Input)** 1.00 **Gain 2 (Lower Input)** 0.0500	**Clock A** 1 MHz (Chip Clock 2)

The first stage of the system is the gain inverter G1, by applying an input voltage v_m, its output v is:

$$v = -\frac{v_i}{100} \tag{13}$$

The sumfilter module Si has and output x_1 described as:

$$x_1 = \dot{x}_1 + hv \tag{14}$$

Notice that $\dot{x}_1$ refers to the last value of x_1. Then, similar to Equation (10), it is possible to convert Equation (14) to:

$$x_1 \cong \int v\, dt x_1 = \int -\frac{v_i}{100}\, dt = -\frac{1}{100}\int v_i\, dt \tag{15}$$

After the first multiplier M1, we can obtain x_2,

$$x_2 = x_1 \cdot x_1 = \frac{1}{100^2}\left(\int v_i\, dt\right)^2 \tag{16}$$

Now, at the output of the second multiplier M2, we have x_3,

$$x_3 = x_2 \cdot v_i = \frac{1}{100^2}\left(\int v\right)^2 \cdot v_i \tag{17}$$

Finally, the sum filter module S1 gives the following result:

$$S = G_1 \cdot x_3 + G_2 \cdot v \tag{18}$$

With G_1= 0.87 and G_2 = 0.0677, and by arranging the terms, we can obtain:

$$\begin{aligned} S &= 0.87 \cdot \tfrac{1}{100^2}\left(\int v_i dt\right)^2 \cdot v_i - 0.0677 \cdot \left(\tfrac{v_i}{100}\right) \\ &= 0.087 \times 10^{-3} \cdot \left(\int v_i\, dt\right)^2 \cdot v_i - 0.0677 \times 10^{-3} \cdot v_i \\ S &= \left(-0.677 \times 10^{-3} + 3 \cdot 0.029 \times 10^{-3} \cdot \left(\int v_i\, dt\right)^2\right) \cdot v_i \end{aligned} \tag{19}$$

By replacing parameters similar to [5], α = 0.677 × 10^{-3}, β = 0.029 × 10^{-3} and $v_i = v_m$ on Equation (19), the result is as follows:

$$i_m(t) = \left(-\alpha + 3 \cdot \beta \cdot \left(\int v_m(t) dt\right)^2\right) \cdot v_m(t)$$

By applying a sinusoidal signal $v(t) = v_o \cdot \sin(2\pi f t)$ with f = 60 Hz, v_o= 1 V, we can obtain the v_m and i_m plots and it's clear that this element really has memristor behavior (Figure 6).

