PHILOSOPHICAL TRANSACTIONS OF THE ROYAL SOCIETY A

Phil. Trans. R. Soc. A (2006) **364**, 2483–2494
doi:10.1098/rsta.2006.1836
Published online 27 July 2006

Exploiting the controlled responses of chaotic elements to design configurable hardware

By Sudeshna Sinha[1] and William L. Ditto[2,3,*]

[1]*The Institute of Mathematical Sciences, Taramani, Chennai 600 113, India*
[2]*J. Crayton Pruitt Family Department of Biomedical Engineering, University of Florida, Gainesville, FL 326611-6131, USA*
[3]*Chaologix Inc., 101 SE 2nd Place, Suite 201-B, Gainesville, FL 32601, USA*

We discuss how threshold mechanisms can be effectively employed to control chaotic systems onto stable fixed points and limit cycles of widely varying periodicities. Then, we outline the theory and experimental realization of fundamental logic-gates from a chaotic system, using thresholding to effect control. A key feature of this implementation is that a single chaotic 'processor' can be flexibly configured (and re-configured) to emulate different fixed or dynamic logic gates through the simple manipulation of a threshold level.

Keywords: threshold mechanism; logic gates; dynamic computer architecture

1. Outline

We will first discuss the general formalism of an easily implementable control strategy, namely, the use of a simple *threshold* mechanism to limit the dynamic range of a state variable, thereby effecting flexible control over the dynamic behaviour of the system. Then, we will focus on the application of thresholding to *flexibly* obtain a wide range of controlled logic-gate responses, from a *single* chaotic element. The 'dynamic logic cells' thus obtained can potentially serve as building blocks for a novel dynamic logic architecture.

2. Threshold control algorithm

Consider a general N-dimensional dynamical system, described by the evolution equation, $\mathrm{d}x/\mathrm{d}t = F(\boldsymbol{x}, t)$, where $\boldsymbol{x} \equiv (x_1, x_2, \ldots x_N)$ are the state variables. In this system, a variable x_i is chosen to be monitored and threshold controlled. The prescription for threshold control is as follows: control will be triggered whenever the value of the monitored variable exceeds a prescribed critical threshold x^* (i.e. when $x_i > x^*$) and the variable x_i will then be reset to x^* (Sinha 1994, 1995, 2002; Glass & Zheng 1994). The dynamics continues until the next occurrence of x_i exceeding the threshold, when control resets its value to x^* again.

* Author for correspondence (william.ditto@bme.ufl.edu).

One Contribution of 15 to a Theme Issue 'Exploiting chaotic properties of dynamical systems for their control'.

© 2006 The Royal Society

Table 1. Threshold values versus periodicity, of a few representative controlled cycles, for the chaotic logistic map $x_{n+1}=4x_n(1-xn)$. (Note that cycles of the same period, but different geometries, can be obtained in different threshold windows.)

threshold	nature of controlled orbit
$x^*<0.75$	period 1 (fixed point)
$0.75<x^*<0.905$	period 2 cycle
$x^*\sim 0.965$	period 3 cycle
$0.905<x^*<0.925$	period 4 cycle
$x^*\sim 0.979$	period 5 cycle
$x^*\sim 0.93$	period 6 cycle
$x^*\sim 0.9355$	period 7 cycle
$x^*\sim 0.932$	period 8 cycle
$x^*\sim 0.981$	period 9 cycle
$x^*\sim 0.95$	period 10 cycle

This method only involves the monitoring and the occasional resetting of a single variable and no parameters are perturbed in the original system. The theoretical basis of the method does not involve stabilizing unstable periodic orbits (Ditto *et al.* 1990; Ott *et al.* 1990), but rather involves *clipping* desired time sequences and enforcing a periodicity on the sequence through the thresholding action, which acts as a resetting of initial conditions. The effect of this scheme is to *limit the dynamic range* slightly, i.e. 'snip' off small portions of the available phase space and this small controlling action is effective in yielding regular dynamics. In fact, chaos is advantageous here, as it possesses a rich range of temporal patterns, which can be clipped to widely ranging stable behaviours. This immense variety is not available from thresholding regular systems.

It has been analytically proven that thresholding can yield stable periodic orbits of all orders in one-dimensional chaotic maps (Sinha 1994, 1995, 2002; see table 1). The analytical results based on symbolic dynamics (Sinha 1994, 1995, 2002; Glass & Zheng 1994) are exactly corroborated in a circuit realization of the logistic map (Murali & Sinha 2003; see figure 1 for traces of representative controlled orbits).

