Crosstalk Avoidance Codes for 3D VLSI

Rajeev Kumar Sunil P. Khatri
Department of ECE, Texas A&M University, College Station TX 77843

Abstract— **In 3D VLSI, through-silicon vias (TSVs) are relatively large, and closely spaced. This results in a situation in which noise on one or more TSVs may deteriorate the delay and signal integrity of neighboring TSVs. In this paper, we first quantify the parasitics in contemporary TSVs, and then come up with a classification of crosstalk sequences as 0C, 1C, ... 8C sequences. Next, we present inductive approaches to quantify the exact overhead for 8C, 6C and 4C crosstalk avoidance codes (CACs) for a $3 \times n$ mesh arrangement of TSVs. These overheads for different CACs for a $3 \times n$ mesh arrangement of TSVs are used to calculate the lower bounds on the corresponding overheads for an $n \times n$ mesh arrangements of TSVs. We also discuss an efficient way to implement the coding and decoding (CODEC) circuitry for limiting the maximum crosstalk to 6C. Our experimental results show that for a TSV mesh arrangement driven by inverters implemented in a $22nm$ technology, the coding based approaches yields improvements which are in line with the theoretical predictions.**

I. Introduction and Previous Work

Through Silicon Via (TSV) based 3D stacking technology has garnered a lot of interest from academia and industry over the past few years. It is expected to alleviate the problem of long interconnects in 2D VLSI ICs by providing another dimension for logic to communicate along. It also allows the possibility of stacking ICs implemented in different technologies, to create heterogeneous 3D structures. As a result, much research is being conducted in all aspects of the 3D IC technology. The works of [1] [2] [3] [4] are targeted towards developing TSV manufacturing and die-to-die as well as wafer-to-die bonding technologies. In the physical design space, placement algorithms for TSV based 3D ICs are proposed in [5] [6]. Several research efforts have been devoted to the 3D routing problem as well [7][8]. Solutions for building 3D clock tress, and testing individual die before bonding are proposed in [9], [10].

The use of TSVs results in a significant capacitive coupling between adjacent TSVs owing to their large dimensions and tight spacing. This causes significant pattern-dependent noise artefacts. This crosstalk noise can affect chip timing by slowing down transitions on a switching signal when a large number of its neighbors perform opposite transitions. It can also cause functionality issues by causing large glitches on a static signal when its aggressing neighbors transition. Analytical expressions for parasitics, including TSV-to-TSV coupling capacitances for simple arrangements of TSVs are presented in [11] [12]. However, very little work has been done to provide systematic solutions for reducing crosstalk among TSVs. The study done in [13] shows that crosstalk between TSVs in a full 3D chip is significant, and the authors propose shielding and buffer insertion based solutions to mitigate the problem. The shielding based solution is employed in a post-placement step, to find the candidate TSVs for shielding. Buffer insertion is performed in a post route environment, when accurate timing information is available. They also show that spacing TSVs apart is an inefficient solution for reducing coupling noise. The solutions proposed in [13] are applied late in the design flow (post-placement, post-routing), and work on a case by case basis to reduce capacitive coupling. No theoretical bounds are provided in [13] for the area overhead of shielding or the maximum capacitive coupling on any TSV after applying these solutions. In contrast, in this paper, we discuss correct-by-construction crosstalk canceling codes for a regular 2D mesh of TSVs, to guarantee an upper bound on the amount of capacitive coupling observed by any victim TSV. This is particularly relevant in the scenario where a regular 2D mesh of TSVs connects one die to another die. One example of such a scenario is when the core logic and caches of a microprocessor are implemented on different dies, and a dense 2D mesh of TSVs is used to interconnect them.

The main contributions of this work are:

- We classify crosstalk patterns into different classes based on the maximum crosstalk they incur (8C, 7C, 6C, 5C ... 0C).
- We present exact overhead of codes to reduce inter-TSV crosstalk for a $3 \times n$ mesh of TSVs, and derive lower bounds on the overheads associated with codes for an $n \times n$ mesh of TSVs.
- We show that by extending the 1D crosstalk avoidance codes proposed in [14][15], we can come up with an efficient 2D code which reduces the maximum crosstalk by 25 percent, with an encoder and decoder of quadratic complexity in circuit area and an asymptotic overhead (in terms of number of additional bits) of $\sim$40%. We also demonstrate the improvement in delay of our 6C, 4C and 2C codes over an uncoded scenario, by performing HSPICE [16] simulations using 22nm PTM [17] transistor models.

II. Approach

In this section, we start with a discussion of our TSV parasitic modeling experiments. This is followed by a classification of data patterns on a 2D mesh of TSVs into multiple classes, based on the maximum crosstalk that is experienced by any TSV in the mesh. Then, we derive mathematical bounds (exact values for a $3 \times n$ mesh, and lower bounds for an $n \times n$ mesh) on the Crosstalk Avoidance Code (CAC) overhead for each class. We end by presenting a technique to implement an efficient CODEC for a 6C CAC.

A. Parasitic Modeling of TSVs

Several analytical models for resistance, capacitance and inductance of TSVs have been proposed in the literature. For our work, we performed our own parasitic extraction using the 3D parasitic extraction tool Raphael. Figure 1 shows the dimensions of each TSV in a 3×3 fragment of the 9×9 TSV mesh we used for parasitic extraction. Based on the TSV dimensions projected for the 2015-2018 timeframe in the 2011 edition of the ITRS [18], TSV diameter, TSV height and TSV-to-TSV spacing are taken as 1μm, 50μm and 1μm respectively. Also, each TSV has an $Si0_2$ insulating dielectric layer of width 0.1μm around it. TSVs are assumed to be made of copper (Cu) with a resistivity of 1.6×10^{-8} Ω-m. The dielectric constants of the silicon substrate in which TSVs are placed, and the $Si0_2$ insulating layers are taken as 3.9 and 12, respectively. We assume that the 3D IC comprises of three stacked dies, and only consider the coupling between TSVs.

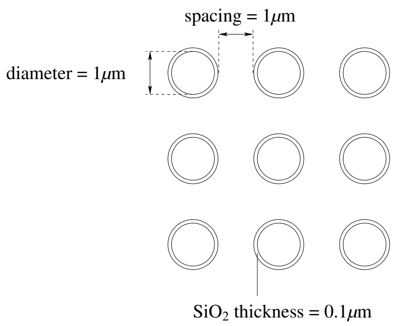


Fig. 1. TSV Mesh Structure used for Parasitic Extraction

Table I shows the coupling capacitances for a mesh of TSVs. The value C_{ij} in any entry (i,j) in the table reports the coupling capacitance of a TSV to another TSV which is i TSVs away horizontally, and j TSVs away vertically. Thus, C_{10} and C_{01} are the coupling capacitances of a TSV to its adjacent TSV to its east (or west) and north (or south) direction respectively. The (0, 0) entry represents the capacitance to ground of any TSV in the mesh. We note that C_{ij}=C_{ji}. It can be also noted from the table that coupling capacitance for a value of i or j greater than 1 is more than two orders of magnitude less than C_{10} and C_{01}, and can be neglected for crosstalk analysis. Also, C_{11} is around $1/5$ of C_{10} and thus has a limited impact on crosstalk. We also extracted inductance, and found that for frequencies up to 10GHz, the oscillation frequency $\frac{1}{\sqrt{LC}}$ of the system was much greater than the operational frequency. Hence, inductance values are not reported. However, inductance parasitics are included in our HSPICE [16] simulations.