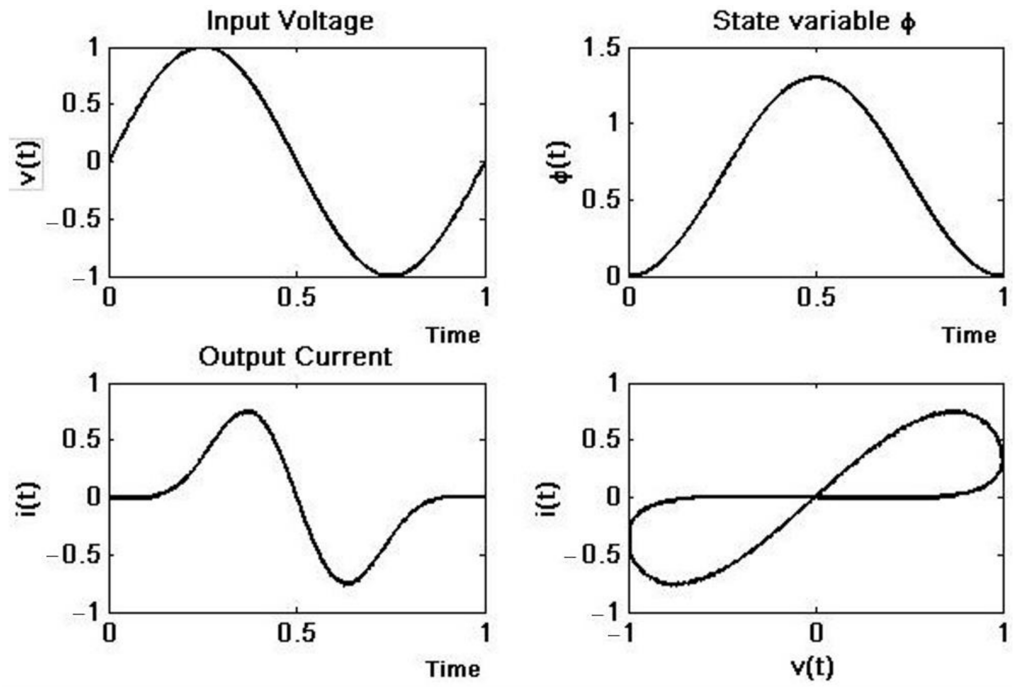


Figure 6. FPAA memristor emulator response.

Since AN221E04 performs analog "calculations" at a certain clock frequency, the values of all settings are limited to a range of values determined by the clock frequency. However, the values can be adjusted by a minimum of 0.01 units, which is acceptable for our needs. Moreover, there are only four CABs for each chip; consequently, an emulator uses almost 50% of the device's available resources.

2.2.2. FPGA Implementation

The second implementation was based on FPGA architecture. Traditionally, analog computers use operational amplifiers to implement basic operations (addition, subtraction, multiplication) and compute time-integral functions [57]. All of these operations can be performed on the FPGA, except computing time integrals. To solve this problem, a device called a Digital Differential Analyzer (DDA) was created to simulate the integral function digitally.

A Terasic DE4-320 Development Board (Figure 7) was chosen for this implementation. The board is equipped with an Altera Stratix IV GX EP4SGX230 FPGA that provides about 228,000 logic elements (LEs), 91,200 adaptive logic modules (ALMs), 14.283-Mb embedded memory, and eight phase-locked loops. The DE4 Development Board provides the ideal hardware platform for system designs that demand high performance, serial connectivity, and advanced memory interfacing [41]. Developed specifically to address the rapidly evolving requirements in many end markets for greater bandwidth, improved jitter performance, and lower power consumption [58].

Figure 7. Altera DE4-320 FPGA Development Board.

A fixed-point version of the DDA was implemented with 32-bit number representation. Bit 32 is the sign bit, and the binary point is between bits 16 and 17 (with bit zero being the least significant) (Table 2). The number range is thus −32,768 to +32,768, and the smallest value to be represented is 1.5259×10^{-5}.

Table 2. Fixed-point 32-bit number representation.

Number	Binary	Hexadecimal
1.0	0_000000000000001.0000000000000000	"00010000"
0.025	0_000000000000000.0000011001100110	"00000666"

The device is composed of three principal modules (Figure 8): the function generator, which contains the input signal, the memristor solver that describes the memristor behavior and the NIOS2 IP core, which is used for control and configuration.

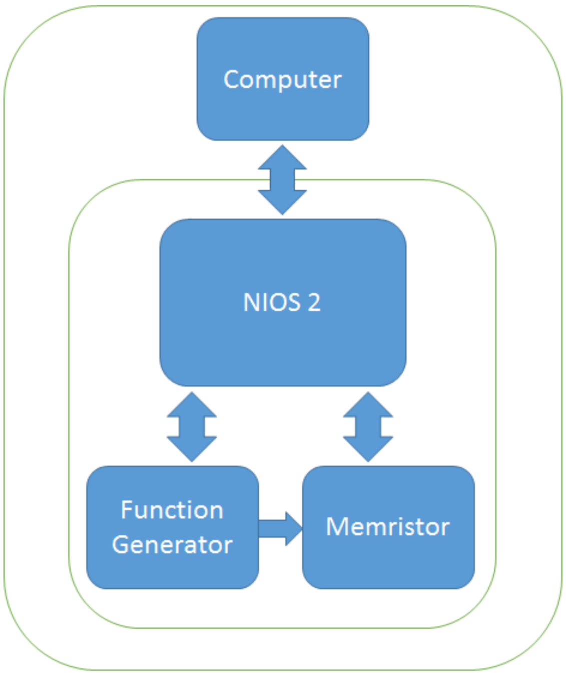


Figure 8. FPGA memristor emulator general diagram.

The hardware implementation is associated with VHDL programming. The "ieee proposed" HDL package [59] was used to obtain a synthesizable fixed-point unit (FPU).

