Construction of a Chaotic Computer Chip

William L. Ditto, K. Murali and Sudeshna Sinha

abstract

Chaotic systems are great pattern generators and their defining feature, sensitivity to initial conditions, allows them to switch between patterns exponentially fast. We exploit such pattern generation by "tuning" representative continuous and discrete chaotic systems to generate all logic gate functions. We then exploit exponential sensitivity to initial conditions to achieve rapid switching between all the logic gates generated by each representative chaotic element. With this as a starting point we will present our progress on the construction of a chaotic computer chip consisting of large numbers of individual chaotic elements that can be individually and rapidly morphed to become all logic gates. Such a chip of arrays of morphing chaotic logic gates can then be programmed to perform higher order functions (such as memory, arithmetic logic, input/output operations, ...) and to rapidly switch between such functions. Thus we hope that our reconfigurable chaotic computer chips will enable us to achieve the flexibility of field programmable gate arrays (FPGA), the optimization and speed of application specific integrated circuits (ASIC) and the general utility of a central processing unit (CPU) within the same computer chip architecture. Results on the construction and commercialization of the ChaoLogix^{TM} chaotic computer chip will also be presented to demonstrate progress being made towards the commercialization of this technology (http://www.chaologix.com).

William L. Ditto
J. Crayton Pruitt Family Department of Biomedical Engineering, University of Florida, Gainesville, FL 32611-6131, USA, and
ChaoLogix, Inc. 101 S.E. 2nd Place, Suite 201 - A, Gainesville, FL 32601, USA
e-mail: william.ditto@bme.ufl.edu

K. Murali
Department of Physics, Anna University, Chennai 600 025, INDIA
e-mail: kmurali@annauniv.edu

Sudeshna Sinha
Institute of Mathematical Sciences, C.I.T. Campus, Taramani, Chennai 600 113, INDIA
e-mail: sudeshna@imsc.res.in

1 Introduction

It was proposed in 1998 that chaotic systems may be utilized to design computing devices [1]. In the early years the focus was on proof-of-principle schemes that demonstrated the capability of chaotic elements to do universal computing. The distinctive feature of this alternate computing paradigm was that they exploited the sensitivity and pattern formation features of chaotic systems.

In subsequent years, it was realized that one of the most promising direction of this computing paradigm is its ability to exploit a single chaotic element to reconfigure into different logic gates through a threshold based morphing mechanism [2, 3]. In contrast to a conventional field programmable gate array element, where reconfiguration is achieved through switching between multiple single purpose gates, reconfigurable chaotic logic gates (RCLGs) are comprised of chaotic elements that morph (or reconfigure) logic gates through the control of the pattern inherent in their nonlinear element. Two input RCLGs have recently been realized and shown to be capable of reconfiguring between all logic gates in discrete circuits [4, 5, 6]. Additionally such RCLGs have been realized in prototype VLSI circuits ($0.13\mu m$ CMOS, $30Mhz$ clock cycles) that employ two input reconfigurable chaotic logic gates arrays (RCGA) to morph between higher order functions such as those found in a typical arithmetic logic unit (ALU) [7].

In this article we first recall the theoretical scheme for flexible implementation of all these fundamental logical operations utilizing low dimensional chaos [2], and the specific realisation of the theory in a discrete-time and a continuous-time chaotic circuit. Then we will present new results on the design of reconfigurable multiple input gates. Note that multiple input logic gates are preferred mainly for reasons of space in circuits and also many combinational and sequential logic operations can be realized with these logic gates, in which one can minimize the propagation delay. Such a multiple input CGA would make RCLGs more power efficient, increase their performance and widen their range of applications. Here we specifically demonstrate a three input RCLG by implementing representative fundamental NOR and NAND gates with a continuous-time chaotic system.

2 Concept

In order to use the rich temporal patterns embedded in a nonlinear time series efficiently one needs a mechanism to extract different responses from the system, in a controlled manner, without much run-time effort. Here we employ a threshold based scheme to achieve this [8].