The success of the threshold method on higher dimensional systems, including hyperchaotic systems, has also been demonstrated through extensive numerical and laboratory experiments (Sinha & Ditto 2001; Murali & Sinha 2003). For instance, it has been implemented in circuit realizations of chaotic jerk systems, which are nonlinear third-order ordinary differential equations (ODEs),

$$\frac{\mathrm{d}^3x}{\mathrm{d}t^3}+A\frac{\mathrm{d}^2x}{\mathrm{d}t^2}+\frac{\mathrm{d}x}{\mathrm{d}t}=G(x), \tag{2.1}$$

where $G(x)$ is a piecewise linear function: $G(x)=B|x|-C$ with $B=1.0$, $C=2.0$ and $A=0.6$ (Sprott 2000). On this system, we implement the threshold mechanism on variable x, i.e. when $x>x^*$, x is clipped to x^*. A precision-clipping circuit (Maddock & Calcutt 1997) is employed for this threshold action.

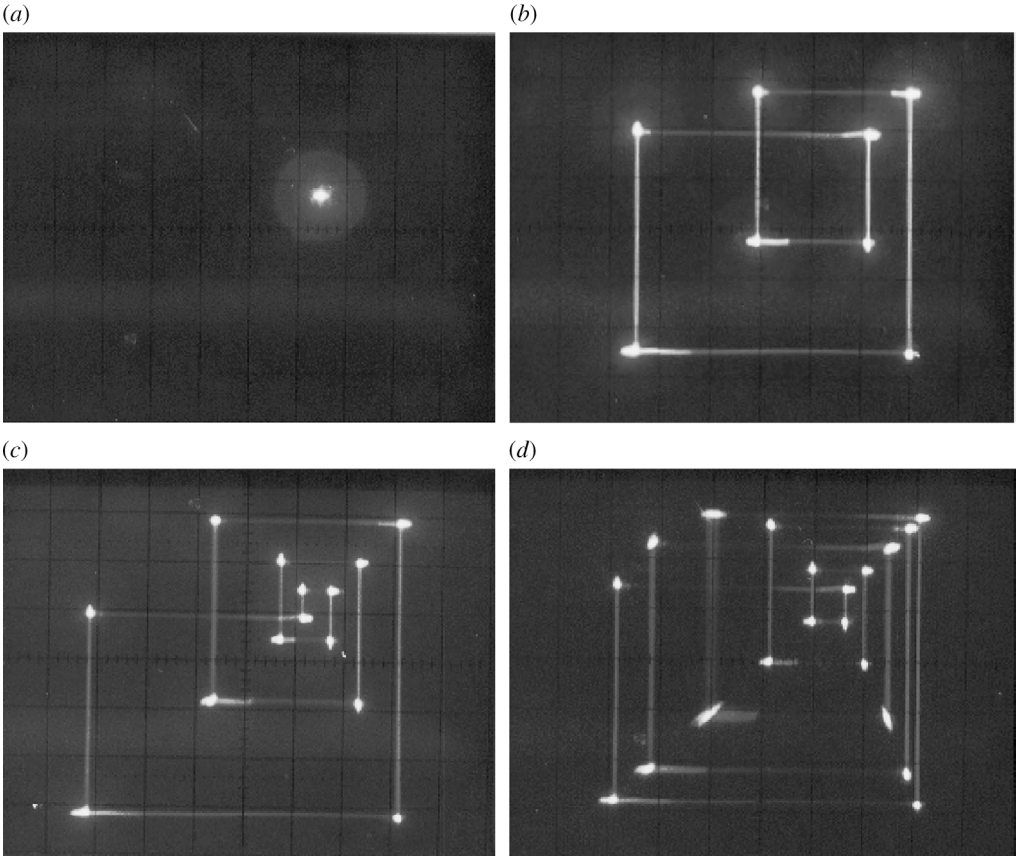

Figure 1. Experimental verification of a range of controlled periods in a circuit realization of the logistic map. The ordinate and abscissa represent traces of x_{n+1} and x_n for (*a*) period 1 cycle, (*b*) period 4 cycle, (*c*) period 7 cycle with threshold and (*d*) period 10 cycle. The threshold levels at which these cycles were obtained coincide exactly with those predicted theoretically.