The package allows float and fixed-point operations to be performed by adding, subtracting, and multiplying. The VHDL multiplier is easily implemented; it has three inputs, one of which can be used for scaling. The adding and subtracting operations were calculated directly. The forward Euler method was used for the integration steep; the step size used was 0.005. The function generator (Figure 9) is used to provide various waveforms, such as square wave, triangle wave, sine wave, etc. These waves are generated and stored in a 32 $\times$ 256 Ram memory. Intermediate values were interpolated at an 8 MHz sampling rate.

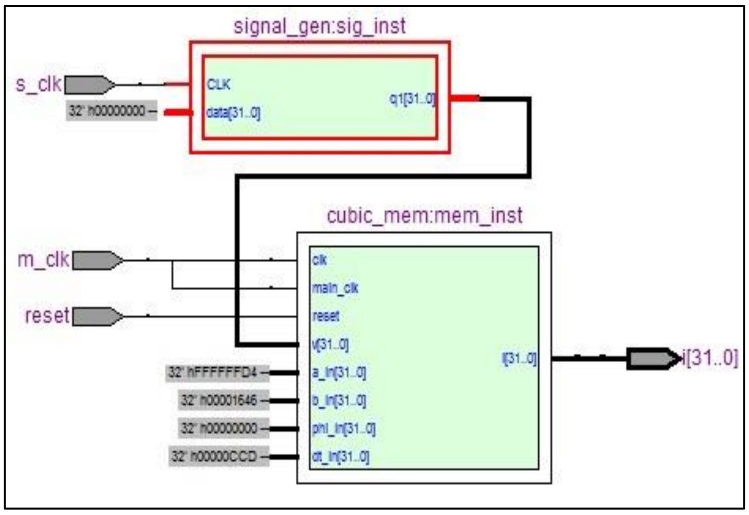
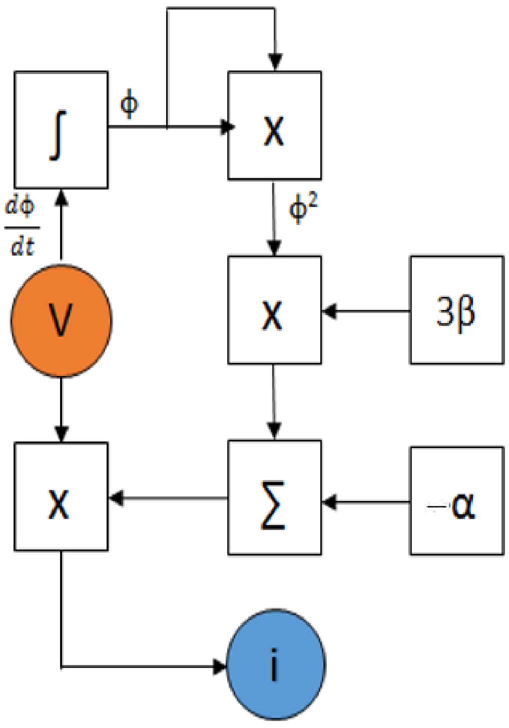
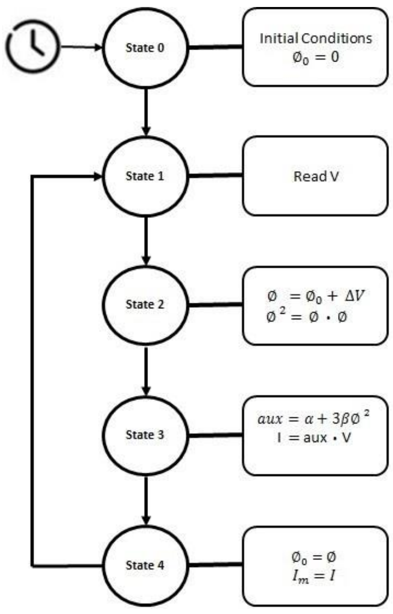


Figure 9. FPGA memristor emulator RTL schematic.

The memristor solver (Figure 10) consists of an FSM that solves Equation (6). It takes four cycles of its 8 MHz clock to solve each sample; other parameters are set by default and could be changed by the Nios 2 core.



