

Efficient Cellular Automata Based Versatile Multiplier for $GF(2^m)$

HUA LI AND CHANG N. ZHANG
*Department of Computer Science, TRILabs
University of Regina
Canada S4S 0A2*

In this paper, a low-complexity Programmable Cellular Automata (PCA) based versatile modular multiplier in $GF(2^m)$ is presented. The proposed versatile multiplier increases flexibility in using the same multiplier in different security environments, and it reduces the user's cost. Moreover, the multiplier can be easily extended to high order of m for more security, and low-cost serial implementation is feasible in restricted computing environments, such as smart cards and wireless devices.

Keywords: finite field multiplication, canonical basis, programmable cellular automata, VLSI, encryption

1. INTRODUCTION

The finite fields $GF(2^m)$ of characteristic 2 are of great interest to researchers studying cryptosystems, as the relevant arithmetic can be implemented in both hardware [1] and software [2, 3]. Finite field multipliers can be classified into fast bit parallel architectures [4] which generally required an area of $O(m^2)$ and low-complexity bit serial multiplier [5] which have an area requirement of $O(m)$. The bit parallel multipliers can get the result in one clock cycle, but bit serial multipliers compute a product in m clock cycles [6]. There have been three main kinds of basis representation of the field elements in $GF(2^m)$: standard (canonical, polynomial) basis, dual basis and normal basis. Different modular multipliers corresponding to these basis representations have their own benefits and trade-offs [7]. In order to reduce the complexity of the hardware and improve the speed of computation, the construction of finite field based on irreducible all one polynomial (AOP) [4, 8-11], equally spaced polynomial (ESP) [8, 9, 11], trinomial [12], and redundant field representations [13] has been proposed.

Most of the proposed finite field multipliers operate over a fixed field, i.e., a new multiplier is needed if there is a change on the irreducible polynomial that defines the field elements. There are some Cellular Automata based multipliers [14-16], but none of them operates on the field of $GF(2^m)$. In this paper, we present a new VLSI array for a versatile modular multiplier based on PCA (Programmable Cellular Automata) and the canonical (standard, polynomial) basis representation where the field set of parameters can be changed according to the application environment. The structure of the proposed multiplier can be easily extended to high order of m for more security, and it

Received August 27, 2001; accepted April 15, 2002.

Communicated by Jang-Ping Sheu, Myongsoon Park and Makoto Takizawa.

can be speeded up by applying more levels of cells. The basic architecture of the multiplier is also suitable for both bit parallel and bit serial schemes. Moreover, the proposed VLSI array has the properties of modularity, simplicity, and regular interconnection, and VLSI implementation is easy.

The remainder of the paper is organized as follows. In section 2, we introduce the algorithm for $GF(2^m)$ multiplication and the theory of Programmable Cellular Automata. Section 3 contains the derivation of the low-complexity PCA based versatile multiplier in $GF(2^m)$. Section 4 shows that the PCA based multiplier is improved for faster multiplication. Section 5 concludes with the improved result and a description of possible areas of application.

2. MULTIPLICATION ON $GF(2^m)$ AND PRELIMINARY CA THEORY

In this section, we derive the algorithm for polynomial multiplication in $GF(2^m)$ and introduce the basic theory for PCA applied in our proposed multiplier.

Let

$$A(x) = a_{m-1}x^{m-1} + \cdots + a_1x + a_0$$

$$B(x) = b_{m-1}x^{m-1} + \cdots + b_1x + b_0$$

be two elements in $GF(2^m)$, let

$$C(x) = c_{m-1}x^{m-1} + c_{m-2}x^{m-2} + \cdots + c_1x + c_0$$

be the product of $A(x) B(x) \bmod P(x)$ in $GF(2^m)$, and let

$$P(x) = x^m + p_{m-1}x^{m-1} + \cdots + p_1x + p_0$$

be the primitive polynomial.

$C(x)$ can be written as follows:

$$\begin{aligned} C(x) &= A(x) B(x) \bmod P(x) \\ &= [(\cdots((0 + a_{m-1}B(x))x + a_{m-2}B(x))x + \cdots + a_1 B(x))x + a_0 B(x)] \bmod P(x). \end{aligned} \quad (1)$$

Equation (1) can be implemented using the following iterative algorithm.

