PRAMANA
— journal of
physics

Vol. 64, No. 3
March 2005
pp. 433–441

Construction of a reconfigurable dynamic logic cell

K MURALI[1], SUDESHNA SINHA[2] and WILLIAM L DITTO[3]
[1]Department of Physics, Anna University, Chennai 600 025, India
[2]The Institute of Mathematical Sciences, Taramani, Chennai 600 113, India
[3]Department of Biomedical Engineering, University of Florida, Gainesville,
FL 326611-6131, USA
E-mail: kmurali@annauniv.edu; sudeshna@imsc.ernet.in; william.ditto@bme.ufl.edu

Abstract. We report the first experimental realization of all the fundamental logic gates, flexibly, using a chaotic circuit. In our scheme a simple threshold mechanism allows the chaotic unit to switch easily between behaviours emulating the different gates. We also demonstrate the combination of gates through a half-adder implementation.

Keywords. Nonlinear circuits; logic gates; computer architecture.

PACS No. 05.45.+b

1. Introduction

Recently there has been a new theoretical direction in harnessing the richness of chaos, namely the exploitation of chaos to do flexible computations [1,2]. The aim is to use a single chaotic element to emulate different logic gates and perform different arithmetic tasks, and further, to have the ability to switch easily between the different operational roles. Such a computing unit may then allow a more dynamic computer architecture and serve as ingredients of a general-purpose device more flexible than statically wired hardware.

A system is capable of universal general purpose computing if it can emulate all logic gates. The necessary and sufficient components of computer architecture today are the logical AND, OR, NOT and XOR (exclusive OR) operations, from which we can directly obtain basic operations like bit-by-bit addition and memory [3]. We first recall the theoretical scheme for flexible implementation of all these fundamental logical operations utilizing low dimensional chaos [2], and then we give details of the specific realization of the theory in a chaotic circuit.

2. Theory

Here we outline a theoretical scheme for obtaining all basic logic gates with a single chaotic system. Consider a chaotic element (our chaotic chip or chaotic processor)

Table 1. 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		

whose state is represented by a value x. In our scheme all the basic logic gate operations, AND, OR, NOT and XOR (see table 1 for the truth table), involve the following steps:

(1) Inputs:

$x \to x_0 + X_1 + X_2$ for the AND, OR and XOR operations, and
$x \to x_0 + X$ for the NOT operation,

where x_0 is the initial state of the system, and $X = 0$ when $I = 0$ and $X = \delta$ (where δ is a positive constant) when $I = 1$.
(2) Chaotic update, i.e. $x \to f(x)$, where $f(x)$ is a chaotic function.
(3) Threshold mechanism to obtain output Z:

$Z = 0$ if $f(x) \leq x^*$, and
$Z = f(x) - x^*$ if $f(x) > x^*$,

where x^* is the threshold. This is interpreted as logic output 0 if $Z = 0$ and logic ouput 1 if $Z = \delta$. 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 will 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 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, 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, as displayed in table 1, we need to satisfy the conditions enumerated in table 2 simultaneously. Note that the symmetry of inputs reduces the four conditions in the truth table 1 to three distinct conditions, with rows 2 and 3 of table 1 leading to condition 2 in table 2.

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 table to be implemented. For instance, table 3 shows the exact solutions of the initial x_0 and threshold x^* which satisfy the conditions in table 2 when

Table 2. 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 3. One specific solution of the conditions in table 2 which yields the logical operations AND, OR, XOR and NOT, with $\delta = \frac{1}{4}$.

Operation	AND	OR	XOR	NOT
x_0	0	$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{2}$
x^*	$\frac{3}{4}$	$\frac{11}{16}$	$\frac{3}{4}$	$\frac{3}{4}$

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

with parameter $a = 1$. The constant $\delta = \frac{1}{4}$ is common to both input and output and to all logical gates. This is the system we will realize through an electrical circuit in this work.

Contrast our use of chaotic elements with the possible use of periodic elements on the one hand, and random elements on the other. It is not possible to extract all the different logic responses from the same element in the case of periodic components, as the temporal patterns are inherently very limited. So periodic elements do not offer much flexibility or versatility. Random elements on the other hand have many different temporal sequences. But they are not deterministic and so one cannot use them to design components. Only chaotic dynamics enjoys both richness of temporal behaviour as well as determinism. Here we have showed how one can select temporal responses corresponding to different logic gate patterns from such dynamics, and this ability allows us to construct flexible hardware.

