Capacitive Coupling Mitigation for TSV-based 3D ICs

Ashkan Eghbal, Pooria M.Yaghini, and Nader Bagherzadeh
Center for Pervasive Communications and Computing
Department of Electrical Engineering and Computer Science, University of California, Irvine
Email:{aeghbal, pooriam, nader}@uci.edu

***Abstract*—TSV-to-TSV capacitive coupling has large disruptive effects on timing requirements of the circuit. The latency effect of TSV-to-TSV capacitive coupling for different characteristics of a TSV using circuit-level model is presented in this article. Two coding approaches are proposed to mitigate capacitive parasitic effects by adjusting the current flow pattern for any given $n \times n$ mesh of TSV arrangement to reduce the number of 8C/7C parasitic capacitance. The experimental results proves the efficacy of the proposed coding methods.**

I. Introduction

IC transistors have reached the fundamental limits of miniaturization at the atomic levels; Three-Dimensional (3D) IC has been proposed as an emerging technology to keep Moore's Law ticking by packing a great deal of functionality into small die area. A 3D IC is actually composed of multiple tiers of thinned-active Two-Dimensional (2D) ICs which are stacked, bonded, and electrically connected with vertical vias formed called Through-Silicon Vias (TSVs) [1]. TSVs support higher bandwidth and core integration with lower power consumption and latency [1]. However, the impact of sub-micron TSVs on future 3D ICs is still under study [2]. TSVs are known as noticeable sources of coupling noise that deteriorate the Signal Integrity (SI) of 3D IC layouts in literature [3]. This is an effect of fine pitch integration on conductive silicon substrate in smaller form factor of 3D ICs [4], introducing TSV coupling as a 3D IC design challenge. The term TSV coupling refers to capacitive and inductive couplings among neighboring TSVs. Electric field results in capacitance coupling and magnetic field is the source of inductive coupling. The capacitance coupling between TSVs depends on the permittivity of the oxide, TSV geometry, the arrangement of surrounding TSVs, and body contacts places. Inductive coupling is disruptive in higher operation frequencies (>5GHz) while capacitive coupling is considerable with application in lower range of frequencies (<5GHz). TSV-to-TSV Capacitive Coupling (TTCC) is considered in this article as one of the major issues of 3D IC design. We have previously proposed solely TSV-to-TSV inductive coupling aware coding for both low and high bandwidth application [5], [6], but they are not proper for TTCC. Based on our analysis, the TSV configuration pattern which is the most interference-free in inductive and capacitive coupling analyses are different.

Many crosstalk-aware methods have been suggested for 2D designs [7]–[9], they are not expandable to 3D designs due to different physical characteristics of 2D wires and TSVs. Furthermore, the number of neighbor TSVs in 3D architectures is not the same as the number of neighbor wires in 2D designs. Many research groups have addressed the effect of capacitive TSV coupling on delay and SI of circuits and interconnects in 3D ICs [10]–[12], but limited systematic solutions have been proposed to reduce TTCC effects in 3D ICs. Five solutions are suggested in [10] to reduce the coupling including: increasing TSV distances, shielding the victim TSVs, inserting buffers at the victim net, decreasing the driver size at the aggressor net, and increasing the load at both victim and aggressor net. The last two approaches have negative implications for timing performance, and others need high effort at post-design time. A crosstalk avoidance coding for 3D VLSI has been proposed in [13]. This coding is suitable for a 2D array of $3 \times n$ by limiting some specific data bit patterns for transmission, but this method is not scalable for larger mesh of TSVs. This is because the computation process of their algorithm is increased exponentially for larger number of n, as they need to enumerate all the patterns and count the valid ones. 3DLAT [14] has been proposed with the goal of capacitive crosstalk reduction and power consumption overhead minimization in the TSV array. In this method they suggest to encode the input data to a codeword which contains limited number of 1's in every 3×3 TSV array. However they have not considered the vertical and horizontal overlap among 3×3 TSV arrays in their proposed technique. Furthermore the overhead of their design for larger mesh of TSVs is not negligible. In this paper, two TTCC mitigation coding for small and large 3D IC bandwidths with affordable overhead are proposed. Our proposed methods are scalable to support any $n \times n$ number of TSVs without limiting any specific data patterns. The main contributions of this work are:

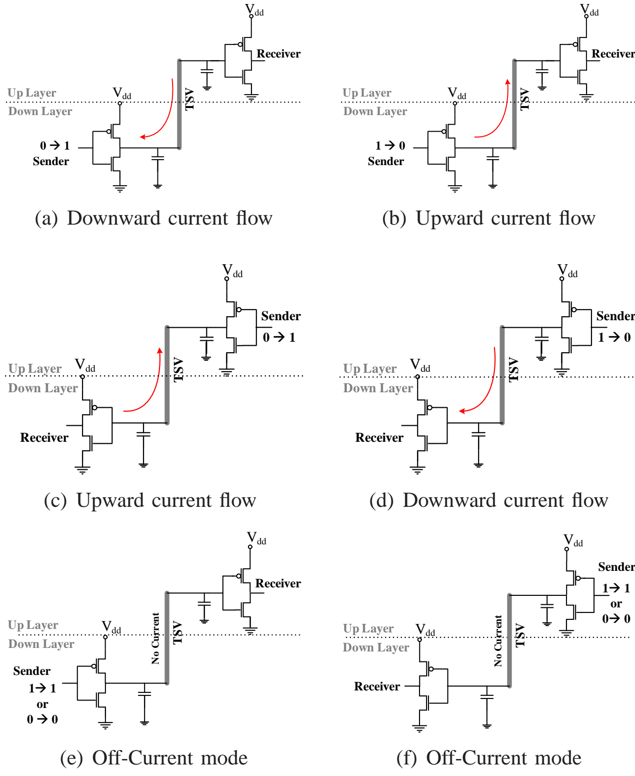
- To introduce the worst class of TTCC by a circuit-level analysis.
- To devise a baseline and an enhanced system-level method to mitigate the TTCC effect for smaller and larger bandwidths.
- To evaluate the efficiency and overhead of both proposed methods.

II. TTCC Characterization

In this section the current flow of TSVs and parasitic capacitance coupling is first discussed and then the presented classes of parasitic capacitanceare compared using circuit-level model.

A. Current flow in TSVs

In order to characterize the effects of capacitive coupling between TSVs used in a CMOS digital circuit, it is first necessary to characterize the direction of current in a given TSV, based on the transmission direction and the the sequential data bit values. Fig. 1 illustrates six possible cases in which a TSV has three possible current directions including: downward, upward, and no-current. For the cases where the data is transmitted from an upper to a lower layer, Fig. 1(a) shows that the TSV current is conducted downward if its voltage makes a high-to-low transition; Fig. 1(b) shows that the TSV current is conducted upward if its voltage



(a) Downward current flow (b) Upward current flow

(c) Upward current flow (d) Downward current flow

(e) Off-Current mode (f) Off-Current mode

Fig. 1. Current flow direction in TSV.

makes a low-to-high transition. For the cases where the data is transmitted from an upper to a lower layer, the currents are in the opposite direction of those indicated in Fig. 1(a) and Fig. 1(b), as shown in Fig. 1(c) and Fig. 1(d), respectively. If there is no output data transition on the TSV, then no current will conduct, as shown in Fig. 1(e) and Fig. 1(f). In the rest of this article, a TSV which does not have any current flow is called an inactive TSV. The ⊙, ⊗, and ○ symbols represent active TSV with upward, downward current flow directions, and inactive TSV, respectively.

The total capacitive coupling voltage on the victim TSV is equal to the sum of voltages coupled by each aggressor on the victim TSV [13]. Furthermore, the value of TTCC depends on the distance of each pair of TSVs. It is proven that the value of TTCC between a victim TSV and its diagonal neighbors is roughly 1/5 of the value of TTCC between a victim TSV and its adjacent TSVs [13]. So only adjacent neighbor TSVs are considered in this experiment for the sake complexity of the proposed algorithms. We use the same 9 presented classification of TTCC for each element of mesh of TSV as discussed in [13] to discuss our proposed methods. In this classification the severity of capacitive coupling voltage between each pair of TSVs is represented by:

- **0C** if they both have the same direction or they are both inactive TSVs (like ⊗⊗ or ○○).
- **1C** if one of them is inactive and the other is active (like ○⊗ or ⊙○).
- **2C** if they have reverse current flow (like ⊙⊗).