Algorithm 1 (Polynomial multiplication in $GF(2^m)$)

Step 1: $C^{-1}(x) = 0$

Step 2: for $i = 0$ to $m - 1$ do

$$C^i(x) = [C^{i-1}(x)x + a_{m-1-i}B(x)] \bmod P(x)$$

Using the fact that

$$x^m \bmod P(x) = p_{m-1}x^{m-1} + \dots + p_1x + p_0.$$

$C^i(x)$ in step 2 of the algorithm 1 can be rewritten as

$$\begin{aligned} C^i(x) &= [c_{m-1}^{i-1}(p_{m-1}x^{m-1} + \dots + p_1x + p_0) + \dots + c_0^{i-1} + a_{m-1-i}B(x)] \bmod P(x) \\ &= (c_{m-1}^{i-1}p_{m-1} + c_{m-2}^{i-1} + a_{m-1-i}b_{m-1})x^{m-1} \\ &\quad + (c_{m-1}^{i-1}p_{m-2} + c_{m-3}^{i-1} + a_{m-1-i}b_{m-2})x^{m-2} \\ &\quad + \dots \\ &\quad + (c_{m-1}^{i-1}p_1 + c_0^{i-1} + a_{m-1-i}b_1)x \\ &\quad + (c_{m-1}^{i-1}p_0 + a_{m-1-i}b_0) \\ &= \sum_{j=0}^{m-1} c_j^i x^j, \end{aligned} \tag{2}$$

where

$$c_j^i = c_{m-1}^{i-1}p_j + c_{j-1}^{i-1} + a_{m-1-i}b_j \text{ and } c_{-1}^{i-1} = 0. \tag{3}$$

Algorithm 1 is a most significant bit (MSB) first algorithm where the multiplication loop begins with the MSB of multiplier A.

In the following, we introduce the preliminary theory of CA, which will be used in the next section. Cellular Automata (CA) is an array of cells, where each cell is in any of the permissible states. For example, in a 2-state CA, each cell's state can be zero or one. In a k -neighborhood CA, at each clock cycle, the evolution of a cell value depends on its rule and the present states of k of its neighbors. The rule of a cell is implemented on the cell's combination logic circuit (CL). Fig. 1 shows a 2-state 4-cell 3-neighborhood CA. The relevant theory, rules and mathematical characteristics of CA can be found in [17-20].

The programmable CA (PCA) was first introduced in [21], where the CL of each cell is not fixed but controlled by a number of control signals such that different functions (rules) can be realized on the same structure. Fig. 2 is a standard 3-neighborhood PCA cell with a non-complemented additive rule.

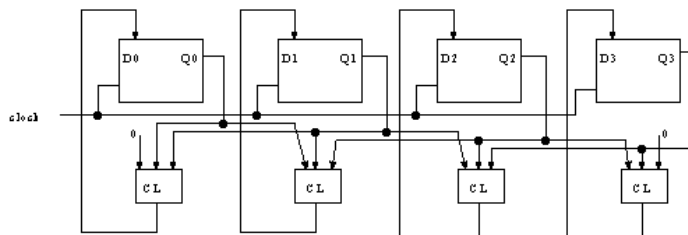


Fig. 1. A 2-state 4-cell 3-neighborhood CA.

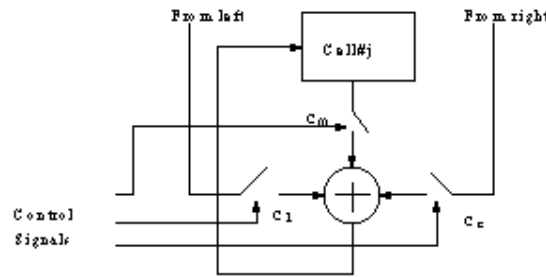


Fig. 2. A 3-neighbor PCA cell with a non-complemented additive rule.

Using a cell structure like those shown in Fig. 2 and Fig. 3, all possible additive rules can be achieved. The combinations of the control signals of C_l , C_m , and C_r , and the corresponding rules are listed in Table 1, where X_s represents the value of a cell. X_r is the value of its right neighbor, and X_l is the value of its left neighbor. From Table 1, it can be seen that by allowing control signals in CL, one can introduce immense flexibility into this programmable structure.

Table 1. PCA rules.

