Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 735189, 7 pages
http://dx.doi.org/10.1155/2013/735189

Research Article

A Reconfigurable Logic Cell Based on a Simple Dynamical System

Lixiang Li,[1,2,3] Chunyu Yang,[2] Sili Hui,[2] Wenwen Yu,[2] Jürgen Kurths,[4] Haipeng Peng,[2] and Yixian Yang[5]

[1] *Institute of Network Coding, The Chinese University of Hong Kong, Shatin N.T., Hong Kong*
[2] *Information Security Center, State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, P.O. Box 145, Beijing 100876, China*
[3] *Shenzhen Research Institute, The Chinese University of Hong Kong, Shenzhen 518063, China*
[4] *Potsdam Institute for Climate Impact Research, 14473 Potsdam, Germany*
[5] *School of Information Engineering, Beijing Institute of Graphic Communication, Beijing 102600, China*

Correspondence should be addressed to Lixiang Li; li_lixiang2006@163.com

Received 21 June 2013; Accepted 7 July 2013

Academic Editor: Ming Li

This paper introduces a new scheme to achieve a dynamic logic gate which can be adjusted flexibly to obtain different logic functions by adjusting specific parameters of a dynamical system. Based on graphical tools and the threshold mechanism, the distribution of different logic gates is studied, and a transformation method between different logics is given. Analyzing the performance of the dynamical system in the presence of noise, we discover that it is resistant to system noise. Moreover, we find some part of the system can be considered as a leaky integrator which has been already widely applied in engineering. Finally, we provide a proof-of-principle hardware implementation of the proposed scheme to illustrate its effectiveness. With the proposed scheme in hand, it is convenient to build the flexible, robust, and general purpose computing devices such as various network coding routers, communication encoders or decoders, and reconfigurable computer chips.

1. Introduction

For years, the construction of integrated circuits has required a vast amount of time and money for combining different logic gates. In 1985, when the first field-programmable gate array (FPGA) was introduced to the world, the era of reusable "field-programming" began which led to a more flexible implementation of integrated circuits. However, the speed of an FPGA reconfigurable scheme is typically slow, since it needs some time for "rewiring" [1].

In 1998, a novel way of configuring dynamic logic gates was introduced by Sinha and Ditto [2]. Based on a threshold mechanism and chaotic maps, they proposed a scheme to construct dynamic computing systems with flexible logic functions. Their method permitted faster switching (typically within only 0.5 clock cycle) between any two kinds of logics. Nowadays, more schemes have been conducted to construct new types of dynamic logic gates, including synchronization of a nonlinear system [3] as well as the interplay of square waves and noise [4]. Recently, piecewise linear systems were also suggested to construct the dynamic logic architecture [5]. The development of dynamic computing has brought about the appearance of commercial chaotic computer [6].

In this work, we propose a scheme to obtain dynamic logic functions by controlling simple dynamical systems. Based on the threshold mechanism, we give a transformation method between different logics and analyze its antinoise and time-delay characteristics. We find that the scheme is robust to system noise. Furthermore, the main part of the system can be designed based on the leaky integrator which has been applied into different research fields, such as in neuronal or cell analysis and filters related to signal processing. Finally, the scheme is proved to be effective by simulation results of a logic gate circuit.