(**a**)



(**b**)

Figure 10. FPGA-based memristor solver: (**a**) Memristor solver block diagram; (**b**) Memristor solver FSM flowchart.

With parameters similar to [5] $\alpha = 0.677 \times 10^{-3}$ and $\beta = 0.029 \times 10^{-3}$, a signal frequency of 60 Hz, and a sampling frequency of 2 MHz, we can obtain the v_m vs. i_m plots (Figure 11); clearly, the results show the memristive behavior of the system.

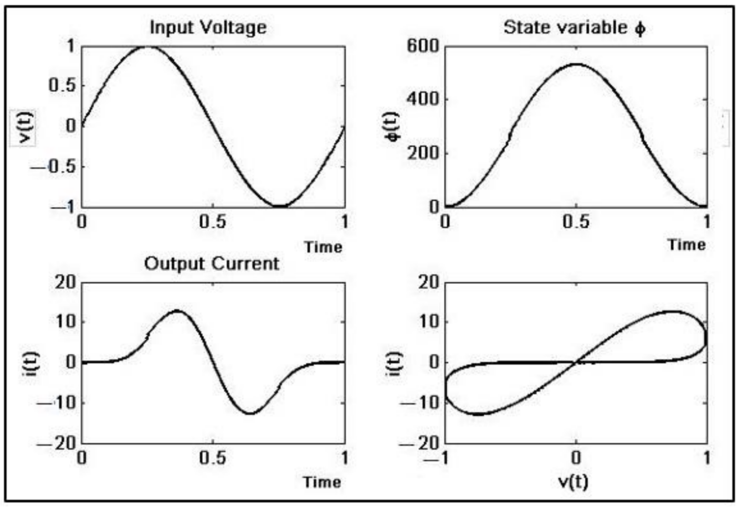


Figure 11. FPGA memristor emulator response.

The EP4SGX230KF40C2 FPGA chip place and route process statistics (Table 3) give us a general idea of the resources used and the potential capacity of this kind of implementation.

Table 3. FPGA chip statistics, including the full system.

FPGA Resource	Used Resources
Combinational ALUTs	10,345
Total Registers	10,345
DSP Block 18-bit Elements	118
Total Block memory bits	4,434,148
Logic utilization	9%

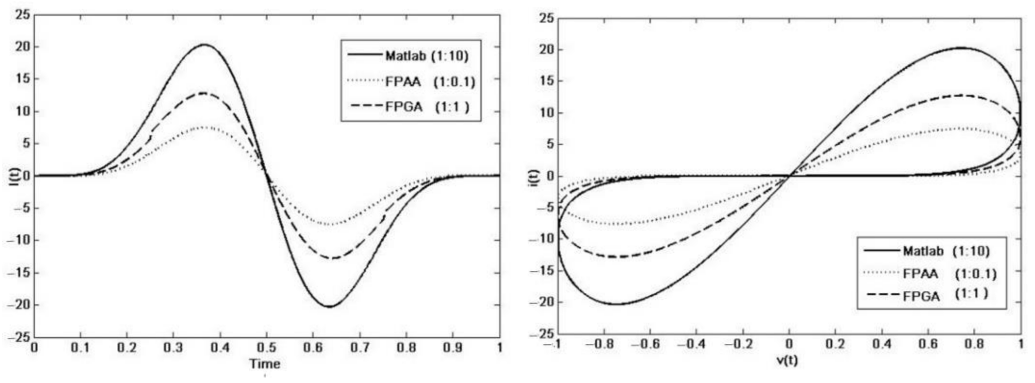
Since the memristor solver is independent, it means that it can work by itself; it is possible to have more memristors managed by the same NIOS 2 core. This gives the possibility of significantly increasing the number of memristors because used resources (Table 4) decrease considerably.

Table 4. FPGA chip statistics memristor solver only.

FPGA Resource	Used Resources
Combinational ALUTs	1298
Total Registers	16,105
DSP Block 18-bit Elements	118
Total Block memory bits	100,352
Logic utilization	1%

3. Results and Discussion

The results (Figure 12) show waves similar to those of the simulation, except for an attenuation factor. Losses are due to the switched capacitor technology, and filtering within the FPAA provides a reduction of 99.65%. Otherwise, the FSM within the FPGA integrates 1/4 of the total samples because it needs 4 clock cycles, resulting in an attenuation of 96% with respect to the original wave. In both cases, the attenuation can be used as the scaling factor needed to obtain current values according to the reality of the electronic circuits. This factor can easily be modified to suit the requirements of the external peripherals in any memristor-based circuit implementation.



