This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

An Enhancement of Crosstalk Avoidance Code Based on Fibonacci Numeral System for Through Silicon Vias

Xiaole Cui, Xiaoxin Cui, Yewen Ni, Min Miao, *Senior Member, IEEE*, and Jin Yufeng

***Abstract*— Through silicon vias (TSVs) play an important role as the vertical electrical connections in 3-D stacked integrated circuits. However, the closely clustered TSVs suffer from the crosstalk noise between the neighboring TSVs, and result in the extra delay and the deterioration of signal integrity. For a 3 × 3 TSV array, the severity of crosstalk noise in the center victim TSV is classified into 11 levels, which is defined as 0C to 10C from low noise to high noise, depending on the combinations of the digital patterns applied to the TSV array. An enhanced code based on the Fibonacci number system (FNS) to suppress the crosstalk noise below 6C level is proposed, in which both the redundancy of numbers and the nonuniqueness of Fibonacci-based binary codeword are utilized to search the proper codeword. Experimental results show that the proposed technique decreases about 22% latency of TSVs comparing with the worst crosstalk cases. This technique is applicable in the large-scale TSV array for it has a quasi-linear hardware overhead, and its system overhead is less than that of the 3-D 4-LAT counterpart if the data width is greater than 18, and it has good usability for it consumes less power per TSV and achieves lower bit error rate at the interested frequency range comparing with that of the original FNS coding technique.**

***Index Terms*— Crosstalk, crosstalk avoidance code (CAC), Fibonacci number system (FNS), redundant codeword, redundant number, through silicon via (TSV).**

I. INTRODUCTION

THE 3-D integrated circuit (3-D IC) is an emerging integration technology to address the challenge of the limit of Moore's Law. The 3-D stacked IC with through silicon vias (TSVs), which is suitable for the high density and heterogeneous integration applications, is one of the most popular process technologies of 3-D IC. The TSV plays an important role as the vertical electrical connection between tiers, and bunch of TSVs are even clustered as the intertier high-speed wideband bus. To regulate the routing rules, the clustered TSVs are usually organized into arrays. However, the coupling noise between TSVs is not negligible because of the relatively large TSV size. This crosstalk noise deteriorates the signal integrity of the closely clustered TSVs, and results in the increase of signal latency and transmission power [1]–[4]. The TSV-to-TSV crosstalk problem is an important reliability concern in the 3-D IC design.

The crosstalk avoidance techniques in the traditional 2-D ICs have been widely studied [5]–[16]. However, only two adjacent wires of the victim wire are usually regarded as the aggressors in the 2-D circuit because of the planar structure, while there are usually eight neighboring aggressor TSVs of the victim TSV in the 3-D TSV array. In the 3-D ICs, those crosstalk avoidance techniques for the 2-D ICs are not effective if they are applied directly. Increasing TSV pitch [3], [17] is a feasible solution to avoid the inter-TSV crosstalk, but it increases the area of TSV array. Aiming at suppression of the coupling noise between TSVs, several design efforts were reported in recent years [18]–[22]. Hu *et al.* [18] suppressed the crosstalk by relayout and shielding techniques, while Chang *et al.* [19] proposed a dynamic shielding technique, which remap the less transitional data bits in a period to the TSVs as shields. Kumar and Khatri [20] designed a 3-D crosstalk avoidance code (CAC) to suppress the inter-TSV crosstalk below certain level. However, this method has a large area overhead. Zou *et al.* [21] utilized the less adjacent transition (LAT) code to avoid the worst capacitive coupling cases in the TSV array. And Eghbal *et al.* [22] presented a coding scheme to mitigate the inductive coupling effects between TSVs by adjusting the current flow pattern.

The code-based methods are usually more efficient than the shielding-based methods both on the crosstalk elimination and the area overhead in the 2-D circuit designs [9]. We focus on the Fibonacci number system (FNS)-based coding technique, which has been proven an effective scheme as a CAC in the 2-D design [10]. Kumar and Khatri [20] have directly applied the FNS-based coding method into the $N \times N$ TSV array in a row by row or column by column style to suppress crosstalk noise. However, this method has some drawbacks: 1) although the intrarow/intracolumn crosstalk is avoided, the

Manuscript received July 3, 2016; revised October 3, 2016 and December 6, 2016; accepted January 8, 2017. This work was supported in part by the State Key Development Program for Basic Research of China (973) under Grant 2015CB057201, in part by the National Natural Science Foundation of China under Grant 61306040 and Grant 61176102, in part by the Beijing Natural Science Foundation under Grant 4152020, in part by the Natural Science Foundation of Guangdong Province under Grant 2015A030313147, in part by the Guangdong Science and Technology Project of China under Grant 2014B090913001, and in part by the Importation and Development of High-Caliber Talents Project of Beijing Municipal Institutions under Grant CIT&TCD20150320. *(Corresponding author: Xiaoxin Cui.)*

X. Cui is with the Key Laboratory of Integrated Microsystems, Peking University Shenzhen Graduate School, Shenzhen 518055, China (e-mail: cuixl@pkusz.edu.cn).

X. Cui and Y. Ni are with the Institute of Microelectronics, Peking University, Beijing 100871, China (e-mail: cuixx@pku.edu.cn).

M. Miao is with the Information Microsystem Institute, Beijing Information Science and Technology University, Beijing 100101, China.

J. Yufeng is with the Key Laboratory of Integrated Microsystems, Peking University Shenzhen Graduate School, Shenzhen 518055, China, and also with the Institute of Microelectronics, Peking University, Beijing 100871, China.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2017.2651141

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

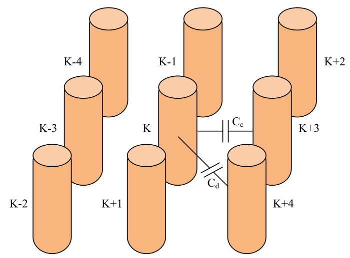


Fig. 1. 3×3 TSV array.

interrow/intercolumn TSVs are still crosstalk prone; 2) in a 3×3 TSV array, only the four adjacent TSVs of the victim TSV are taken as the aggressors, and the crosstalk noise from the four diagonal TSVs is not taken into account; 3) the hardware overhead of FNS-based codec is exponential to the width of the dataword [10]. It limits the application of the FNS-based code in [20] only to small TSV arrays. Taking the crosstalk noise from the eight aggressor TSVs into account, this paper proposes an enhanced FNS-based code to suppress the crosstalk in $3 \times N$ TSV array with low hardware overhead, and overcomes these drawbacks.

The rest of this paper is organized as follows. Considering the crosstalk effects from the diagonal TSVs, the crosstalk noise in TSV array is classified into 11 levels according to its severity in Section II, and the relationship of the crosstalk levels and the data patterns are discussed. Section III proposes our enhanced FNS-based code in detail, and Section IV presents the experimental results. Finally, the conclusions are drawn in Section V.

II. Classification of Crosstalk Noise in TSV Array

A. Crosstalk Noise Levels

The capacitive coupling effects are the dominant parts in the total coupling effects, because the thickness of stacked die in current 3-D IC is rarely greater than 100 μm. Only the capacitive coupling noise is considered in the classification of the crosstalk severity in this paper. As shown in Fig. 1, for the victim TSV$_k$ in the center of a 3×3 TSV array, we define TSV$_{k-1}$, TSV$_{k+1}$, TSV$_{k-3}$, and TSV$_{k+3}$ as the adjacent aggressors, and TSV$_{k-2}$, TSV$_{k+2}$, TSV$_{k-4}$, and TSV$_{k+4}$ as the diagonal aggressors. It uses C_c and C_d to represent the coupling capacitance of the adjacent aggressor TSV and the diagonal aggressor TSV to the victim TSV, respectively. Obviously, the four adjacent aggressors are closer to the victim TSV, implying $C_c > C_d$. The previous study in [20] has found out that $C_c \approx 4 \times C_d$, it is adopted in this paper.