	$j=0$	$j=1$	$j=2$	$j=3$	$j=4$
$i=0$	1.2×10^{-19}	1.2×10^{-14}	2.0×10^{-16}	4.8×10^{-18}	1.2×10^{-19}
$i=1$	1.2×10^{-14}	3.0×10^{-15}	1.0×10^{-16}	2.8×10^{-18}	7.4×10^{-20}
$i=2$	2.0×10^{-16}	1.0×10^{-16}	8.3×10^{-18}	3.8×10^{-19}	1.5×10^{-20}
$i=3$	4.8×10^{-18}	2.8×10^{-18}	3.8×10^{-19}	2.8×10^{-20}	2.2×10^{-21}
$i=4$	1.2×10^{-19}	7.4×10^{-20}	1.5×10^{-20}	2.2×10^{-21}	5.1×10^{-22}

TABLE I
CAPACITANCES FOR A 9X9 MESH OF TSVS (IN FARADS)

B. Crosstalk Classification

In this section, we classify the data patterns in a 2D mesh of TSVs into different crosstalk classes. For this classification, we ignore the coupling capacitances C_{ij} (where i, $j \geq 2$). The coupling capacitance C_{11} of a TSV to any of its diagonally neighbouring TSV is roughly $1/5$ of its coupling capacitance (C_{01} or C_{10}) to any of its adjacent TSVs. Clearly, the sum of the coupling capacitances of a TSV to its four diagonally neighbouring TSVs, though not negligible, is less than its coupling capacitance to any of its adjacent TSVs. Thus, the overhead of including the four diagonally neighbouring TSVs in any coding scheme is not justified by the corresponding crosstalk improvement achieved. Hence, for our classification, we do not consider the diagonal neighbours. For any TSV in a data pattern, only its adjacent (East, West, North and South) TSVs are considered. In our results, all the diagonally neighbouring TSVs are assumed to be contributing the worst-case crosstalk by switching in a direction opposite to that of the TSV under consideration.

Consider a $n \times n$ mesh consisting of signals b_{11}, b_{12} $\cdots$ $b_{n-1,n}$ $\cdots$ $b_{n,n}$.

Definition 1: A pattern P is an assignment of values v_{ij} to b_{ij} such that $v_{ij} \in [0,1]$

Let P_k and P_{k+1} be two successive patterns (sequence) applied on the TSV mesh. Let v_{ij}^{k} and v_{ij}^{k+1} be the values of signal b_{ij} in P_k and P_{k+1} respectively. We now define the eight type of crosstalk conditions below:

A sequence is an **8C sequence** if $\exists i,j,k$ such that

$$v_{i,j}^{k} = v_{i-1,j}^{k+1} = v_{i+1,j}^{k+1} = v_{i,j-1}^{k+1} = v_{i,j+1}^{k+1} = v$$
and
$$v_{i,j}^{k+1} = v_{i-1,j}^{k} = v_{i+1,j}^{k} = v_{i,j-1}^{k} = v_{i,j+1}^{k} = \bar{v}$$

Intuitively, it means that from P_k to P_{k+1}, all four of the adjacent neighbors of b_{ij} switch in a direction opposite to its direction of switching, which results in b_{ij} charging a load of $8C$ during its transition, where C is C_{01}, the capacitance between a TSV and any one of its adjacent neighbours.

A sequence is a **7C sequence** if it is not a 8C sequence and $\exists i,j,k$ such that from P_k to P_{k+1}, three of the four adjacent neighbors of b_{ij} switch in the direction opposite to its direction of switching and one of them remains static.

A sequence is a **6C sequence** if it is not an 8C or 7C sequence and $\exists i,j,k$ such that from P_k to P_{k+1}, out of the four neighbors of b_{ij} a) three switch in the direction opposite to its direction of switching and one switches in the same direction, or b) two switch in the direction opposite to its direction of switching and two remain static

A sequence is a **5C sequence** if it is not an 8C or 7C or 6C sequence and $\exists i,j,k$ such that from P_k to P_{k+1}, out of the four neighbors of b_{ij} a) two switch in the direction opposite to its direction of switching, one switches in the same direction and one remains static, or b) one switches in the direction opposite to its direction of switching and three remain static.

A sequence is a **4C sequence** if it is not a 8C or 7C or 6C or 5C sequence and $\exists i,j,k$ such that from P_k to P_{k+1}, out of the four neighbors of b_{ij} a) two switch in the direction opposite to its direction of switching and two switch in the same direction, or b) one switches in the direction opposite to its direction of switching, one switches in the same direction and two remain static, or c) all four remain static.

A sequence is a **3C sequence** if it is not a 8C or 7C or 6C or 5C or 4C sequence and $\exists i,j,k$ such that from P_k to P_{k+1}, out of the four neighbors of b_{ij} a) one switches in the direction opposite to its direction of switching, two switch in the same direction and one remains static, or b) one switches in the same direction and three remain static.

A sequence is a **2C sequence** if it is not a 8C or 7C or 6C or 5C or 4C or 3C sequence and $\exists i,j,k$ such that from P_k to P_{k+1}, out of the four neighbors of b_{ij} a) one switches in the direction opposite to its direction of switching and three switch in the same direction, or b) two switch in the same direction and two remain static.

A sequence is a **1C sequence** if it is not a 8C or 7C or 6C or 5C or 4C or 3C or 2C sequence and $\exists i,j,k$ such that from P_k to P_{k+1}, out of the four neighbors of b_{ij}, three switch in the same direction and one remains static.

A sequence is a **0C sequence** if it is not a 8C or 7C or 6C or 5C or 4C or 3C or 2C or 1C sequence.

Note that for b_{ij} which lie on the the edges of the mesh and have less than four adjacent neighbors, the absent neighbor can be assumed to be switching in the same direction as that of b_{ij} so that it does not contribute any load, and the definitions stated above hold.

C. Crosstalk Avoidance Code (CAC)

A kC ($k = 0,1..8$) Crosstalk Avoidance Code (CAC) for a $n \times n$ mesh is a set of patterns such that no crosstalk sequence of $(k+1)C$ or above occurs on any TSV for a transition between any two patterns in the set. If $T^{kC}(n)$ is the number of valid patterns for a kC CAC, then, number of bits m which can be actually transmitted using this CAC is given by:-

$$m = \lfloor log_2(T^{kC}(n)) \rfloor \tag{1}$$

The overhead (i.e. the number of additional bits used per useful bit) of using this kC CAC is defined as:-

$$Ovh = (n^2 - m)/m \tag{2}$$

Before discussing CACs for different k, we define the following types of patterns:-

8C: A pattern P is a 8C pattern if $\exists i,j$ such that all the four adjacent neighbors of b_{ij} have a value which is the complement of its value.