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

Figure 12. Memristor emulators' current: (**a**) Current vs. Time plot; (**b**) Voltagevs. Current plot.

The time for one integration step depends on the main clock frequency on the FPAA, and on the sampling clock on FPGA at 4 MHz and 2 MHz, respectively (Table 5).

Table 5. Time performance comparison.

	Fs	One Period Time	
	MHz	Nro Samples	Time
Matlab	8	133,366	64.6 ms
FPAA	4	66,667	16.7 ms
FPGA	2	32,347	16.7 ms

Notice that, on simulation, the time for one period is proportional to the number of samples. On the other hand, in the case of hardware implementation, the total time depends on the signal period (Table 5). It is possible to increase the number of samples by increasing the sampling frequency while the time remains. However, the maximum sampling frequency is limited by the physical constraints, overall, on the FPAA device.

4. Conclusions

In this work, two different emulator models were proposed. First, a simple FPAA programmable analog circuit has replaced dozens of standard discrete components. It has become an effective solution to problems of rapid prototyping and has simplified the task of designing similar electronic circuits. However, resources are limited, and at most, it is possible to obtain two emulators for each CI. Likewise, a powerful tool called DDA has given us the possibility of performing a digital integrator on an FPGA, allowing a relatively high number of elements regarding the occupied area density; however, analog interaction is not possible directly, and it will require additional features to obtain this capacity. The next step is to use them as a part of more complex systems, such as chaotic oscillators, voltage-controlled sources, or neural network circuits. Finally, since the discovery of the memristor, researchers have used mathematical models, simulations, and emulators of memristors and, more recently, commercial memristors in their experiments. Simulations are useful but do not consider all factors, especially physical ones; emulators mimic real memristor behavior; however, they are not practical to use in more complex systems such as a neural network due to their design and physical proportions, while the new commercial memristors are still too expensive and make their use difficult, especially when there are budget limitations. Therefore, the memristor emulators proposed in this work become viable alternatives, especially when the applications demand a large number of them.

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

Funding: This research was funded by the Technical University of Ambato, Research and Development Directorate.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

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

Acknowledgments: The authors gratefully acknowledge the financial support of the Research and Development Directorate from the Technical University of Ambato.

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

References

1. Chua, L. Memristor-The missing circuit element. *IEEE Trans. Circuit Theory* **1971**, *18*, 507–519. [CrossRef]
2. Mohanty, S. Memristor: From Basics to Deployment. *IEEE Potentials* **2013**, *32*, 34–39. [CrossRef]