Consider the discrete-time chaotic map, with its state represented by a variable x, as our *chaotic chip* or *chaotic processor*. In our scheme all the basic logic gate operations (AND, OR, XOR, NAND, NOR, NOT) involve the following simple steps:

1. Inputs:
   $x \to x_0 + I_1 + I_2$ for 2-input gates such as the AND, OR, XOR, NAND and NOR operations, and
   $x \to x_0 + I$ for the 1-input gate such as the NOT operation.

   Here x_0 is the initial state of the system, and the input value $I = 0$ when logic input is 0 and $I = V_{in}$ when logic input is 1 (where V_{in} is a positive constant).

2. Dynamical update, i.e. $x \to f(x)$

   where $f(x)$ is a strongly nonlinear function.

3. Threshold mechanism to obtain output V_0 :

   $V_0 = 0$ if $f(x) \leq E$, and

   $V_0 = f(x) - E$ if $f(x) > E$

   where E is the threshold.

   This is interpretated as logic output 0 if $V_0 = 0$ and Logic Ouput 1 if $V_0 \sim V_{in}$.

Since the system is chaotic, in order to specify the inital x_0 accurately one needs a controlling mechanism. Here we will employ a threshold controller to set the inital x_0. So in this example we use the clipping action of the threshold controller to achieve the initialization, and subsequently to obtain the output as well.

Note that in our implementation we demand that the input and output have equivalent definitions (i.e. 1 unit is the same quantity for input and output), as well as among various logical operations. This requires that constant V_{in} assumes the same value throughout a network, and this will allow the output of one gate element to easily couple to another gate element as input, so that gates can be "wired" directly into gate arrays implementing compounded logic operations.

In order to obtain all the desired input-output responses of the different gates, we need to satisfy the conditions enumerated in Table 1 simultaneously. So given a dynamics $f(x)$ corresponding to the physical device in actual implementation, one must find values of threshold and initial state satisfying the conditions derived from the Truth Tables to be implemented.

For instance, Table 2 shows the exact solutions of the initial x_0 and threshold E which satisfy the conditions in Table 1 when

$$f(x) = 4x(1-x)$$

The constant $V_{in} = \frac{1}{4}$ is common to both input and output and to all logical gates.

Logic Operation	Input Set (I_1, I_2)	Output	Necessary and Sufficient Condition
AND	(0,0)	0	$f(x_0) < E$
	(0,1)/(1,0)	0	$f(x_0 + V_{in}) < E$
	(1,1)	1	$f(x_0 + 2V_{in}) - E = V_{in}$
OR	(0,0)	0	$f(x_0) < E$
	(0,1)/(1,0)	1	$f(x_0 + V_{in}) - E = V_{in}$
	(1,1)	1	$f(x_0 + 2V_{in}) - E = V_{in}$
XOR	(0,0)	0	$f(x_0) < E$
	(0,1)/(1,0)	1	$f(x_0 + V_{in}) - E = V_{in}$
	(1,1)	0	$f(x_0 + 2V_{in}) < E$
NOR	(0,0)	1	$f(x_0) - E = V_{in}$
	(0,1)/(1,0)	0	$f(x_0 + V_{in}) < E$
	(1,1)	0	$f(x_0 + 2V_{in}) < E$
NAND	(0,0)	1	$f(x_0) - E = V_{in}$
	(0,1)/(1,0)	1	$f(x_0 + V_{in}) - E = v_{in}$
	(1,1)	0	$f(x_0 + 2V_{in}) < E$
NOT	0	1	$f(x_0) - E = V_{in}$
	1	0	$f(x_0 + V_{in}) < E$

Table 1 Necessary and sufficient conditions, derived from the logic truth tables, to be satisfied simultaneously by the nonlinear dynamical element, in order to have the capacity to implement the logical operations AND, OR, XOR, NAND, NOR and NOT with the same computing module.

