Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2011, Article ID 380345, 12 pages
doi:10.1155/2011/380345

Research Article

Constructing Dynamic Multiple-Input Multiple-Output Logic Gates

Haipeng Peng,[1] Gang Hu,[2] Lixiang Li,[1] Yixian Yang,[1] and Jinghua Xiao[3]

[1] *Information Security Center, Beijing University of Posts and Telecommunications, P.O.Box 145, Beijing 100876, China*
[2] *Department of Physics, Beijing Normal University, Beijing 100875, China*
[3] *School of Science, Beijing University of Posts and Telecommunications, Beijing 100876, China*

Correspondence should be addressed to Lixiang Li, li_lixiang2006@yahoo.com.cn

Received 22 May 2011; Revised 9 August 2011; Accepted 16 August 2011

Academic Editor: Kwok W. Wong

Investigation of computing devices with dynamic architecture which makes devices have reconfigurable ability is an interesting research direction for designing the next generation of computer chip. In this paper, we present a window threshold method to construct such dynamic logic architecture. Here, dynamic multiple-input multiple-output (MIMO) logic gates are proposed, analyzed, and implemented. By using a curve-intersections-based graphic method, we illustrate the relationships among the threshold, the control parameter, and the functions of logic gates. A noise analysis on all the parameters is also given. The chips based on the proposed schemes can be transformed into different arrangements of logic gates within a single clock cycle. With these schemes in hand, it is conceivable to build more flexible, robust, cost effective, yet general-purpose computing devices.

1. Introduction

In today's processor designs, transistors are locked down for specific functions. Can we overcome the limitation of fixed structures of static architecture in the next generation of computer? This is an important issue to the practical applications. A reconfigurable technique with dynamic architecture has made it possible to break through the fixed limitations of the current computer systems [1, 2]. In dynamic architecture, systems can flexibly change their hardware configurations during the course of computation according to the demands of various functions.

Using reconfigurable techniques, one can envision future processor architecture to morph into distinct functions, each is suitable for an application at hand [3]. The dynamic

architecture currently used in the field programmable gate array (FPGA) technique which constructs dynamic architecture by "rewiring" tiles or computer elements may be termed as dynamic rewiring architecture [4–8]. Recently, another technique based on theories of chaos computing which is different from FPGA was proposed to construct dynamic reconfigurable architecture by harnessing dynamical systems [8–10]. In chaos computing, the programmable gate can be modified by adjusting the system parameters of chaos dynamics [11]. By changing the system parameters, chaotic elements of computing can act as different logic elements and perform various computing tasks [12, 13]. Recently, piecewise-linear systems are also suggested for constructing such dynamic logic architecture [14, 15]. In 2008, a prototype VLSI chip (TSMC CMOS, 0.18 μ, 30 Mhz clock) has been designed and developed incorporating proof of concept on chaos computing which establishes the technical feasibility of the chaos-based dynamic logic architecture [16].

In our previous works [14, 15], 2-input 1-output logic gates were considered to construct such dynamical architecture. However, the dynamic MIMO logic gate has received little attention. In this paper, we propose schemes to construct such dynamic MIMO logic gates based on a window threshold mechanism, which can emulate different logic gates, perform different arithmetic tasks, and further have the ability to switch among different operational roles by changing the control instruction. By using a curve-intersections-based method, we analyze the different logic distribution on parameters. Noise plays an important role in designing logic architecture. Here, a detailed noise analysis on all the parameters of dynamic MIMO logic gate is also given. The proposed schemes in this paper are efficient in computation and available in engineering implementations.

2. Schemes for Dynamic MIMO Logic Gates

Our basic scheme is represented by the following M-input N-output logic cell:

$$\begin{aligned} y &= \sum_{i=0}^{M-1} C_i I_i - k, \\ I_{\text{out}j} &= 0, \quad \text{if } -\beta_j < y < \beta_j \\ I_{\text{out}j} &= 1, \quad \text{else}, \end{aligned} \tag{2.1}$$

where I_i is the input signals, C_i is the weights of I_i, $I_{\text{out}j}$ is the output signals, β_j is the window thresholds ($0 \le i \le M-1$, $0 \le j \le N-1$), and k is the control instruction which acts as a controller for dynamic MIMO logic gate.

Now we show how to obtain different logic gates by simply changing the values of parameter k. We firstly consider an example of 3-input 1-output logic gate whose parameters are selected as $C_0 = C_1 = C_2 = 1$, $\beta_0 = 1.75$, and $k = 1$. If we input (000) for $(I_2 I_1 I_0)$, then we have $y = -1, |y| = 1$. Since $1 < 1.75$, the output is 0. Inputting (001/010/100) for $(I_2 I_1 I_0)$ results in $|y| = 0$. Since $0 < 1.75$, the output is 0. Similarly if we input (011/101/110) or (111) for $(I_2 I_1 I_0)$, the output is identified to 0 and 1, respectively. Thus, the logic cell performs a 3-input AND gate. It is easy to justify that we can change from AND gate to NOR gate by simply changing parameter k from 1 to 2 (for more details, please see Figure 1). Seen from Figure 1, we can know that, if we choose different values of β_0, the logical performance will change accordingly.