6C: A pattern P is a 6C pattern if $\exists i,j$ such that exactly three out of the four adjacent neighbors of b_{ij} have a value which is the complement of its value.

4C: A pattern P is a 4C pattern if $\exists i,j$ such that exactly two out of the four adjacent neighbors of b_{ij} have a value which is the complement of its value.

2C: A pattern P is a 2C pattern if $\exists i,j$ such that exactly one out of the four adjacent neighbors of b_{ij} have a value which is the complement of its value.

Lemma 2.1: If a CAC CC does not have any pattern of the type 8C, then it is a 6C CAC.

Proof: By the definition of an 8C crosstalk sequence, *both* patterns P_k and P_{k+1} should be of the type 8C. Since CC has no 8C patterns, it cannot have an $8C$ crosstalk sequence. Similarly, by definition, a 7C crosstalk sequence must have one of the patterns P_k or P_{k+1} of the type 8C. Hence, CC can not have a 7C crosstalk sequence either. Thus, CC is a 6C CAC. ■

By arguing along these lines, it can be shown that:-

1. If a CAC does not have any pattern of the type 8C and 6C, it is a 4C CAC.
2. If a CAC does not have any pattern of the type 8C, 6C and 4C, it is a 2C CAC.

Calculating the exact overhead of 6C, 4C and 2C CACs for a $n \times n$ mesh requires enumerating all the patterns and counting the valid ones, which is computationally infeasible. Hence, in the following subsections, we derive the exact overheads of 6C, 4C and 2C CACs for a $3 \times n$ mesh, and then extend these results to derive the lower bounds on the overheads of 6C, 4C and 2C CACs for a $n \times n$ mesh. A $3 \times n$ mesh is used as the building block because the inductive equations for counting the number of valid patterns are of lower complexity as compared to that of a $4 \times n$ or

$5 \times n$ mesh, while still allowing the calculation of a tighter lower bound as compared to using a $2 \times n$ mesh.

1) Number of 6C CAC Patterns for a $3 \times n$ Mesh (exact): In this section, an exact formulation for the number of valid 6C bit patterns for a $3 \times n$ mesh is derived. A $3 \times n$ mesh refers to a mesh with 3 rows and n columns.The formulation is inductive, and derives expressions for valid 6C bit patterns for a $3 \times n$ mesh from the expressions for valid 6C bit patterns for a $3 \times n-1$ mesh. To simplify notation, when we say that a bit in a pattern has 6C crosstalk, it implies that exactly three out of it's four West, North, East and South neighboring bits have a value which is the complement of it's bit value. The terminology for the expressions is written below:-

$n =$ number of columns in a $3 \times n$ mesh

corner = for a $3 \times n$ mesh, corner refers to the bits with the coordinates (1, n) and (3, n)

$T_6(n) =$ total number of valid 6C patterns for a $3 \times n$ mesh

$T^{0C}(n) =$ total number of valid 6C patterns in which bit (2, n) has a crosstalk of 0C

$T^{2C}(n) =$ total number of valid 6C patterns in which bit (2, n) has a crosstalk of 2C

$T^{4C}(n) =$ total number of valid 6C patterns in which bit (2, n) has a crosstalk of 4C

$T^{6C}(n) =$ total number of valid 6C patterns in which bit (2, n) has a crosstalk of 6C

Based on the definitions, we know that:

$$T_6(n) = T^{0C}(n) + T^{2C}(n) + T^{4C}(n) + T^{6C}(n) \quad (3)$$

We now present the inductive step:

$$T^{0C}(n) = T^{0C}(n-1) + T^{2C}(n-1) + T^{4C}(n-1) + T^{6C}(n-1) \quad (4)$$

$$T^{2C}(n) = 3T^{0C}(n-1) + 3T^{2C}(n-1) + 3T^{4C}(n-1) + 2T^{6C}(n-1) \quad (5)$$

$$T^{4C}(n) = 3T^{0C}(n-1) + 3T^{2C}(n-1) + 3T^{4C}(n-1) + T^{6C}(n-1) \quad (6)$$

$$T^{6C}(n) = T^{0C}(n-1) + T^{2C}(n-1) + T^{4C}(n-1) \quad (7)$$

Figure 2 illustrates the procedure for deriving the inductive equation for $T^{6C}(n)$. Figure 2(a) depicts how a $T^{0C}(n-1)$ pattern is extended to create a single $T^{6C}(n)$ pattern. By definition of $T^{6C}(n)$, bit (2, n) is 6C. Thus, if the value of bit (2, $n-1$) is b, then the values of bits (1, n), (2, n) and (3, n) should be b, $\bar{b}$ and b respectively. Clearly, every $T^{0C}(n-1)$ pattern can be extended to create a single $T^{6C}(n)$ pattern. A similar analysis can be applied to create $T^{6C}(n)$ patterns by extending $T^{2C}(n-1)$ and $T^{4C}(n-1)$ patterns as well.
Figure 2(b) shows how a $T^{6C}(n-1)$ pattern is extended to create a $T^{6C}(n)$ pattern. By definition of $T^{6C}(n-1)$, bit (2, $n-1$) is 6C, and by definition of $T^{6C}(n)$, bit (2, n) is 6C as well. Clearly, this is an invalid condition, since as a crosstalk of 6C on bit (2, n) will result in a crosstalk of 8C on bit (2, $n-1$) as shown in Figure 2(b). Thus, no $T^{6C}(n)$ pattern can be created by extending a $T^{6C}(n-1)$ pattern. A similar analysis can be done to derive the inductive equations for $T^{0C}(n)$, $T^{2C}(n)$ and $T^{4C}(n)$ as well.

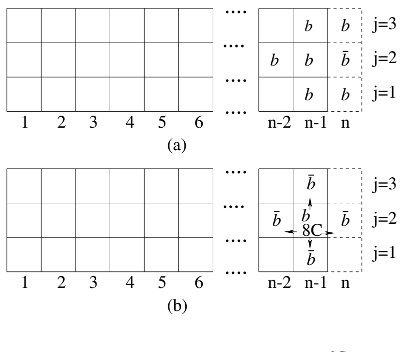


Fig. 2. Illustration of Inductive Derivation for $T^{6C}(n)$ for a 6C CAC

2) Number of 4C CAC Patterns for a $3 \times n$ Mesh (exact): In this section, we derive an exact formulation for the number of valid 4C bit patterns for a $3 \times n$ mesh. The formulation is inductive, and derives expressions for valid 4C bit patterns for a $3 \times n$ mesh from the expressions for valid 4C bit patterns for a $3 \times n-1$ mesh. To simplify notation, when we say that a bit in a pattern has 4C crosstalk, it implies that exactly two out of it's four West, North, East and South neighboring bits have a value which is the complement of it's bit value. The terminology for the expressions is written below:-

$n =$ number of columns in a $3 \times n$ mesh

corner = for a $3 \times n$ mesh, corner refers to the bits with the coordinates (1, n) and (3, n)