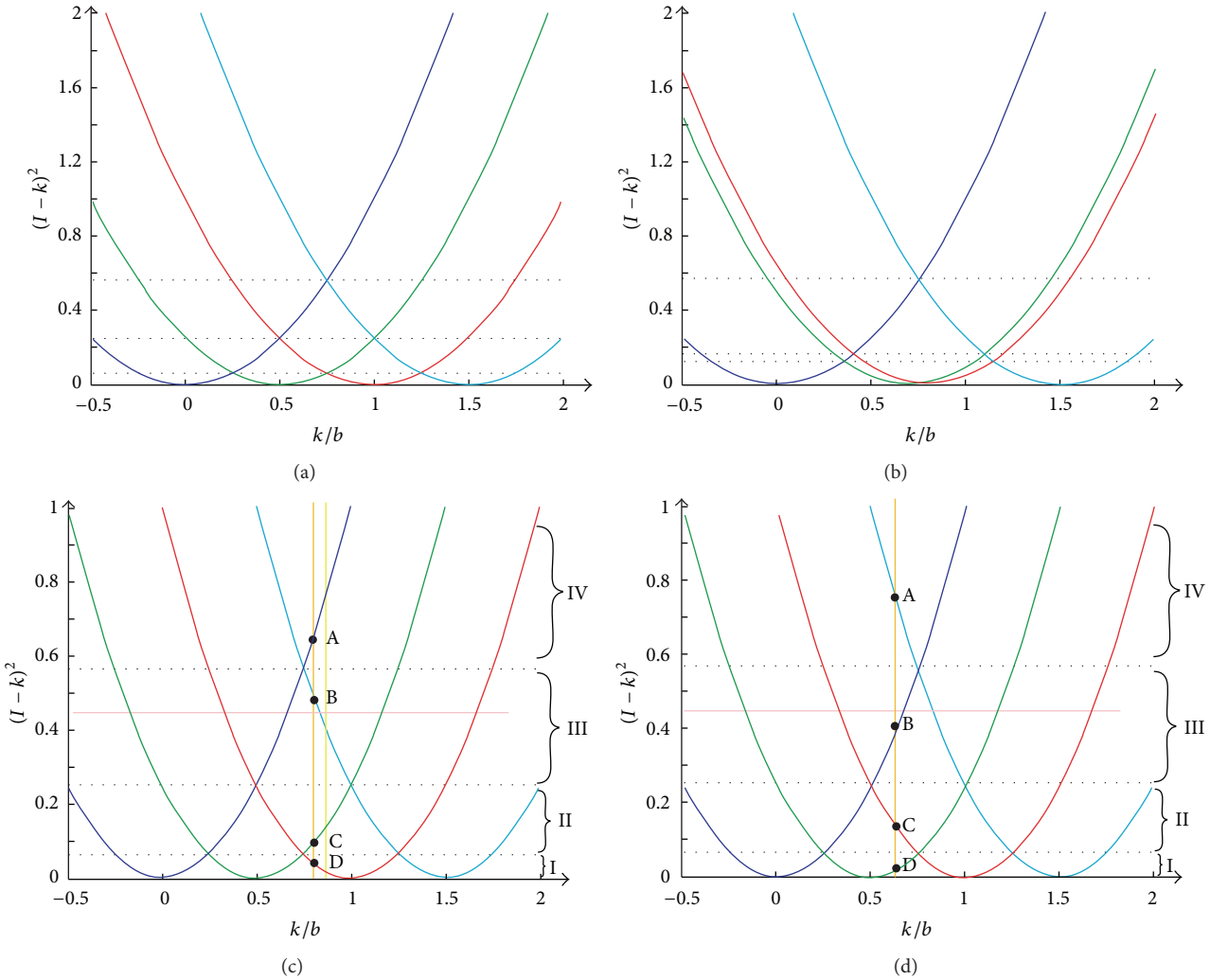

FIGURE 1: (Color online) the judging method of the system output, where the navy blue, green, red, and baby blue lines represent the situations of inputs (0, 0), (0, 1), (1, 0), and (1, 1), respectively.

2. A Scheme of Dynamic Logic Gate

We now propose a new method to change the function of a logic gate flexibly by altering only one parameter or two specific parameters. The formula of its implementation is

$$\dot{x} = -px + p(I - k)^2, \tag{1}$$

where x is the state of system (1), I is the input of the logic gate, $p > 1$ determines the convergence rate of the system, and k is the control parameter to achieve a transformation between different logics.

When system (1) is stable, its state x will converge to the constant as follows:

$$x = (I - k)^2. \tag{2}$$

Based on the threshold mechanism introduced by Murali et al. [4], the output of the logic gate can be determined by

$$I_{\text{out}} = 0 \quad \text{if } x < \beta; \quad I_{\text{out}} = 1 \quad \text{else.} \tag{3}$$

To implement the dynamic logics, the most significant step is to set p, k, and β based on some specific applications. A general situation will be discussed in this paper.

3. Explanation and Discussion of the Proposed Scheme

Typically, we consider that a logic gate has two inputs and one output, for example, we suppose that

$$I = a \cdot I_0 + b \cdot I_1, \tag{4}$$

where I_0 and I_1 are two logic inputs being either 0 or 1, and a and b are the parameters. (I_0, I_1) has four possible values. The relationship between the inputs and the output is shown in Table 1.