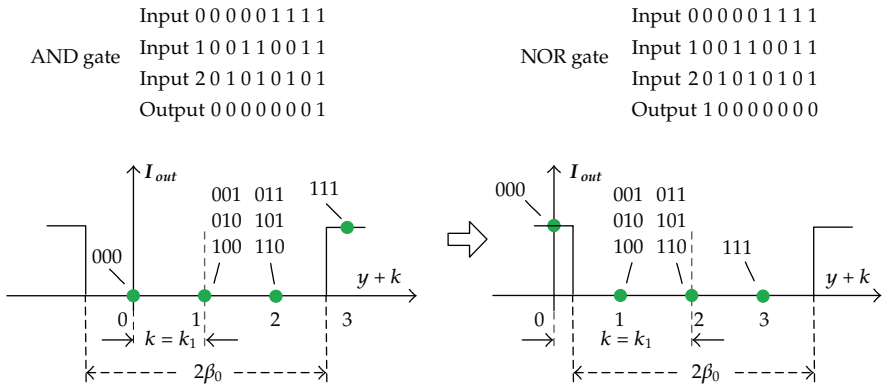

Figure 1: By changing the parameter k from k_1 to k_2, logic function of the cell can transform from logic AND to logic NOR.

In order to analyze different gate distribution of logic cell (2.1), we propose an analysis method, called CIA method (curve-intersection-based analysis method), from which we can obtain the relationships among parameters I_i, C_i, $I_{\text{out}j}$, β_j, and k, and then obtain different regions for different logical functions. There are four steps in the proposed analysis method. First, we determine the domain of k, calculate, and draw curves of $|y|$ for different inputs $(I_{M-1} \cdots I_0)$, where the x-coordinate is k and the y-coordinate is $|y|$. Second, different values of β_j are used to divide the region of $|y|$ into two parts: for the upper part, $I_{\text{out}j} = 1$, where $|y| \geq \beta_j$, and, for the lower part, $I_{\text{out}j} = 0$, where $|y| < \beta_j$. In the second step, we can determine different β_j regions according to the intersections of curves $|y|$ and boundaries of k. Third, the positions of intersections for curves of β_j and $|y|$ are used to discriminate different regions of parameter k which represent different logic functions. Finally, with different values of parameter k, our proposed logic cell can transform among different logic functions.

Now we use CIA method to analyze the 3-input 1-output logic gate. The relationships among parameters $|y|, k$, and β_0 are shown in Figure 2 with the domain $-1 \leq k \leq 4$, where parameter values $C_0 = C_1 = C_2 = 1$. Figure 2(a) shows the curves of $|y|$ with different combinations of $(I_2 I_1 I_0)$. Figure 2(b) shows different gate distribution of β_0. In Figure 2(c), the black points mark intersections of the line $\beta_0 = 0.75$ with various curves of $|y|$ against k for different combinations of $(I_2 I_1 I_0)$. Thus, different regions of logic functions are produced with different values of k. Here, the intersection points are termed as the critical points. If parameter k is selected at or near these critical points, a small variation of k may make the logic function transform from one gate to another. This brings disadvantages for designing robust logic gates against noise. Figure 2(d) shows that logic gates can change its function from one to the other. Seen from Figure 2, we know that the logic cell can change flexibly among different kinds of logic functions by changing parameters k and β_0.

In our scheme, C_i is important to distinguish different inputs [17]. For 3-input 1-output logic gate, the curves of $|y|$ with different combinations of $(C_2 C_1 C_0)$ are shown in Figure 3, from which we can see that different combinations of C_i can lead to different logic outputs. In order to further illustrate the proposed method, a 3-input 2-output logic gate is considered in Figure 4 which shows different k regions for different logic functions. Seen from Figure 2 to Figure 4, we know that the proposed curve-intersection-based analysis method is a general and legible tool to analyze different logic gates.