$T_4(n) =$ total number of valid 4C patterns for a $3 \times n$ mesh

$T^{0C}(n) =$ total number of valid 4C patterns in which bit (2, n) has a crosstalk of 0C

$T^{2C}_{v_{2,n} \neq v_{2,n-1}}(n) =$ total number of valid 4C patterns in which bit (2, n) has a crosstalk of 2C and the value of bit (2, n) is the complement of the value of bit (2, $n-1$)

$T^{2C}_{v_{2,n} = v_{2,n-1}, Ocorner}(n) =$ total number of valid 4C patterns in which bit (2, n) has a crosstalk of 2C, the value of bit (2, n) is the same as that of bit (2, $n-1$) and none of the corner bits has a crosstalk of 4C

$T^{2C}_{v_{2,n} = v_{2,n-1}, 1corner}(n) =$ total number of valid 4C patterns in which bit (2, n) has a crosstalk of 2C, the value of bit (2, n) is the same as that of bit (2, $n-1$) and exactly one of the corner bits has a crosstalk of 4C

$T^{4C}_{v_{2,n} = v_{2,n-1}, Ocorner}(n) =$ total number of valid 4C patterns in which bit (2, n) has a crosstalk of 4C, the value of bit (2, n) is the same as that of bit (2, $n-1$) and none of the corner bits has a crosstalk of 4C

$T^{4C}_{v_{2,n} = v_{2,n-1}, 1corner}(n) =$ total number of valid 4C patterns in which bit (2, n) has a crosstalk of 4C, the value of bit (2, n) is the same as that of bit (2, $n-1$) and exactly one of the corner bits has a crosstalk of 4C

$T^{4C}_{v_{2,n} = v_{2,n-1}, 2corner}(n) =$ total number of valid 4C patterns in which bit (2, n) has a crosstalk of 4C, the value of bit (2, n) is the same as that of bit (2, $n-1$) and both of the corner bits have a crosstalk of 4C

$T^{4C}_{v_{2,n} \neq v_{2,n-1}, Ocorner}(n) =$ total number of valid 4C patterns in which bit (2, n) has a crosstalk of 4C, the value of bit (2, n) is the complement of the value of bit (2, $n-1$) and none of the corner bits has a crosstalk of 4C

$T^{4C}_{v_{2,n} \neq v_{2,n-1}, 1corner}(n) =$ total number of valid 4C patterns in which bit (2, n) has a crosstalk of 4C, the value of bit (2, n) is the complement of the value of bit (2, $n-1$) and exactly one of the corner bits has a crosstalk of 4C

Based on the definitions, we know that:

$$\begin{aligned} T_4(n) = {} & T^{0C}(n) + T^{2C}_{v_{2,n} \neq v_{2,n-1}}(n) + T^{2C}_{v_{2,n} = v_{2,n-1}, Ocorner}(n) \\ & + T^{2C}_{v_{2,n} = v_{2,n-1}, 1corner}(n) + T^{4C}_{v_{2,n} = v_{2,n-1}, Ocorner}(n) \\ & + T^{4C}_{v_{2,n} = v_{2,n-1}, 1corner}(n) + T^{4C}_{v_{2,n} = v_{2,n-1}, 2corner}(n) \\ & + T^{4C}_{v_{2,n} \neq v_{2,n-1}, Ocorner}(n) + T^{4C}_{v_{2,n} \neq v_{2,n-1}, 1corner}(n) \end{aligned} \quad (8)$$

We now present the inductive step:

$$\begin{aligned} T^{0C}(n) = {} & T^{0C}(n-1) + T^{2C}_{v_{2,n-1} \neq v_{2,n-2}}(n-1) + T^{2C}_{v_{2,n-1} = v_{2,n-2}, Ocorner}(n-1) \\ & + T^{4C}_{v_{2,n-1} = v_{2,n-2}, Ocorner}(n-1) + T^{4C}_{v_{2,n-1} \neq v_{2,n-2}, Ocorner}(n-1) \end{aligned} \quad (9)$$

$$\begin{aligned} T^{2C}_{v_{2,n} \neq v_{2,n-1}}(n) = {} & T^{0C}(n-1) + T^{2C}_{v_{2,n-1} \neq v_{2,n-2}}(n-1) \\ & + T^{2C}_{v_{2,n-1} = v_{2,n-2}, Ocorner}(n-1) + \\ & T^{2C}_{v_{2,n-1} = v_{2,n-2}, 1corner}(n-1) \end{aligned} \quad (10)$$

$$\begin{aligned} T^{2C}_{v_{2,n} = v_{2,n-1}, Ocorner}(n) = {} & T^{2C}_{v_{2,n-1} = v_{2,n-2}, Ocorner}(n-1) \\ & + T^{2C}_{v_{2,n-1} = v_{2,n-2}, 1corner}(n-1) \\ & + 2T^{4C}_{v_{2,n-1} = v_{2,n-2}, Ocorner}(n-1) \\ & + T^{4C}_{v_{2,n-1} = v_{2,n-2}, 1corner}(n-1) \\ & + T^{4C}_{v_{2,n-1} \neq v_{2,n-2}, Ocorner}(n-1) \\ & + T^{4C}_{v_{2,n-1} \neq v_{2,n-2}, 1corner}(n-1) \end{aligned} \quad (11)$$

$$\begin{aligned} T^{2C}_{v_{2,n} = v_{2,n-1}, 1corner}(n) = {} & 2T^{0C}(n-1) + 2T^{2C}_{v_{2,n-1} \neq v_{2,n-2}}(n-1) \\ & + T^{2C}_{v_{2,n-1} = v_{2,n-2}, Ocorner}(n-1) \\ & + T^{4C}_{v_{2,n-1} \neq v_{2,n-2}, Ocorner}(n-1) \end{aligned} \quad (12)$$

$$\begin{aligned} T^{4C}_{v_{2,n}=v_{2,n-1},0corner}(n) \;=\; & T^{4C}_{v_{2,n-1}=v_{2,n-2},0corner}(n-1) \\ & + T^{4C}_{v_{2,n-1}\neq v_{2,n-2},1corner}(n-1) \\ & + T^{4C}_{v_{2,n-1}=v_{2,n-2},2corner}(n-1) \end{aligned} \tag{13}$$

$$\begin{aligned} T^{4C}_{v_{2,n}=v_{2,n-1},1corner}(n) \;=\; & T^{2C}_{v_{2,n-1}=v_{2,n-2},0corner}(n-1) \\ & + T^{2C}_{v_{2,n-1}=v_{2,n-2},1corner}(n-1) \\ & + T^{4C}_{v_{2,n-1}\neq v_{2,n-2},0corner}(n-1) \\ & + T^{4C}_{v_{2,n-1}\neq v_{2,n-2},1corner}(n-1) \end{aligned} \tag{14}$$

$$T^{4C}_{v_{2,n}=v_{2,n-1},2corner}(n) = T^{0C}(n-1) + T^{2C}_{v_{2,n-1}\neq v_{2,n-2}}(n-1) \tag{15}$$

$$\begin{aligned} T^{4C}_{v_{2,n}=v_{2,n-1},0corner}(n) \;=\; & 2T^{0C}(n-1) + 2T^{2C}_{v_{2,n-1}\neq v_{2,n-2}}(n-1) \\ & + T^{2C}_{v_{2,n-1}=v_{2,n-2},0corner}(n-1) \\ & + T^{2C}_{v_{2,n-1}=v_{2,n-2},1corner}(n-1) \end{aligned} \tag{16}$$

$$T^{4C}_{v_{2,n}=v_{2,n-1},1corner}(n) = T^{2C}_{v_{2,n-1}=v_{2,n-2},0corner}(n-1) \tag{17}$$