The severity of the crosstalk between TSVs is usually presented with its effective capacitance, calculated from [20]

$$\begin{aligned} C_{\text{eff}} = C_L[1 &+ \lambda_1(\delta_{k,k-3} + \delta_{k,k-1} + \delta_{k,k+1} + \delta_{k,k+3}) \\ &+ \lambda_2(\delta_{k,k-4} + \delta_{k,k-2} + \delta_{k,k+2} + \delta_{k,k+4})] \end{aligned} \quad (1)$$

where

$$\begin{cases} \delta_{k,j} = abs((\Delta V_k - \Delta V_j)/V_{dd}) \\ \Delta V_k = V_k(t^+) - V_k(t^-) \\ \lambda_1 = C_c/C_L \\ \lambda_2 = C_d/C_L. \end{cases}$$

In formula (1), $V_k(t^+)$ and $V_k(t^-)$ are the voltages of the initial pattern and that of its subsequent pattern, respectively. The value of $\delta_{k,j}$ is 2 if the reverse signal transitions occur on TSV$_k$ and TSV$_j$; it is 0 if transitions of the same direction occur on these two TSVs, and the value is 1 if only one TSV in the TSV pair transits, while the other TSV holds its value. According to the relative direction of transition between the aggressor and the victim, the value of $\delta_{k,j}$ is assigned as 0, 1, or 2, respectively, so the range of the effective capacitance is $[C_L,\ C_L(1 + 8\lambda_1 + 8\lambda_2)]$. Considering the relation of $C_c \approx 4 \times C_d$, we obtain that $C_L \leq C_{\text{eff}} \leq C_L (1 + 10\lambda_1)$, in which the range of the normalized effective crosstalk capacitance with respect to $C_L\lambda_1$ is [0, 10]. It means that the severity of the crosstalk noise between TSVs is classified into 11 levels according to the pattern sequences.

For the relationship of the pattern sequence and the effective crosstalk capacitance, we define the pattern sequences as a sextuple $(A_r,\ A_h,\ A_s,\ D_r,\ D_h,\ D_s)$, where A_r is the number of reverse transitions in the pattern sequences between the adjacent aggressors and the victim, A_h is the number of adjacent aggressors that hold their values in the pattern sequences, A_s is the number of the adjacent aggressors with the same direction transitions with respect to that of the victim in the pattern sequences, D_r is the number of reverse transitions in the pattern sequences between the diagonal aggressors and the victim, D_h is the number of diagonal aggressors that hold their values in the pattern sequences, D_s is the number of the diagonal aggressors with the same direction transitions with respect to that of the victim in the pattern sequences. The crosstalk noise is classified into 11 levels based on the severity value defined as

$$\begin{aligned} &\text{Severity value} \\ &\quad = (2 \cdot A_r + 1 \cdot A_h + 0 \cdot A_s) + (2 \cdot D_r + 1 \cdot D_h + 0 \cdot D_s)/4. \end{aligned} \quad (2)$$

The $10C$ level crosstalk noise has a severity value greater than 9.5, and the xC level crosstalk noise has a severity value of $(x - 0.5, x + 0.5]$, $x \in [2, 9]$; the range of severity value is (0, 1.5] for the $1C$ crosstalk noise; and the severity value is 0 for the $0C$ crosstalk noise. For example, the pattern sequences of $10C$ crosstalk noise include two cases, i.e., (4, 0, 0, 4, 0, 0) and (4, 0, 0, 3, 1, 0), presented in the $(A_r,\ A_h,\ A_s,\ D_r,\ D_h,\ D_s)$ style, with the severity value of 10 and 9.75, respectively. We define the pattern sequence results in xC crosstalk noise level as a xC pattern sequence. The relationships of the crosstalk noise levels and the corresponding pattern sequences are summarized in the Appendix.

B. Classification of Patterns

The memory-less codes, which code the pattern in one cycle only based on its present value, are more efficient to be implemented comparing with the memory-based counterparts. To discuss the patterns to be coded explicitly, the xC pattern in a 3×3 TSV array, $x \in [0, 10]$, is defined according to the numbers of the adjacent and diagonal aggressor TSVs with the same/complement logic values, respectively, with respect to that of the victim TSV. Fig. 2 shows the instances of the xC

TABLE I
RANGES OF SEVERITY VALUE FOR VARIOUS PATTERN SEQUENCES

The class of subsequent pattern	The class of initial pattern: 10C	9C	8C	7C	6C	5C	4C	3C	2C	1C	0C
10C	[0,10]										
9C	[0.25,9.5]	[0,9.5]									
8C	[1,9]	[0.5,8.5]	[0,8]								
7C	[1.5,8.5]	[1,8]	[0.5,7.5]	[0,7.5]							
6C	[2,8]	[1.5,7.5]	[1,7]	[0.5,6.5]	[0,6]						
5C	[2.5,8]	[2,7.5]	[1.5,7]	[1,6.5]	[0.5,6]	[0,5.5]					
4C	[3,7]	[2.5,6.5]	[2,6]	[1.5,5.5]	[1,5]	[0.5,4.5]	[0,4]				
3C	[3.5,6.5]	[3,6]	[2.5,5.5]	[2,5]	[1.5,4.5]	[1,4]	[0.5,3.5]	[0,3.5]			
2C	[4,6]	[3.5,5.5]	[3,5]	[2.5,4.5]	[2,4]	[1.5,3.5]	[1,3]	[0.5,2.5]	[0,2]		
1C	[4.5,5.5]	[4,5]	[3.5,4.5]	[3,4]	[2.5,3.5]	[2,3]	[1.5,2.5]	[1,2]	[0.5,1.5]	[0,1.5]	
0C	[4.75,5]	[4.25,4.5]	[3.75,4]	[3.25,3.5]	[2.75,3]	[2.25,2.5]	[1.75,2]	[1.25,1.5]	[0.75,1]	[0.25,0.5]	[0,0]

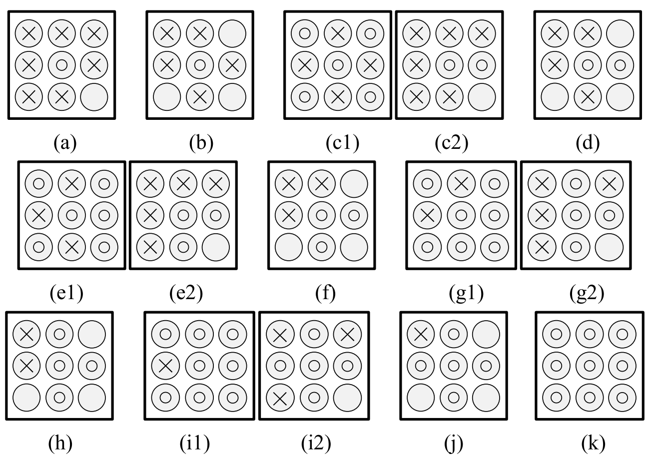


Fig. 2. Examples of xC patterns ($x \in [0, 10]$). (a) $10C$ pattern. (b) $9C$ pattern. (c1) $8C$ pattern: Case1. (c2) $8C$ pattern: Case2. (d) $7C$ pattern. (e1) $6C$ pattern: Case1. (e2) $6C$ pattern: Case2. (f) $5C$ pattern. (g1) $4C$ pattern: Case1. (g2) $4C$ pattern: Case2. (h) $3C$ pattern. (i1) $2C$ pattern: Case1. (i2) $2C$ pattern: Case2. (j) $1C$ pattern. (k) $0C$ pattern.

pattern, $x \in [0, 10]$, where the logic value of the center victim TSV is labeled as "O," and the complement logic value with respect to that of the victim TSV is labeled as "×," and the do not care logic values are not labeled.