With this definition the maximum capacitive coupling voltage on a victim TSV in this representation is 8C. Neglecting the diagonal neighbor TSVs, there are $3^5 = 243$ possible TSV configurations of active (upward or downward) and inactive TSVs, while many of them are similar as long as the capacitive coupling value is concerned. Table I summarizes all possible TSV configuration patterns with their occurrence frequency for each capacitive coupling class from the range of 0C to 8C. Although table I shows the configuration 3C parasitic capacitance is most frequent, this does not necessarily mean that the probability of occurrence of 3C parasitic capacitance is the highest as this will be strongly dependent on the application and data-transfer flow.

B. *Circuit-level model*

As discussed earlier, TSV coupling deteriorates the delay and SI of neighboring TSVs. To evaluate the effect of TTCC on timing of 3D IC, a victim and its four adjacent neighbors are modeled in HSPICE. In simulations, the top and bottom end of each TSV is connected to a D flip-flop to monitor the effect of TTCC on their sampling time. Predictive Technology Model (PTM) [15] FinFET transistor models are employed to implement D flip-flops in this experiment. The severity of each class of TTCC from 0C to 8C are reported for different range of process technologies (20nm to 7nm) in Fig. 2. The radius, length, pitch, and t_{ox} parameter values of TSVs in this experiment are $5\mu m$, 15μ, $21\mu m$, and $2\mu m$, respectively extracted from reported values in ITRS reports [16].

Timing Violation (TV) is defined as the additional delay, relative to the clock period, caused by the parasitic capacitive coupling which is given by:

$$TV = \frac{APD - NPD}{T_{clk}} = f_{clk}\,(APD - NPD) \qquad (1)$$

where APD refers to actual path delay (when there is CTTC), NPD refers to nominal path delay (when there is no CTTC), T_{clk} is the clock period, and f_{clk} is the clock frequency. According to the experimental results, the effects of 8C and 7C parasitic capacitance are more critical than the other types, specially in higher frequencies, which are targeted in the proposed TSV-to-TSV Capacitive Coupling Mitigation Algorithm (TCMA).

III. Proposed Coding Approaches

The main goal of our proposed TCMA , is to reduce the probability of 7C and 8C parasitic capacitance emergence by adjusting the transmitting data bits. Mitigation is chosen in this experiment since eliminating all 7C and 8C parasitic capacitance imposes a complex architecture which is not scalable for any size of TSV meshes [13]. In this Section, our baseline TCMA is discussed

Table I
TSV-to-TSV Capacitive Coupling Categorization

Types	0C	1C	2C	3C	4C	5C	6C	7C	8C
Sample pattern	⊙ ⊙ ⊙ ⊙ ⊙	⊙ ⊙ ⊙ ⊙ ○	⊙ ⊙ ⊙ ⊙ ⊗	⊙ ⊙ ⊙ ○ ⊗	⊙ ⊙ ⊙ ⊗ ⊗	○ ⊙ ⊙ ⊗ ⊗	⊗ ⊙ ⊙ ⊗ ⊗	⊗ ○ ⊙ ⊗ ⊗	⊗ ⊗ ⊙ ⊗ ⊗
Occurrence frequency	3	16	44	64	54	32	20	8	2
Occurrence probability	0.01	0.07	0.18	0.26	0.22	0.13	0.08	0.03	0.01

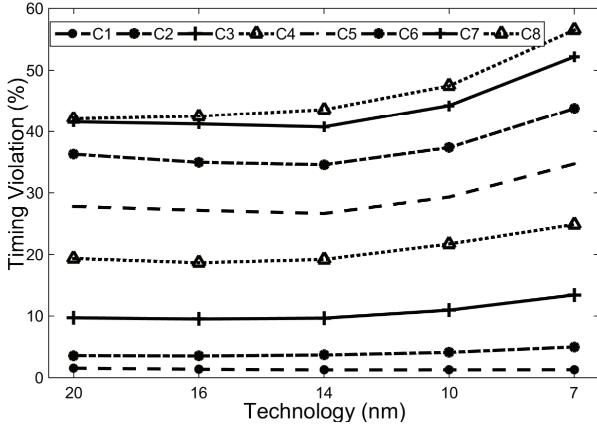


Fig. 2. Severity of TTCC of each classes

for small interconnections and then issues of the baseline method for large interconnections are highlighted. Finally, the enhanced TCMA is presented which supports large mesh of TSVs.