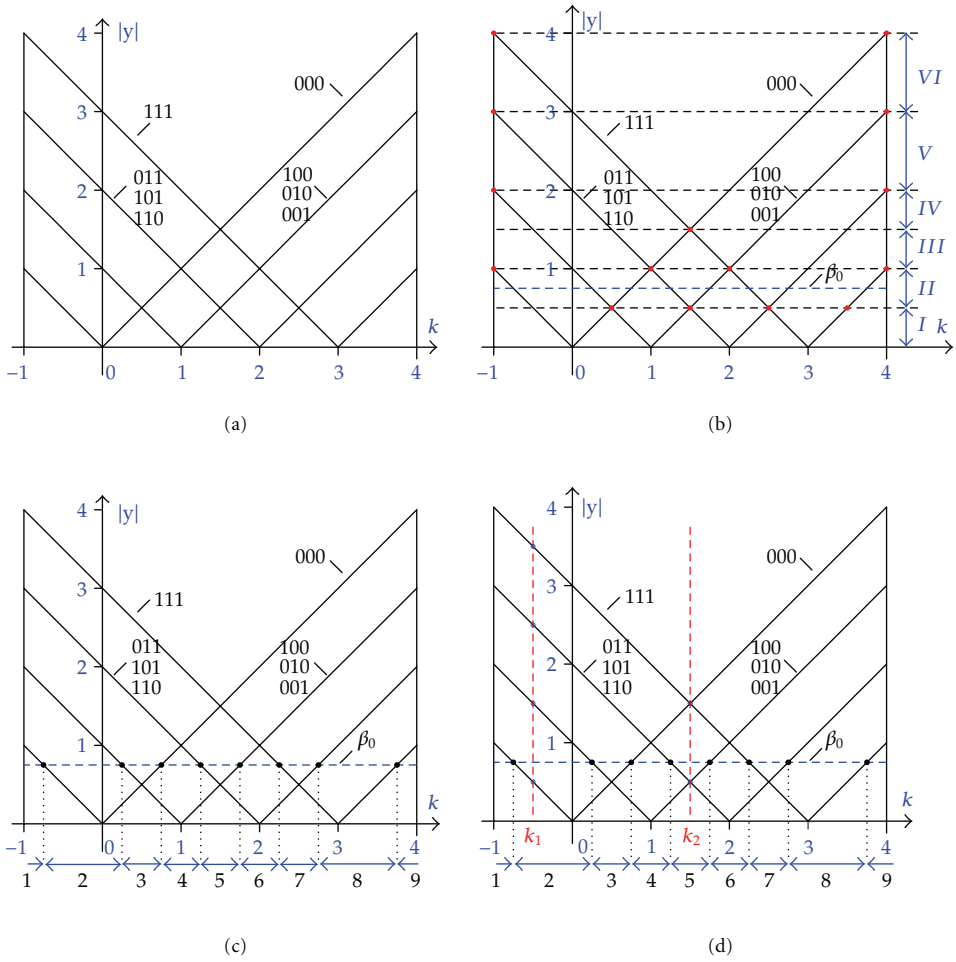

Figure 2: Different k regions for different logics: (a) curves of $|y|$ for different combinations (000), (001/010/100), (110/101/011), and (111), respectively. (b) Different β_0 regions according to the intersections of the curves. In each region, β_0 has similar character. (c) $\beta = 0.75$. According to the intersections (black points), we achieve 9 regions of k. (d) For different k, the logic cell can perform different logic gates.

For logic gate with 2-input 1-output, a special case of multiple-input multiple-output logic gate, there are 16 possible boolean algebraic functions which are shown in Tables 1 and 2. The details of different logic functions when β and k belong to different regions are shown in Table 3, where $C_1 = 1, C_0 = 0.5$, and $y = 0.5I_0 + I_1 - k$, if $-\beta < y < \beta$, $I_{\text{out}} = 1$, else $I_{\text{out}} = 0$. From Table 3, we can see that the gate can change within logic functions NOR, XOR, AND, 0, X_5, X_6, X_7, and X_8 by changing values of parameter k. If we use opposite output of the gate, we have contrary gate which means that the gate can change within OR, XNOR, NAND, 1, X_2, X_1, X_8 and X_7. Moreover, if we exchange inputs I_1 and I_0, we obtain its symmetric gate, that is, the gate can change within NOR, XOR, AND, 0, X_6, X_5, X_3, and X_4. The above analysis demonstrates that we can use the same cell to produce all the basic logics by adjusting parameter k (for more detailed definitions about contrary-gate and symmetric-gate, please see [15]).