For example, a pattern is a $10C$ pattern if all the four adjacent aggressor TSVs are with the complement logic values with respect to that of the victim TSV, and at least three in the four diagonal aggressors are with the complement logic values with respect to that of the victim. And if a pattern is not a $10C$ pattern, it is a $9C$ pattern if all the four adjacent aggressor TSVs are with the complement logic values with respect to that of the victim TSV, and one or two in the four diagonal aggressors are with the complement logic values with respect to that of the victim, and the other diagonal aggressors are with the same logic values with respect to that of the victim. Similarly, the $7C$, $5C$, $3C$, $1C$, and $0C$ patterns can be defined.

However, the $8C$, $6C$, $4C$, and $2C$ patterns have two conditions. For example, if a pattern is not a $9C$ or $10C$ pattern, it is an $8C$ pattern if it satisfies any one of the two following conditions: 1) all the four adjacent aggressors are with the complement logic values with respect to that of the victim, and all the four diagonal aggressors are with the same logic values with respect to that of the victim and (2) at least three in the four adjacent aggressors are with the complement logic values with respect to that of the victim, and the other adjacent aggressor is with the same logic value with respect to that of the victim, and at least three in the four diagonal aggressors are with the complement logic values with respect to that of the victim. The instances of the former case and the latter case are shown in Fig. 2(c1) and (c2), respectively. The $6C$, $4C$, and $2C$ patterns can be defined in the similar style.

For each TSV, transition occurs if the initial value and the subsequent value are different. The severity values of all the combinations of different patterns are collected in Table I. For instance, as shown in the third column of Table I, assume the initial pattern is a $10C$ pattern, the severity value of the pattern sequence is in the range [0, 10] if the subsequent pattern is a $10C$ pattern, and [0.25, 9.5] for a $9C$ subsequent pattern.

From Table I, it is observed that the pattern sequence is not an xC pattern sequence if none of the patterns in the pattern sequence is the xC pattern, i.e., the xC pattern in the consecutive pattern sequence is the necessity condition of an xC pattern sequence. It implies that the xC pattern sequence is avoided if the xC pattern is absent, which assures the feasibility of the memory-less CAC.

III. Proposed Enhancement of FNS-Based Crosstalk Avoidance Code

A. FNS-Based Crosstalk Avoidance Code

Fibonacci sequence is a sequence of natural numbers generated by

$$f_m = \begin{cases} 0 & \text{if } m = 0 \\ 1 & \text{if } m = 1 \\ f_{m-1} + f_{m-2} & \text{if } m \geq 2. \end{cases} \tag{3}$$

A number system is a framework where numbers are represented by numerals in a consistent manner. For instance, the binary numeral system, which is defined as formula (4), is one of the most widely used numeral systems to represent information. The binary numeral system is complete and unambiguous, and it means that each number has one and only one representation in this numeral system. The FNS is such a numeral system that the Fibonacci sequence is used as base. Similar to the binary numeral system, the FNS is also complete, i.e., each number can be represented in this numeral system, as presented in (5)

$$v = \sum_{k=1}^{n} b_k \cdot 2^{k-1} \quad b_k \in \{0, 1\} \tag{4}$$

$$= \sum_{k=1}^{m} d_k \cdot f_k \quad d_k \in \{0, 1\}. \tag{5}$$

Following the notations in [10], the vector $d_0 d_1 \ldots d_{m-1} d_m$ and $b_0 b_1 \ldots b_{n-1} b_n$ are referred as the Fibonacci code and the binary code, in which d_0 and b_0 are LSBs, respectively. For clearness, we label the bit sequence of the Fibonacci code and binary code with the subscripts "F" and "B," respectively. The Fibonacci sequence has an important property as presented in

$$f_m = \sum_{k=0}^{m-2} f_k + 1. \tag{6}$$

Formula (6) implies that the range of m-bit Fibonacci code is $[0, f_{m+2} - 1]$, the m-bit Fibonacci code is able to represent totally f_{m+2} distinct values. While the range of n-bit binary code is $[0, 2^n - 1]$, and it is able to represent 2^n distinct values in total.

However, FNS is ambiguous because some numbers have nonunique representation in FNS. All Fibonacci codes that represent the same value are equivalent. For instance, Fibonacci code 001_{F} and 110_{F} are equivalent, because both of them represent the decimal number "1," the decimal number "19" even has seven equivalent Fibonacci codes. Actually, all the segments of "001_{F}" in a Fibonacci code can be equivalently substituted by the segments of "110_{F}."

Duan *et al.* [10] has proven that the FNS-based codes are forbidden pattern free code, which avoid the worst crosstalk cases such as 101 or 010, in the 2-D circuit designs. This property of nonuniqueness of the FNS code provides great flexibility, and for an equivalent mutation code can be chosen if the initial code does not meet the requirements of crosstalk avoidance.

The FNS and the binary number system have a relation presented in

$$2^n \le f_{m+2} \le 2 f_{m+1}. \tag{7}$$

It shows that $m + 2$ bit Fibonacci code represents n bit binary code, there are redundant bits in the Fibonacci code because $n < m$. For example, 9-bit Fibonacci code represents the number of decimal digits as 6-bit binary code does, it introduces $(9 - 6)/6 = 50\%$ system overhead.

B. Proposed Method

An enhanced FNS-based code is designed to suppress the crosstalk noise below $6C$ level. The original input pattern of the TSV array is defined as dataword, and the coded pattern is referred as codeword.

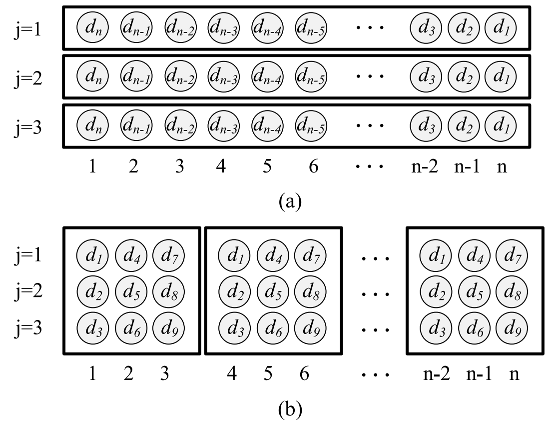


Fig. 3. (a) FNS-based coding technique in [20]. (b) Proposed FNS-based coding technique.

Kumar and Khatri [20] have applied the FNS-based code into the TSV array by a unit of single row/column, as shown in Fig. 3(a). For the worst cases, it suppresses the crosstalk noise below $6C$ level not considering the crosstalk noise from the diagonal TSVs, i.e., below $8C$ level following the definition in Section II-A. In [20], the hardware overhead of the FNS CAC codec is very heavy for the large scaled TSV array, because the FNS code is simply applied by the unit of row or column.

To overcome these drawbacks, in the proposed scheme, the $3 \times N$ TSV array is divided into several subarrays, and the FNS-based coding technique is applied within each subarray, as shown in Fig. 3(b). Based on the FNS-based coding technique, the dataword of a 3×3 subarray is coded into a 9-bit FNS codeword, and the weight of each bit of the FNS codeword is 1, 1, 2, 3, 5, 8, 13, 21, 34 from LSB to MSB, respectively. The 9-bit FNS-based codeword represents 89 decimal numbers, the maximum number is 88, and the minimum number is 0. It represents all the decimal numbers of a 6-bit binary codeword, for $63 < 88 < 127$. The dataword of a $3 \times N$ array is coded from the left to the right by the unit of 3×3 subarray. If the width of the dataword *mod* $6 = 5$ or 0, the right most subarray in the $3 \times N$ array is a 3×3 subarray. If the width of the dataword *mod* $6 = 1{\sim}4$, a 3×2, instead of a 3×3, subarray is the right most subarray, it saves 3 bit of the FNS-based codeword with respect to the 9-bit counterpart. A 6-bit FNS-based codeword is able to represent decimal number in the range of [0, 20], and it corresponds to a 4-bit binary codeword.