A. Baseline TCMA

The TTCC is data-dependent as described in Section II. The basic idea of the baseline TCMA is to encode, if necessary, the consecutive data bits transmitting over the TSVs in order to mitigate the frequency of 7C and 8C parasitic capacitance. This method does not limit any pattern of data transmission bits by encoding them before transmission and decoding them in receiver side, if needed. The inversion operation is chosen as a simple but light and efficient practical coding method in TCMA in order to keep the overhead low, while mitigating TTCC noise. In a mesh of TSVs, a single bit per row is needed in TCMA to determine whether the inversion process is needed or not at the receiver side. TCMA stores the last transmitted data bit of each TSV and compares it with the available data bit which has not been transmitted yet. The current direction matrix of all TSVs is generated by comparing these successive data bits as described in Section II. Then the parasitic capacitance for each of TSVs are calculated based on the the current flow of its neighbor TSVs. Each row of 2D array of TSVs including 8C or 7C parasitic capacitance values is nominated for the data encoding process. By encoding the ready to transmit data bits, 8C parasitic capacitance will be 4C and 7C parasitic capacitance will be 1C or 2C in this method.

B. Enhanced TCMA

Although the baseline TCMA reduces the quantity of 8C and 7C parasitic capacitance values, but it may have some undesirable

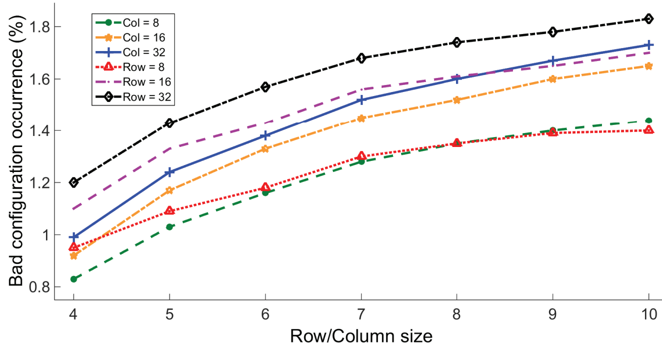


Fig. 3. Probability of bad configuration occurrence

Table II
CURRENT FLOW OF TSVS BEFORE AND AFTER ENCODING

Sent data	Ready to send data	CF_{bi}	CF_{ai}
0	0	○	⊙
0	1	⊙	○
1	0	⊗	○
1	1	○	⊗

side effects by converting a row of data bits. For some special data patterns, converting a single row of data bits may generate unexpected 8C or 7C parasitic capacitance values, which happens in a mesh of TSV with more than 3 rows or 6 columns. We refer to these special cases as bad configuration in the rest of this article.

A bad configuration is a subset of TSV mesh which potentially generates unexpected 8C or 7C parasitic capacitance values by converting a single row of data bits. In more details, the row encoding affects the other data bits of the same row or the data bits in predecessor or successor rows in 2D matrix of TSVs. However, since the probability of bad configuration occurrence is low, specially for smaller matrices of TSVs, the baseline coding is still efficient for smaller data buses (less than 64 bits) which are considered in 3D Network-on-Chip (3D NoC applications). Fig. 3 shows the probability of bad configuration occurrences in different mesh size of TSVs. This experiment is done by running the Monte Carlo simulations for 10000 iterations for different row/column dimensions. According to experimental results the reported percentage of bad configuration for all of the experimented dimensions is less than 2%.

However, the baseline coding is not scalable for larger data buses (more than 64 bits) which are applied in 3D memory applications according to the increasing trend in Fig. 3. The enhanced version of TCMA is devised for these sorts of application to make sure the encoding process of a selected data bit of TSVs does not worsen the total capacitive coupling. First, we explore the bad configuration in detail and then present our solution.

Table II summarizes the TSV current flow direction before and after encoding its ready to send data bit. CF_{bi} shows the current flow of TSV before inverting the ready to send data, while CF_{ai} represents the current flow of TSV after inversion. Based on this table an inactive TSV current flow (○) may convert to active TSV (either ⊙ or ⊗), while an active TSV (either of the ⊙ or ⊗) is converted into an inactive one (○) after inverting the ready to send data bits. Based on our analysis, a bad configuration occurs in five cases, while two of them are potential to generate unwanted 8C parasitic capacitance and the other three may generate unwanted 7C parasitic capacitance. They are called bad_config$_{8_1}$, bad_config$_{8_2}$, bad_config$_{7_1}$, bad_config$_{7_2}$, and bad_config$_{7_3}$. Fig. 4 illustrates these five cases in top view of 2D array of TSVs in a 3x3 mesh of TSVs. The candidate row for inversion is recognized by dashed lines in this figure. It also shows the parasitic capacitance value of middle TSV in the recognized row by dashed lines before and after encoding process. Each of these bad configurations affects the result of baseline coding with some conditions which are discussed in the following.