We will now discuss the procedure for the inductive derivation of $T^{0C}(n)$. By definition, in a $T^{0C}(n)$ pattern, the bit (2, n) is 0C. This implies that all three of its neighboring bits (1, n), (3, n) and (2, $n-1$) have the same value b. We go through each pattern for $3 \times n-1$ mesh one by one, and discuss how they can be extended to generate $T^{0C}(n)$ patterns for a $3 \times n$ mesh.

a) $T^{0C}(n-1)$:- By definition, in a $T^{0C}(n-1)$ pattern, all the bits in the column $n-1$ have the same value. As illustrated in Figure 3(a), each $T^{0C}(n-1)$ pattern can be extended to generate a single $T^{0C}(n)$ pattern.

b) $T^{2C}_{v_{2,n-1}\neq v_{2,n-2}}(n-1)$:- By definition, in a $T^{2C}_{v_{2,n-1}\neq v_{2,n-2}}(n-1)$ pattern, bit (2, $n-1$) is 2C and the value of bit (2, $n-2$) ($\bar{b}$) is the complement of the value of bit (2, $n-1$) (b). This implies that bits (1, $n-1$) and (3, $n-1$) have the value b. Hence, as illustrated in Figure 3(b), each $T^{2C}_{v_{2,n-1}\neq v_{2,n-2}}(n-1)$ pattern can be extended to create a single $T^{0C}(n)$ pattern.

c) $T^{2C}_{v_{2,n-1}=v_{2,n-2},0corner}(n-1)$:- By definition, in a $T^{2C}_{v_{2,n-1}=v_{2,n-2},0corner}(n-1)$ pattern, bit (2, $n-1$) is 2C, the value of bit (2, $n-2$) (b) is same as that of bit (2, $n-1$) (b) and none of the bits (1, $n-1$) and (3, $n-1$) is 4C. Clearly, as illustrated in Figure 3(c), each $T^{2C}_{v_{2,n-1}=v_{2,n-2},0corner}(n-1)$ pattern results in a single $T^{0C}(n)$ pattern.

d) $T^{2C}_{v_{2,n-1}=v_{2,n-2},1corner}(n-1)$:- The difference among $T^{2C}_{v_{2,n-1}=v_{2,n-2},1corner}(n-1)$ and $T^{2C}_{v_{2,n-1}=v_{2,n-2},0corner}(n-1)$ patterns discussed in c) is as follows. In $T^{2C}_{v_{2,n-1}=v_{2,n-2},1corner}(n-1)$, one of the bits (1, $n-1$) and (3, $n-1$) is 4C. This implies that no $T^{0C}(n)$ patterns can be generated from any of the $T^{2C}_{v_{2,n-1}=v_{2,n-2},1corner}(n-1)$ patterns, as illustrated in Figure 3(d), since this would result in 6C crosstalk on (1, $n-1$) or (3, $n-1$). Hence, this term does not appear in Equation 9.

e) $T^{4C}_{v_{2,n-1}=v_{2,n-2},0corner}(n-1)$:- By definition, in a $T^{4C}_{v_{2,n-1}=v_{2,n-2},0corner}(n-1)$ pattern, bit (2, $n-1$) is 4C, the value of bit (2, $n-2$) (b) is same as that of bit (2, $n-1$) (b), and none of the bits (1, $n-1$) and (3, $n-1$) is 4C. Thus, similar to the $T^{2C}_{v_{2,n-1}\neq v_{2,n-2}}(n-1)$ case, each $T^{4C}_{v_{2,n-1}=v_{2,n-2},0corner}(n-1)$ pattern results in a single $T^{0C}(n)$ pattern, as shown in Figure 3(e) .

f) $T^{4C}_{v_{2,n-1}=v_{2,n-2},1corner}(n-1)$:- A $T^{4C}_{v_{2,n-1}=v_{2,n-2},1corner}(n-1)$ pattern is same as a $T^{4C}_{v_{2,n-1}=v_{2,n-2},0corner}(n-1)$ pattern except for the fact that one of the bits (1, $n-1$) and (3, $n-1$) in a $T^{4C}_{v_{2,n-1}=v_{2,n-2},1corner}(n-1)$ pattern is 4C. Clearly, this results in an invalid pattern, as illustrated in Figure 3(f), since (1, $n-1$) or (3, $n-1$) becomes 6C. Hence, this term does not appear in Equation 9.

g) $T^{4C}_{v_{2,n-1}=v_{2,n-2},2corner}(n-1)$:- A $T^{4C}_{v_{2,n-1}=v_{2,n-2},2corner}(n-1)$ pattern is same as a $T^{4C}_{v_{2,n-1}=v_{2,n-2},0corner}(n-1)$ pattern except that both of the bits (1, $n-1$) and (3, $n-1$) in a $T^{4C}_{v_{2,n-1}=v_{2,n-2},2corner}(n-1)$ pattern are 4C. Thus, no $T^{0C}(n)$ pattern can be generated from it, as shown in Figure 3(g).

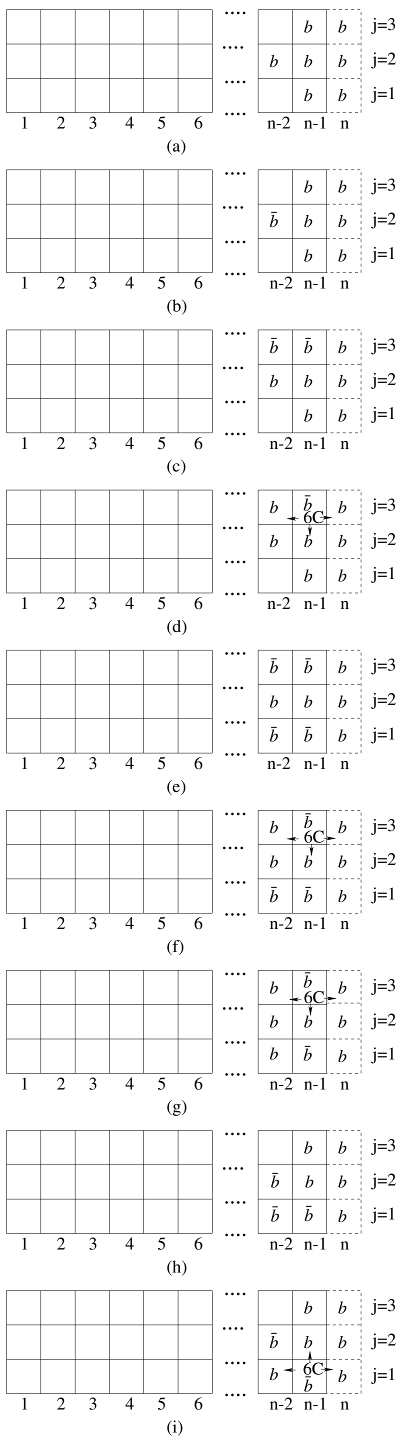


Fig. 3. Illustration of Inductive Derivation for $T^{2C}_{v_{2,n}=v_{2,n-1},1corner}$ for a 4C CAC