Figure 1(a) shows the situation of $b = 2a$, while in Figure 1(b), a and b do not have to have any relationships. We can see that both Figures 1(a) and 1(b) can be divided into four logical areas based on the intersections among these four

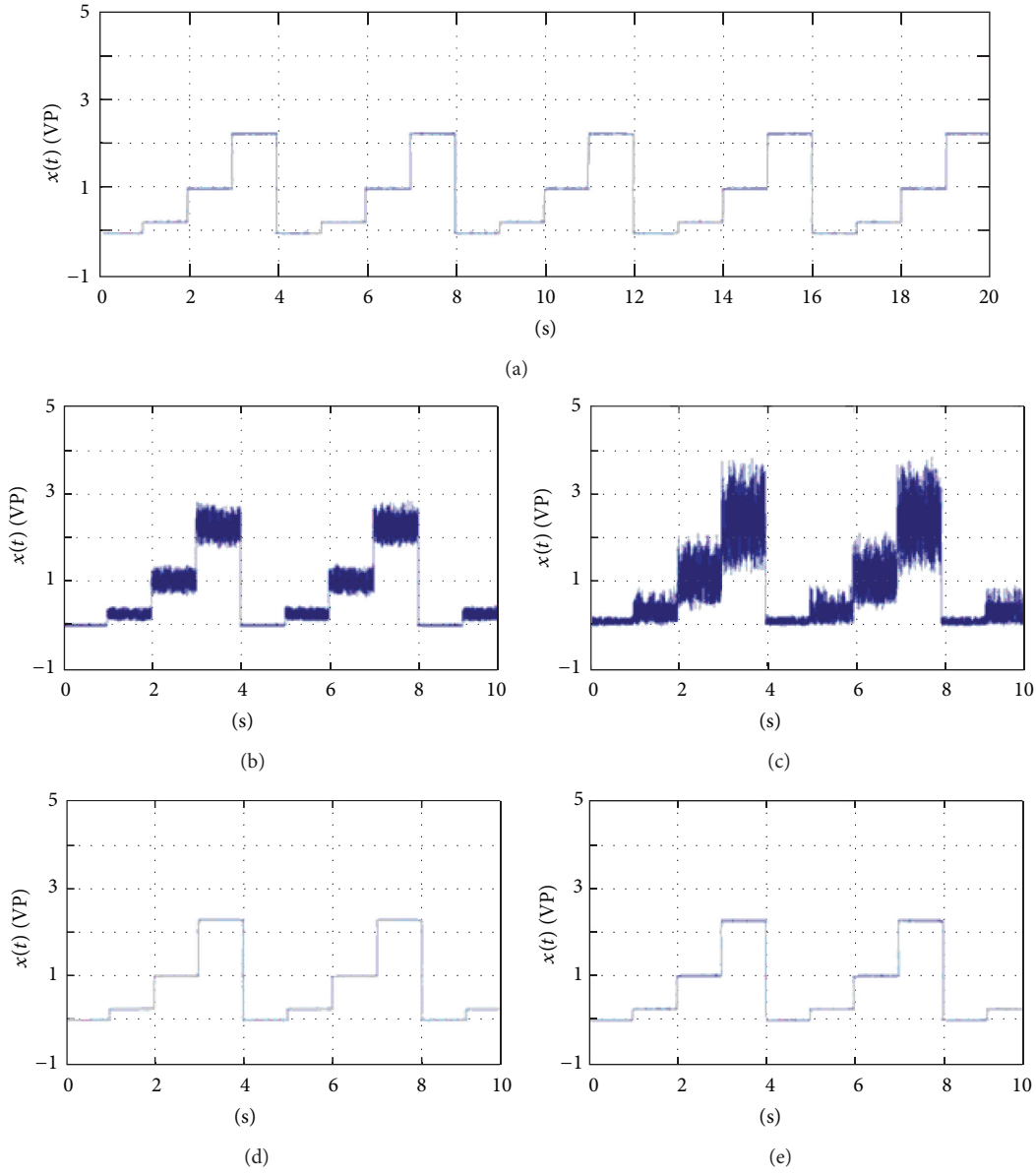

Figure 2: (Color online) the influences of input and system noises on the performance of the system, where (a) in a noiseless case; (b) in the presence of input noise whose range is [−0.2, +0.2] V; (c) in the presence of input noise whose range is [−0.5, +0.5] V; (d) in the presence of system noise whose range is [−0.5, +0.5] V; and (e) in the presence of system noise whose range is [−10, +10] V.