Another drawback of the scheme in [20] is that the interrow/intercolumn TSVs are crosstalk prone, although the intrarow/intracolumn crosstalk is avoided. To solve this problem, we select each FNS-based codeword by a proper pattern criterion. As analyzed in Section II-B, the $6C$ pattern condition is required if the aim is to suppress the crosstalk below $6C$ level.

However, the $3 \times N$ TSV array is built by the cascading subarrays, as shown in Fig. 3(b), and all the TSVs in the second row may become the victim. The $6C$ pattern condition is sufficient only if the TSV d_5 in each subarray is regarded as the victim. As shown in Fig. 4, if TSV d_2 in the target subarray

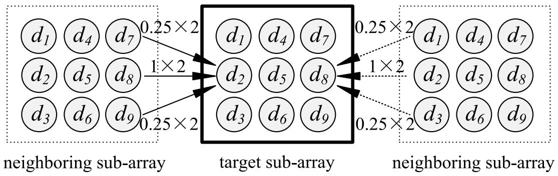


Fig. 4. Worst case of the victim TSV d_2 and d_8.

is taken as the victim, in the worst case, i.e., d_7, d_8, and d_9 of its left neighboring subarray are with the complement logic values with respect to that of d_2 in the target subarray, it consumes $(0.25 + 1 + 0.25) \times 2C = 3C$ severity value. So within the target subarray, the total severity value is below $3C$, i.e., at least two TSVs in d_1, d_3, and d_5 are with the same logic values with respect to that of d_2, because only d_1, d_3, d_4, d_5, and d_6 in the target subarray affect the voltage of d_2. A similar result can be drawn if we take TSV d_8 as the victim.

In summary, for any 3×3 TSV subarray, the pattern criterion is stated as follows.

1) For TSV d_5, the pattern results in an effective capacitance below $6C$ level.
2) For TSV d_2 and d_8, the pattern results in an effective capacitance below $3C$ level.

Similarly, for the right most 3×2 TSV subarray, the pattern criterion is stated as follows.

1) For TSV d_5, this pattern results in an effective capacitance below $6C$ level.
2) For TSV d_2, this pattern results in an effective capacitance below $3C$ level.

We utilize the redundancy of the FNS-based code to substitute the codeword that is not able to meet the pattern criterion. There are two types of redundancy in our FNS-based code.

The first type of redundancy comes from the nonuniqueness of the FNS-based codeword, and it is referred as the codeword redundancy. For an input dataword, if a specific FNS-based codeword does not satisfy the pattern criterion, an alternative equivalent FNS-based codeword is used to substitute the current codeword. The mutation codeword can be generated by iteratively applying the rule "$110_F \Leftrightarrow 001_F$."

The second type of redundancy comes from the system overhead of our FNS-based coding technique, and it is referred as the number redundancy. For the 3×3 subarray, a 6-bit binary dataword is coded into a 9-bit FNS-based codeword. A 9-bit FNS-based codeword represents 89 decimal numbers, and however, a 6-bit binary dataword only represents 64 decimal numbers, and 25 redundant numbers are introduced. For the 3×2 subarray, the 6-bit FNS-based codeword represents 21 decimal numbers, while the corresponding 4-bit binary dataword only represents 16 decimal numbers, and there are five redundant numbers. For a dataword, if a specific FNS-based codeword does not satisfy the pattern criterion, the FNS-based codeword of the redundant number can be utilized as a substitution. This technique adds lots of candidates of the mutation codeword, because each redundant number may have several equivalent FNS-based codewords.

A $3 \times N$ TSV array provides a $3N$-bit-wide channel. To suppress the crosstalk below $6C$ level, we propose a four-step data transmission method based on the FNS-based coding technique, as shown in Fig. 5.

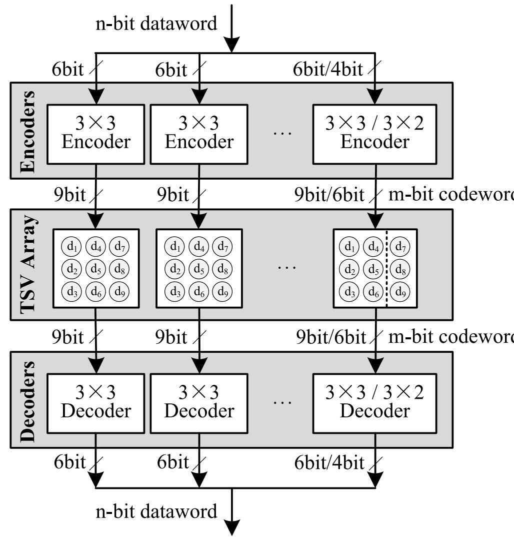


Fig. 5. Proposed data transmission method.

1) *Step 1 (Slicing the Input Dataword):* For the n-bit binary dataword, execute operation n *mod* 6, the quotient is k, and the remainder of the operation is referred as the tail of the input dataword. The input dataword is divided into k 6-bit binary slices and a tail slice.
2) *Step 2:* Each 6-bit binary slice is coded into a 9-bit FNS-based codeword. The tail slice is coded into a 9-bit FNS-based codeword if the remainder of n *mod* 6 is 5 or 0; otherwise, it is coded into a 6-bit FNS-based codeword.
3) *Step 3:* Each 9-bit FNS-based codeword is transmitted through a 3×3 TSV subarray, and the 6-bit FNS-based codeword is transmitted through a 3×2 TSV subarray.
4) *Step 4:* The received FNS-based codewords are decoded and merged into an n-bit binary sequence.

This enhanced FNS coding scheme is named 6CmFNS for it suppress the crosstalk below $6C$ level. The four-step data transmission method assures the crosstalk noise is below $6C$ level in the $3\times N$ TSV array. For the $M\times N$ TSV array, $M > 3$, this scheme is applied by the unit of $3\times N$ TSV subarray. And a relatively larger pitch is required between each $3 \times N$ TSV subarray, and for the proposed 6CmFNS coding scheme is not able to suppress the crosstalk between the $3 \times N$ subarrays below $6C$ level.

C. Generation of the Codeword Table

To implement the proposed coding technique, a codeword table is required. For any given dataword to be transmitted through a TSV subarray, the corresponding FNS-based codeword is generated by the algorithm shown in Fig. 6. In the algorithm, an initial FNS-based codeword is generated for the input dataword. If this codeword satisfies the pattern criterion in Section III-B, it is written into the codeword table. Otherwise, it uses the redundant FNS-based codewords as the substitution codewords, and the unused equivalent codeword