h) $T^{4C}_{v_{2,n-1}\neq v_{2,n-2},0corner}(n-1)$:- By definition, in a $T^{4C}_{v_{2,n-1}\neq v_{2,n-2},0corner}(n-1)$ pattern, bit(2, $n-1$) is 4C, the value of bit(2, $n-2$) ($\bar{b}$) is the complement of the value of bit (2, $n-1$) (b) and none of the bits (1, $n-1$) and (3, $n-1$) is 4C. Clearly, each $T^{4C}_{v_{2,n-1}\neq v_{2,n-2},0corner}(n-1)$ pattern can be extended to generate a single $T^{0C}(n)$ pattern, as illustrated in Figure 3(h).

i) $T^{4C}_{v_{2,n-1}\neq v_{2,n-2},1corner}(n-1)$:- In a $T^{4C}_{v_{2,n-1}\neq v_{2,n-2},1corner}(n-1)$ pattern, one of the bits (1, $n-1$) and (3, $n-1$) is 4C. This implies that no $T^{0C}(n)$ patterns can be generated from it, since one corner would become 6C after the induction step. This is shown in Figure 3(i).

Based on the discussions in paragraph a) through i) above, Equation 9 is derived by adding the terms in paragraphs a), b), c), e) and h). The same approach as discussed for $T^{0C}(n)$ patterns can be used to derive expressions for Equations 10 through 17.

3) Number of 2C CAC Patterns for a $3 \times n$ Mesh (exact): In this section, we present an exact formulation for the number of valid 2C patterns for a $3 \times n$ mesh. The formulation is inductive, and derives expressions for valid 2C patterns for a $3 \times n$ mesh from the expressions for valid 2C patterns for a $3 \times n-1$ mesh. A bit in a bit pattern has 2C crosstalk, if exactly one of it's four West, North, East and South neighboring bits has a value which is the complement of it's bit value. The terminology we use is given below:

$n =$ number of columns in a $3 \times n$ mesh

corner = for a $3 \times n$ mesh, corner refers to the bits with the coordinates (1, n) and (3, n)

$T_2(n) =$ total number of valid 2C patterns for a $3 \times n$ mesh

$T^{0C}_{0corner}(n) =$ total number of valid 2C patterns in which bit (2, n) has a crosstalk of 0C and none of the corner bits has a crosstalk of 2C

$T^{0C}_{1corner}(n) =$ total number of valid 2C patterns in which bit (2, n) has a crosstalk of 0C and only one of the two corner bits has a crosstalk of 2C

$T^{0C}_{2corner}(n) =$ total number of valid 2C patterns in which bit (2, n) has a crosstalk of 0C and both of the two corner bits have a crosstalk of 2C

$T^{2C}_{v_{2,n} \neq v_{2,n-1}}(n) =$ total number of valid 2C patterns in which bit (2, n) has a crosstalk of 2C and the value of bit (2, n) is the complement of the value of bit (2, $n-1$)

$T^{2C}_{v_{2,n} = v_{2,n-1}}(n) =$ total number of valid 2C patterns in which bit (2, n) has a crosstalk of 2C and the value of bit (2, n) is same as that of bit (2, $n-1$)

Based on the definitions, we know that:

$$T_2(n) = T^{0C}_{0corner}(n) + T^{0C}_{1corner}(n) + T^{0C}_{2corner}(n) + T^{2C}_{v_{2,n} \neq v_{2,n-1}}(n) + T^{2C}_{v_{2,n} = v_{2,n-1}}(n) \tag{18}$$

We now present the inductive step:

$$T^{0C}_{0corner}(n) = T^{0C}_{0corner}(n-1) + T^{2C}_{v_{2,n-1} \neq v_{2,n-2}}(n-1) + T^{2C}_{v_{2,n-1} = v_{2,n-2}}(n-1) \tag{19}$$

$$T^{0C}_{1corner}(n) = 0 \tag{20}$$

$$T^{0C}_{2corner}(n) = 0 \tag{21}$$

$$T^{2C}_{v_{2,n} != v_{2,n-1}}(n) = T^{0C}_{0corner}(n-1) \tag{22}$$

$$T^{2C}_{v_{2,n} = v_{2,n-1}}(n) = T^{2C}_{v_{2,n-1} = v_{2,n-2}}(n-1) \tag{23}$$

Due to the space limitations, we do not discuss the derivation of inductive Equations 19 through 23. A procedure similar to that used in Sections II-C.1 and II-C.2 for deriving the inductive equations is used in this case as well.

Having presented exact induction based formulae for 6C, 4C and 2C crosstalk for a $3 \times n$ mesh of TSVs, we now present bounds for a $n \times n$ mesh of TSVs, for 6C, 4C and 2C CACs.

4) Lower Bound on Overhead of a 6C CAC for a $n \times n$ Mesh: A lower bound on the overhead of a 6C CAC for an $n \times n$ mesh can be calculated from an upper bound on the number of 6C CAC patterns for the same mesh, by using Equation 2. To calculate an upper bound on the number of 6C CAC patterns for an $n \times n$ mesh, we note that an $n \times n$ mesh can be generated by arranging (n/3) $3 \times n$ meshes one of the top of other. Clearly, (n/3) times the number of 6C CAC patterns for a $3 \times n$ mesh upper bounds the number of 6C CAC patterns for an $n \times n$ mesh as some patterns will become invalid at the boundaries between any two adjacent $3 \times n$ meshes. Thus, the exact overhead for a $3 \times n$ mesh is a lower bound the overhead for an $n \times n$ mesh.

5) Lower Bound on Overhead of a 4C and 2C CAC for a $n \times n$ Mesh: The argument presented above can applied in the case of 4C and 2C CAC patterns for an $n \times n$ mesh, to deduce that the overhead of 4C and 2C CAC patterns for a $3 \times n$ mesh are lower bounds on the overhead of 4C and 2C CAC patterns for an $n \times n$ mesh respectively.