Operation	AND	OR	XOR	NAND	NOT
x_0	0	1/8	1/4	3/8	1/2
E	3/4	11/16	3/4	11/16	3/4

Table 2 One specific solution of the conditions in Table 1 which yields the logical operations AND, OR, XOR, NAND and NOT, with $V_{in} = \frac{1}{4}$. Note that these theoretical solutions have been fully verified in a discrete electrical circuit emulating a logistic map [4].

3 Continuous-time Nonlinear System

We now present a somewhat different scheme for obtaining logic responses from a continuous-time nonlinear system. Our processor is now a continuous time system described by the evolution equation $d\,\mathbf{x}\,/dt = \mathbf{F}\,(\mathbf{x},t)$, where $\mathbf{x} = (x_1, x_2, \ldots x_N)$ are the state variables and $\mathbf{F}$ is a nonlinear function. In this system we choose a variable, say x_1, to be thresholded. Whenever the value of this variable exceeds a threshold E it resets to E, i.e. when $x_1 > E$ then (and only then) $x_1 = E$.

Now the basic 2-input 1-output logic operation on a pair of inputs I_1, I_2 in this scheme simply involves the setting of an inputs-dependent threshold, namely the threshold voltage

$$E = V_C + I_1 + I_2$$

where V_C is the dynamic control signal determining the functionality of the processor. By switching the value of V_C one can switch the logic operation being performed.

Again I_1/I_2 has value 0 when logic input is 0 and has value V_{in} when logic input is 1. So the theshold E is equal to V_C when logic inputs are $(0,0)$, $V_C + V_{in}$ when logic inputs are $(0,1)$ or $(1,0)$, and $V_C + 2V_{in}$ when logic inputs are $(1,1)$.

The output is interpreted as logic output 0 if $x_1 < E$, i.e. the excess above threshold $V_0 = 0$. The logic output is 1 if $x_1 > E$, and the excess above threshold $V_0 = (x_1 - E) \sim V_{in}$. The schematic diagram of this method is displayed in Fig. 1.

Now for a NOR gate implementation ($V_C = V_{NOR}$) the following must hold true:

(i) when input set is $(0,0)$, output is 1, which implies that for threshold $E = V_{NOR}$, output $V_0 = (x_1 - E) \sim V_{in}$

(ii) when input set is $(0,1)$ or $(1,0)$, output is 0, which implies that for threshold $E = V_{NOR} + V_{in}$, $x_1 < E$ so that output $V_0 = 0$.

(iii) when input set is $(1,1)$, output is 0, which implies that for threshold $E = V_{NOR} + 2V_{in}$, $x_1 < E$ so that output $V_0 = 0$.

For a NAND gate ($V_C = V_{NAND}$) the following must hold true:

(i) when input set is $(0,0)$, output is 1, which implies that for threshold $E = V_{NAND}$, output $V_0 = (x_1 - E) \sim V_{in}$

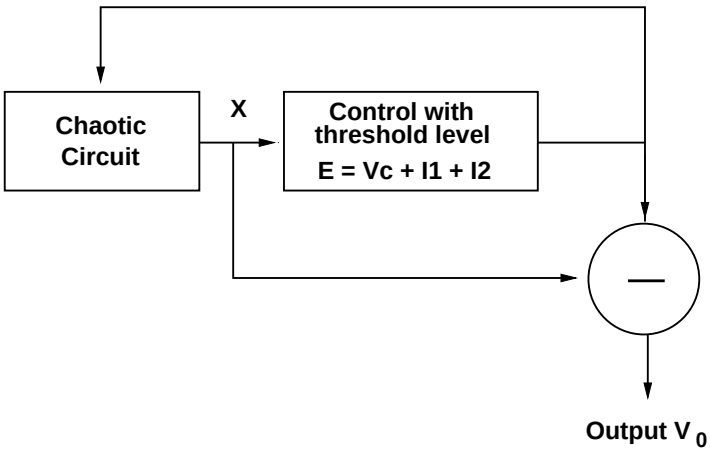

Fig. 1 Schematic diagram for implementing a morphing 2 input logic cell with a continuous time dynamical system. Here V_C determines the nature of the logic response, and the 2 inputs are $I1, I2$.