3. Strukov, D.B.; Snider, G.S.; Stewart, D.R.; Williams, R.S. The missing memristor found. *Nature* **2008**, *453*, 80–83. [CrossRef] [PubMed]
4. Yang, C.; Choi, H.; Park, S.; Sah, M.P.; Kim, H.; O Chua, L. A memristor emulator as a replacement of a real memristor. *Semicond. Sci. Technol.* **2014**, *30*, 015007. [CrossRef]
5. Muthuswamy, B. Implementing Memristor Based Chaotic Circuits. *Int. J. Bifurc. Chaos* **2010**, *20*, 1335–1350. [CrossRef]
6. Itoh, M.; Chua, L.O. Memristor Oscillators. *Int. J. Bifurc. Chaos* **2008**, *18*, 3183–3206. [CrossRef]
7. Pershin, Y.; Di Ventra, M. Memristive circuits simulate memcapacitors and meminductors. *Electron. Lett.* **2010**, *46*, 517–518. [CrossRef]
8. Muthuswamy, B.; Chua, L.O. Simplest Chaotic Circuit. *Int. J. Bifurc. Chaos* **2010**, *20*, 1567–1580. [CrossRef]
9. Pershin, Y.V.; Di Ventra, M. Practical Approach to Programmable Analog Circuits With Memristors. *IEEE Trans. Circuits Syst. I: Regul. Pap.* **2010**, *57*, 1857–1864. [CrossRef]
10. Bilotta, E.; Chiaravalloti, F.; Pantano, P. Spontaneous Synchronization in Two Mutually Coupled Memristor-Based Chua's Circuits: Numerical Investigations. *Math. Probl. Eng.* **2014**, *2014*, e594962. [CrossRef]
11. Ho, Y.; Huang, G.M.; Li, P. Nonvolatile Memristor Memory: Device Characteristics and Design Implications. In Proceedings of the 2009 IEEE/ACM International Conference on Computer-Aided Design-Digest of Technical Papers, San Jose, CA, USA, 28 December 2009; pp. 485–490.
12. Wang, L.; Duan, M.; Duan, S. Memristive Perceptron for Combinational Logic Classification. *Math. Probl. Eng.* **2013**, *2013*, e625790. [CrossRef]
13. Tran, T.; Rothenbuhler, A.; Smith, E.H.B.; Saxena, V.; Campbell, K.A. Reconfigurable Threshold Logic Gates using memristive devices. In Proceedings of the IEEE Subthreshold Microelectronics Conference (SubVT), Waltham, MA, USA, 9–10 October 2012; pp. 1–3. [CrossRef]
14. Kim, H.; Sah, M.P.; Yang, C.; Cho, S.; Chua, L.O. Memristor Emulator for Memristor Circuit Applications. *IEEE Trans. Circuits Syst. I: Regul. Pap.* **2012**, *59*, 2422–2431. [CrossRef]
15. Dong, Z.; Duan, S.; Hu, X.; Wang, L.; Li, H. A Novel Memristive Multilayer Feedforward Small-World Neural Network with Its Applications in PID Control. *Sci. World J.* **2014**, *2014*, e394828. [CrossRef] [PubMed]
16. Afifi, A.; Ayatollahi, A.; Raissi, F. STDP implementation using memristive nanodevice in CMOS-Nano neuromorphic networks. *IEICE Electron. Express* **2009**, *6*, 148–153. [CrossRef]
17. Jo, S.H.; Chang, T.; Ebong, I.; Bhadviya, B.B.; Mazumder, P.; Lu, W. Nanoscale Memristor Device as Synapse in Neuromorphic Systems. *Nano Lett.* **2010**, *10*, 1297–1301. [CrossRef]