TABLE II
CODEWORD TABLE OF A 3 × 3 SUBARRAY

Dataword		Codeword	Dataword		Codeword	Dataword		Codeword
Decimal	Binary	$d_1d_4d_7$ $d_2d_5d_8$ $d_3d_6d_9$	Decimal	Binary	$d_1d_4d_7$ $d_2d_5d_8$ $d_3d_6d_9$	Decimal	Binary	$d_1d_4d_7$ $d_2d_5d_8$ $d_3d_6d_9$
0	000000	000_000_000	22	010110	111_001_000	44	101100	001_001_001
1	000001	100_000_000	23	010111	001_001_100	45	101101	011_110_110
2	000010	001_000_000	24	011000	011_011_000	46	101110	111_110_110
3	000011	001_000_100	25	011001	111_001_100	47	101111	000_011_110
4	000100	100_100_000	26	011010	001_101_100	48	110000	100_100_111
5	000101	000_110_110	27	011011	100_101_001	49	110001	001_100_111
6	000110	111_011_011	28	011100	000_011_011	50	110010	011_011_110
7	000111	111_100_000	29	011101	100_000_111	51	110011	011_010_110
8	001000	000_000_111	30	011110	001_000_111	52	110100	110_111_110
9	001001	100_001_000	31	011111	000_100_111	53	110101	011_111_110
10	001010	001_001_000	32	100000	000_001_111	54	110110	111_111_110
11	001011	011_110_000	33	100001	000_110_111	55	110111	111_001_001
12	001100	111_110_000	34	100010	000_000_001	56	111000	100_001_111
13	001101	000_000_100	35	100011	100_000_001	57	111001	001_001_111
14	001110	100_000_100	36	100100	001_000_001	58	111010	000_011_111
15	001111	110_011_000	37	100101	111_100_100	59	111011	011_011_011
16	010000	000_100_100	38	100110	100_100_001	60	111100	110_011_011
17	010001	100_100_100	39	100111	111_011_000	61	111101	110_110_011
18	010010	001_100_100	40	101000	110_111_000	62	111110	110_010_011
19	010011	011_111_000	41	101001	111_100_001	63	111111	000_110_011
20	010100	111_111_000	42	101010	000_001_001			
21	010101	110_110_000	43	101011	100_001_001			

with minimum code weight is written into the codeword table. If all the redundant FNS-based codewords of the dataword do not satisfy the pattern criterion, the redundant numbers are used as the substitutions until the proper codeword is found. With this algorithm, the codeword table of the 3×3 and 3×2 TSV subarrays is generated, as shown in Tables II and III, respectively, where the shaded codewords are those substituted with the redundancies.

IV. ANALYSIS AND EXPERIMENTAL RESULTS

A. Delay

The TSVs are made of copper, tungsten [23], or carbon nanotube [24], and the copper is the most mature TSV material at present. The circuit model of the copper TSV was evolved from the simple *RC* model [25] to the fullwave *RLGC* model [26], and the more accurate metal–insulator–semiconductor (MIS) transmission line TSV model was proposed in recent years [27]. Fig. 7(a) shows the MIS model of TSV, where R_{TSV}, L_{TSV}, and C_{TSV} are the resistance, inductance, and load capacitance of TSV, respectively, and C_{i} is the side wall capacitance. Fig. 7(b) shows the circuit model of a 3×3 TSV subarray, where the parallel of the capacitance and conductance presents the coupling between TSVs.

Assume that parameters R_{TSV}, L_{TSV}, C_{TSV}, and C_i in the TSV model are 2×10^{-1} Ω, 1×10^{-11} H, 5×10^{-14} F, and 9×10^{-14} F, the value of crosstalk capacitances C_c and C_d are 3×10^{-13} and 7.5×10^{-14} F, respectively, and the crosstalk conductances G_c and G_d are 1×10^{-5} and 7.1×10^{-6} S, respectively, according to the TSV parameter ranges in [28]. The delay of the victim TSV caused by the aggressor TSVs in a 3×3 subarray is greatly relevant to the input pattern.

TABLE III
CODEWORD TABLE OF A 3 × 2 SUBARRAY

Dataword		Codeword	Dataword		Codeword
Decimal	Binary	d_1d_4 d_2d_5 d_3d_6	Decimal	Binary	d_1d_4 d_2d_5 d_3d_6
0	0000	000_000	8	1000	000_001
1	0001	100_000	9	1001	100_001
2	0010	001_000	10	1010	001_001
3	0011	000_100	11	1011	011_110
4	0100	100_100	12	1100	111_110
5	0101	001_100	13	1101	000_011
6	0110	111_011	14	1110	011_011
7	0111	111_100	15	1111	110_011

A pulse is input into the victim TSV with $0C$ to $10C$ configurations of its neighboring TSVs, and the simulation results on signal delays with SMIC 65-nm technology file are collected in Table IV.

The improvement on signal delays is measured by formula (8) for the worst case of $10C$, where t_{delay} is the maximal signal delay if the crosstalk avoidance method is applied

$$\gamma = \frac{(t_{10C} - t_{\text{delay}})}{t_{10C}}. \tag{8}$$

Table V compares the worst signal delays obtained by different CAC methods. It is seen that both the 3-D 4-LAT and our proposed 6CmFNS are able to reduce about 22% signal delay with respect to that of the worst cases.

B. System Overhead

The CAC schemes introduce redundancy bits. As analyzed earlier, for the proposed 6CmFNS coding method, 6-bit binary

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE IV
TYPICAL SIGNAL DELAYS OF VARIOUS CROSSTALK LEVELS

Crosstalk Level	0C	1C	2C	3C	4C	5C	6C	7C	8C	9C	10C
Signal Delay (ps)	96	108	110	119	121	129	137	143	149	157	175

Algorithm: The generation algorithm of FNS based codeword of a sub-array

Input: Binary dataword $pattern_B$ to be transmitted through the TSV array;
Output: The FNS based codeword $code_F$;

```
Binary String GenFNScode (binary dataword pattern_B) {
    code_F =Translate (pattern_B);
                          // Find an initial FNS based codeword of pattern_B
    if(Check (code_F) == pass) {
                          // if code_F meet the pattern criterion
      return (code_F);    // return the FNS based codeword
    }
    else {
      while (there are untried redundant number) {
        while (there are untried redundant equivanlent codeword
                of current number) {
          if (Check (code_F) == pass) {
                          // if code_F meet the pattern criterion
            break;
          }
          else {
            code_F ← Select the unused equivalent redundant codeword
    of code_F with the minimum code weight
          }
        }
        if (Check (code_F) == pass) {
                          // if code_F meet the pattern criterion
          break;
        }
        else {
          code_F ← Select the unused codeword of the redundant number
    of code_F with the minimum code weight
        }
      }
    }
    if (Check (code_F) == fail) {
                          // if all code_Fs do not meet the pattern criterion
      abort ( );
    }
    else  return (code_F);  // return the FNS based codeword
}
```

Fig. 6. Generation algorithm of FNS-based codeword of a subarray.

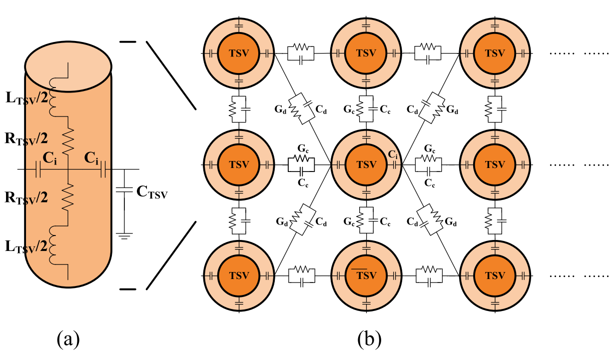


Fig. 7. (a) Model of single TSV. (b) Model of TSV array.

code requires 9-bit Fibonacci code, and three extra TSVs are needed. System overhead, caused by the fact that n bit binary dataword is coded into m bit codeword, is used to evaluate the code efficiency. It is calculated by

$$\text{overhead} = \frac{m-n}{n}. \tag{9}$$

TABLE V
COMPARISON ON SIGNAL DELAYS OF DIFFERENT CAC TECHNIQUES FOR TSV ARRAYS

CAC Techniques	Uncoded Data	Original FNS [20]	3D-LAT [21]	6CmFNS
Worst Signal Delay (ps)	175	149	137	137

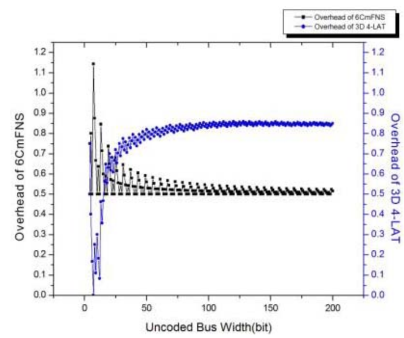


Fig. 8. Comparison on system overhead between the 3-D 4-LAT technique and the proposed 6CmFNS technique.