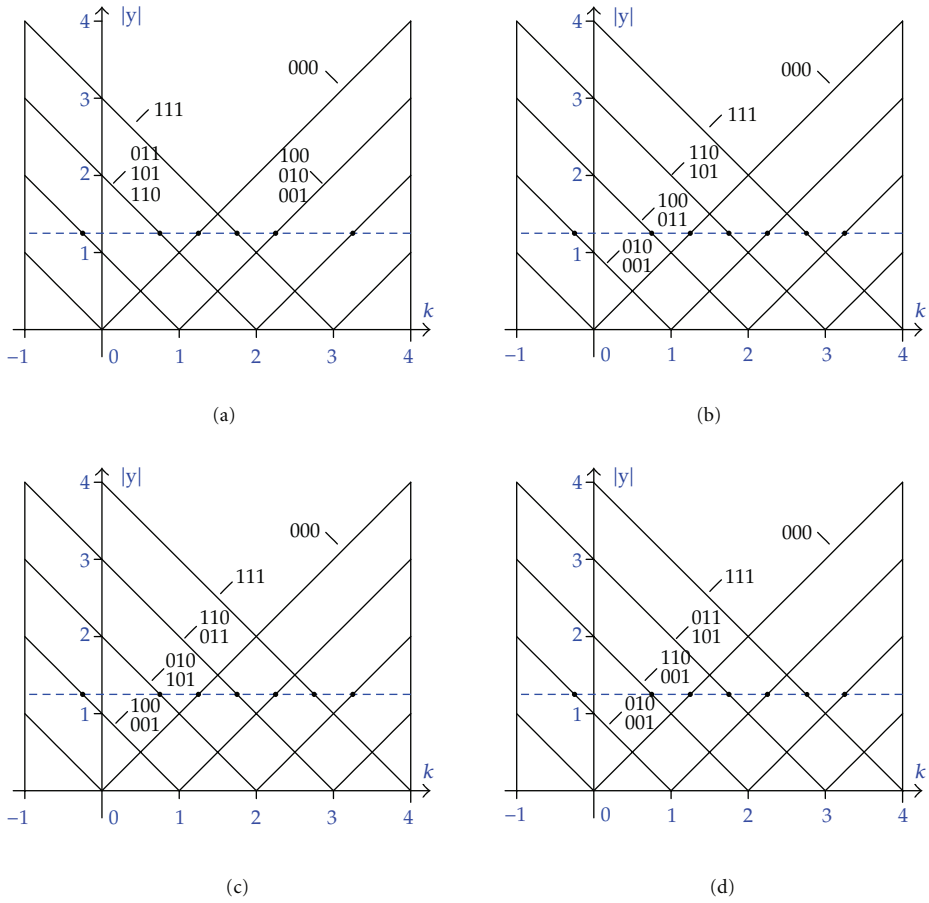

Figure 3: Curves of $|y|$ for different combination of $(C_2C_1C_0)$ and $(I_2I_1I_0)$ in (2.1). (a) $y = I_2+I_1+I_0-k$, $(C_2 = C_1 = C_0 = 1)$. The inputs (001/010/100) can be shown by a curve, and the inputs (101/011/110),(000), and (111) can be shown by other different curves, respectively. (b) $y = 2I_2 + I_1 + I_0 - k$, $(C_2 = 2, C_1 = C_0 = 1)$. The inputs (001/010) can be shown by one curve, and the inputs (100/011),(110/101), (000), and (111) can be shown by other different curves, respectively. (c) $y = I_2 + 2I_1 + I_0 - k$, $(C_2 = C_0 = 1, C_1 = 2)$. The inputs (001/100) can be shown by one curve, and for the inputs (010/101),(110/011), (000), and (111) can be shown by other different curves, respectively. (d) $y = I_2 + I_1 + 2I_0 - k$, $(C_2 = C_1 = 1, C_0 = 2)$. The inputs (100/010) can be shown by one curve, and the inputs (001/110),(101/011),(000), and (111) can be shown by other different curves, respectively. Different logic functions can be obtained from Figures 3(a), 3(b), 3(c) and 3(d) with the same β_0 and k.

Table 1: Truth table of basic logical NOR, NAND, XOR, OR, AND, 0, and 1.

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

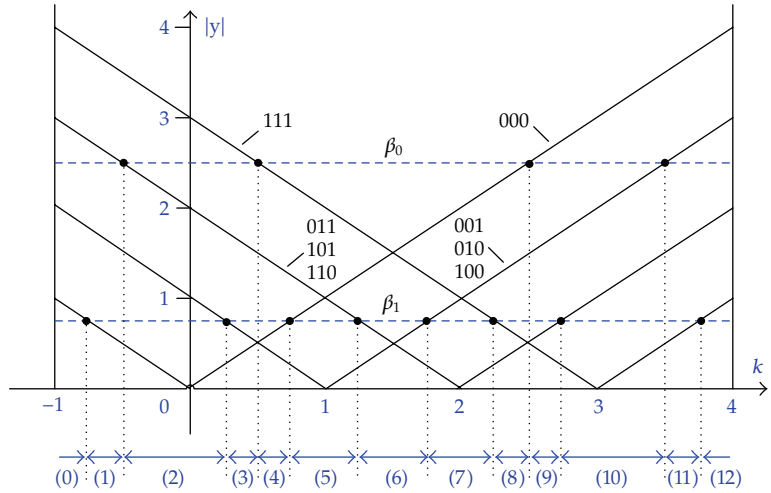

Figure 4: Different k regions for different three-input and two-output logic gates where $y = I_2 + I_1 + I_0 - k$, with the rules $-\beta_0 < y < \beta_0$, $I_{\text{out0}} = 0$; *else* $I_{\text{out0}} = 1$ and $-\beta_1 < y < \beta_1$, $I_{\text{out1}} = 0$; *else* $I_{\text{out1}} = 1$. We can obtain different kinds of multiple-input multiple-output logic gates by changing k.