18. Saadeldeen, H.; Franklin, D.; Long, G.; Hill, C.; Browne, A.; Strukov, D.; Sherwood, T.; Chong, F.T. Memristors for Neural Branch Prediction: A Case Study in Strict Latency and Write Endurance Challenges. In Proceedings of the ACM International Conference on Computing Frontiers, Association for Computing Machinery, Sicily, Italy, 11–13 May 2013; pp. 1–10. [CrossRef]
19. Ebong, I.E.; Mazumder, P. CMOS and Memristor-Based Neural Network Design for Position Detection. *Proc. IEEE* **2011**, *100*, 2050–2060. [CrossRef]
20. Ziegler, M.; Soni, R.; Patelczyk, T.; Ignatov, M.; Bartsch, T.; Meuffels, P.; Kohlstedt, H. An Electronic Version of Pavlov's Dog. *Adv. Funct. Mater.* **2012**, *22*, 2744–2749. [CrossRef]
21. Chen, L.; Li, C.; Huang, T.; Wang, X. Quick noise-tolerant learning in a multi-layer memristive neural network. *Neurocomputing* **2014**, *129*, 122–126. [CrossRef]
22. Zamarreño-Ramos, C.; Camuñas-Mesa, L.A.; Pérez-Carrasco, J.A.; Masquelier, T.; Serrano-Gotarredona, T.; Linares-Barranco, B. On Spike-Timing-Dependent-Plasticity, Memristive Devices, and Building a Self-Learning Visual Cortex. *Front. Behav. Neurosci.* **2011**, *5*, 26. [CrossRef]
23. Duan, S.; Zhang, Y.; Hu, X.; Wang, L.; Li, C. Memristor-based chaotic neural networks for associative memory. *Neural Comput. Appl.* **2014**, *25*, 1437–1445. [CrossRef]
24. Sheri, A.M.; Hwang, H.; Jeon, M.; Lee, B.-G. Neuromorphic Character Recognition System with Two PCMO Memristors as a Synapse. *IEEE Trans. Ind. Electron.* **2013**, *61*, 2933–2941. [CrossRef]
25. Chen, L.; Li, C.; Huang, T.; Chen, Y.; Wen, S.; Qi, J. A synapse memristor model with forgetting effect. *Phys. Lett. A* **2013**, *377*, 3260–3265. [CrossRef]
26. Cruz-Albrecht, J.M.; DeRosier, T.; Srinivasa, N. A scalable neural chip with synaptic electronics using CMOS integrated memristors. *Nanotechnology* **2013**, *24*, 384011. [CrossRef] [PubMed]
27. Knowm. Available online: https://knowm.org/ (accessed on 17 May 2022).
28. Zhu, L.J.; Wang, F.Q. Design and analysis of new meminductor model based on Knowm memristor. *Acta Phys. Sin.* **2019**, *68*, 198501. [CrossRef]
29. Memristors, KnowM. Available online: https://knowm.org/downloads/Knowm_Memristors.pdf (accessed on 17 May 2022).
30. Pershin, Y.; Di Ventra, M. SPICE Model of Memristive Devices with Threshold. *Radioengineering* **2012**, *22*, 2.
31. Wang, Y.; Fei, W.; Yu, H. SPICE simulator for hybrid CMOS memristor circuit and system. In Proceedings of the 2012 13th International Workshop on Cellular Nanoscale Networks and Their Applications, Turin, Italy, 29–31 August 2012; pp. 1–6. [CrossRef]
32. Biolek, D.; Biolek, Z.; Biolkova, V. SPICE modeling of memristive, memcapacitative and meminductive systems. In Proceedings of the 2009 European Conference on Circuit Theory and Design, Antalya, Turkey, 23–27 August 2009; pp. 249–252. [CrossRef]