C_l	C_m	C_r	Noncomplemented Rules	Complemented Rules
0	0	1	X_r	X_r
0	1	0	X_s	X_s
1	1	1	$X_s \oplus X_r$	$X_s \oplus X_r$
1	0	0	X_l	X_l
1	0	1	$X_l \oplus X_r$	$X_l \oplus X_r$
1	1	0	$X_l \oplus X_s$	$X_l \oplus X_s$
1	1	1	$X_l \oplus X_s \oplus X_r$	$X_l \oplus X_s \oplus X_r$

The objective of this work is to develop a high speed versatile modular multiplier which has good cascability and security performance. This is achieved by using an extended PCA structure. The major difference between the standard PCA and extended PCA is that the three neighbor cells of a standard PCA structure are the cell itself and its two nearest neighbor cells, while in the extended PCA structure, the neighbor cells can be either the nearest neighbor cells or the left/right most cell. Fig. 3 shows the logic structure of an extended PCA cell, in which the three neighbor cells are the left most neighbor cell, the nearest left neighbor cell, and right most neighbor cell. The PCA based structure can also be applied in symmetric-key cryptosystems. A high-speed CA based cipher for multimedia applications was presented in [22].

3. PCA BASED VERSATILE MULTIPLIER IN $GF(2^m)$

Based on Algorithm 1, Eq. (2) and Eq. (3), a new theorem is presented which sug-

gests that a multiplication in $GF(2^m)$ can be implemented by an extended PCA.

Theorem 1. If the degree of $P(x)$ is m , then $C(x) = A(x)B(x) \bmod P(x)$ can be implemented by running 1-D (a single row of) m PCA cell through m clock cycles. At the i th clock cycle, the j th cell of the PCA has the rule $R = c_{m-1}^{i-1}p_j \oplus c_{j-1}^{i-1} \oplus a_{m-1-i}b_j$, where a , b , c , and p are coefficients of $A(x)$, $B(x)$, $C(x)$ and $P(x)$, respectively ($0 \leq i, j \leq m - 1$).

Based on Theorem 1, an extended m -cell 3-neighborhood PCA is presented for the multiplication operation in $GF(2^m)$. The proposed extended PCA stores the coefficients of $C(x)$, and its cell logic structure is shown in Fig. 3. For the j th cell, the state transition at the i th clock cycle depends on the values of the cell's three fixed neighbors: c_{m-1} from the right most cell, c_{j-1} from the left cell, and a_{m-1-i} from the boundary of the left most cell.

The three control signals of each cell are set as follows: C_l is always set to '1', C_m is configured by the coefficients of $B(x)$, and C_r is configured by the coefficients of $P(x)$. By controlling c_{m-1} using p_j and a_{m-1-i} using b_j , the extended PCA implements rule R in Theorem 1.

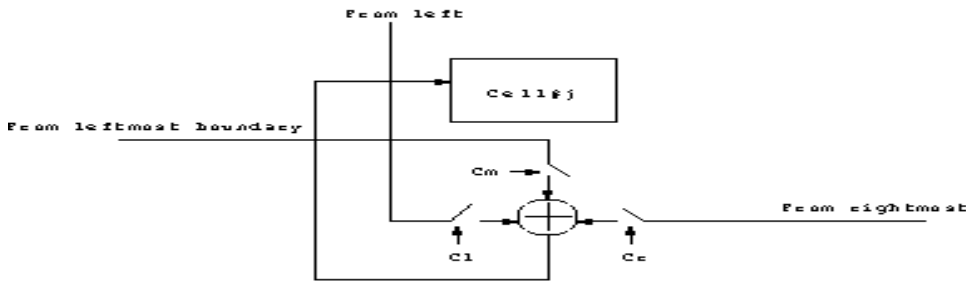


Fig. 3. Logic structure of an extended PCA cell.

The algorithm for the modular multiplication $C(x) = A(x)B(x) \bmod P(x)$ using the extended PCA is as follows.

Algorithm 2 (PCA based modular multiplication)

Input: Coefficients of $A(x)$, $B(x)$ and $P(x)$.

Output: $A(x)B(x) \bmod P(x)$.

Step 1: Reset PCA.

Step 2: Configure coefficients of $B(x)$ as C_m , and coefficients of $P(x)$ as C_r .

Step 3: Run PCA m clock cycles. The result is in PCA.

For example, in order to implement $A(x)B(x) \bmod P(x)$, where $B(x) = (x^5 + x + 1)$ and $P(x) = (x^6 + x^5 + x^4 + x^3 + 1)$, firstly the control signals C_m and C_r should be set as shown in Fig. 4. Then the coefficients of $A(x)$ should be loaded into the PCA. After 6 clock cycles, the result will be on PCA.