Another representative example is the double-scroll chaotic Chua's attractor given by the following set of (rescaled) three coupled ODEs (Dmitriev *et al.* 2001)

$$\dot{x}_1 = \alpha(x_2 - x_1 - g(x_1)), \tag{2.2}$$

$$\dot{x}_2 = x_1 - x_2 + x_3, \tag{2.3}$$

$$\dot{x}_3 = -\beta x_2, \tag{2.4}$$

where $\alpha = 10$ and $\beta = 14.87$, and the piecewise linear function $g(x) = bx + (1/2) \times (a-b)(|x+1| - |x-1|)$ with $a = -1.27$ and $b = -0.68$. In this system, we implement an even more minimal thresholding. Instead of demanding that the x_1 variable be reset to x^*, if it exceeds x^*, we demand this *only* in equation (2.3). This has an easy implementation, as it avoids modifying the value of x_1 in the nonlinear element, which is hard to do. Thus, all we do then is to implement $x_2 = x^* - x_2 + x_3$ instead of equation (2.3), when $x_1 > x^*$, and there is no controlling action if $x_1 > x^*$.

Table 2. Threshold ranges (in V) versus periodicity of the controlled cycle for the chaotic system given by (I) equation (1) and (II) equations (2.2)–(2.4).

threshold for system I	threshold for system II	nature of controlled orbit
$x^* < -2.00$	$x^* < 1.84375$	fixed point
$-2.00 < x^* < 1.477$	$1.84375 < x^* < 2.235$	period 1 cycle
$1.477 < x^* < 2.242$	$2.235 < x^* < 2.258$	period 2 cycle
$2.242 < x^* < 2.321$	$2.258 < x^* < 2.264$	period 4 cycle
$2.321 < x^* < 2.325$	$2.264 < x^* < 2.265$	period 8 cycle
$2.325 < x^* < 2.331$	$2.265 < x^* < 2.2653$	period 16 cycle

The results of the threshold mechanism in these systems are summarized in table 2. It is clear that the chaotic dynamics gets clipped to different stable regular cycles for different threshold values.

The control transience is very short here (typically of the order of 10^{-3} times the controlled cycle). This makes the control practically instantaneous. The underlying reason for this is that the system does not have to be close to any particular unstable fixed point before control comes into effect, as in schemes based on the Ott–Grebogi–Yorke concept (Ditto *et al.* 1990; Ott *et al.* 1990). Once a specified state variable exceeds the threshold, it is caught immediately in a stable orbit. Hence, there is no significant interval between the onset of control action and the achievement of control. In addition, thresholding does not entail any run-time computation during control, which reduces control latencies. Lastly, threshold control is highly robust with respect to noise, both in the threshold setting and in the dynamics. This is easy to see analytically for one-dimensional systems, since thresholding by design creates a *super-stable* orbit (as the derivative of the effective map which determines the stability of the controlled system is exactly zero (Sinha 1994, 1995, 2002)). This robustness is also borne out in numerical and experimental realizations for higher dimensional systems.

(*a*) *Hyperchaotic system*

The method has also been demonstrated on a hyperchaotic electrical circuit (Murali & Sinha 2003). This constitutes a stringent test of the control method since the system possesses more than one positive Lyapunov exponent, hence *more than one unstable eigen direction* has to be reigned in by thresholding a *single* variable. In particular, we consider the realization of four coupled nonlinear (rescaled) ODEs of the form

$$\dot{x}_1 = (k-2)x_1 - x_2 - G(x_1 - x_3), \tag{2.5}$$

$$\dot{x}_2 = (k-1)x_1 - x_2, \tag{2.6}$$

$$\dot{x}_3 = -x_4 + G(x_1 - x_3), \tag{2.7}$$

$$\dot{x}_4 = \beta x_3, \tag{2.8}$$

where $G(x_1 - x_3) = 1/2b[|x_1 - x_3 - 1| + (x_1 - x_3 - 1)]$, with $k = 3.85$, $b = 88$ and $\beta = 18$ (Murali *et al.* 2001). Again, we implement a *partial* thresholding on variable