Among the previous CAC techniques, only 3-D LAT [21] is a true 3-D CAC method, which takes the crosstalk noise from the diagonal TSVs into account. We compare the system overhead of 3D 4-LAT with our proposed 6CmFNS with various width of dataword n, and for 3-D 4-LAT, it is also able to suppress the crosstalk below $6C$ level. Fig. 8 shows that the system overhead of 6CmFNS technique is less than that of 3-D 4-LAT technique if the width of the input dataword is greater than 18 bit. It implies that the proposed method is more applicable in the large scale TSV arrays.

C. Hardware Overhead

Duan *et al.* [10] has found out that the hardware overhead of the FNS codec is exponential with the width of input dataword. To avoid the inhibitive hardware overhead, we implement 6CmFNS codec with a lookup table for the 3×3 and 3×2 subarrays. Synthesis results with SMIC 65-nm technology file show that the encoder and decoder of 6CmFNS for 3×3 subarray are 483 and 438 μm^2, while the areas of encoder and decoder of 6CmFNS for 3×2 subarray are 135 and 127 μm^2, respectively. Fig. 9 compares the gate count of the proposed 6CmFNS codec with that of the original FNS [20] codec. In Fig. 9, the gate count is normalized by the area of the 3×3

TABLE VI

SEXTUPLED PRESENTATIONS OF *xC* PATTERN SEQUENCES AND THE CORRESPONDING SEVERITY VALUES

Pattern Sequence Level	Severity Value	Sextupled Presentation (A_r,A_h,A_s,D_r,D_h,D_s)
10C	(9.5-10]	(4,0,0,4,0,0) (4,0,0,3,1,0)
9C	(8.5-9.5]	(4,0,0,3,0,1) (4,0,0,2,2,0) (4,0,0,2,1,1) (4,0,0,2,0,2) (4,0,0,1,3,0) (4,0,0,1,2,1) (4,0,0,1,1,2) (4,0,0,0,4,0) (4,0,0,0,3,1) (3,1,0,4,0,0) (3,1,0,3,1,0)
8C	(7.5-8.5]	(4,0,0,1,0,3) (4,0,0,0,2,2) (4,0,0,0,1,3) (4,0,0,0,0,4) (3,1,0,3,0,1) (3,1,0,2,2,0) (3,1,0,2,1,1) (3,1,0,2,0,2) (3,1,0,1,3,0) (3,1,0,1,2,1) (3,1,0,1,1,2) (3,1,0,0,4,0) (3,1,0,0,3,1) (3,0,1,4,0,0) (3,0,1,3,1,0) (2,2,0,4,0,0) (2,2,0,3,1,0)
7C	(6.5-7.5]	(3,1,0,1,0,3) (3,1,0,0,2,2) (3,1,0,0,1,3) (3,1,0,0,0,4) (3,0,1,3,0,1) (3,0,1,2,2,0) (3,0,1,2,1,1) (3,0,1,2,0,2) (3,0,1,1,3,0) (3,0,1,1,2,1) (3,0,1,1,1,2) (3,0,1,0,4,0) (3,0,1,0,3,1) (2,2,0,3,0,1) (2,2,0,2,2,0) (2,2,0,2,1,1) (2,2,0,2,0,2) (2,2,0,1,3,0) (2,2,0,1,2,1) (2,2,0,1,1,2) (2,2,0,0,4,0) (2,2,0,0,3,1) (2,1,1,4,0,0) (2,1,1,3,1,0) (1,3,0,4,0,0) (1,3,0,3,1,0)
6C	(5.5-6.5]	(3,0,1,0,2,2) (3,0,1,1,0,3) (3,0,1,0,1,3) (3,0,1,0,0,4) (2,2,0,0,2,2) (2,2,0,1,0,3) (2,2,0,0,1,3) (2,2,0,0,0,4) (2,1,1,3,0,1) (2,1,1,2,2,0) (2,1,1,2,1,1) (2,1,1,2,0,2) (2,1,1,1,3,0) (2,1,1,1,2,1) (2,1,1,1,1,2) (2,1,1,0,4,0) (2,1,1,0,3,1) (1,3,0,3,0,1) (1,3,0,2,2,0) (1,3,0,2,1,1) (1,3,0,2,0,2) (1,3,0,1,3,0) (1,3,0,1,2,1) (1,3,0,1,1,2) (1,3,0,0,4,0) (1,3,0,0,3,1) (2,0,2,4,0,0) (2,0,2,3,1,0) (1,2,1,4,0,0) (1,2,1,3,1,0) (0,4,0,4,0,0) (0,4,0,3,1,0)
5C	(4.5-5.5]	(2,1,1,0,2,2) (2,1,1,1,0,3) (2,1,1,0,1,3) (2,1,1,0,0,4) (1,3,0,0,2,2) (1,3,0,1,0,3) (1,3,0,0,1,3) (1,3,0,0,0,4) (2,0,2,3,0,1) (2,0,2,2,2,0) (2,0,2,2,1,1) (2,0,2,2,0,2) (2,0,2,1,3,0) (2,0,2,1,2,1) (2,0,2,1,1,2) (2,0,2,0,4,0) (2,0,2,0,3,1) (1,2,1,3,0,1) (1,2,1,2,2,0) (1,2,1,2,1,1) (1,2,1,1,3,0) (1,2,1,2,0,2) (1,2,1,1,2,1) (1,2,1,1,1,2) (1,2,1,0,4,0) (1,2,1,0,3,1) (0,4,0,3,0,1) (0,4,0,2,2,0) (0,4,0,2,1,1) (0,4,0,2,0,2) (0,4,0,1,3,0) (0,4,0,1,2,1) (0,4,0,1,1,2) (0,4,0,0,4,0) (0,4,0,0,3,1) (0,3,1,4,0,0) (0,3,1,3,1,0) (1,1,2,4,0,0) (1,1,2,3,1,0)
4C	(3.5-4.5]	(2,0,2,1,0,3) (2,0,2,0,2,2) (2,0,2,0,1,3) (2,0,2,0,0,4) (1,2,1,1,0,3) (1,2,1,0,2,2) (1,2,1,0,1,3) (1,2,1,0,0,4) (0,4,0,1,0,3) (0,4,0,0,2,2) (0,4,0,0,1,3) (0,4,0,0,0,4) (0,3,1,3,0,1) (0,3,1,2,2,0) (0,3,1,2,1,1) (0,3,1,2,0,2) (0,3,1,1,3,0) (0,3,1,1,2,1) (0,3,1,1,1,2) (0,3,1,0,4,0) (0,3,1,0,3,1) (1,1,2,3,0,1) (1,1,2,2,2,0) (1,1,2,2,1,1) (1,1,2,2,0,2) (1,1,2,1,3,0) (1,1,2,1,2,1) (1,1,2,1,1,2) (1,1,2,0,4,0) (1,1,2,0,3,1) (1,0,3,4,0,0) (1,0,3,3,1,0) (0,2,2,4,0,0) (0,2,2,3,1,0)
3C	(2.5-3.5]	(0,3,1,1,0,3) (0,3,1,0,2,2) (0,3,1,0,1,3) (0,3,1,0,0,4) (1,1,2,1,0,3) (1,1,2,0,2,2) (1,1,2,0,1,3) (1,1,2,0,0,4) (1,0,3,3,0,1) (1,0,3,2,2,0) (1,0,3,2,1,1) (1,0,3,2,0,2) (1,0,3,1,3,0) (1,0,3,1,2,1) (1,0,3,1,1,2) (1,0,3,0,4,0) (1,0,3,0,3,1) (0,2,2,3,0,1) (0,2,2,2,2,0) (0,2,2,2,1,1) (0,2,2,2,0,2) (0,2,2,1,3,0) (0,2,2,1,2,1) (0,2,2,1,1,2) (0,2,2,0,4,0) (0,2,2,0,3,1) (0,1,3,4,0,0) (0,1,3,3,1,0)
2C	(1.5-2.5]	(1,0,3,1,0,3) (1,0,3,0,2,2) (1,0,3,0,1,3) (1,0,3,0,0,4) (0,2,2,0,2,2) (0,2,2,1,0,3) (0,2,2,0,1,3) (0,2,2,0,0,4) (0,1,3,3,0,1) (0,1,3,2,2,0) (0,1,3,2,1,1) (0,1,3,2,0,2) (0,1,3,1,3,0) (0,1,3,1,2,1) (0,1,3,1,1,2) (0,1,3,0,4,0) (0,1,3,0,3,1) (0,0,4,4,0,0) (0,0,4,3,1,0)
1C	(0-1.5]	(0,1,3,0,2,2) (0,1,3,1,0,3) (0,1,3,0,1,3) (0,1,3,0,0,4) (0,0,4,3,0,1) (0,0,4,2,2,0) (0,0,4,2,1,1) (0,0,4,2,0,2) (0,0,4,1,3,0) (0,0,4,1,2,1) (0,0,4,1,1,2) (0,0,4,1,0,3) (0,0,4,0,4,0) (0,0,4,0,3,1) (0,0,4,0,2,2) (0,0,4,0,1,3)
0C	[0]	(0,0,4,0,0,4)