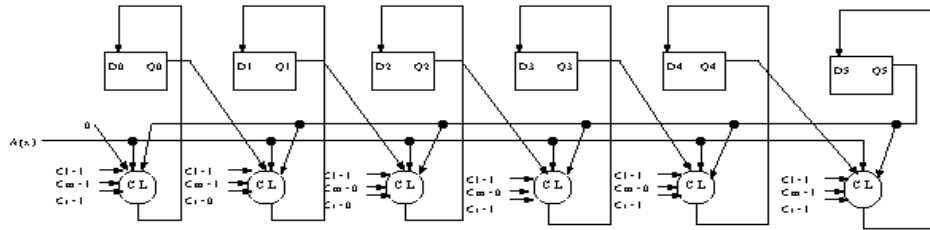


Fig. 4. The extended PCA structure for $A(x)(x^5 + x + 1) \bmod (x^6 + x^5 + x^4 + x^3 + 1)$.

From the above example, it can be seen that for modular multiplication, the extended PCA uses only m D flip-flops, and that the irreducible polynomial $P(x)$ is not fixed, which enables $P(x)$ to be changed after a period of time. Other advantages of this architecture are that the proposed PCA structure can be easily modified, and that two or more such VLSI chips can be cascaded together to form a multiplier of large operands.

4. AN IMPROVED PCA MULTIPLIER IN $GF(2^m)$

In this section, we propose an improved extended PCA in short (improved PCA) for faster modular multiplication, which uses fewer clock cycles.

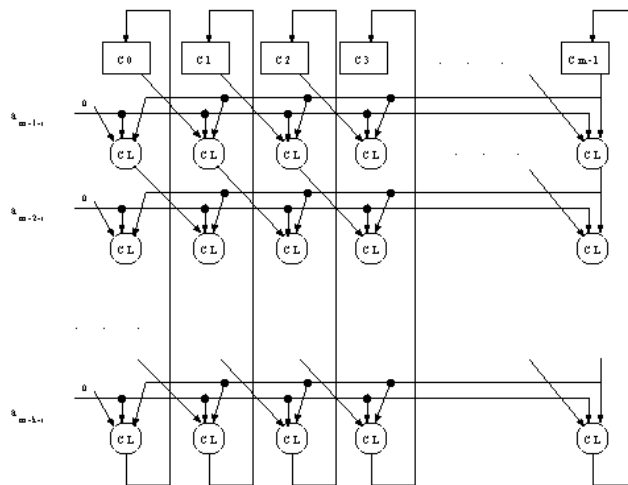


Fig. 5. An improved PCA modular multiplier.

The proposed multiplier can be improved by incorporating as many CLs as possible to increase computation speed. The architecture is shown in Fig. 5, where k logic circuits have been cascaded together, which include $k * m$ CLs. Here k is decided as fol-

lows. Suppose the system clock cycle time is t_s , that one D flip-flop delay is t_d , and that one CL delay is t_x . Then at each clock cycle, the number of CL operations that can be done is $k = \lfloor (t_s - t_d) / t_x \rfloor$. By using k logic circuits, k coefficients of $A(x)$ from a_{m-1-i} to a_{m-k-i} can be loaded simultaneously into the PCA in one clock cycle. Thus, instead of only getting $C^i(x)$ from $C^{i-1}(x)$, $C^{i+k-1}(x)$ can be obtained after one clock cycle. The improved PCA greatly increases the multiplier speed when the system is operating at low clock rates. Suppose the original extended PCA uses m clock cycles for one modular multiplication. Then the time needed for an improved PCA is $t = \lceil \frac{m}{k} \rceil$ clock cycles.

In the case where different clock rates can be selected, we can let $k = m$ and select a clock with a cycle time of $\lceil mt_x + t_d \rceil$. Thus, the multiplication operation can be done in one clock cycle. Fig. 6 illustrates this optimal implementation.