However, note that while nonlinearity is absolutely necessary for implementing all the logic gates, chaos may not be always necessary. In the representative example of the logistic map presented here, solutions for all the gates exist only at the fully chaotic limit of the logistic map. But the degree of nonlinearity necessary for obtaining all the desired logic responses will depend on the system at hand and on

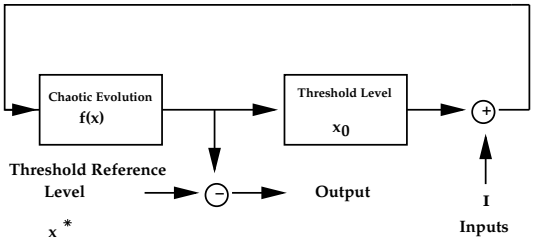
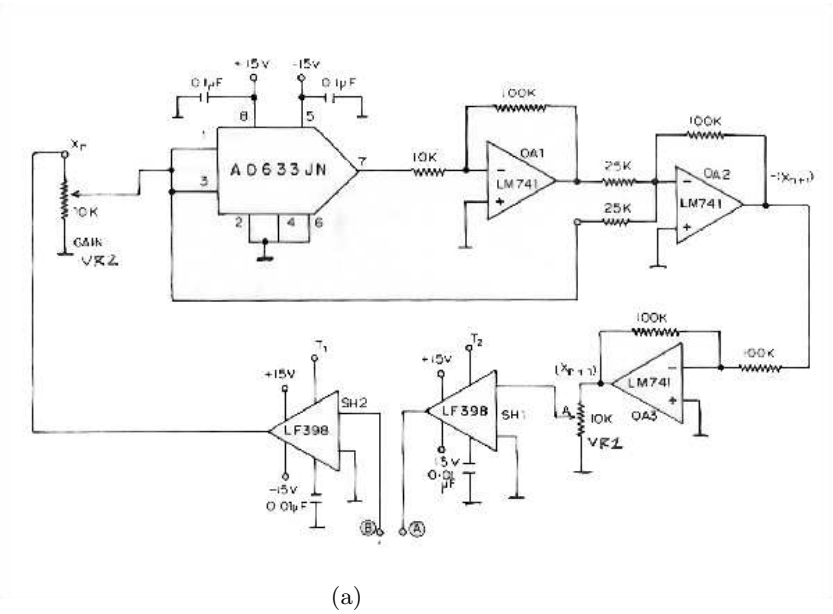
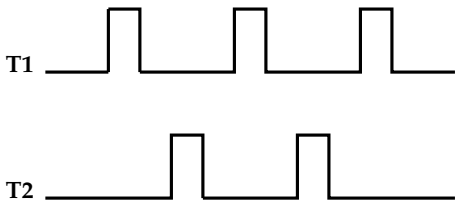


Figure 1. Schematic of the circuit.



(a)



(b)

Figure 2. Circuit implementation of (**a**) logistic map module and (**b**) the timing pulses T1 and T2 generated from the clock generator providing a delay of feedback. The output voltage of OA3 becomes a new input voltage to the multiplier AD633 after passing through two sample-and-hold circuits provided the terminals A and B are connected together. The sample-and-hold circuits are constructed with LF398 or ADG412 ICs and they are triggered by T1 and T2. See text for more details.

the specific scheme employed to obtain the input–output mapping. It may happen that certain nonlinear systems will allow a wide range of logic responses without actually being chaotic.

Now in the section below we will present the experimental realization of the theory above, in a discrete electrical circuit.

3. Experiment

Here we will present an implementation of the theory discussed in the section above, and we will verify that theoretical solutions, such as table 3, are indeed realizable in electronic circuits. The schematic of the circuit is given in figure 1. Here a