Table 2: Truth table of logical XNOR, X_1, X_2, X_3, X_4, X_5, X_6, X_7, and X_8.

I_0	I_1	XNOR	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8
I_0	I_1	$\overline{I_0 \oplus I_1}$	$I_0 + \overline{I_1}$	$\overline{I_0} + I_1$	$\overline{I_0}$	I_0	$I_0\overline{I_1}$	$\overline{I_0}I_1$	$\overline{I_1}$	I_1
0	0	1	1	1	1	0	0	0	1	0
0	1	0	0	1	1	0	0	1	0	1
1	0	0	1	0	0	1	1	0	1	0
1	1	1	1	1	0	1	0	0	0	1

Since noise cannot be avoided in designing robust logic cells for implementing gate functions successfully, we must discuss the optimal selections of parameter k in presence of noise. Now we begin to discuss a thorough noise analysis on all the parameters k, β_j, and the inputs. When k is influenced by noise, we have $y = \sum_{i=0}^{M-1} C_i I_i + D\eta(t) - k$, where $\eta(t)$ is an additive zero mean noise and D is the noise strength. The additive noise will cause some confusions. Based on CIA method, we know that some confusion domains come into being around curves of $|y|$ in presence of noise. Figure 5 shows confusion domains of 3-input 1-output logic gate where $C_0 = C_1 = C_2 = 1$ and the band of the domain is $2D$. When the combination of (k, β_0) is selected in the grey belts (e.g., the red point), the discrimination of logic gate is confused. In order to avoid the influence of noise, the combination of (k, β_0) should be selected near or at the centers of white belts (e.g., the green point). Since $w = \sqrt{2}/2$ (please see Figure 5), we can see that when D increases to $D = \sqrt{2}/4$, the white belts will disappear, that is to say, to perform a robust gate, D should be less than $\sqrt{2}/4$ in this case. Suppose that the mth input is influenced by noise, we have $y = \sum_{i=0}^{m-1} C_i I_i + C_m(I_m + D\eta(t)) + \sum_{i=m+1}^{M-1} C_i I_i - k = \sum_{i=0}^{M-1} C_i I_i + C_m D\eta(t) - k$, then the influence of noise on I_m is similar to that on k. In this condition, we can know that the band of confusion domain is $2C_m D$ and D should be less than $\sqrt{2}/(4C_m)$. Figure 6 shows the confusion domains when β_0 is influenced, from which we know that k should be selected so that intersections of curves k and $|y|$ should not fall into the confusion domains.

Table 3: Logical gates for different values of parameters β, k.

Region	β	k	Gate
(I)	$(0, 1/4]$	$(-1/2, -\beta)$	0
		$(-\beta, \beta)$	NOR
		$(\beta, 1/2-\beta)$	0
		$(1/2-\beta, 1/2+\beta)$	X_5
		$(1/2+\beta, 1-\beta)$	0
		$(1-\beta, 1+\beta)$	X_6
		$(1+\beta, 3/2-\beta)$	0
		$(3/2-\beta, 3/2+\beta)$	AND
		$(3/2+\beta, 2)$	0
(II)	$(1/4, 1/2]$	$(-1/2, -\beta)$	0
		$(-\beta, 1/2-\beta)$	NOR
		$(1/2-\beta, \beta)$	X_7
		$(\beta, 1-\beta)$	X_5
		$(1-\beta, 1/2+\beta)$	XOR
		$(1/2+\beta, 3/2-\beta)$	X_6
		$(3/2-\beta, 1+\beta)$	X_8
		$(1+\beta, 3/2+\beta)$	AND
		$(3/2+\beta, 2)$	0
(III)	$(1/2, 3/4]$	$(-1/2, 1/2-\beta)$	NOR
		$(1/2-\beta, 1-\beta)$	X_7
		$(1-\beta, \beta)$	NAND
		$(\beta, 3/2-\beta)$	XOR
		$(3/2-\beta, 1/2+\beta)$	OR
		$(1/2+\beta, 1+\beta)$	X_8
		$(1+\beta, 2)$	AND
(IV)	$(3/4, 1]$	$(-1/2, 1/2-\beta)$	NOR
		$(1/2-\beta, 1-\beta)$	X_7
		$(1-\beta, 3/2-\beta)$	NAND
		$(3/2-\beta, \beta)$	1
		$(\beta, 1/2+\beta)$	OR
		$(1/2+\beta, 1+\beta)$	X_8
		$(1+\beta, 2)$	AND
(V)	$(1, 3/2]$	$(-1/2, 1-\beta)$	X_7
		$(1-\beta, 3/2-\beta)$	NAND
		$(3/2-\beta, \beta)$	1
		$(\beta, 1/2+\beta)$	OR
		$(1/2+\beta, 2)$	X_8
(VI)	$(3/2, 2]$	$(-1/2, 3/2-\beta)$	NAND
		$(3/2-\beta, \beta)$	1
		$(\beta, 2)$	OR