Table 1: The relationship between the inputs and the output.

(a) For arbitrary a and b

Input 0	0	0	1	1
Input 1	0	1	0	1
Output	0	b	a	$a + b$

(b) For $b = 2a$

Input 0	0	0	1	1
Input 1	0	1	0	1
Output	0	$2a$	a	$3a$

curves, which states that there are four possible ranges for the threshold β. Similarly, it can be easily concluded that when $b = 2a$ or $a = 2b$, possible logic functions can be uniformly distributed along the k-axis. Hence, when $b = 2a$ or $a = 2b$, the logic values are clearly determined, and a confusion is less likely to occurr. To simplify the problem and avoid some confusion, $b = 2a$ will be used in this paper. Then, the value of I is simply $I = a \cdot (I_0 + 2I_1)$.

Figures 1(c)-1(d) show the judging method for the system output, where the logic value of the system is determined by a curve intersection method, for example, different points in these figures show different states of the system, and the

Table 2: All available logic gates for the system.

Region	β	k/a	Logic gate
I	$\left(0, \frac{1}{16}\right]$	$\left(-\infty, -\sqrt{\beta}\right)$	1
		$\left(-\sqrt{\beta}, \sqrt{\beta}\right)$	OR
		$\left(\sqrt{\beta}, \frac{1}{2} - \sqrt{\beta}\right)$	1
		$\left(\frac{1}{2} - \sqrt{\beta}, \frac{1}{2} + \sqrt{\beta}\right)$	$I_1 + I_0'$
		$\left(\frac{1}{2} + \sqrt{\beta}, 1 - \sqrt{\beta}\right)$	1
		$\left(1 - \sqrt{\beta}, 1 + \sqrt{\beta}\right)$	$I_1' + I_0$
		$\left(1 + \sqrt{\beta}, \frac{3}{2} - \sqrt{\beta}\right)$	1
		$\left(\frac{3}{2} - \sqrt{\beta}, \frac{3}{2} + \sqrt{\beta}\right)$	NAND
		$\left(\frac{3}{2} + \sqrt{\beta}, +\infty\right)$	1
II	$\left(\frac{1}{16}, \frac{1}{4}\right]$	$\left(-\infty, -\sqrt{\beta}\right)$	1
		$\left(-\sqrt{\beta}, \frac{1}{2} - \sqrt{\beta}\right)$	OR
		$\left(\frac{1}{2} - \sqrt{\beta}, \sqrt{\beta}\right)$	I_1
		$\left(\sqrt{\beta}, 1 - \sqrt{\beta}\right)$	$I_1 + I_0'$
		$\left(1 - \sqrt{\beta}, \frac{1}{2} + \sqrt{\beta}\right)$	XNOR
		$\left(\frac{1}{2} + \sqrt{\beta}, \frac{3}{2} - \sqrt{\beta}\right)$	$I_1' + I_0$
		$\left(\frac{3}{2} - \sqrt{\beta}, 1 + \sqrt{\beta}\right)$	I_1'
		$\left(1 + \sqrt{\beta}, \frac{2}{3} + \sqrt{\beta}\right)$	NAND
		$\left(\frac{2}{3} + \sqrt{\beta}, +\infty\right)$	1
III	$\left(\frac{1}{4}, \frac{9}{16}\right]$	$\left(-\infty, -\sqrt{\beta}\right)$	1
		$\left(-\sqrt{\beta}, \frac{1}{2} - \sqrt{\beta}\right)$	OR
		$\left(\frac{1}{2} - \sqrt{\beta}, 1 - \sqrt{\beta}\right)$	I_1
		$\left(1 - \sqrt{\beta}, \sqrt{\beta}\right)$	AND
		$\left(\sqrt{\beta}, \frac{3}{2} - \sqrt{\beta}\right)$	XNOR
		$\left(\frac{3}{2} - \sqrt{\beta}, \frac{1}{2} + \sqrt{\beta}\right)$	NOR
		$\left(\frac{1}{2} + \sqrt{\beta}, 1 + \sqrt{\beta}\right)$	I_1'
		$\left(1 + \sqrt{\beta}, \frac{3}{2} + \sqrt{\beta}\right)$	NAND
		$\left(\frac{3}{2} + \sqrt{\beta}, +\infty\right)$	1
IV	$\left(\frac{9}{16}, +\infty\right)$	$\left(-\infty, -\sqrt{\beta}\right)$	1
		$\left(-\sqrt{\beta}, \frac{1}{2} - \sqrt{\beta}\right)$	OR
		$\left(\frac{1}{2} - \sqrt{\beta}, 1 - \sqrt{\beta}\right)$	I_1
		$\left(1 - \sqrt{\beta}, \frac{3}{2} - \sqrt{\beta}\right)$	AND
		$\left(\frac{3}{2} - \sqrt{\beta}, \sqrt{\beta}\right)$	0
		$\left(\sqrt{\beta}, \frac{1}{2} + \sqrt{\beta}\right)$	NOR
		$\left(\frac{1}{2} + \sqrt{\beta}, 1 + \sqrt{\beta}\right)$	I_1'
		$\left(1 + \sqrt{\beta}, \frac{3}{2} + \sqrt{\beta}\right)$	NAND
		$\left(\frac{3}{2} + \sqrt{\beta}, +\infty\right)$	1