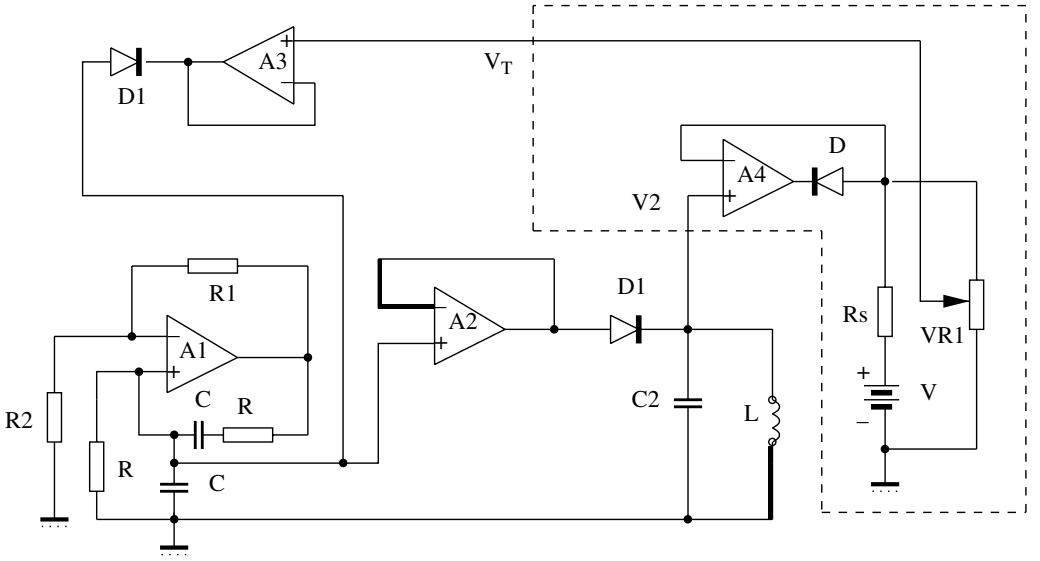

Figure 2. Circuit implementation of equations (2.5)–(2.8). The precision-clipping circuit, affecting threshold control, is in the dotted box. V_T is the threshold-controlled signal.

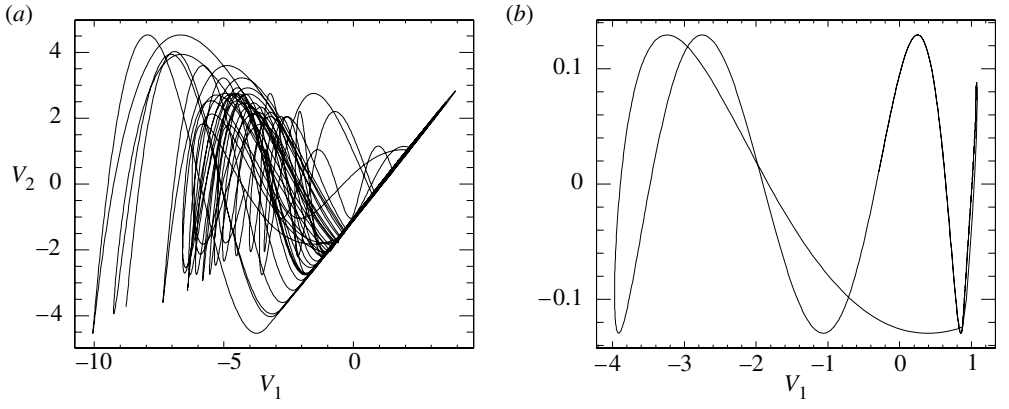

Figure 3. (*a*) Uncontrolled hyperchaotic attractor and (*b*) controlled attractor for threshold $=0$ *V*, in the V_1-V_2 plane, corresponding to the x_1-x_3 plane of equations (2.5)–(2.8).

x_3; whenever $x_3 > x^*$ in the system, $G(x_1-x_3)$ in equation (2.5) becomes $G(x_1-x^*)$, i.e. we have $x_1=(k-2)x_1-x_2-G(x_1-x^*)$, while equations (2.6)–(2.8) are unchanged. When $x_3 \leq x^*$, there is no action at all (see figure 2).

Both our experiments and numerical simulations (which are in complete agreement) show that this scheme successfully yields regular *stable* cycles under a wide range of thresholds (Murali & Sinha 2003). A representative example with threshold set at $0\,V$ is shown in figure 3, which shows the controlled cycle in the V_1-V_2 plane corresponding to the rescaled x_1-x_3 plane of equations (5)–(8).

Thresholding, then, is especially useful in the situation where one wishes to *design* controllable components that can switch flexibly between different behaviours. Calibrating the system characteristics at the outset, with respect to threshold, gives one a *look-up table* to directly and simply effect control at all

consequent times, at no run-time cost. Thus, this scheme has considerable potential for use in chaos-based applications. Further, the simplicity of the controller implies low complexity costs, which is important in technical applications seeking to exploit the richness of chaos in a direct and an efficient way. In the section below, we shall delineate one exciting application of thresholding, namely obtaining different logic responses from a chaotic element.

3. Computing with chaotic elements

Here, we will demonstrate how the threshold controller, which clips chaos into different temporal patterns, can serve as a basis for a dynamic logic unit (Sinha & Ditto 1998, 1999). The aim is to use a single chaotic element to emulate different logic gates, with the ability to switch easily between the different operational roles. Such a computing unit may then allow a more dynamic computer architecture, which is more flexible than fixed hardware. Unlike existing paradigms that try to achieve flexibility by flexible wiring, we envisage here the flexibility arising from the computational modules themselves. Such flexible logic modules can serve as the basis for a programmable general-purpose computer, as all the fundamental gates, which are the necessary and sufficient components of a universal computing machine, can be realized by them.