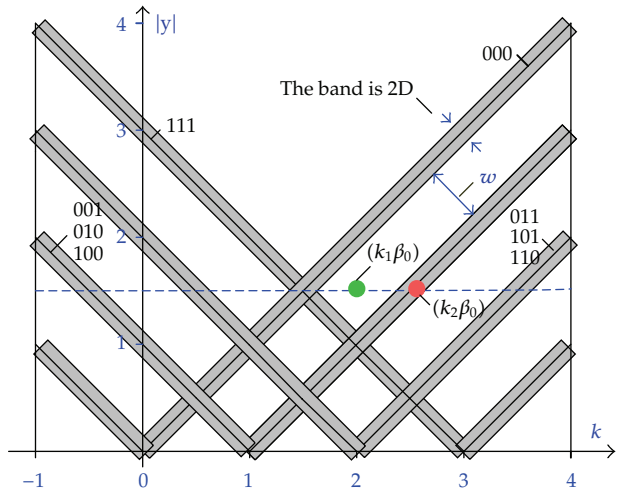

Figure 5: Confusing processing domains (grey belts) caused by additive noise. When combination (k, β_0) is selected in the grey belts (e.g., the red point), the discrimination of logic gate is confused. In order to avoid the influence of such noise, the combination (k, β_0) should be selected at the centers of the white belts (e.g., the green point).

Physical implementation of the proposed scheme is an important work for successful engineering applications. Figure 7 shows the simulation circuit of a 3-input 1-output logic gate, where UA741 and OP37CZ are operational amplifiers and ZDX1F and ZPD5.1 are diodes. In the circuit, the operational amplifier of OP37CZ is used to calculate the value of y, and the other operational amplifiers and diodes are used to determine the output. Figure 8 shows simulation results of inputs, k, output, and y, respectively. In practical applications, chips based on our schemes can be designed based on the existing semiconductor technology with no retooling requirement.

Note that there are two types of multiple-input multiple-output (MIMO) logic gates. The first type with one-control instruction which is given in (2.1). The second type with multicontrol instructions is described as follows:

$$\begin{aligned} y_j &= \sum_{i=0}^{M-1} C_{ij} I_i - k_j, \\ I_{\text{out}j} &= 0, \quad \text{if } -\beta_j < y_j < \beta_j \\ I_{\text{out}j} &= 1, \quad \text{else.} \end{aligned} \tag{2.2}$$

For the M-input N-output logic gate with multicontrol instructions, we can also use the proposed CIA method to analyze the gate distribution. The structure of dynamic MIMO logic gate with multicontrol instructions is more complex than that with only one control instruction. However, the logic functions of dynamic MIMO logic gate with multicontrol instructions are richer than that with only one control instruction.

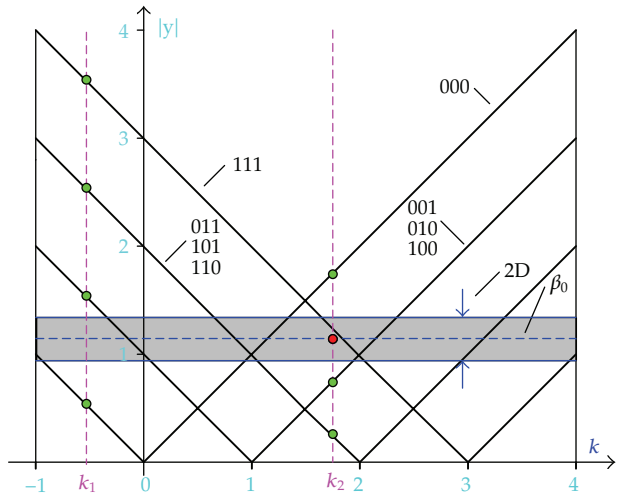

Figure 6: Confusing processing domains (grey belt) caused by additive noise. When the intersections of curves k and $|y|$ fall into the grey belts (e.g., the red point), the discrimination of logic gate is confused. In order to avoid the influence of such noise, the intersections of k and $|y|$ should be far away from the grey belt.