D. A Simple 6C CAC Coding Scheme for an $n \times n$ Mesh

The implementation of a CAC for an $n \times n$ mesh requires an encoder at the transmitting end and a decoder at the receiving end. A simple-minded way to build an encoder and decoder would be to enumerate all the valid 6C patterns for an $n \times n$ mesh, and map every valid 6C pattern to an input binary word. The mappings can be chosen in a manner so as to minimize the circuit size. However, in general, the size of such an encoder and decoder circuitry increases exponentially with increasing n [15]. Also, enumerating all valid 6C patterns is feasible only for small values of n. To the best of the author's knowledge, there is no encoding/decoding scheme for a 4C or 2C CAC for an $n \times n$ mesh, which results in an encoder/decoder circuit size complexity that grows polynomially with n. This remains an open problem. However, a polynomial complexity encoding/decoding scheme for a 6C code can be implemented by observing that if every column in an $n \times n$ mesh is encoded using a one dimensional 2C CAC, then the resulting CAC for the $n \times n$ mesh is a 6C CAC. A coding scheme for a 2C CAC for a one dimensional bus, based on the Fibonacci Number System (FNS), was proposed in [15]. It results in an encoder and decoder with a circuit size complexity which grows as n^2. Figure 4 shows an example of how the FNS can be used to encode a one dimensional bus to ensure a maximum crosstalk of 2C. In the FNS based representation, the weight of any bit is equal to the sum of the weights of two preceeding bits. The binary represenation of 27 results in the bit with the weight of 4 experiencing a crosstalk of 4C, whereas the FNS based representation results in a maximum crosstalk of 2C on any bit. Figure 4 presents the binary (top) and FNS based representation (middle and bottom) of the number 26. In the binary representation, the bit with weight 2 experiences a crosstalk of 4C. The FNS based representation (Figure 4-middle) has a crosstalk of 4C occuring on the bit with the weight of 5. However, this FNS based representation can be transformed (Figure 4-bottom) into an equivalent representation, which has a maximum crosstalk of 2C occurring on any bit, by setting the two highlighted bits to 0 and the next higher bit to 1.

By using the FNS based one dimensional 2C encoding along each of the n columns of an $n \times n$ mesh, a coding scheme with the circuit size complexity of n^3 can be implemented. The asymptotic overhead for such a coding scheme is 0.44 [15], but for n up to 20, the maximum overhead for any n was found to be around 0.35 [15]. The encoder/decoder logic area overhead for such a coding scheme, extrapolated to 22*nm* technology from the 90*nm* technology area numbers published in [15], is 386μm^2 for a 10×10 mesh.

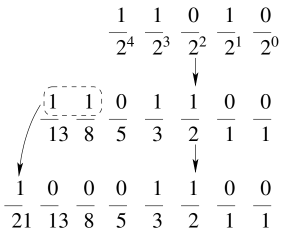
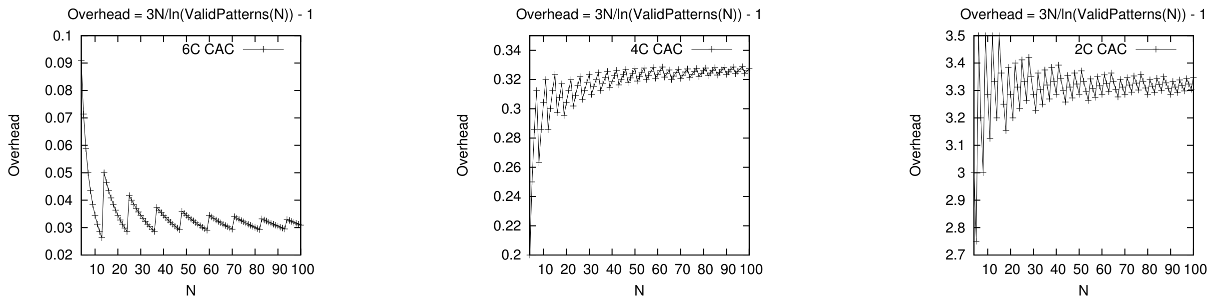


Fig. 4. Fibonacci Number System Based Coding for a 2C CAC on a one Dimensional Bus

III. Experimental Results

We implemented the inductive equations for the number of 6C, 4C and 2C CAC patterns for a $3 \times n$ mesh, using n=2 as the base case. The base case is derived by enumerating the patterns. Figure 5 a), b) and c) show the plots of overhead versus n for 6C, 4C and 2C CAC for a $3 \times n$ mesh respectively. The asymptotic overheads for 6C, 4C and 2C CAC are 3%, 33% and 335% respectively. It should be noted that these overheads are lower bounds on the overhead of corresponding CACs on an $n \times n$ mesh. The extremely high overhead of 335% for 2C CAC makes them unsuitable for any practical use.

To validate our theoretical predictions that the worst delay through an inverter (buffer) driving a TSV is proportional to the maximum coupling capacitance, we extracted the parasitics for a 9×9 mesh of TSVs by using Raphael [19], and simulated the 8C, 6C, 4C, 2C and 0C crosstalk sequences for the center TSV (victim) in the mesh by applying different logic transitions to its adjacent East, West, North and South TSVs. All the other TSVs are made to cause worst crosstalk on the victim by always making them switch in a direction opposite to that of the victim. All circuit simulations are done using HSPICE [16]. The coupling capacitance values used are as reported in Table I. Each TSV is driven by a chain of 3 cascaded inverters. The input slew used is 10ps, and sizes of the first, second and third inverter are $3\times$, $10\times$ and $30\times$ of the minimum size respectively. These sizes are selected so as to get a maximum slewrate of



(a) 6C (b) 4C (c) 2C

Fig. 5. Overhead for 6C, 4C and 2C CACs for a 3×n mesh

$100ps$ at the output of third inverter which drives the TSV. This allows us to operate the 3D stack at $\sim 2GHz$. 22nm Predictive Technology Model (PTM) [17] is used for simulations. Table II presents the delay from the input of third inverter to the far end of the TSV for different crosstalk sequences. The 0C sequence corresponds to the case of no capacitive coupling between the center TSV and its four adjacent neighbors. The incremental delay (d_{xtalk}) from the 0C to the 8C crosstalk sequence is $42.1ps$. d_{xtalk} represents the maximum increase in delay due to crosstalk from the adjacent TSVs. Table III shows the incremental delays from 0C to 2C, 2C to 4C, 4C to 6C and 6C to 8C crosstalk sequences, and their contribution to d_{xtalk}. Clearly, the results show that the increase in delay due to crosstalk is proportional to the increase in coupling capacitance. 2C CAC patterns achieve the highest improvement in delay (the 0C CAC is a degenerate code), albeit at a much higher overhead.

Crosstalk Sequence	Delay (ps)
8C	61.8
6C	51.9
4C	41.6
2C	30.2
0C	19.7

TABLE II
DELAYS FOR DIFFERENT CROSSTALK SEQUENCES

Transition	Incremental Delay (ps)	% of d_{xtalk}
6C CAC→8C CAC	13.0	23.4
4C CAC→6C CAC	14.5	24.4
2C CAC→4C CAC	15.8	27.0
0C CAC→2C CAC	9.5	24.8

TABLE III
INCREMENTAL DELAYS BETWEEN CACS

In practice, the complex design of today are heavily pipelined. In case of such systems, the maximum TSV mesh data transfer rate is significantly improved by using coding. In such pipelined systems, the encoding/decoding delays are hidden in the additional pipelined stages, provided the delays are smaller than the clock period.