(ii) when input set is $(0,1)$ or $(1,0)$, output is 1, which implies that for threshold $E = V_{in} + V_{NAND}$, output $V_0 = (x_1 - E) \sim V_{in}$

(iii) when input set is $(1,1)$, output is 0, which implies that for threshold $E = V_{NAND} + 2V_{in}$, $x_1 < E$ so that output $V_0 = 0$.

In order to design a dynamic NOR/NAND gate one has to find values of V_C that will satisfy all the above input-output associations in a robust and consistent manner.

A proof-of-principle experiment of the scheme was realized with the double scroll chaotic Chua's circuit given by the following set of (rescaled) 3 coupled ODEs [9]

$$\dot{x_1} = \alpha(x_2 - x_1 - g(x_1)) \quad (1)$$

$$\dot{x_2} = x_1 - x_2 + x_3 \quad (2)$$

$$\dot{x_3} = -\beta x_2 \quad (3)$$

where $\alpha = 10.$ and $\beta = 14.87$ and the piecewise linear function $g(x) = bx + \frac{1}{2}(a-b)(|x+1| - |x-1|)$ with $a = -1.27$ and $b = -0.68$. We used the ring structure configuration of the classic Chua's circuit [9].

In the experiment we implemented minimal thresholding on variable x_1 (this is the part in the "control" box in the schematic figure). We clipped x_1 to E, if it exceeded E, only in Eqn. 2. This has very easy implementation, as it avoids modifying the value of x_1 in the nonlinear element $g(x_1)$, which is harder to do. So then all we need to do is to implement $\dot{x_2} = E - x_2 + x_3$ instead of Eqn. 2, when $x_1 > E$, and there is no controlling action if $x_1 \leq E$.

A representative example of a dynamic NOR/NAND gate can be obtained in this circuit implementation with parameters: $V_{in} = 2V$. The NOR gate is realized

around $V_C = 0V$. At this value of control signal, we have the following: for input (0,0) the threshold level is at 0, which yields $V_0 \sim 2V$; for inputs (1,0) or (0,1) the threshold level is at 0, which yields $V_0 \sim 0V$; and for input (1,1) the threshold level is at $2V$, which yields $V_0 = 0$ as the threshold is beyond the bounds of the chaotic attractor. The NAND gate is realized around $V_C = -2V$. The control signal yields the following: for input (0,0) the threshold level is at $-2V$, which yields $V_0 \sim 2V$; for inputs (1,0) or (0,1) the threshold level is at $2V$, which yields $V_0 \sim 2V$; and for input (1,1) the threshold level is at $4V$, which yields $V_0 = 0$ [5].

So the knowledge of the dynamics allowed us to design a control signal that can select out the temporal patterns emulating the NOR and NAND gates [6]. For instance in the example above, as the dynamic control signal V_C switches between $0V$ to $-2V$, the module first yields the NOR and then a NAND logic response. Thus one has obtained a dynamic logic gate capable of switching between two fundamental logic reponses, namely the NOR and NAND.

4 Design and Construction of a Three-Input Reconfigurable Chaotic Logic Gate

As in Section 3, consider a single chaotic element (for inclusion into a RCLG) to be a continuous time system described by the evolution equation: $d\ \mathbf{x}/dt = \mathbf{F}\ (\ \mathbf{x};t)$ where $\mathbf{x} = (x_1, x_2, \ldots, x_N)$ are the state variables, and $\mathbf{F}$ is a strongly nonlinear function. Again in this system we choose a variable, say x_1, to be thresholded. So whenever the value of this variable exceeds a critical threshold E (i.e. when $x_1 > E$), it re-sets to E.

In accordance to our basic scheme, the logic operation on a set of inputs I_1, I_2 and I_3 simply involves the setting of an inputs-dependent threshold, namely the threshold voltage $E = V_C + I_1 + I_2 + I_3$, where V_C is the dynamic control signal determining the functionality of the processor. By switching the value of V_C, one can switch the logic operation being performed.