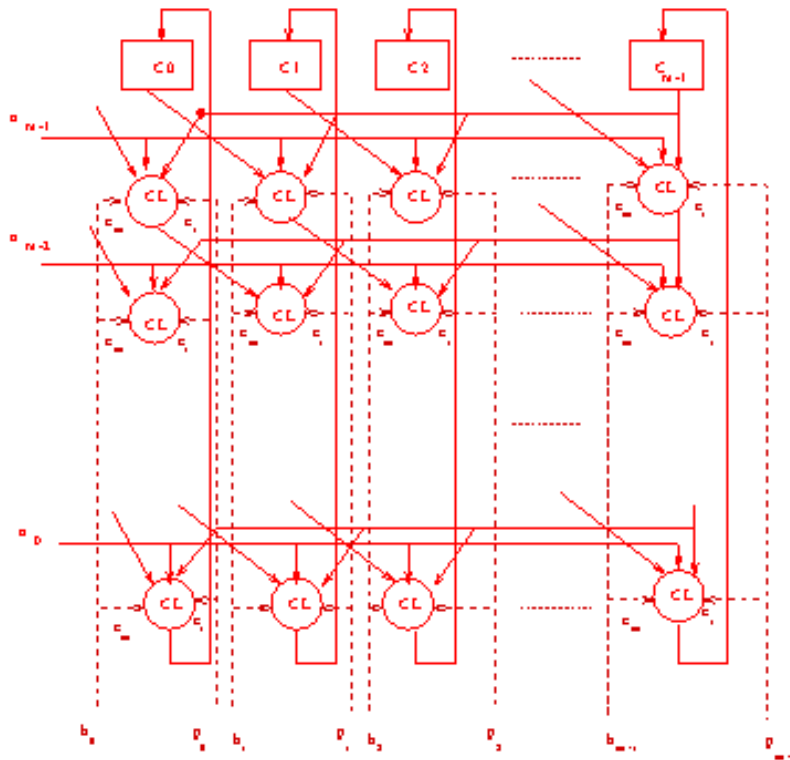


Fig. 6. Optimal PCA multiplier in $GF(2^m)$.

Our proposed PCA based serial/parallel multipliers have modular structures and regular interconnections which are suitable for high speed or restricted space VLSI implementation. Table 2 compares the space and time complexity of the proposed serial multiplier and the optimal parallel multiplier (assuming that the D flip-flop delay is t_d , and that the CL delay is t_x).

Table 2. Space and time complexity of PCA based multipliers.

Multiplier	# of CL	Delay Time	Pipeline Scheme
Parallel PCA Multiplier	m^2	$t_d + mt_x$	Applicable
Serial PCA Multiplier	m	$m(t_d + t_x)$	Not Applicable

5. CONCLUSIONS

In this paper, architectures for parallel/serial multiplication based on PCA have been proposed. These architectures require simple control signals and have regular interconnections. As a consequence, they are very suitable for VLSI implementation. The advantages of this VLSI array modular multiplier are as follows: it is versatile, i.e., the irreducible polynomial can be changed without requiring any hardware changes, which is a valuable feature for some applications; the multiplier can be easily extended to high order of m to achieve greater more security; it is very fast since more levels of cells are applied in the optimal PCA multiplier; and the hardware is very efficient for VLSI implementation. Also, it should be noted that we can make the serial multiplier by using only one level of cells and repeatedly use the cells to get the final result. These low-cost, low-complexity performances are feasible in restricted computing environments.

REFERENCES

1. G. B. Agnew, R. C. Mullin, and S. A. Vanstone, "An implementation of elliptic curve cryptosystems over $F_{2^{155}}$," *IEEE Journal on Selected Areas in Communications*, Vol. 11, 1993, pp. 804-813.
2. R. Schroepel, H. Orman, S. O'Malley, and O. Spatschek, "Fast key exchange with elliptic curve systems," *Advances in Cryptology-CRYPTO 95*, LNCS, No. 963, Springer-Verlag, Berlin, 1995, pp. 43-56, 1995.
3. G. Harper, A. Menezes, and S. Vanstone, "Public-key cryptosystems with very small key lengths," *Advances in Cryptology Proceedings Eurocrypt '92*, LNCS 658, pp. 163-173, Springer-Verlag, 1993.
4. C. K. Koc and B. Sunar, "Low-complexity bit-parallel canonical and normal basis multipliers for a class of finite fields," *IEEE Transactions on Computer*, Vol. 47, 1998, pp. 353-356.
5. C. C. Wang, T. K. Truong, H. M. Shao, L. J. Deutsch, and J. K. Omura, "VLSI architectures for computing multiplications and inverses in $GF(2^m)$," *IEEE Transactions on Computers*, Vol. C-34, 1985, pp. 709-716.
6. C. Paar, P. Fleischmann, and P. Soria-Rodriguez, "Fast arithmetic for public-key algorithms in Galois fields with composite exponents," *IEEE Transactions on Computers*, Vol. 48, 1999, pp. 1025-1034.
7. I. S. Hsu, T. K. Truong, L. J. Deutsch, and I. S. Reed, "A comparison of VLSI architecture of finite field multipliers using dual, normal, or standard bases," *IEEE Transactions on Computers*, Vol. 37, 1988, pp. 735-739.
8. T. Itoh and S. Tsujii, "Structure of parallel multipliers for a class of finite fields $GF(2^m)$," *Information and Computation*, Vol. 83, 1989, pp. 21-40.

