Contents lists available at ScienceDirect

Physics Letters A

www.elsevier.com/locate/pla

Dynamic logic architecture based on piecewise-linear systems

Haipeng Peng [a,b,c], Fei Liu [a], Lixiang Li [a,b,c,*], Yixian Yang [a,b,c], Xue Wang [a]

[a] Information Security Center, State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, PO Box 145, Beijing 100876, China
[b] Key Laboratory of Network and Information Attack and Defence Technology of Ministry of Education, Beijing University of Posts and Telecommunications, Beijing 100876, China
[c] National Engineering Laboratory for Disaster Backup and Recovery, Beijing University of Posts and Telecommunications, Beijing 100876, China

ARTICLE INFO

Article history:
Received 23 June 2009
Received in revised form 29 December 2009
Accepted 16 January 2010
Available online 22 January 2010
Communicated by C.R. Doering

Keywords:
Chaotic computing
Piecewise-linear system
Logic operations

ABSTRACT

This Letter explores piecewise-linear systems to construct dynamic logic architecture. The proposed schemes can discriminate the two input signals and obtain 16 kinds of logic operations by different combinations of parameters and conditions for determining the output. Each logic cell performs more flexibly, that makes it possible to achieve complex logic operations more simply and construct computing architecture with less logic cells. We also analyze the various performances of our schemes under different conditions and the characteristics of these schemes.

1. Introduction

The investigation of computing devices with the feature of dynamical architecture which makes the devices to have the ability of reconfigurable is an interesting research direction for designing the next generation of computer chip [1]. Traditionally, the field programmable gate array (FPGA) is used to construct such architecture. Recently, much attention has been paid to the construction of dynamic logic architecture through harnessing nonlinear (chaotic) dynamical systems [2–8]. The aim is to use the cells based on dynamical systems to emulate different logic gates and perform various computing tasks. It can switch among different logic operational roles by changing its parameters. Such a reconfigurable logic unit may then serve as an ingredient for the construction of general purpose programmable hardware. Almost all other existing computing paradigms do not have such flexibility, which may be termed as static architecture.

The chaos computing was firstly proposed by Sinha and Ditto [9]. In chaos computing, chaotic elements can act as different logic elements by changing parameters. Such scheme by chaos computing may be called dynamic logic architecture [1], and this is essentially different from the FPGA. Recently, reliable logic circuit elements that exploit nonlinearity in the presence of a noise-floor are developed [10]. Fault tolerance and detection in chaotic computers are investigated in Ref. [11], and where the idea of distinguishing inputs by weighting is given. Piecewise-linear systems are also used to construct the dynamic logic architecture [12]. Computing devices based on such schemes may be more flexible than statically wired hardware.

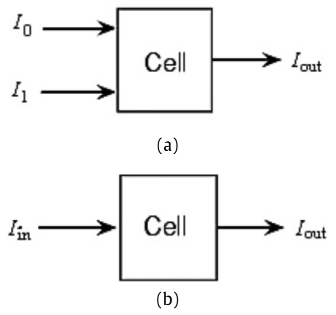

Fig. 1. Two types of input/output configurations: (a) Logical AND, OR, XOR, NOR and NAND; (b) Logical NOT.

In this Letter, we exploit piecewise-linear systems to construct the dynamic logic architecture. Different from the previous works on dynamic logic architecture [1,3,10,12], the proposed schemes can discriminate the two input signals, so that more logical performances can be involved to fit in the needs of various conditions. The newly gained logic operations could improve the logic cells to become more efficient and flexible in practical implementations.

2. The logic cells

Logic cells for basic logical operations can be classified into two types, as shown in Fig. 1(a) and (b). Type (a), with two input sig-

[*] Corresponding author at: Information Security Center, State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, PO Box 145, Beijing 100876, China. Tel.: +86 10 62282264 (office).
E-mail address: lixiang@bupt.edu.cn (L. Li).

0375-9601/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.physleta.2010.01.031

Table 1
The truth table of basic logic gates with two input signals.

		NOR	NAND	XOR	OR	AND	XNOR	0	1
I_0	I_1	$\overline{I_0+I_1}$	$\overline{I_0 I_1}$	$I_0 \oplus I_1$	I_0+I_1	$I_0 I_1$	$\overline{I_0 \oplus I_1}$		
0	0	1	1	0	0	0	1	0	1
0	1	0	1	1	1	0	0	0	1
1	0	0	1	1	1	0	0	0	1
1	1	0	0	0	1	1	1	0	1

Table 2
The truth table of rest logic gates with two input signals.

		G_0	G_1	NG_0	NG_1	F_0	F_1	T_0	T_1
I_0	I_1	$\overline{I_0} I_1$	$I_0 \overline{I_1}$	$I_0+\overline{I_1}$	$\overline{I_0}+I_1$	$\overline{I_0}$	$\overline{I_1}$	I_0	I_1
0	0	0	0	1	1	1	1	0	0
0	1	1	0	0	1	1	0	0	1
1	0	0	1	1	0	0	1	1	0
1	1	0	0	1	1	0	0	1	1

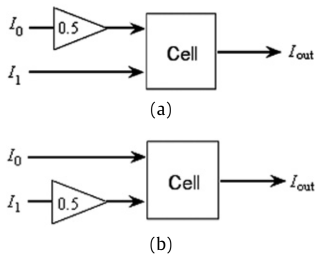

Fig. 2. The type of input/output configurations, which can discriminate I_0 and I_1.

nals, is employed for basic logic gates of NOR (from which all gates may be constructed), AND, OR, XOR, and NAND. Type (b), with one input, is employed for basic logic gate NOT. The gates of AND, OR, XOR, NAND, and NOT are the basic logic gates to build a computer.