functionality of the system can be altered by changing k. For each case of these four possible ranges of β, when k is known, the logic value can be determined. For example, when $1/4 \le \beta < 9/16$, for $\sqrt{\beta} < k < 2/3 - \sqrt{\beta}$ and $1 - \sqrt{\beta} < k < \sqrt{\beta}$, which are shown in Figures 1(c)-1(d), respectively. There are four intersection points between the straight line of k and these four curves. In Figure 1(c), since the values of point A and point B are higher than that of β, we get $I_{\text{out}} = 1$ by (3); similarly, since the values of point C and point D are lower than that of β, we get $I_{\text{out}} = 0$. Hence, for inputs $(0, 0)$ and $(1, 1)$, the output is 1; while for inputs $(0, 1)$ and $(1, 0)$, the output is 0. The corresponding logic gate is an XNOR gate. Similarly, for Figure 1(d), the output for input $(1, 1)$ is 1 and that for inputs $(0, 0)$, $(0, 1)$, and $(1, 0)$, is 0. That is, the function of AND gate is achieved. It is worth noting that the transformation from XNOR gate to AND can be realized by only changing the control parameter k.

By a similar analytical method, all the logic gates can be achieved by the proposed scheme as summarized in Table 2. It is clearly seen that, by only altering the value of k, eight types of logic gates can be achieved. For example, when $\beta \in (1/16, 1/4]$, the possible logic functions that can be achieved are 1, OR, I_1, $I_1 + I_0'$, XNOR, $I_1' + I_0$, I_1', NAND.

3.1. Analysis in the Presence of Noise. In reality, noise is unavoidable. Generally, there are two types of noise: input noise and system noise. Figure 2(a) shows the output results of system without noise. When the system is added with input noise, then $\dot{x} = -px + p(I + D\eta(t) - k)^2$, and its steady state is

$$x = \left[I + D\eta\left(t\right) - k\right]^2, \tag{5}$$

where $\eta(t)$ is the additive white Gaussian noise (AWGN) and D is its intensity. Figures 2(b)-2(c) show the simulation results

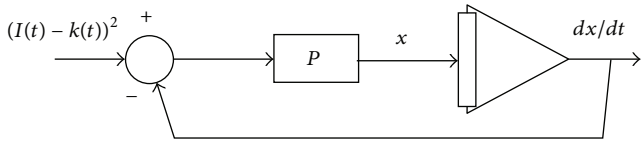

FIGURE 3: The system block diagram of a leaky integrator.

in the presence of input noise, where (b) the noise range is [−0.2, +0.2] V and (c) the noise range is [−0.5, +0.5] V. It can be clearly seen that when the input noise increases, the system becomes more fluctuating.