9. M. A. Hasan, M. Z. Wang, and V. K. Bhargava, "Modular construction of low complexity parallel multiplier for a class of finite fields $GF(2^m)$," *IEEE Transactions on Computers*, Vol. 41, 1992, pp. 962-971.
10. M. A. Hasan, M. Wang, and V. K. Bhargava, "A modified massey-omura parallel multiplier for a class of finite fields," *IEEE Transactions on Computers*, Vol. 42, 1993, pp. 1278-1280.
11. H. Wu and M. A. Hasan, "Low complexity bit-parallel multiplier for a class of finite fields," *IEEE Transactions Computers*, Vol. 47, 1998, pp. 883-887.
12. H. Wu, "Low complexity bit-parallel finite field arithmetic using polynomial basis," *Cryptographic Hardware and Embedded Systems '99*, LNCS, Vol. 1717, 1999, pp. 280-291.
13. H. Wu, M. A. Hasan, and I. F. Blake, "Highly regular architectures for finite field computation using redundant basis," *Cryptographic Hardware and Embedded Systems '99*, LNCS, Vol. 1717, 1999, pp. 269-279.
14. T. A. York, B. Srisuchinwong, Ph. Tsalides, P. J. Hicks, and A. Thanailakis, "Design and VLSI implementation of mod-127 multiplier using cellular automaton-based data compression techniques," *IEE Proceedings-E*, Vol. 138, 1991, pp. 351-356.
15. B. Srisuchinwong, Ph. Tsalides, T. A. York, P. J. Hicks, and A. Thanailakis, "VLSI implementation of mod-p multiplier using homomorphisms and hybrid cellular automata," *IEE Proceedings-E*, Vol. 139, 1992, pp. 486-490.
16. R. Squier, K. Steiglitz, and M. Jakubowski, "Implementation of parallel arithmetic in a cellular automaton," in *Proceedings of the 1995 International Conference on Application Specific Array Processors*, 1995, pp. 238-245.
17. S. Wolfram, *Theory and Applications of Cellular Automata*, World Scientific, 1986.
18. A. K. Das and A. Ganguly, "Efficient characterisation of cellular automata," *IEE Proceedings Pt. E*, Vol. 137, 1990, pp. 81-85.
19. P. P. Chaudhuri et al, "Additive cellular automata: Theory and applications," *IEEE Computer Society Press*, 1997.
20. S. Sarkar, "A brief history of cellular automata," *ACM Computing Surveys*, Vol. 32, 2000, pp. 80-107.
21. S. Nandi, B. K. Kar, and P. Pal Chaudhuri, "Theory and applications of cellular automata in cryptography," *IEEE Transactions on Computers*, Vol. 43, 1994, pp. 1346-1356.
22. C. N. Zhang, C. Lai, and A. Kostiuk, "High-speed software cellular automata cryptosystem for multimedia applications," *12th International Conference on Wireless Communications*, 2000, pp. 583-591.



Hua Li received the B.E. and M. Sc. degrees from Beijing Polytechnic University and Peking University. He is currently working toward the Ph.D. degree in Department of Computer Science, University of Regina, and TRILabs, Canada. His research interests include parallel systems, reconfigurable computing, fault-tolerant, VLSI design, and information and network security. He is a student member of IEEE.



Chang Nian Zhang received the BSc degree in Applied Math from University of Science Technology, China, and the Ph.D. degree in Computer Science and Engineering from Southern Methodist University. In 1998, he joined Concordia University as a research assistant professor in Department of Computer Science. Since 1990, he has been with University of Regina, Canada, in Department of Computer Science. Currently he is a full professor and leads a research group in parallel processing, data security and neural networks.