All the logic operations, which may be obtained by the logic gates with two input signals are listed in Tables 1 and 2. We can see that there are totally 16 different logics, and that the obtained basic logical gates by the previous work [1,3,10,12] are listed in Table 1. In Table 2, we list eight kinds of logic gates which are not basic logic gates with two input signals. And G_0, G_1, NG_0 and NG_1 can be employed as comparators; F_0 and F_1 can be employed as NOT gates:

- G_0: when I_1 is greater than I_0, $I_{out}=1$;
- G_1: when I_0 is greater than I_1, $I_{out}=1$;
- NG_0: when I_1 is not greater than I_0, $I_{out}=1$;
- NG_1: when I_0 is not greater than I_1, $I_{out}=1$;
- F_0: I_{out} is contrary to I_0;
- F_1: I_{out} is contrary to I_1;
- T_0: I_{out} equals I_0;
- T_1: I_{out} equals I_1.

For the logic gates with two input signals, previous works used I_0+I_1 as the input parameters of the dynamical systems. However, these methods cannot discriminate the difference of the two inputs (I_0 and I_1) because of the symmetry of addition. This can be easily found in gates AND, OR, XOR and NAND, which give the same results when (I_0, I_1) is $(1,0)$ or $(0,1)$.

In this Letter, a type of logic cells with various logical performances is proposed with the difference from the previous ones, that it can make a distinction between the two input signals by halving the value of I_0 or I_1, as shown in Fig. 2. This means that the input parameter of the dynamical systems will be $0.5I_0+I_1$ or $I_0+0.5I_1$, which is different from I_0+I_1 in the previous works. This change of input parameter may make the logic analysis of the

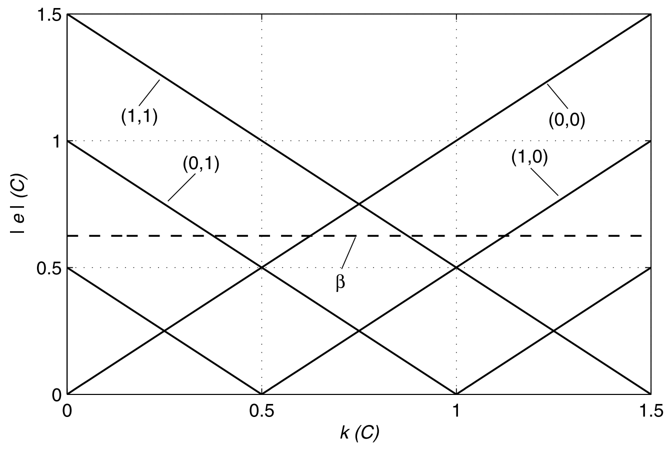

Fig. 3. The curves of $|e|$ for different combination of (I_0, I_1, k) when $m=1$.

systems more complex. However, the proposed schemes also make it possible to achieve more kinds of logic computing operations by this discrimination of the two input signals.

3. Schemes for obtaining logic gates by linear systems

In this scheme, a logic cell with two input signals is constructed based on the discrete linear systems as follows:

$$\begin{cases} x(n+1) = (0.5I_0C + I_1C)x(n), \\ y(n+1) = ky(n), \\ I_{out} = 1, \quad \text{if } |e| = |y(m) - x(m)| < \beta, \\ I_{out} = 0, \quad \text{else} \end{cases} \tag{1}$$

where C and β are positive constants; k is the parameter which acts as the logic gates controller; m is the number of iterative steps; I_0, I_1 are the input logic signals; and I_{out} is the output signal. We choose the cell (a) in Fig. 2 in this scheme, and the cell (b) in Fig. 2 will be discussed in Section 4.

All the fundamental logical operations involve the following three steps:

- External inputs and initialization of states. Initiating the states of $x(0) = y(0) = 1$, the input set (I_0, I_1) determines the parameter $0.5I_0C + I_1C$ of the x-subsystem in Eq. (1). There are two inputs for AND, OR, XOR, NOR and NAND gates, and one input for NOT gate.
- Run the linear system for m steps.
- Obtain the output I_{out}.

Now we show how to obtain different kinds of logic gates. For example, if we set $C=1$, $\beta=0.625$, $k=0.75$, and $m=1$, which means that the computing will only involve one step. When we input $(0,0)$ or $(1,1)$ for (I_0, I_1), we have $|e| = |y(1) - x(1)| = |0.75| = 0.75$. Since $0.625 < 0.75$, the output is 0. If we input $(1,0)$ for (I_0, I_1), we have $|e| = |y(1) - x(1)| = |0.25| = 0.25$. Since $0.25 < 0.625$, the output is 1. Else if we input $(0,1)$ for (I_0, I_1), we have $|e| = |y(1) - x(1)| = |-0.25| = 0.25$. Since $0.25 < 0.625$, the output is 1. Thus, under this condition, the cell performs a XOR logic gate. If we only change parameter k from 0.75 to 0.5, then the cell will change from XOR gate to NAND gate. We can achieve the transformations among the F_1, NAND, XOR, OR and T_1 gates by changing the parameter k.

When we choose different values of m, the logical performance will also change. Several logic operations with different values of m are indicated as follows.