$I_{1,2,3}$ has value $\sim 0V$ when logic input is zero, and $I_{1,2,3}$ has value V_{in} when logic input is one. So for input (0,0,0) the threshold level is at V_C; for inputs (0,0,1) or (0,1,0) or (1,0,0) the threshold level is at $V_C + V_{in}$; for input (0,1,1) or (1,1,0) or (1,1,0) the threshold level is at $V_C + 2V_{in}$ and for input (1,1,1) the threshold level is $V_C + 3V_{in}$.

As before, the output is interpreted as logic output 0 if $x_1 < E$, and the excess above threshold $V_0 \sim 0$. The logic output is 1 if $x_1 > E$, and $V_0 = (x_1 - E) \sim V_{in}$. Now for the 3-inputs NOR and the NAND gate implementations the input-output relations given in Tables 3 and 4 must hold true. Again, in order to design the NOR or NAND gates, one has to use the knowledge of the dynamics of the nonlinear system to find the values of V_C and V_0 that will satisfy all the input-output associations in a consistent and robust manner.

Consider again the simple realization of the double-scroll chaotic Chua's attractor represented by the set of (rescaled) 3-coupled ODEs given in Eqns. 1-3. This

Input Set (I_1, I_2, I_3)	Threshold E	Output	Logic Output
(0,0,0)	V_{NOR}	$V_0 = (x_1 - E) \sim V_{in}$	1
(0,0,1) or (1,0,0) or (0,1,0)	$V_{NOR} + V_{in}$	$V_0 \sim 0V$ as $x_1 < E$	0
(0,1,1) or (1,1,0) or (1,0,1)	$V_{NOR} + 2V_{in}$	$V_0 \sim 0V$ as $x_1 < E$	0
(0,0,0)	$V_{NOR} + 3V_{in}$	$V_0 \sim 0V$ as $x_1 < E$	0

Table 3 Truth table for NOR gate implementation ($V_{in} = 1.84V$, $V_{NOR} = 0V$)

Input Set (I_1, I_2, I_3)	Threshold E	Output	Logic Output
(0,0,0)	V_{NAND}	$V_0 = (x_1 - E) \sim V_{in}$	1
(0,0,1) or (1,0,0) or (0,1,0)	$V_{NAND} + V_{in}$	$V_0 = (x_1 - E) \sim V_{in}$	1
(0,1,1) or (1,1,0) or (1,0,1)	$V_{NAND} + 2V_{in}$	$V_0 = (x_1 - E) \sim V_{in}$	1
(0,0,0)	$V_{NAND} + 3V_{in}$	$V_0 \sim 0V$ as $x_1 < E$	0

Table 4 Truth table for NAND gate implementation ($V_{in} = 1.84V$, $V_{NOR} = -3.68V$)

system was implemented by the circuit shown in Fig.3, with circuit component values: [$L = 18mH$, $R = 1710\Omega$, $C_1 = 10nF$, $C_2 = 100nF$, $R_1 = 220\Omega$, $R_2 = 220\Omega$, $R_3 = 2.2k\Omega$, $R_4 = 22k\Omega$, $R_5 = 22k\Omega$, $R_3 = 3.3k\Omega$, $D =$ IN4148, B_1, B_2 = Buffers, OA1 - OA3 : opamp μA741]. The x_1 dynamical variable (corresponding to the voltage V1 across the capacitor C1) is thresholded by a control circuit shown in the dotted box in Fig. 3, with voltage E setting varying thresholds. In the circuit, VT corresponds to the output signal from the threshold controller. Note that, as in the implementation of 2-input gates, we are only replacing $dx_2/dt = x_1 - x_2 + x_3$ by $dx_2/dt = E - x_2 + x_3$ in Eq.(2), when $x_1 > E$, and there is no controlling action if $x_1 \leq E$.