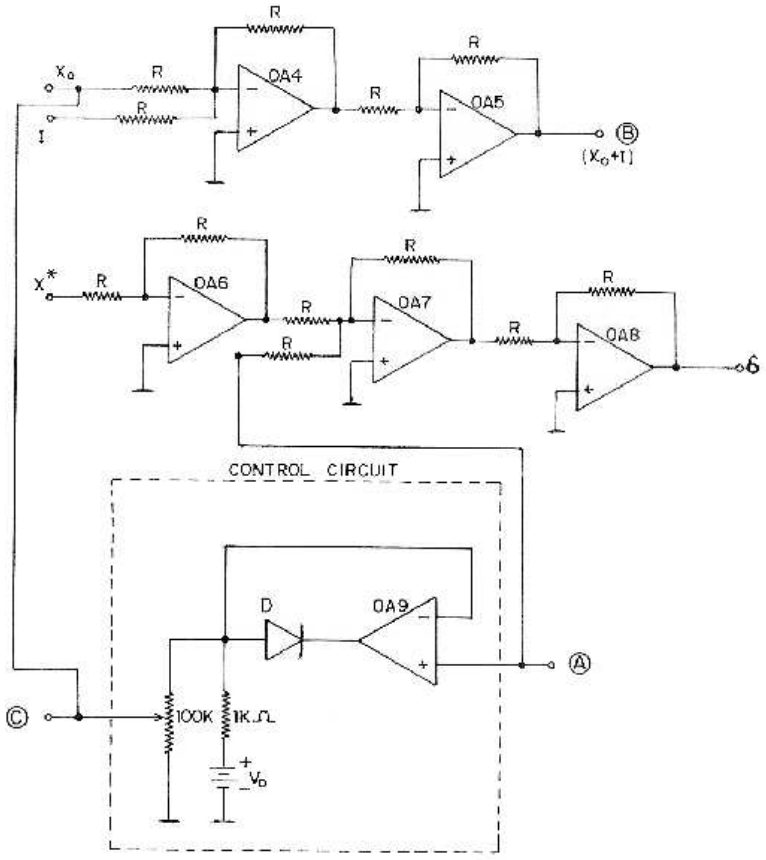


Figure 3. Circuit implementation of the flexible gates module. See text for details.

threshold controller with threshold level set by voltage level x_0 is used. The logic level input $I = I_1 + I_2$ is added to x_0 and used as the new input to the logistic map iteration to generate x_{n+1}. This implements step 1 of the scheme. By using another threshold reference level voltage signal x^*, the signal difference between x_{n+1} and x^* is monitored as the logic level output. This constitutes the third (and final) step of the scheme.

In figure 2a the circuit realization of the chaotic logistic map is depicted. In the circuit implementation x_{n-1}, x_n and x_{n+1} denote voltages normalized by 10 V as the unit. An analog multiplier IC AD633 is used as a squarer and it produces the output voltage of $x_n^2/10$ V for the given x_n as the input. By using suitable scale changer, summing amplifier and an invertor the voltage proportional to x_{n+1} is available at the output of op-amp (OA3) circuit. A variable resistor VR1 is employed to control the parameter a from 0 to 1 in the logistic map. The output voltage of OA3 becomes a new input voltage to the multiplier AD633 after passing through two sample-and-hold circuits (SH1 and SH2) provided the terminals A and B are connected together. The sample-and-hold circuits are constructed with