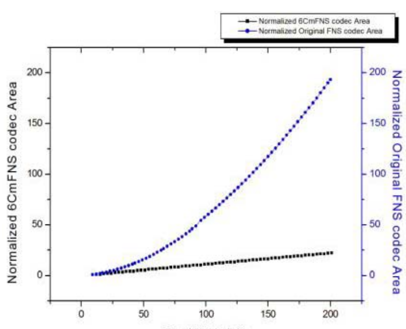


Fig. 9. Comparison on gate counts of codec hardware between the proposed 6CmFNS technique and original FNS CAC method.

6CmFNS codec. It shows that the 6CmFNS codec consumes less gate count than that of the original FNS codec, and the hardware overhead of 6CmFNS codec increases quasi-linearly with the increase of dataword width.

The 3-D 4-LAT technique codes a 5-bit dataword into a 9-bit codeword, it has a heavier system overhead comparing with our proposed technique, and it consumes more 3×3 subarray for the same input dataword. With SMIC 65-nm technology file, the lookup table of 3-D 4-LAT codec consumes 1290 μm^2 if the data width is 8, and the area increases to 342 184 μm^2 if the data width grows to 16. The area overhead of 3-D 4-LAT grows exponential with the width of input dataword, even worse than that of the original FNS codec.

D. Power

Simulations are conducted on the transmission power per TSV, because different CAC technique leads to different system overheads. In the power simulation, we use the same TSV parameters in Section IV-A, and 10 000 random uncoded 90-bit datawords are generated. If we apply this set of uncoded dataword directly to a 3×30 TSV array, it generates 2.65×10^{-6} W average transmission power per TSV. And if the datawords are applied to a 3×45 TSV array with the 6CmFNS coding technique and a 3×43 TSV array with the original FNS coding technique, respectively, 2.55×10^{-6} and 2.61×10^{-6} W average transmission power per TSV were generated. It shows that both the 6CmFNS and the original FNS coding techniques are able to reduce the average transmission power per TSV, and the former one generates less power than the latter one does.

E. Bit Error Rate

The reliability of the data transmission is presented by the bit error rate (BER). The 10 000 randomly generated uncoded

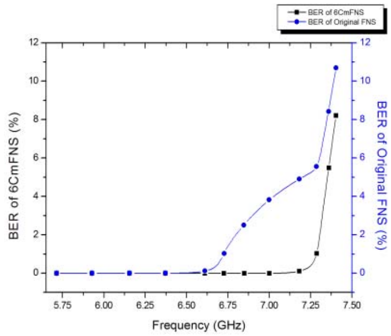


Fig. 10. Comparison on BER between the proposed 6CmFNS technique and the original FNS CAC technique.

90-bit datawords were applied into a 3 × 45 TSV array with the 6CmFNS coding technique and a 3 × 43 TSV array with the original FNS coding technique, respectively, and the results on the BER are presented in Fig. 10. According to the simulation results on signal delay in Section IV-A, the delays of 10*C* and 6*C* patterns are 175 and 137 ps, respectively, correspond to the frequency of 5.7 and 7.3 GHz. The BERs of the 6CmFNS and the original FNS coding techniques at the frequency range of 5.5–7.5GHz are shown in Fig. 10, because these two techniques can suppress the crosstalk noise below 6*C* level and 8*C* level, respectively. Fig. 10 shows that the BERs of the 6CmFNS coding technique are less than that of the original FNS coding technique.

V. Conclusion

This paper proposes an enhanced FNS-based code technique to suppress the crosstalk noise in the TSV array below 6*C* level. Both the redundant numbers and redundant codewords of the FNS-based code are utilized to generate the codeword tables. This is a true 3-D CAC, because the crosstalk noise from all the eight aggressors in a 3 × 3 TSV array is taken into account. Simulation results show that the proposed coding technique is able to reduce about 22% signal delay with respect to that of the worst cases. The proposed technique has advantage on system overhead and hardware overhead, which are important for the applications on the large scale TSV arrays. And this CAC technique is low power and low BER, showing a good usability in the 3-D IC designs.

Appendix

See Table VI.

References

[1] Z. Zhang, R. Fang, G. Wang, X. Sun, Y. Jin, and M. Miao, "Fast time domain crosstalk analysis of through silicon vias based on equivalent circuit model," in *Proc. IEEE Electron. Packag. Technol. (ICEPT)*, Chengdu, China, Aug. 2014, pp. 134–137.

[2] I. Savidis and E. G. Friedman, "Closed-form expressions of 3-D via resistance, inductance, and capacitance," *IEEE Trans. Electron Devices*, vol. 56, no. 9, pp. 1881–1973, Sep. 2009.

[3] D. H. Kim, S. Mukhopadhyay, and S. K. Lim, "Fast and accurate analytical modeling of through-silicon-via capacitive coupling," *IEEE Trans. Compon. Packag. Manuf. Technol.*, vol. 1, no. 2, pp. 168–180, Feb. 2011.

[4] C. Liu, T. Song, J. Kim, J. Kim, S. K. Lim, and J. Cho, "Full-chip TSV-to-TSV coupling analysis and optimization in 3D IC," in *Proc. ACM Design Autom. Conf. (DAC)*, New York, NY, USA, Jun. 2011, pp. 783–788.

[5] Z. Xu *et al.*, "Crosstalk evaluation, suppression and modeling in 3D through-strata-via (TSV) network," in *Proc. IEEE Int. 3D Syst. Integr. Conf.*, Nov. 2010, pp. 1–8.

[6] H. Guo and S. Parameswaran, "Shifted gray encoding to reduce instruction memory address bus switching for low-power embedded systems," *J. Syst. Archit.*, vol. 56, pp. 263–270, Jun. 2010.