The schematic diagram for the NAND/NOR gate implementation is depicted in Fig.2. In the representative example shown here, $V_{in} = 1.84V$. The NOR gate is

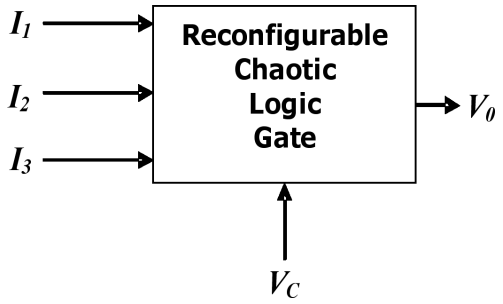

Fig. 2 Symbolic diagram for dynamic 3-Input NOR/NAND logic cell. Dynamic control signal V_C determines the logic operation. In our example, V_C can switch between V_{NAND} giving a NAND gate, and V_{NOR} giving a NOR gate.

realized around $V_C = V_{NOR} = 0V$ and the NAND gate is realized with $V_C = V_{NAND} = -3.68V$ (See Tables 3 and 4).

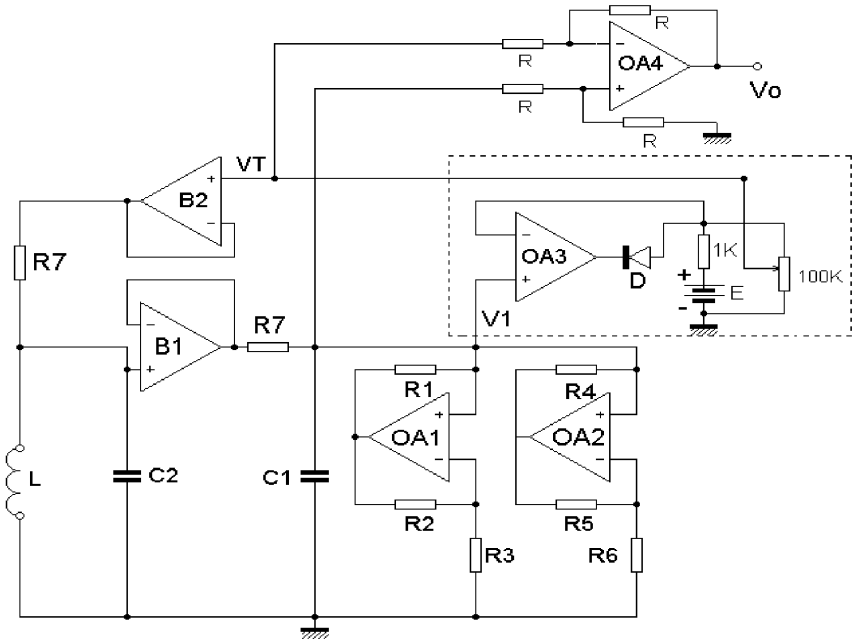

Fig. 3 Circuit module implementing a RCLG that morphs between NAND and NOR logic gates. The diagram represented in the dotted region is the threshold controller. Here $E = V_C + I_1 + I_2 + I_3$ is the dynamically varying threshold voltage. VT is the output signal from the threshold controller and V_0 is the difference voltage signal.