The necessary and sufficient components of computer architecture to date are the logical AND, OR, NOT and Exclusive OR (XOR) operations from which we can directly obtain all basic operations, like bit-by-bit addition and memory (Mano 1993; Bartee 1991). In conventional computer architectures, all gates can be constructed by combining the fundamental NOR (or NAND) operation. For example, AND can be realized by: AND(X,Y)=NOR(NOR(X,Y),NOR(X,Y)) and XOR(X,Y) = NOR(NOR(NOR(X,NOR(X,Y))),NOR(NOR(NOR(X,X),Y))).

Clearly, this conversion process is inefficient in comparison with direct implementation (which would require only one unit and no cascaded operations). This is especially significant considering, perhaps, that such fundamental operations may be performed billions of times. Therefore, the direct and flexible implementations of gates are useful and could prove highly cost effective.

Here, we will show the *direct and flexible implementation* of *all* these logical operations by thresholding a single chaotic element.

Consider a single chaotic element, whose state is represented by a value x, as our *chaotic chip* or *chaotic processor*. This element receives two inputs I_1 and I_2 (for AND, OR and XOR), or one input I (in case of NOT) and outputs a signal O. The logical operations are defined by patterns of input-to-output mapping represented by the truth table in table 3. Our aim is to design a scheme such that the chaotic elements will yield the appropriate output for all possible sets of inputs.

In our scheme, all logic gate operations involve the following steps (figure 4):

(i) Inputs:

$x \rightarrow x_0 + X_1 + X_2$ for the AND, OR and XOR operations, and
$x \rightarrow x_0 + X$ for the NOT operation
where x_0 is the initial state of the system, and $X=0$ when $I=0$ and $X=\delta$ when $I=1$.

Table 3. The truth table of the basic logic operations. (Column 1 shows AND (I_1, I_2), Column 2 shows OR (I_1, I_2) and Column 3 shows XOR (I_1, I_2), where the 2 inputs are I_1 and I_2. Column 4 shows the NOT gate, where there is 1 input: I.)

I_1	I_2	AND	OR	XOR	I	NOT
0	0	0	0	0	0	1
0	1	0	1	1	1	0
1	0	0	1	1		
1	1	1	1	0		

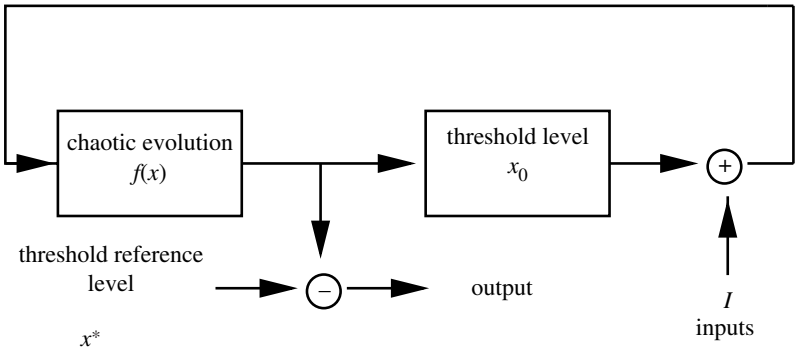

Figure 4. Scheme for obtaining different logic responses from chaotic dynamics.

(ii) Chaotic update: $x \rightarrow f(x)$

where $f(x)$ is a chaotic function.

(iii) Threshold mechanism to obtain *output*, i.e. the output Z is:

$Z=0$ if $f(x) \leq x^*$, and
$Z=f(x)-x^*$ if $f(x)>x^*$
where x^* is the threshold.
This is interpreted as $O=0$ if $Z=0$ and $O=1$ if $Z=\delta$.

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). This is also demanded among the different logical operations. Therefore, constant δ 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. The elements can then be 'wired' directly into gate arrays, implementing compound logic operations.

In order to obtain the desired input–output response, we need to satisfy the conditions enumerated in table 4 for the different gates. Note that the symmetry of inputs reduces the four conditions in the truth table 3 to three distinct conditions, with rows 2 and 3 of table 3 leading to condition 2 in table 4. Hence, given a dynamics $f(x)$, corresponding to the physical device in actual implementation, one must find values of x^* and x_0 that satisfy all the conditions in table 4 simultaneously, in a robust manner (Munakata *et al.* 2002).