Owing to the small resistance of a TSV and the fact that it mainly has a capacitive nature, the inverters can be sized up further to reduce d_{xtalk} at the cost of power. However, this will require a longer, more power-hungry chain of inverters, which may offset the delay improvement achieved. Also, with decreasing TSV diameter and TSV-to-TSV spacing in future technologies, the TSV resistance and coupling capacitance will increase, resulting in a more distributed RC parasitic behavior of a TSV, making it even more important to use coding on the 2D mesh of TSVs.

IV. CONCLUSIONS

In this paper, we present techniques to alleviate crosstalk in 3D VLSI. The through-silicon vias (TSVs) used in 3D VLSI are significantly capacitive, and therefore the switching of neighboring TSVs can degrade the speed as well as signal integrity of a victim TSV. After quantifying 3D parasitics in this context, we classify crosstalk patterns as 8C, 7C, 6C, ... 0C patterns. We present inductive approaches to quantify the exact overhead for 6C, 4C and 2C crosstalk avoidance codes (CAC) for a $3 \times n$ mesh arrangement of TSVs. We also provide lower bounds of the overheads for an $n \times n$ mesh arrangement of TSVs. An efficient method to realize the 6C CODECs is presented as well. Our results show that the asymptotic overheads for 6C, 4C and 2C codes for a $3 \times n$ bus are 3%, 33% and 335% respectively. By means of HSPICE experiments conducted on the 22nm technology node, we demonstrate that our coding based improvements match with simulation results.

REFERENCES

[1] E. Beyne, P. De Moor, W. Ruythooren, R. Labie, A. Jourdain, H. Tilmans, D. Tezcan, P. Soussan, B. Swinnen, and R. Cartuyvels, "Through-silicon via and die stacking technologies for microsystems-integration," in *Electron Devices Meeting, 2008. IEDM 2008. IEEE International*, pp. 1 –4, Dec. 2008.

[2] F. Liu, R. Yu, A. Young, J. Doyle, X. Wang, L. Shi, K.-N. Chen, X. Li, D. Dipaola, D. Brown, C. Ryan, J. Hagan, K. Wong, M. Lu, X. Gu, N. Klymko, E. Perfecto, A. Merryman, K. Kelly, S. Purushothaman, S. Koester, R. Wisnieff, and W. Haensch, "A 300-mm wafer-level three-dimensional integration scheme using tungsten through-silicon via and hybrid Cu-adhesive bonding," in *Electron Devices Meeting, 2008. IEDM 2008. IEEE International*, pp. 1 –4, Dec. 2008.

[3] J. Van Olmen, A. Mercha, G. Katti, C. Huyghebaert, J. Van Aelst, E. Seppala, Z. Chao, S. Armini, J. Vaes, R. Teixeira, M. Van Cauwenberghe, P. Verdonck, K. Verhemeldonck, A. Jourdain, W. Ruythooren, M. de Potter de ten Broeck, A. Opdebeeck, T. Chiarella, B. Parvais, I. Debusschere, T. Hoffmann, B. De Wachter, W. Dehaene, M. Stucchi, M. Rakowski, P. Soussan, R. Cartuyvels, E. Beyne, S. Biesemans, and B. Swinnen, "3D stacked IC demonstration using a through silicon via first approach," in *Electron Devices Meeting, 2008. IEDM 2008. IEEE International*, pp. 1 –4, Dec. 2008.

[4] D. Chen, W. Chiou, M. Chen, T. Wang, K. Ching, H. Tu, W. Wu, C. Yu, K. Yang, H. Chang, M. Tseng, C. Hsiao, Y. Lu, H. Hu, Y. Lin, C. Hsu, W. Shue, and C. Yu, "Enabling 3D-IC foundry technologies for 28 nm node and beyond: through-silicon-via integration with high throughput die-to-wafer stacking," in *Electron Devices Meeting (IEDM), 2009 IEEE International*, pp. 1 –4, Dec. 2009.

[5] B. Goplen and S. Sapatnekar, "Efficient thermal placement of standard cells in 3D ICs using a force directed approach," in *Computer Aided Design, 2003. ICCAD-2003. International Conference on*, pp. 86 – 89, Nov. 2003.

[6] K. Athikulwongse, A. Chakraborty, J.-S. Yang, D. Pan, and S. K. Lim, "Stress-driven 3D-IC placement with TSV keep-out zone and regularity study," in *Computer-Aided Design (ICCAD), 2010 IEEE/ACM International Conference on*, pp. 669 –674, Nov. 2010.

[7] T. Zhang, Y. Zhan, and S. S. Sapatnekar, "Temperature-aware routing in 3D ICs," in *in Proceedings of the Asia-South Pacific Design Automation Conference*, pp. 309–314, 2006.

[8] C.-J. Chang, P.-J. Huang, T.-C. Chen, and C.-N. J. Liu, "ILP-based inter-die routing for 3D ICs," in *Proceedings of the 16th Asia and South Pacific Design Automation Conference*, ASPDAC '11, (Piscataway, NJ, USA), pp. 330–335, IEEE Press, 2011.

[9] X. Zhao, D. Lewis, H.-H. Lee, and S. K. Lim, "Pre-bond testable low-power clock tree design for 3D stacked ICs," in *Computer-Aided Design - Digest of Technical Papers, 2009. ICCAD 2009. IEEE/ACM International Conference on*, pp. 184 –190, Nov. 2009.

[10] X. Zhao, J. Minz, and S. K. Lim, "Low-power and reliable clock network design for through-silicon via (TSV) based 3D ICs," *Components, Packaging and Manufacturing Technology, IEEE Transactions on*, vol. 1, pp. 247 –259, Feb. 2011.

[11] I. Savidis and E. Friedman, "Closed-form expressions of 3-D via resistance, inductance, and capacitance," *Electron Devices, IEEE Transactions on*, vol. 56, pp. 1873 –1881, Sept. 2009.

[12] D. H. Kim, S. Mukhopadhyay, and S. K. Lim, "Fast and accurate analytical modeling of through-silicon-via capacitive coupling," *Components, Packaging and Manufacturing Technology, IEEE Transactions on*, vol. 1, pp. 168 –180, Feb. 2011.

[13] C. Liu, T. Song, J. Cho, J. Kim, J. Kim, and S. K. Lim, "Full-chip TSV-to-TSV coupling analysis and optimization in 3D IC," in *Design Automation Conference (DAC), 2011 48th ACM/EDAC/IEEE*, pp. 783 –788, June 2011.

[14] C. Duan, C. Zhu, and S. Khatri, "Forbidden transition free crosstalk avoidance CODEC design," in *Design Automation Conference, 2008. DAC 2008. 45th ACM/IEEE*, pp. 986 –991, June 2008.

[15] C. Duan, V. Calle, and S. Khatri, "Efficient on-chip crosstalk avoidance CODEC design," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 17, pp. 551 –560, April 2009.

[16] HSPICE http://www.synopsys.com/products/mixedsignal/hspice/hspice.html.

[17] PTM http://www.eas.asu.edu/~ptm.

[18] "The International Technology Roadmap for Semiconductors." `http://public.itrs.net/`, 2003.

[19] "Raphael Interconnect Analysis Tool: User's Guide,"