[7] M. R. Stan and W. P. Burleson, "Bus-invert coding for low-power I/O," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 3, no. 1, pp. 49–58, Mar. 1995.

[8] J. Gu and H. Guo, "A segmental bus-invert coding method for instruction memory data bus power efficiency," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Taipei, Taiwan, May 2009, pp. 137–140.

[9] B. Victor and K. Keutzer, "Bus encoding to prevent crosstalk delay," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, San Jose, CA, USA, Nov. 2001, pp. 57–63.

[10] C. Duan, V. H. C. Calle, and S. P. Khatri, "Efficient on-chip crosstalk avoidance CODEC design," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 4, pp. 551–560, Apr. 2009.

[11] X. Wu and Z. Yan, "Efficient CODEC designs for crosstalk avoidance codes based on numeral systems," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 4, pp. 548–558, Apr. 2011.

[12] P. Subrahmanya, R. Manimegalai, M. Madhu, and V. Kamakoti, "A bus encoding technique for power and crosstalk minimization," in *Proc. 17th Int. Conf. VLSI Design*, Mumbai, India, Jan. 2004, pp. 443–448.

[13] L. Li, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin, "A crosstalk aware interconnect with variable cycle transmission," in *Proc. IEEE Int. Conf. Design Autom. Test Eur. (DATE)*, Paris, France, Feb. 2004, pp. 102–107.

[14] B. Halak, "Partial coding algorithm for area and energy efficient crosstalk avoidance codes implementation," *IET Comput. Digit. Techn.*, vol. 8, no. 2, pp. 97–107, Mar. 2014.

[15] W. N. Flayyih, K. Samsudin, S. J. Hashim, F. Z. Rokhani, and Y. I. Ismail, "Crosstalk-aware multiple error detection scheme based on two-dimensional parities for energy efficient network on chip," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 61, no. 7, pp. 2034–2047, Jul. 2014.

[16] G. Kornaros, "Temporal coding schemes for energy efficient data transmission in systems-on-chip," in *Proc. 7th Workshop Intell. Solutions Embedded Syst.*, Regensburg, Germany, Jun. 2009, pp. 111–118.

[17] R. Arunachalam, E. Acar, and S. R. Nassif, "Optimal shielding/spacing metrics for low power design," in *Proc. IEEE Int. Annu. Symp. VLSI (ISVLSI)*, Lafayette, LA, USA, Feb. 2003, pp. 167–172.

[18] J. Hu, Q. Wang, J. Jiang, J. Xie, and Z. Mao, "A crosstalk avoidance scheme based on re-layout of signal TSV," in *Proc. IEEE 11th Int. Conf. ASIC (ASICON)*, Chengdu, China, Nov. 2015, pp. 1–4.

[19] Y.-Y. Chang, Y. S.-C. Huang, V. Narayanan, and C.-T. King, "ShieldUS: A novel design of dynamic shielding for eliminating 3D TSV crosstalk coupling noise," in *Proc. IEEE 18th Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Yokohama, Japan, Jan. 2013, pp. 675–680.

[20] R. Kumar and S. P. Khatri, "Crosstalk avoidance codes for 3D VLSI," in *Proc. IEEE Int. Conf. Design Autom. Test Eur. (DATE)*, Grenoble, France, Mar. 2013, pp. 1673–1678.

[21] Q. Zou, D. Niu, Y. Cao, and Y. Xie, "3DLAT: TSV-based 3D ICs crosstalk minimization utilizing less adjacent transition code," in *Proc. IEEE 19th Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Singapore, Jan. 2014, pp. 762–767.

[22] A. Eghbal, P. M. Yaghini, S. S. Yazdi, and N. Bagherzadeh, "TSV-to-TSV inductive coupling-aware coding scheme for 3D network-on-chip," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst. (DFT)*, Amsterdam, The Netherland, Oct. 2014, pp. 92–97.

[23] C. Cassidy *et al.*, "Through silicon via reliability," *IEEE Trans. Device Mater. Rel.*, vol. 12, no. 2, pp. 285–295, Jun. 2012.

[24] L. Qian, Y. Xia, and G. Shi, "Electrical modeling and analysis of a mixed carbon nanotube based differential through silicon via in 3-D integration," *IEEE Trans. Nanotechnol.*, vol. 15, no. 2, pp. 155–163, Mar. 2016.

[25] R. Weerasekera, M. Grange, D. Pamunuwa, H. Tenhunen, and L.-R. Zheng, "Compact modelling of through-silicon vias (TSVs) in three-dimensional (3-D) integrated circuits," in *Proc. IEEE Int. Conf. 3D Syst. Integr. (3DIC)*, San Francisco, CA, USA, Sep. 2009, pp. 1–8.

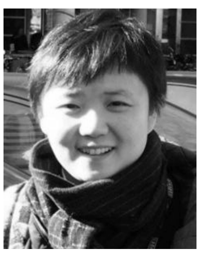
[26] Z. Xu and J. Q. Lu, "Through-strata-via (TSV) parasitics and wideband modeling for three-dimensional integration/packaging," *IEEE Electron Device Lett.*, vol. 32, no. 9, pp. 1278–1280, Sep. 2011.

[27] A. E. Engin and S. R. Narasimhan, "Modeling of crosstalk in through silicon vias," *IEEE Trans. Electromagn. Compat.*, vol. 55, no. 1, pp. 149–157, Feb. 2013.

[28] G. Katti, M. Stucchi, K. de Meyer, and W. Dehaene, "Electrical modeling and characterization of through silicon via for three-dimensional ICs," *IEEE Trans. Electron Devices*, vol. 57, no. 1, pp. 256–262, Jan. 2010.

Xiaole Cui received the B.Sc. degree in communication engineering from Xidian University, Xi'an, China, in 1997, the M.Sc. degree in computer engineering from the Xi'an Institute of Microelectronics, Xi'an, China, in 2000, and the Ph.D. degree in system engineering from Beihang University, Beijing, China, in 2004.

He is currently an Associate Professor with the Peking University Shenzhen Graduate School, Shenzhen, China. His current research interests include VLSI testing, reliability and safety of electronic systems, and dependable computing.



Xiaoxin Cui received the B.Sc. degree in cybernation from Beihang University, Beijing, China, in 2002, and the Ph.D. degree in microelectronic from Peking University, Beijing, China, in 2007.

She is currently an Associate Professor of Peking University, Beijing, China. Her current research interests include low-power IC design and hardware security.

Yewen Ni received the B.Sc. degree in microelectronics from Peking University, Beijing, China in 2015. He is currently pursuing the M.Sc. degree in microelectronic from Peking University.

His current research interests include SoC and NoC design.

Min Miao (SM'16) received the B.Sc. degree from Beihang University, Beijing, China, in 1995, the M.Sc. degree from the Beijing Institute of Vacuum Electronics, Beijing, China, in 1998, and the Ph.D. degree from Peking University (PKU), Beijing, China, in 2004.

He is currently a Professor of Beijing Information Science and Technology University, and also a Guest Professor of PKU. His current research interests include 3D heterogeneous Micro/nano system integration, advanced IC packaging, RF MEMS/NEMS device, and micro-sensor networks.

Dr. Miao is the Secretary–General of the IEEE CPMT Society, Beijing Section.

Jin Yufeng received the Ph.D. degree in physical and optical-electronics engineering from Southeast University, Nanjing, China, in 1999.

He was a Post-Doctor, Associated Professor and Professor with Peking University. He was with the Joinning Group of GINTIC/SIMTech as a Visiting Research Fellow from 2001 to 2004. He has directed the National Key Laboratory of Science and Technology on Micro/Nano Fabrication since 2005. His research interests focus on MEMS sensors and TSV related 3D integration of microsystems and its application systems.