When the system is added with system noise, then $\dot{x} = -px + p(I-k)^2 + D\eta(t)$, and the steady state of the system is

$$x = [I-k]^2 + \frac{D\eta(t)}{p}. \tag{6}$$

Figures 2(d)-2(e) show the simulation results in the presence of system noise, where (d) the noise range is [−0.5, +0.5] V and(e) the noise range is [−10, +10] V. We can see from Figures 2(d)-2(e) that the system is strongly resistant to system noise which is one of its most important advantage. Therefore, if we want to build a robust logic gate, then we should put the best effort to minimize the input noise.

3.2. Analysis of Delay. The parameter p has an important influence on the response time of the system. The system equation $\dot{x} = -px + p(I-k)^2$ can be rearranged into

$$\frac{1}{p} \cdot \frac{dx}{dt} = -x(t) + (I(t) - k(t))^2. \tag{7}$$

Hence, we can obtain a system of a leaky integrator whose block diagram is shown in Figure 3. It was proposed as a vital digital signal processing filter which has been very popular in different areas. It has been used to investigate biological and artificial learning processes. Moreover, its famous application in neuron network has made the computation much easier and more powerful [7]. The system here is applicable to study further details in timing and delays.

Note that $1/p$ is the time constant of the system. Then, we get from (7)

$$x(t) = e^{-t/\tau}x(0) + p\int_0^t e^{-p(t-\tau)}(I(t-\tau) - k(t-\tau))^2 d\tau. \tag{8}$$

If $(I(t) - k(t))^2$ is defined as a constant (e.g., C), then the relationship between p and $x(t)$ is

$$x(t) = e^{-pt}x(0) + C\left(1 - e^{-pt}\right). \tag{9}$$

Figure 4 shows the evolution of $x(t)$ as $p = 1, 3, 5, 10$. We can see from Figure 4 that the larger the value of p is, the shorter the time lag to reach a constant result is, or the more accurate the desired output is. If p is very small, the system may not fully response to an input and cannot reach a stable state before the next input starts. This indicates that

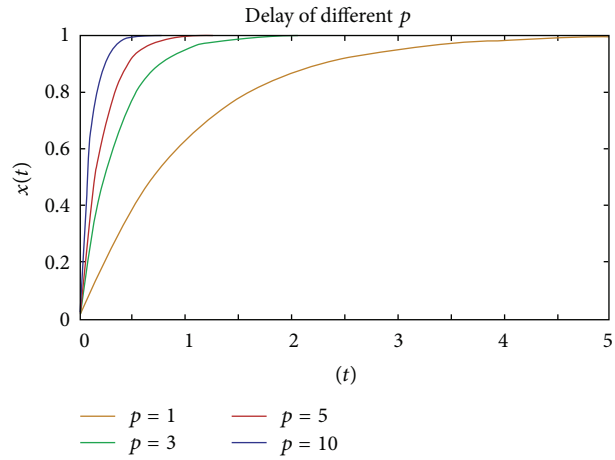

FIGURE 4: (Color online) the evolution of $x(t)$ as $p = 1, 3, 5, 10$, respectively.

the response time of each transformation will be longer than the output intervals, and errors will occurr. Therefore, we can say that larger p means a faster response time and a more accurate output. However, it is not the larger the better. In practice, larger p also means higher energy-consuming amplifier. Users should design a system based on specific applications to make the system work more effectively.

3.3. Circuit Implementation. The physical implementation of a logic cell is an important step for successful applications [8, 9]. Figure 5 shows the equivalent circuit of system (1) by simulation with multisim software.

In Figure 5, there are two parts which are the computation part and the judgment one serving for computing the solution of system (1) as well as judging whether the solution exceeds the threshold, respectively. The left part of Figure 5 corresponds to the computation part in which there are two subtractors, one multiplexer, one amplifier, and one integrator. The right part of Figure 5 corresponds to the judgment one in which there is an operational amplifier serving as the voltage comparator. For subtractor 1, the output is $V_t = I_0 + 2I_1 - k$ and, for subtractor 2, the output is $(I-k)^2 - x$. For the voltage comparator, we can change the threshold by altering the value of the DC source.