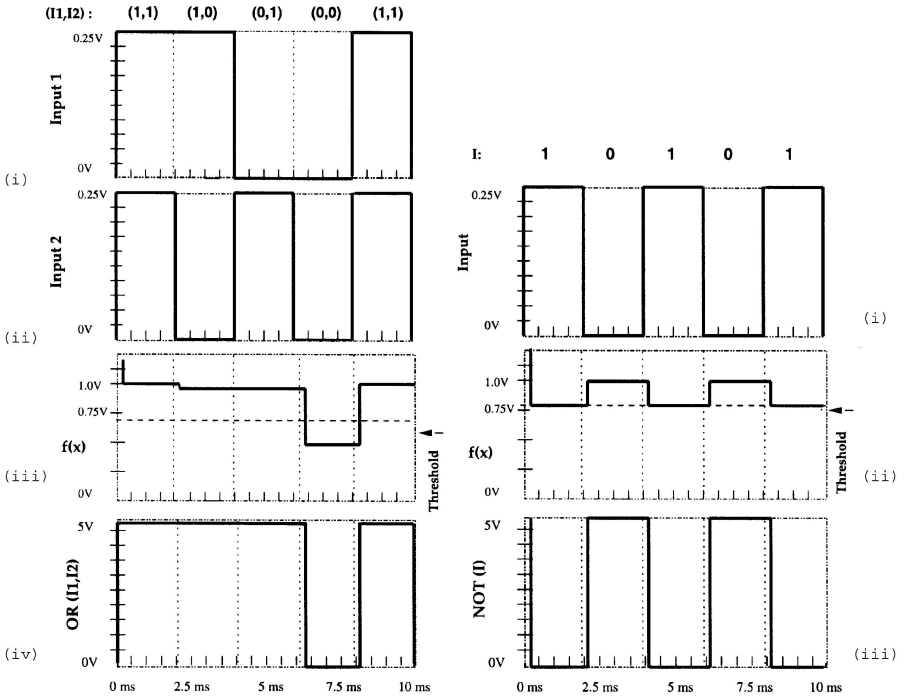


Figure 4. Timing sequences of the OR gate implementation (left): (i) First input I_1, (ii) second input I_2, (iii) state after chaotic update $f(x)$, and (iv) output obtained by thresholding; and timing sequences of the NOT gate implementation (right): (i) Input I, (ii) state after chaotic update $f(x)$, and (iii) output obtained by thresholding (accuracy within 5 mV).

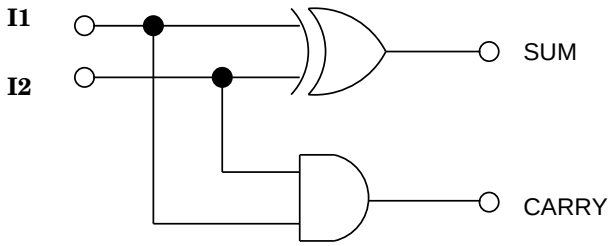


Figure 5. Schematic representation of the half adder implementation on inputs I_1 and I_2, with SUM $\equiv$ XOR (I_1, I_2) and CARRY $\equiv$ AND (I_1, I_2).

LF398 or ADG412 ICs and they are triggered by suitable delayed timing pulses T1 and T2 (shown in figure 2b). The timing pulses are usually generated from the clock generator providing a delay of feedback and the delay is essential for obtaining the solution x_{n+1} of the logistic map. Usually the clock rate of either 5 kHz or 10 kHz is used.

If the terminals A and B in figure 2a are connected to the respective terminals of the circuit of figure 3, we have the general circuit configuration for the flexible

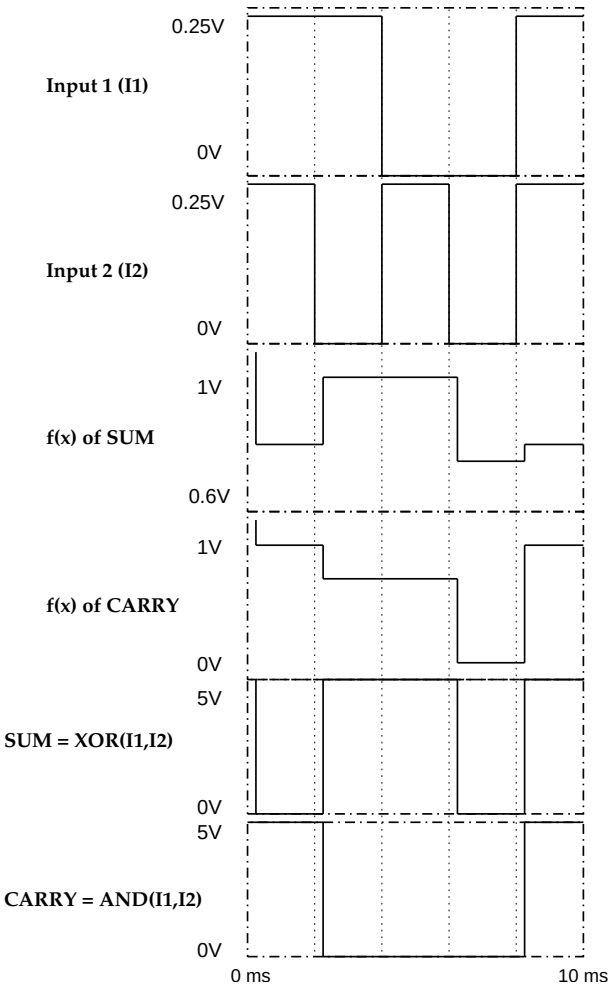


Figure 6. Timing sequences of the half adder implemented by the schematic circuit in figure 5 (accuracy within 5 mV).

logic gate implementation. In the circuit, all input and output variables are again normalized by 10 V. The control circuit (dotted line box) is the threshold control unit which generates the signal x_0 at terminal C corresponding to the input signal x_{n+1} at A under the threshold control voltage V_0. The input voltage $I = 0$ V, 0.25 V or 0.5 V correspond to different logic gates. Here x^* is another reference threshold voltage used to produce the difference voltage δ from the x_{n+1} signal. This δ and the input signal I determine the logic condition of the different gates. Here op-amps OA4 to OA9 are implemented with μA741 or AD712. The resistor R = 100 kΩ and diode D = IN4148 or IN34A.