Table 4. Necessary and sufficient conditions to be satisfied by a chaotic element in order to implement the logical operations AND, OR, XOR and NOT.

operation	AND	OR	XOR	NOT
condition 1	$f(x_0)\leq x^*$	$f(x_0)\leq x^*$	$f(x_0)\leq x^*$	$f(x_0)-x^*=\delta$
condition 2^*	$f(x_0+\delta)\leq x^*$	$f(x_0+\delta)-x^*=\delta$	$f(x_0+\delta)-x^*=\delta$	$f(x_0+\delta)\leq x^*$
condition 3^*	$f(x_0+2\delta)-x^*=\delta$	$f(x_0+2\delta)-x^*=\delta$	$f(x_0+2\delta)\leq x^*$	

Table 5. Example of the implementation of the logical AND, OR, XOR and NOT operations, using the logistic map $f(x)=4x(1-x)$. (The value of δ is 1/4 in this example.)

operation	AND	OR	XOR	NOT
x_0	0	1/8	1/4	1/2
x^*	3/4	11/16	3/4	3/4

Now, we give an explicit example of the basic procedure laid out earlier. As a representative chaotic function, we take $f(x)$ to be the prototypical logistic map, a map known to be of widespread relevance to physical and biological chaotic phenomena $f(x)=4x\,(1-x)$, where $x\in[0,1]$. We select the constant δ to be 1/4. This value is common to both input and output, and to all logic gates. Table 5 shows a set of x_0 and x^* values that simultaneously satisfy all the conditions in table 4.

For instance, for the AND operation, $x_0=0$ and $x^*=3/4$. This satisfies the three conditions in table 4 as follows:

$$f(x_0) = f(0) = 0 \leq x^* (= 3/4),$$

$$f(x_0 + \delta) = f(1/4) = 0 \leq x^* (= 3/4),$$

$$f(x_0 + 2\delta) - x^* = f(1/2) - 3/4 = 1 - 3/4 = 1/4 = \delta.$$

We would also like to underscore that the different chaos control and synchronization schemes in existence, could possibly be exploited similarly, to obtain different dynamic computing schemes.

The use of chaotic elements is compared with the possible use of periodic elements on one hand, and random elements on the other. It is not possible to extract *all* the different logic responses from the *same* element in case of periodic components, as the temporal patterns are inherently very limited. Therefore, periodic elements do not offer flexibility. Random elements, on the other end, have many different temporal sequences; but they are *not deterministic* and hence one cannot use them to *design* components. Only chaotic dynamics enjoys both richness of temporal behaviour and determinism. Here, we have shown how one can select temporal responses corresponding to

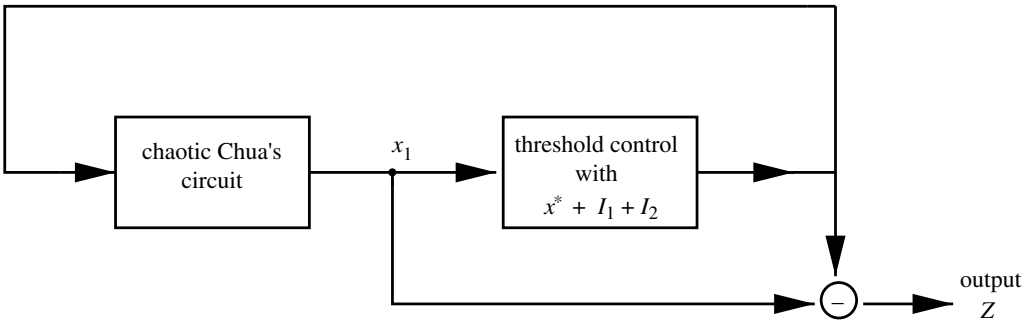

Figure 5. Scheme for implementing the fundamental NOR gate with a Chua's circuit.

different logic-gate patterns from such dynamics, and this ability allows us to construct flexible hardware (our efforts are in contrast to the earlier efforts of Toth (1995)[1]).

The above theoretical scheme of implementing logic gates has been completely verified in proof-of-principle experiments on logistic map circuits. We have also extended the scheme to realize combination logic circuits, such as half-adders. Therefore, the robust cascading of these dynamic logic units allows the construction of basic computation modules, which are important steps towards implementing bit-by-bit arithmetic operations and computer memory.

One can also use continuous time-chaotic systems, for instance, the Chua's circuit given by equations (2.2)–(2.4), to implement logic gates via the simple thresholding discussed in the section above (Murali *et al.* 2003*a*,*b*). A representative example is the implementation of the fundamental NOR operation on a pair of inputs (I_1, I_2). This simply involves setting an input-dependent threshold $x^* + X_1 + X_2$, where $X = 0$ when $I = 0$ and $X = \delta$ when $I = 1$ (see figure 5 for a schematic). That is, the threshold level is: $x^\star$ if the input set is (0, 0), $x^* + \delta$ if the input set is (0, 1)/(1, 0) and $x^* + 2\delta$ if the input set is (1, 1). The output is interpreted as 0 if the thresholded variable x_i is below threshold. The output is interpreted as 1 if the thresholded variable x_i is above threshold, and $x_i - x^* = \delta$.