By (1) and (3), all the parameters in the circuit of Figure 5 can be calculated. Therefore, certain circuits can be designed according to specific applications. For example, if all the parameters are set properly, then we can achieve an OR logic gate. Figure 6 gives the stream of input signals I_0, I_1 and the output I_{out} for the OR logic gate. The inputs I_0 and I_1 are square waves with 10 Vp amplitudes and 2 kHz and 4 kHz frequencies, respectively. The voltage values of I_0, I_1, and I_{out} are shown in Figure 6 with 200 us/Div time base and 5 V/Div scale.

Similar results can be achieved as the frequencies of the inputs increases. When the frequencies of the inputs become large enough, the lag time must be taken into consideration. As discussed above, the increment of p leads to smaller lag

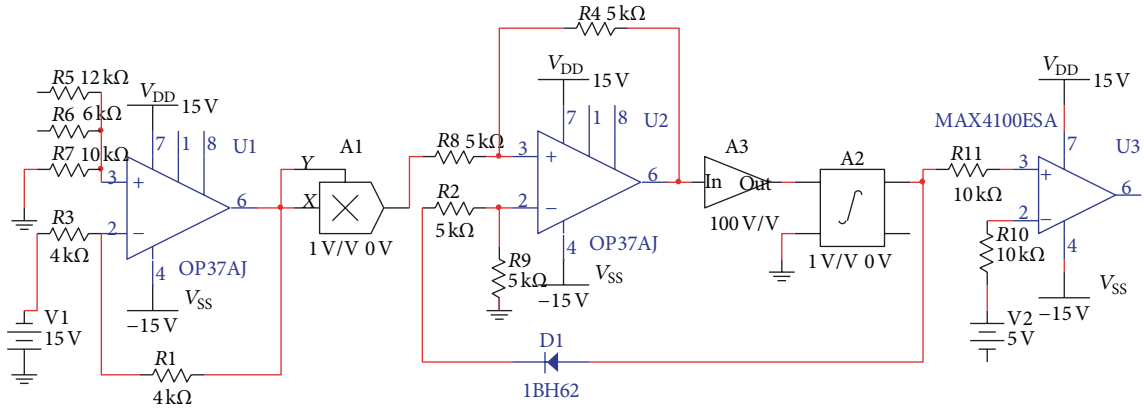

Figure 5: (Color online) the circuit diagram of the system (1).

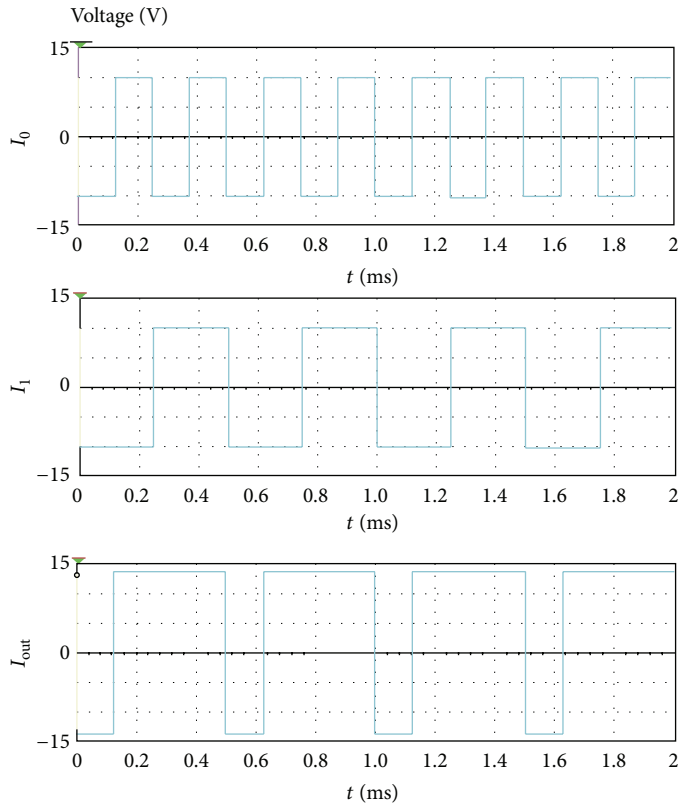

Figure 6: (Color online) simulation results of an OR gate.

time and faster convergence. This phenomenon is obvious for larger frequency inputs.