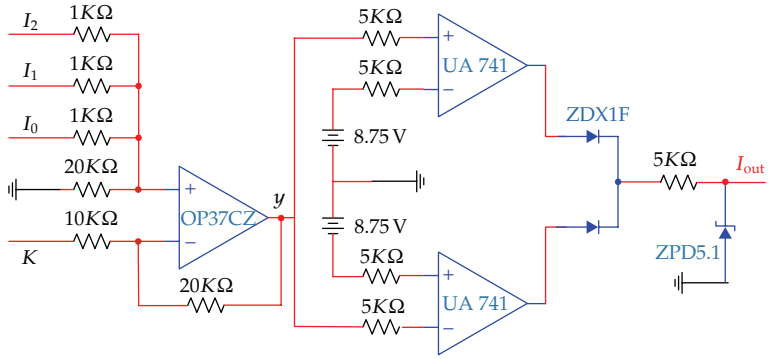

Figure 7: Circuit diagram of 3-input 1-output logic gate where $y = I_0 + I_1 + I_2 - 2k$ and $\beta_0 = 8.75$.

3. Discussion and Conclusion

Arrays of such morphing logic gates can be conceivably programmed on the run (e.g., by an external program) with satisfactory optimization for tasks at hand. For instance, they may serve flexibly as arithmetic processing units or memory units and can be swapped from one to another as demands.

The computing scheme proposed here is a kind of technique for dynamic logic architecture, and it has an important and practical advantage of flexibility over all the previous computing paradigms of static architecture. Moreover, the architecture of dynamic logic is essentially different from that of FPGA [7, 12]. FPGA contains programmable interconnects that

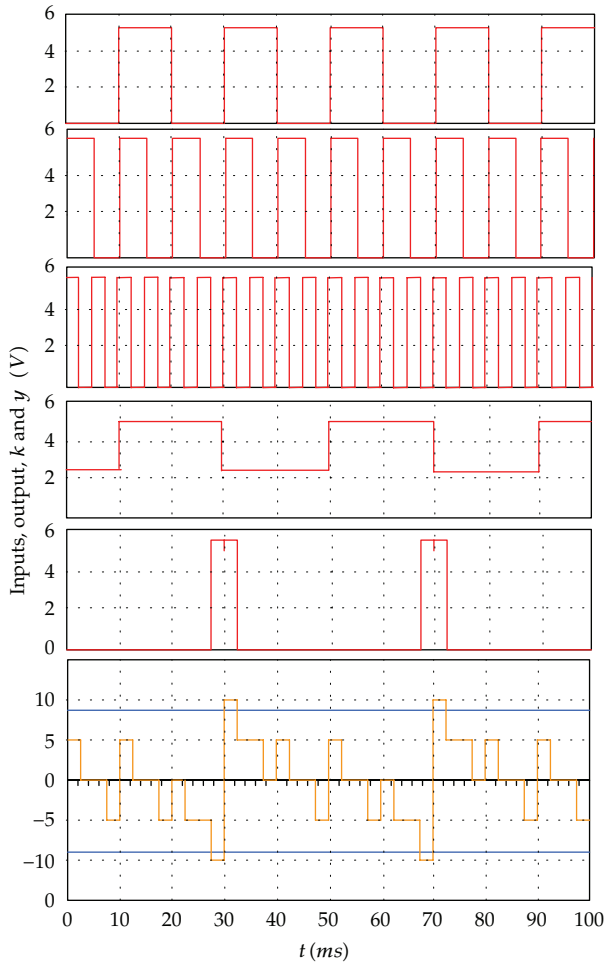

Figure 8: Simulation results of time sequences from top to bottom: panel 1, panel 2, and panel 3 show a stream of input signals I_0, I_1, and I_2, determining input set $(I_0I_1I_2)$. Panel 4 shows the control signal of k for different logics. Panel 5 shows the output signal I_{out}. Both these logical functions are consistent with the corresponding k values indicated in Panel 4. Panel 6 shows the signals of y, β_0, and $-\beta_0$, respectively.

can be rewired to perform different functions [17–19]. The chips based on dynamic logic architecture can be transformed into different arrangements of logic gates in single clock cycles. FPGA is relatively slow to reconfigure, typically taking milliseconds for each rewiring, or about one million times slower than chips of dynamic logic architecture. ChaoLogix, a semiconductor company, has gotten to the stage where it can create any kind of gate from a small circuit of about 30 transistors, and this circuit is then repeated across the chip. The use of a single circuit has huge advantages over FPGA [11]. The way FPGA designed takes up more silicon real estate and consumes more resources than chips of dynamic logic architecture [11]. In schemes of dynamic logic architecture, there is no special difference between a memory element and a processing element. Hence, the duties of damaged cells may be efficiently distributed among other elements [17–20]. The reconfigurable computing systems based on dynamic logic architecture may be more robust than those based on FPGA.