In the baseline method and in case of encoding, the 3C parasitic capacitance, if any, is converted into 7C (see Fig. 4(a))by encoding the second row of 2D array of TSVs with following four conditions:

- There are exactly two inactive TSV next to each other in

potential row for encoding process as in TSV5 and TSV6.

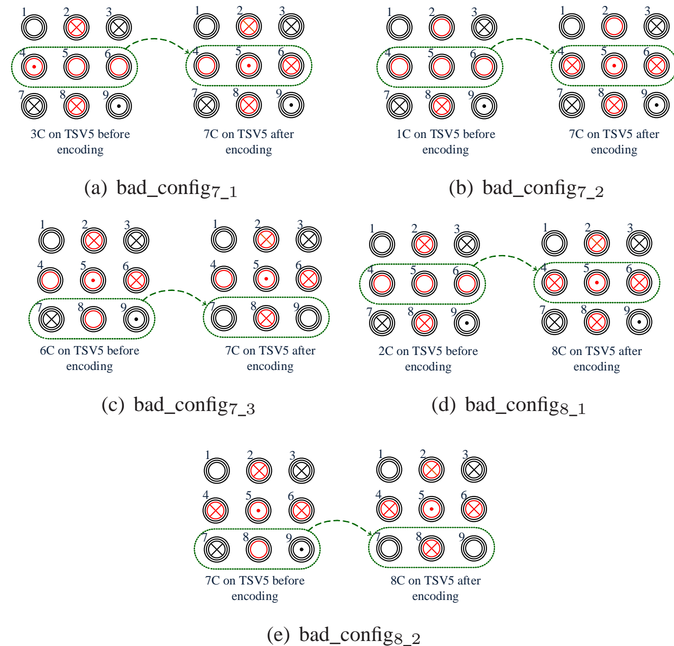
- TSV2 and TSV8 are active with the same current direction.
- The current direction of TSV6 after encoding should be the same as the current direction of TSV2 and TSV8.
- The current direction of TSV5 should be reverse of the current direction in TSV2, TSV6, and TSV8 after encoding.

The 1C parasitic capacitance is converted into 7C (see Fig. 4(b)) by encoding the second row of 2D array of TSVs with following four conditions:

- There are at least three inactive TSV next to each other in potential row for encoding process as in TSV4, TSV5, and TSV6.
- Either of TSV2 or TSV8 is inactive and the other should be active.
- The current direction of TSV4 and TSV6 after encoding should be the same as the current direction of either TSV2 or TSV8 which was active.
- The current direction of TSV5 after encoding should be reverse of the current direction of TSV4, TSV6, and either TSV2 or TSV8 which was active.

The 6C parasitic capacitance is converted into 7C (see Fig. 4(c)) by encoding the third row of the 2D array of TSVs with following four conditions:

- In capacitive matrix there is a 6C parasitic capacitance in predecessor row which is selected for encoding in a way that TSV5 has reverse current direction of TSV2 and either of TSV4 or TSV6.
- TSV8 which is in the nominated row for encoding is inactive.
- One of TSV4 or TSV6 is inactive and the other should should be active with reverse current direction of TSV5.
- The current direction of TSV8 after encoding should be same as current direction of TSV2 and either of TSV4 or TSV6 which was active.



(a) bad_config$_{7_1}$ (b) bad_config$_{7_2}$

(c) bad_config$_{7_3}$ (d) bad_config$_{8_1}$

(e) bad_config$_{8_2}$

Fig. 4. Potential configurations to generate 7C and 8C parasitic capacitance

Algorithm 1 Enhanced TCMA coding algorithm

1: AMAT ← Sent data bits
2: BMAT ← To be sent data bits
3: CMAT ← Current direction of each TSV generated by AMAT & BMAT
4: CAPMAT ← Capacitive parasitic noise of each TSV generated by CMAT
5: INV ← Redundant vector for inversion process decision at receiver side
6: **for each** $R \in Rows$ **do**
7: **for each** $C \in Columns$ **do**
8: **if** CAPMAT[R][C] == 8 or CAPMAT[R][C] == 7 **then**
9: $78C_counter++$
10: **end if**
11: **if** (there is a bad configuration bad_config$_{7_1}$ or bad_config$_{7_2}$ or bad_config$_{7_3}$) **then**
12: $bad_config_7_counter++$
13: **end if**
14: **if** (there is a bad_config$_{8_1}$ or bad_config$_{8_2}$) **then**
15: $bad_config_8_counter++$
16: **end if**
17: **end for**
18: **if** $(78C_counter > bad_config_7_counter + bad_config_8_counter)$ **then**
19: Encode the BMAT[R]
20: INV[R]=1
21: **end if**
22: **end for**