The proposed method can be used to other systems such as fractional oscillators [10, 11], and we may obtain some potential interesting results.

4. Conclusion

To sum up, a scheme to realize a dynamic logic gate is introduced in this paper. Based on the proposed scheme, all available logics and its transformation method are discussed. Besides, the noise and lag characteristics of the system are studied. We find that the system is resistant to system noise and its response time can be easily controlled. Finally, a circuit implementation for an OR logic gate is provided as an example. Other feasible logic gates can be achieved similarly. The scheme is both straightforward and robust which enables a strong flexible hardware implementation with very low cost. This dynamic logic gate can be applied as a universal basic hardware element to build various kinds of communication encoders and decoders, network coding routers, specific reconfigurable computer chips, graphics processor units, reconfigurable multimedia video cards, or specific systems that require frequent transformations between different

logics. Moreover, there are some further significant directions to be investigated such as all kinds of reconfigurable network coding routers and reconfigurable cyclic code encoder or decoder based on the proposed reconfigurable dynamic logic gate. Communication and computer hardware devices based on such dynamic logic scheme may be more flexible and robust than the existing statically wired hardware.

Acknowledgments

The authors would like to thank the reviewers for their helpful advices. This paper is supported in part by the AoE grant E-02/08 from the University Grants Committee of the Hong Kong Special Administration Region, China, the Hong Kong Scholars Program (Grant no. HJ2012005), the China Postdoctoral Science Foundation Funded Project (Grant no. 2012T50209), the National Natural Science Foundation of China (Grant nos. 61070209, 61100204, 61272402), the Beijing Higher Education Young Elite Teacher Project, the Shenzhen Municipal Key Laboratory of Key Technology and Application (Grant no. C.02.12.00301) and the Fundamental Research Funding of Shenzhen, China (Grant no. C.02.13.00701).

References

[1] T. Munakata, S. Sinha, and W. L. Ditto, "Chaos computing: implementation of fundamental logical gates by chaotic elements," *IEEE Transactions on Circuits and Systems I*, vol. 49, no. 11, pp. 1629–1633, 2002.

[2] S. Sinha and W. L. Ditto, "Dynamics based computation," *Physical Review Letters*, vol. 81, no. 10, pp. 2156–2159, 1998.

[3] K. Murali and S. Sinha, "Using synchronization to obtain dynamic logic gates," *Physical Review E*, vol. 75, no. 2, Article ID 025201, 2007.

[4] K. Murali, S. Sinha, W. L. Ditto, and A. R. Bulsara, "Reliable logic circuit elements that exploit nonlinearity in the presence of a noise floor," *Physical Review Letters*, vol. 102, no. 10, Article ID 104101, 2009.

[5] H. Peng, F. Liu, L. Li, Y. Yang, and X. Wang, "Dynamic logic architecture based on piecewise-linear systems," *Physics Letters A*, vol. 374, no. 13-14, pp. 1450–1456, 2010.

[6] D. Graham-Rowe, "How To Be Human: call centers might be able to teach "chat bots" a thing or two about passing the Turing Test," in MIT Technology Review Website, 2006.

[7] D. Hansel, G. Mato, C. Meunier, and L. Neltner, "On numerical simulations of integrateand-fire neural networks," *Neural Computation*, vol. 10, no. 2, pp. 467–483, 1998.

[8] H. Peng, Y. Yang, L. Li, and H. Luo, "Harnessing piecewise-linear systems to construct dynamic logic architecture," *Chaos*, vol. 18, no. 3, Article ID 033101, 2008.

[9] G. Taubes, "Computer design meets Darwin," *Science*, vol. 277, no. 5334, pp. 1931–1932, 1997.

[10] M. Li, "Approximating ideal filters by systems of fractional order," *Computational and Mathematical Methods in Medicine*, vol. 2012, Article ID 365054, 6 pages, 2012.

[11] M. Li, S. C. Lim, and S. Chen, "Exact solution of impulse response to a class of fractional oscillators and its stability," *Mathematical Problems in Engineering*, vol. 2011, Article ID 657839, 9 pages, 2011.