Now, for the NOR gate, we must have an output 1 for input set (0, 0), and output 0 for input sets (0, 1), (1, 0) and (1, 1). This is obtained robustly for $x^\star \sim 0$ and $\delta \sim 1.84$, as evident from figure 6.

Therefore, unlike conventional *static* architecture-based computing paradigms, these dynamical computing elements have flexibility and re-configurable capability (Sinha *et al.* 2002*a*,*b*). Thus, it can yield a gate architecture that can dynamically switch between different gates, without rewiring the circuit. Such configuration changes can be implemented either by a predetermined schedule or by the outcome of computation. Therefore, the flexibility of obtaining different logic operations using varying thresholds on the same physical element may lead to new dynamic architecture concepts (Taubes 1997).

[1] Our efforts are in contrast to the earlier efforts of Toth, A. & Showalter, K. 1995 *J. Chem. Phys.* **103**, 2058, who obtain gates from chemical systems by delicately tuning many parameters (involving both the construction of the apparatus, as well as the geometric configuration and timing of the input and output waves). Fine adjustments of these lead to the desired phenomena. In contrast here we have an adjustable threshold defining all the gates from the same system.

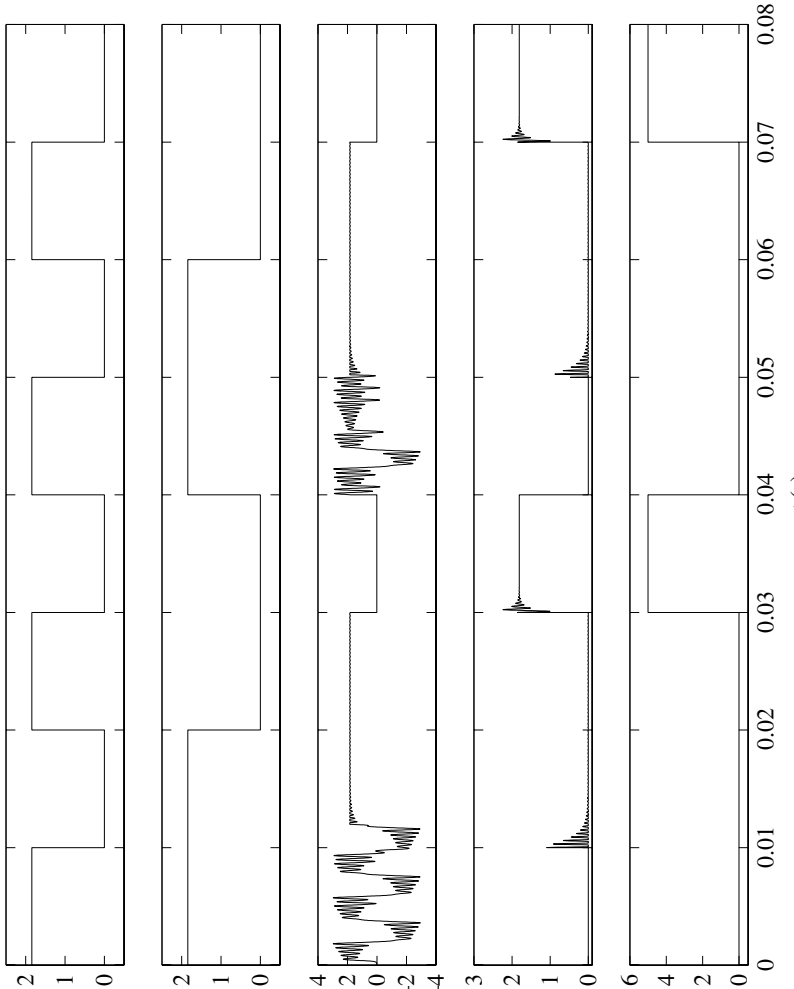

Figure 6. Timing sequences: (*a*) first input I_1, (*b*) second input I_2, (*c*) thresholded signal x_1 in equation (2.3) $\equiv V_T$, (*d*) output signal V_0, which is equivalent to Z in the schematic and (*e*) logic output trace, where $z \sim \delta$ is interpreted as 1 and $z \sim 0$ is interpreted as 0.

Further, since all logic gates are obtained from identical single chaotic circuits, one can envision that this will enable more efficient packing, in analogue programmable gate implementations, as well as more efficient design. Another significant potential advantage is also related to making all elements identical. With clever programming approaches that take advantage of identical elements, elements that endure damage or errors can be effectively disconnected from the whole bath of chaotic elements and the remaining elements can continue normal operations.