33. Zaplatilek, K. Memristor Modeling in MATLAB®&Simulink®. In Proceedings of the 5th European Computing Conference, Paris, France, 28–30 April 2011; pp. 62–67.
34. Borgese, G.; Vena, S.; Pantano, P.; Pace, C.; Bilotta, E. Simulation, Modeling, and Analysis of Soliton Waves Interaction and Propagation in CNN Transmission Lines for Innovative Data Communication and Processing. *Discret. Dyn. Nat. Soc.* **2015**, *2015*, 1–13. [CrossRef]
35. Pershin, Y.V.; Di Ventra, M. Experimental demonstration of associative memory with memristive neural networks. *Neural Networks* **2010**, *23*, 881–886. [CrossRef]
36. Biolek, D.; Bajer, J.; Biolkova, V.; Kolka, Z. Mutators for transforming nonlinear resistor into memristor. In Proceedings of the 2011 20th European Conference on Circuit Theory and Design (ECCTD), Linkoping, Sweden, 29–31 August 2011; pp. 488–491. [CrossRef]
37. Fitch, A.; Lu, H.; Wang, X.; Sreeram, V.; Qi, W. Realization of an analog model of memristor based on light dependent resistor. In Proceedings of the 2012 IEEE International Symposium on Circuits and Systems (ISCAS), Seoul, Korea, 20–23 May 2012; pp. 1139–1142. [CrossRef]
38. Valsa, J.; Biolek, D.; Biolek, Z. An analogue model of the memristor. *Int. J. Numer. Model. Electron. Netw. Devices Fields* **2010**, *24*, 400–408. [CrossRef]
39. Ila, V.; Batlle, J.; Cufi, X.; Garcia, R. Recent Trends in Fpaa Devices. In *Multimedia Information Processing*; World Scientific: Singapore, 2002; pp. 186–190. [CrossRef]
40. Borgese, G.; Pace, C.; Pantano, P.; Bilotta, E. Reconfigurable Implementation of a CNN-UM Platform for Fast Dynamical Systems Simulation. *Appl. Electron. Pervading Ind. Environ. Soc.* **2014**, *289*, 85–101. [CrossRef]
41. Borgese, G.; Pace, C.; Pantano, P.; Bilotta, E. FPGA-based distributed computing microarchitecture for complex physical dynamics investigation. *IEEE Trans. Neural Networks Learn. Syst.* **2013**, *24*, 1390–1399. [CrossRef]
42. Ananth, R.S. *Signal Processing Structures for Power-Supervisory and Low-Frequency Applications (Dissertation)*; National Library of Canada: Ottawa, ON, Canada, 1996.
43. Garcia, P.; Compton, K.; Schulte, M.; Blem, E.; Fu, W. An Overview of Reconfigurable Hardware in Embedded Systems. *EURASIP J. Embed. Syst.* **2006**, *2006*, 1–19. [CrossRef]
44. Hulub, M.; Frasca, M.; Fortuna, L.; Arena, P. Implementation and synchronization of 3×3 grid scroll chaotic circuits with analog programmable devices. *Chaos: Interdiscip. J. Nonlinear Sci.* **2006**, *16*, 013121. [CrossRef] [PubMed]
45. Kilic, R.; Dalkiran, F.Y. Programmable Design and Implementation of a Chaotic System Utilizing Multiple Nonlinear Functions. *Turk. J. Electr. Eng. Comput. Sci.* **2010**, *18*, 647–656. [CrossRef]
46. Volos, C.; Kyprianidis, I.; Stavrinides, S.G.; Stouboulos, I.N.; Anagnostopoulos, A.N. Memristors: A New Approach in Nonlinear Circuits Design. In Proceedings of the 14th WSEAS International Conference on Communications, World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, WI, USA, 23–25 July 2010; pp. 25–30.
47. Fitch, A.L. *Development of Memristor Based Circuits*; World Scientific: Singapore, 2013.
48. Kavehei, O.; Kim, Y.-S.; Iqbal, A.; Eshraghian, K.; Al-Sarawi, S.; Abbott, D. The fourth element: Insights into the memristor. In Proceedings of the 2009 International Conference on Communications, Circuits and Systems, Milpitas, CA, USA, 23–25 July 2009; pp. 921–927. [CrossRef]
49. Chua, L.O.; Kang, S.M. Memristive devices and systems. *Proc. IEEE* **1976**, *64*, 209–223. [CrossRef]
50. Kavehei, O.; Iqbal, A.; Kim, Y.S.; Eshraghian, K.; Al-Sarawi, S.; Abbott, D. The fourth element: Characteristics, modelling and electromagnetic theory of the memristor. *Proc. R. Soc. A Math. Phys. Eng. Sci.* **2010**, *466*, 2175–2202. [CrossRef]
51. Moshnyaga, V.; Esseling, M.; Sudheendra, L.; Lebedev, O.I.; Gehrke, K.; van Tendeloo, G.; Samwer, K. Memristor Behaviour in Nano-Sized Vertical Lsmo/Lsmo Tunnel Junctions. *arXiv* **2010**, arXiv:1002.0495.
52. Muthuswamy, B.; Kokate, P. Memristor-Based Chaotic Circuits. *IETE Technol. Rev.* **2009**, *26*, 417. [CrossRef]
53. Mutlu, R.; Karakulak, E. A methodology for memristance calculation. *Turk. J. Electr. Eng. Comput. Sci.* **2014**, *22*, 121–131. [CrossRef]
54. Kilic, R. *A Practical Guide for Studying Chua's Circuits*; World Scientific: Singapore, 2010; Volume 71.
55. Butcher, J.C. *Numerical Methods for Ordinary Differential Equations*; Wiley: Hoboken, NJ, USA, 2008.
56. Zhao, J.; Kim, Y.-B. Circuit implementation of FitzHugh-Nagumo neuron model using Field Programmable Analog Arrays. In Proceedings of the 2007 50th Midwest Symposium on Circuits and Systems, Montreal, QC, Canada, 5–8 August 2007; pp. 772–775. [CrossRef]
57. Guerrero-Rivera, R.; Morrison, A.; Diesmann, M.; Pearce, T.C. Programmable logic construction kits for hyper-real-time neuronal modeling. *Neural Comput.* **2006**, *18*, 2651–2679. [CrossRef]
58. Terasic Technologies. Terasic-All FPGA Boards-Stratix IV-Altera DE4 Development and Education Board. Available online: http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=138&No= (accessed on 12 December 2021).
59. EDA Playground. Available online: https://www.edaplayground.com/ (accessed on 12 December 2021).