The 2C parasitic capacitance is converted into 8C (see Fig. 4(d)) by encoding the second row of 2D array of TSVs with following four conditions:

- There are at least three inactive TSVs beside each other in potential row for encoding process like TSV4, TSV5, and TSV6.
- TSV2 and TSV8 are active with same current direction.
- The current direction of TSV4 and TSV6 after encoding should be the same as the current direction of TSV2 and TSV8.
- The current direction of TSV5 should be reverse of the current direction in TSV2, TSV4, TSV6, and TSV8 after encoding.

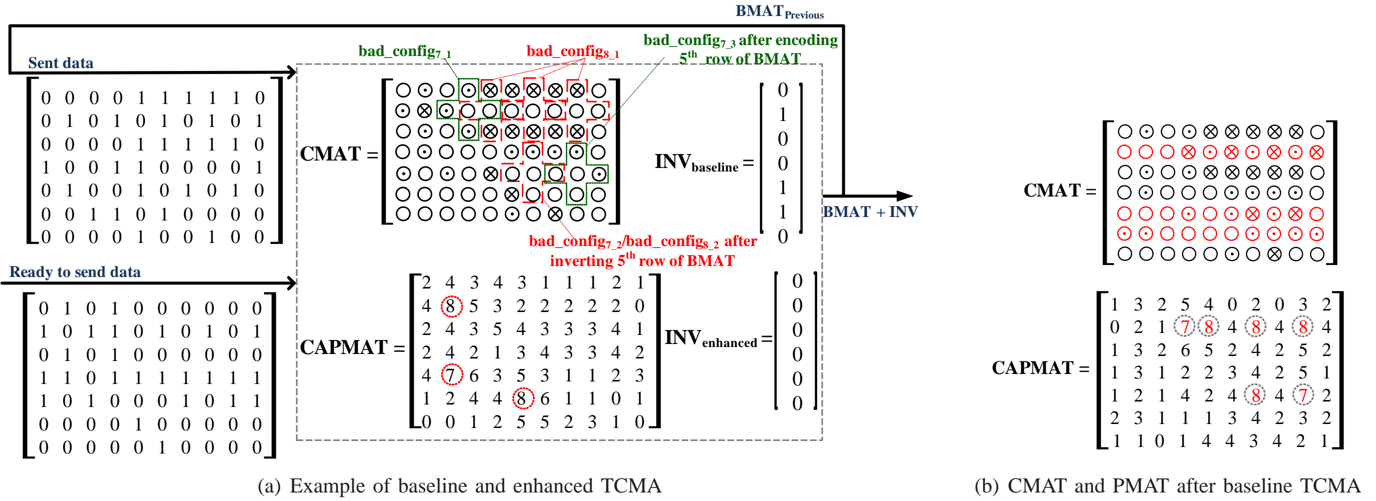
The 7C parasitic capacitance is converted into 8C (see Fig. 4(e)) by encoding the third row of 2D array TSV, if the following conditions are satisfied:

- In capacitive matrix there is a 7C parasitic capacitance in predecessor row which is selected for encoding. The inactive TSV should be also in the selected row for encoding.
- TSV8 has the reverse current direction of TSV5 after encoding.

The probability of bad configuration presence in a mesh of TSVs is very low since all the discussed conditions should be satisfied simultaneously. However, the goal of the enhanced TCMA, which is summarized in Algorithm 1 is to guarantee the encoding process will not worsen the total number of 7C and 8C parasitic capacitance in a 2D array of TSVs. In the enhanced version of TCMA the encoding process will be done if the total number of 7C and 8C parasitic capacitance in capacitive matrix is higher than the total number of bad configuration in each row.