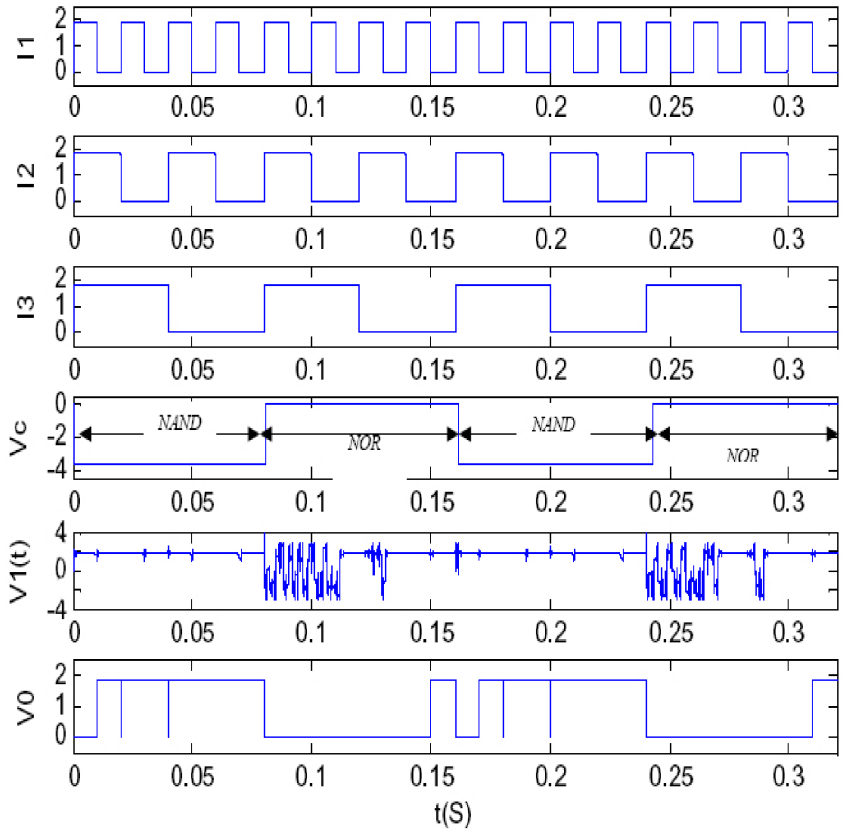

Fig. 4 Voltage timing sequences from top to bottom (PSPICE simulation): (a) First input $I1$, (b) Second input $I2$, (c) Third input $I3$, (d) Dynamic control signal V_C, where V_C switches between $V_{NAND} = -3.68V$ and $V_{NOR} = 0V$ (e) Output signal $V1$ (corresponding to $x_1(t)$) from the Chua's circuit, (f) Recovered logic output signal from $V0$. The fundamental period of oscillation of this circuit is $0.33mS$.

Thus the nonlinear evolution of the element has allowed us to obtain a control signal that selects out temporal patterns corresponding to NOR and NAND gates. For instance in Fig.4, as the dynamic control signal V_C switches between $-3.68V$ to $0V$, the element yields first a NAND gate and then morphs into a NOR gate. The fundamental period of oscillation of the Chua's circuit is $0.33ms$. The average latency of morphing between logic gates is 48% of this period.

5 VLSI Implementation of Chaotic Computing Architectures – Proof of Concept

Recently ChaoLogix Inc. designed and fabricated a proof of concept chip that demonstrates the feasibility of constructing reconfigurable chaotic logic gates, henceforth ChaoGates, in standard CMOS based VLSI ($0.18\mu m$ TSMC process operating at $30Mhz$ with a $3.1 \times 3.1mm$ die size and a $1.8V$ digital core voltage). The basic building block ChaoGate is shown schematically in Fig. 5. ChaoGates were then incorporated into a ChaoGate Array in the VLSI chip to demonstrate higher order morphing functionality including:

1. A small Arithmetic Logic Unit (ALU) that morphs between higher order arithmetic functions (multiplier and adder/accumulator) in *less than one clock cycle*. An ALU is a basic building block of computer architectures.

2. A Communications Protocols (CP) Unit that morphs between two different complex communications protocols *in less than one clock cycle*: Serial Peripheral Interface (SPI, a synchronous serial data link) and an Inter Integrated Circuit Control bus implementation (I2C, a multi-master serial computer bus).

While the design of the ChaoGates and ChaoGate Arrays in this proof of concept VLSI chip was not optimized for performance, it clearly demonstrates that ChaoGates can be constructed and organized into reconfigurable chaotic logic gate arrays capable of morphing between higher order computational building blocks. Current efforts are focused upon optimizing the design of a single ChaoGate to levels where they are comparable or smaller to a single NAND gate in terms of power and size yet are capable of morphing between all gate functions in under a single computer clock cycle. Preliminary designs indicate that this goal is achievable and that all gates currently used to design computers may be replaced with ChaoGates to provide added flexibility and performance.

Acknowledgments

We acknowledge the support of the Office of Naval Research [N000140211019].