In this paper, we use a threshold mechanism to obtain dynamic MIMO logic cell. Such simple computing units may then support a dynamic computer architecture and serve as ingredients of general-purpose device more flexibly than statically wired hardware as well as dynamic hardware based on dynamical systems. Possible applications of such reconfigurable hardware include digital signal processing, software-defined radio, aerospace and defense systems, ASIC prototyping, cryptography, computer vision, speech recognition, computer hardware emulation, and a growing range of other related areas [21, 22]. Further advantages of reconfigurable hardware include the ability to reprogram in the field, to fix bugs, lower nonrecurring engineering costs, and implement coarse-grained architecture approaches [4].

Acknowledgments

The authors would like to thank the Editor and all the anonymous reviewers for their helpful advices. This paper is supported by the National Natural Science Foundation of China (Grant nos. 61070209 and 61100204), the Specialized Research Fund for the Doctoral Program of Higher Education (Grant no. 200800131028), the Chinese Universities Scientific Fund (Grant no. BUPT2011RC0211), the Fok Ying-Tong Education Foundation for Young Teachers in the Higher Education Institutions of China (Grant no. 121062), the Program for New Century Excellent Talents in University of the Ministry of Education of China (Grant no. NCET-10-0239).

References

[1] T. C. Bartee, *Computer Architecture and Logic Design*, McGraw-Hill, New York, NY, USA, 1991.
[2] M. M. Mano, *Computer System Architecture*, Prentice-Hall, Englewood Cliffs, NJ, USA, 3rd edition, 1993.
[3] D. Tabak, "Dynamic architecture and LSI modular computer systems," *IEEE Micro*, vol. 4, no. 2, pp. 48–66, 1984.
[4] G. Taubes, "Computer design meets Darwin," *Science*, vol. 277, no. 5334, pp. 1931–1932, 1997.
[5] J. R. Heath, P. J. Kuekes, G. S. Snider, and R. S. Williams, "A defect-tolerant computer architecture: opportunities for nanotechnology," *Science*, vol. 280, no. 5370, pp. 1716–1721, 1998.
[6] C. P. Collier, E. W. Wong, M. Belohradský et al., "Electronically configurable molecular-based logic gates," *Science*, vol. 285, no. 5426, pp. 391–394, 1999.
[7] S. Lange and M. Middendorf, "Multi-level reconfigurable architectures in the switch model," *Journal of Systems Architecture*, vol. 56, no. 2-3, pp. 103–115, 2010.
[8] S. Sinha and W. L. Ditto, "Dynamics based computation," *Physical Review Letters*, vol. 81, no. 10, pp. 2156–2159, 1998.
[9] K. Chlouverakis and M. J. Adams, "Optoelectronic realization of NOR logic gate using chaotic two-section lasers," *Electronics Letters*, vol. 41, no. 6, pp. 359–360, 2005.
[10] D. Kuo, "Chaos and its computing paradigm," *IEEE Potentials*, vol. 24, no. 2, pp. 13–15, 2005.
[11] D. Graham-Rowe, "Logic from chaos: new chips use chaos to produce potentially faster, more robust computing," Technology Review, Massachusetts Institute of Technology, Cambridge, Mass, USA, 2006, http://www.technologyreview.com/business/16989/.
[12] 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: Fundamental Theory and Applications*, vol. 49, no. 11, pp. 1629–1633, 2002.
[13] 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, p. 104101, 2009.
[14] H. Peng, Y. Yang, L. Li, and H. Luo, "Harnessing piecewise-linear systems to construct dynamic logic architecture," *Chaos*, vol. 18, no. 3, article 033107, 2008.
[15] 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.

[16] W. L. Ditto, K. Murali, and S. Sinha, "Exploiting the controlled responses of chaotic elements to design configurable hardware," *Philosophical Transactions of the Royal Society A*, vol. 364, no. 1846, pp. 2483–2494, 2006.

[17] D. Normile, "Artificial life gets real as scientists meet in Japan," *Science*, vol. 272, no. 5270, pp. 1872–1873, 1996.

[18] M. R. Jahed-Motlagh, B. Kia, W. L. Ditto, and S. Sinha, "Fault tolerance and detection in chaotic computers," *International Journal of Bifurcation and Chaos in Applied Sciences and Engineering*, vol. 17, no. 6, pp. 1955–1968, 2007.

[19] M. Edwards and P. Green, "Run-time support for dynamically reconfigurable computing systems," *Journal of Systems Architecture*, vol. 49, no. 4–6, pp. 267–281, 2003.

[20] P. M. Frank, "Analytical and qualitative model-based fault diagnosis—a survey and some new results," *European Journal of Control*, vol. 2, no. 1, pp. 6–28, 1996.

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

[22] K. Murali, A. Miliotis, W. L. Ditto, and S. Sinha, "Logic from nonlinear dynamical evolution," *Physics Letters A*, vol. 373, no. 15, pp. 1346–1351, 2009.