In conclusion, we have demonstrated the basic principles of a universal programmable chaotic logic unit that can potentially provide the starting point for more mature approaches of flexibly dynamic computational platforms, based on the principle of large numbers of identical, re-configurable and re-programmable units.

This work was supported under a grant from the US Office of Naval Research (N00014-02-1019).

References

Bartee, T. C. 1991 *Computer architecture and logic design*. New York, NY: Mc-Graw Hill.

Dmitriev, A. S., Kyarginsky, B. Ye, Panas, A. I. & Starkov, S. O. 2001 Direct chaotic communications schemes in microwave band. *J. Commun. Technol. Electron.* **46**, 224–233.

Ditto, W., Rauseo, S. N. & Spano, M. L. 1990 Experimental control of chaos. *Phys. Rev. Lett.* **65**, 3211. (doi:10.1103/PhysRevLett.65.3211)

Glass, L. & Zheng, W. 1994 Bifurcations in flat-topped maps and the control of cardiac chaos. *Int. J. Bif. Chaos* **4**, 1061–1067. (doi:10.1142/S0218127494000770)

Maddock, R. J. & Calcutt, D. M. 1997 *Electronics: a course for engineers*, p. 542. London, UK: Addison-Wesley Longman.

Mano, M. M. 1993 *Computer system architecture*, 3rd edn. Englewood Cliffs, NJ: Prentice Hall.

Munakata, T., Sinha, S. & Ditto, W. L. 2002 Chaoscomputing: implementation of fundamental logicgates by chaotic elements. *IEEE Trans. Circ. Syst.* **49**, 1629–1633. (doi:10.1109/TCSI.2002.804551)

Murali, K. & Sinha, S. 2003 Experimental realization of chaos control by thresholding. *Phys. Rev. E* **68**, 016 210. (doi:10.1103/PhysRevE.68.016210)

Murali, K., Lindberg, E. & Leung, H. 2001 Design principles of hyperchaotic circuits. *AIP Conf. Proc.* **622**, 15–26.

Murali, K., Sinha, S. & Ditto, W. L. 2003*a* Realization of the fundamental NOR gate using a chaotic circuit. *Phys. Rev. E* **68**, 016 205. (doi:10.1103/PhysRevE.68.016205)

Murali, K., Sinha, S. & Ditto, W. L. 2003*b* Implementation of NOR gate by a chaotic Chua's circuit. *Int. J. Bif. Chaos* **13**, 2669. (doi:10.1142/S0218127403008053)

Ott, E., Grebogi, C. & Yorke, J. 1990 Controlling chaos. *Phys. Rev. Lett.* **64**, 1196–1199. (doi:10.1103/PhysRevLett.64.1196)

Sinha, S. 1994 Unidirectional adaptive dynamics. *Phys. Rev. E* **49**, 4832. (doi:10.1103/PhysRevE.49.4832)

Sinha, S. 1995 Adaptive dynamics on circle maps. *Phys. Rev. E* **199**, 365.

Sinha, S. 2002 Controlling chaos with threshold mechanisms. In *Nonlinear systems* (ed. R. Sahadevan & M. L. Lakshmanan), p. 309. New Delhi, India: Narosa.

Sinha, S. & Ditto, W. L. 1998 Dynamics based computation. *Phys. Rev. Lett.* **81**, 2156–2159. (doi:10.1103/PhysRevLett.81.2156)

Sinha, S. & Ditto, W. L. 1999 Computing with distributed chaos. *Phys. Rev. E* **59**, 363–377. (doi:10.1103/PhysRevE.60.363)

Sinha, S. & Ditto, W. 2001 Controlling neuronal spikes. *Phys. Rev. E* **63**, 056 209. (doi:10.1103/PhysRevE.63.056209)

Sinha, S., Munakata, T. & Ditto, W. L. 2002*a* Parallel computing with extended dynamical systems. *Phys. Rev. E* **65**, 036 214. (doi:10.1103/PhysRevE.65.036214)

Sinha, S., Munakata, T. & Ditto, W. L. 2002*b* Flexible parallel implementation of logic gates using chaotic elements. *Phys. Rev. E* **65**, 036 216. (doi:10.1103/PhysRevE.65.036216)

Sprott, J. C. 2000 A new class of chaotic circuit. *Phys. Lett. A* **266**, 19. (doi:10.1016/S0375-9601(00)00026-8)

Taubes, G. 1997 After 50 years, self-replicating silicon. *Science* **277**, 1935. (doi:10.1126/science.277.5334.1935)