Figure 4 shows the timing sequences of the implementation of two representative gates. The output waveforms are generated with both PSPICE circuit simulations

and through hardware implementations. Figure 5 shows the schematic of the circuit implementing the half adder and figure 6 shows its corresponding timing sequences. These waveforms are again straightforwardly obtained in both simulation and experiments.

4. Conclusions

In summary, we have experimentally demonstrated the direct and flexible implementation of all the basic logic gates utilizing nonlinear dynamics. The richness of the dynamics allows us to reverse engineer and select all the different gate responses from the same processor by simply setting suitable threshold levels. These threshold levels are known exactly from theory and thus available as a look-up table [4]. Arrays of such logic gates can conceivably be programmed on the run (for instance, with a stream of threshold values being sent in by an external program) to be optimized for the task at hand. For instance they may serve flexibly as an arithmetic processing unit or a unit of memory, and can be swapped, as the need demands, to be one or the other. Thus architectures based on such logic implementations may serve as ingredients of a general-purpose computing device more flexible than statically wired hardware.

Finally, what advantages could universal programmable analog logic gates convey over existing approaches and technologies? Where could it potentially be better? Of course we cannot answer this question without defining 'better' in this context. We can define better in many ways: (1) faster, (2) easier to fabricate (and cheaper), (3) reprogrammable, (4) robust to damage or errors and (5) reconfigurable. The easiest metric is faster. It has been demonstrated that high speed electrical [5,6] and optical [6–8] chaotic systems can be fabricated and controlled with frequencies higher than 1 GHz. Obviously, implementations of threshold controllers at these frequencies are a natural future extension. Another metric is the ease of fabrication. Universal programmable analog logic gates can be constructed from a single chaotic circuit. Thus we can envision that all elements can be identical. This could enable more efficient packing in analog programmable logic gate implementations as well as more efficient designs. Another significant potential advantage is also related to making all elements identical. With clever programming approaches that take advantage of identical elements, elements that can endure damage or error can be effectively disconnected from the whole bath of elements and the remaining elements can continue normal operations (albeit with possibly slower operation). Finally, such computing approaches are fundamentally reconfigurable. Specifically since nonlinear electronic and optical components have been shown to be controllable with very small latencies (in nanosecond and below range), we can reasonably imagine logic gates that can be reconfigured, in principle, at GHz frequencies (and above) to change function and operation subject only to the latencies of the hardware, and software and hardware controllers. In conclusion, we have demonstrated a universal programmable logic gate that could potentially provide the starting point for more mature approaches to faster, robust and reconfigurable computational platforms based on the principle of large numbers of identical, reconfigurable and programmable dynamical logic cells.

References

[1] S Sinha and W L Ditto, *Phys. Rev. Lett.* **81**, 2156 (1998)
S Sinha and W L Ditto, *Phys. Rev.* **E59**, 363 (1999)
S Sinha, T Munakata and W L Ditto, *Phys. Rev.* **E65**, 036216 (2002)
[2] S Sinha, T Munakata and W L Ditto, *Phys. Rev.* **E65**, 036214 (2002)
T Munakata, S Sinha and W L Ditto, *IEEE Trans. Circ. Systems* **49**, 1629 (2002)
[3] M M Mano, *Computer system architecture*, 3rd edition (Prentice Hall, Englewood Cliffs, 1993)
T C Bartee, *Computer architecture and logic design* (Mc-Graw Hill, New York, 1991)
[4] Our efforts are in contrast to the earlier efforts of A Toth and K Showalter, *J. Chem. Phys.* **103**, 2058 (1995), 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 only an adjustable threshold to obtain all the gates from the same system
[5] K Myneni *et al*, *Phys. Rev. Lett.* **83**, 2175 (1999)
[6] N J Corron, *et al*, *Int. J. Bifurcat. Chaos* **13**, 957 (2003)
[7] D W Sukow *et al*, *Chaos* **7**, 560 (1997)
[8] J N Blakely *et al*, *Phys. Rev. Lett.* **92**, 193901 (2004)
[9] J N Blakely *et al*, *IEEE J. Quantum Electron.* **40**, 299 (2004)