References

1. Sinha, S. and Ditto, W.L. *Phys. Rev. Lett*. **81** (1998) 2156.
2. Sinha, S., Munakata, T. and Ditto, W.L, *Phys. Rev. E* **65** (2002) 036214; Munakata, T., Sinha, S. and Ditto, W.L, IEEE *Trans. Circ. and Systems* **49** (2002) 1629.
3. Sinha, S. and Ditto, W.L. *Phys. Rev. E* **59** (1999) 363; Sinha, S., Munakata, T. and Ditto, W.L *Phys. Rev. E* **65** 036216.

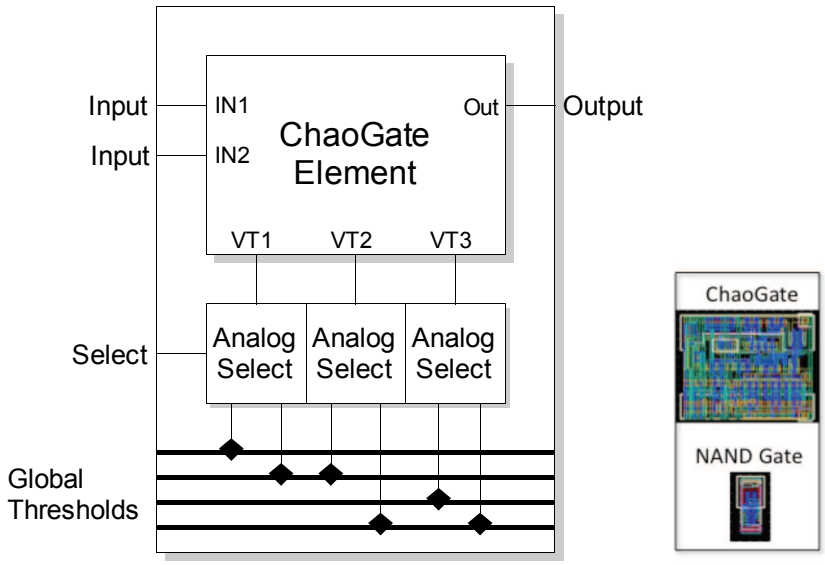

Fig. 5 (Left) Schematic of a two-input, one output morphable ChaoGate. The gate logic functionality (NOR, NAND, XOR,) is controlled (morphed), in the current VLSI design, by global thresholds connected to VT1, VT2 and VT3 through analog multiplexing circuitry and (Right) a size comparison between the current ChaoGate circuitry implemented in the ChaoLogix VLSI chaotic comuting chip and a typical NAND gate circuit (Courtesy of ChaoLogix Inc.)

4. Murali, K., Sinha, S. and Ditto, W.L., Proceedings of the STATPHYS-22 Satellite conference Perspectives in Nonlinear Dynamics Special Issue of Pramana 64 (2005) 433
5. Murali, K., Sinha, S. and Ditto, W.L., *Int. J. Bif. and Chaos (Letts)* **13** (2003) 2669; Murali, K., Sinha S., and I. Raja Mohamed, I.R., *Phys. Letts. A* **339** (2005) 39.
6. Murali, K., Sinha, S., Ditto, W.L., Proceedings of *Experimental Chaos Conference* (ECC9), Brazil (2006) published in *Philosophical Transactions of the Royal Society of London* (Series A) (2007)
7. W. Ditto, S. Sinha and K. Murali, US Patent Number 07096347 (August 22, 2006).
8. Sinha, S., *Nonlinear Systems*, Eds. R. Sahadevan and M.L. Lakshmanan, (Narosa, 2002) 309-328; Murali, K. and Sinha, S., *Phys. Rev. E* **68** (2003) 016210 ; Ditto, W.L. and Sinha, S., *Philosophical Transactions of the Royal Society of London* (Series A) **364** (2006) 2483-2494.
9. Dimitriev, A.S. *et al*, J. Comm. Tech. Electronics, **43** (1998) 1038.