IV. TCMA Elaboration and Evaluation

Fig. 5(a) illustrates an example of the baseline and enhanced algorithm for 7×10 given AMAT and BMAT matrices. These matrices and the ones which are used in following sentences are defined in Algorithm 1. This dimension has been chosen to show the advantages of the enhanced approach over the baseline technique for higher bandwidth data buses. First, CMAT and then CAPMAT matrices are generated form the sent (AMAT) and not sent yet (BMAT) data lines. The current flow of each TSV is presented with the same method as discussed in Section II. Then, CAPMAT is generated from CMAT by counting the total mutual capacitive parasitic difference between each TSV and its adjacent neighbors. The INV matrix is evaluated in the receiver side to



(a) Example of baseline and enhanced TCMA (b) CMAT and PMAT after baseline TCMA

Fig. 5. An example that shows baseline algorithm issues

extract the original data values if they are encoded. $\text{INV}_{baseline}$ of this example shows that the second, fifth, and sixth rows of the BMAT matrix have been encoded since there are 8C or 7C parasitic capacitance values in these rows of CAPMAT matrix. Since the number of 7C and 8C parasitic capacitance are not higher than the number of bad configuration in enhanced method, the $\text{INV}_{enhanced}$ shows none of the rows the BMAT has been encoded. Fig. 5(b) represents the updated CMAT and CAPMAT matrices in the baseline approach after encoding the second, fifth, and sixth rows of BMAT matrix in which the total number of 7C and 8C parasitic capacitance increases from 3 to 6. This example illustrates all 5 possible bad configurations. The bad_config_{7_1} and bad_config_{8_1} are depicted in second row of CMAT in Fig. 5(a), resulting in three 8C and one 7C after encoding second row of BMAT. The bad_config_{7_2} of fifth row is highlighted in CMAT matrix of Fig. 5(a). After encoding fifth row of BMAT, the undesirable 7C will be generated in CAPMAT[5][7], which is also bad_config_{8_2}. Furthermore, encoding fifth row of BMAT generates a bad_config_{7_3} in CAMPMAT[5][9]. Since the encoding decision is supposed to be done row by row in one direction (from top to bottom in this example) or reverse, the unwanted generated 7C and 6C in fifth row of CAPMAT are potential to generate 8C and 7C, respectively by encoding the sixth row of BMAT. Due to the presence of 8C in sixth row of CAPMAT, it is selected for encoding process and both of bad_config_{7_3} and bad_config_{8_2} generate undesirable 8C and 7C in fifth row of CAPMAT which is shown in Fig. 5(b). However, the enhanced algorithm prevents all of these bad effects by predicting them.

To evaluate the advantages of the baseline TCMA for smaller mesh size, Monte Carlo simulations for 10000 iterations on different sizes of TSV mesh are examined. The total number of 7C and 8C parasitic capacitance before and after applying the baseline TCMA for different mesh size of TSVs is shown in Fig. 6. It is depicted that the mitigation rate of 7C and 8C parasitic capacitance after applying the baseline TCMA are almost 98%, 94%, and 90% for 4×4, 6×6, and 8×8 mesh of TSVs. The information redundancy of the baseline TCMA method for these sizes of mesh of TSVs are 25%, 16%, and 12%. However, the mitigation rate of the baseline TCMA is increased for large mesh of TSVs, as expected. This is because of the probability of bad configuration occurrence rises by increasing the sizes of TSV meshes. The Monte Carlo simulations for 10000 iterations for larger mesh of TSVs are also examined for both baseline and enhanced TCMA to show the advantages of enhanced TCMA. Although the mitigation rate of total number of 7C and 8C parasitic capacitance values is increasing by using larger mesh of TSVs, enhanced TCMA prevents encoding process if the result is worsen. This is shown in Fig. 7(a), in which the mitigation rate of 7C and 8C parasitic capacitance occurrence by applying enhanced TCMA are always higher than baseline approach.

PARSEC benchmark [17] as a realistic data traffic for large size of mesh of TSVs are also applied to check the performance of the baseline and enhanced TCMA. Memory traces of PARSEC applications have been employed in this experiment, which are extracted by the PIN tool [18], a dynamic binary instrumentation framework for the IA-32 and x86-64 instruction-set architectures. The total number of 7C and 8C parasitic capacitance values for memory traces of PARSEC application workloads through the TSVs are reported for a 8×32 mesh of TSVs in Fig. 7(b). The migration rate of TCMA for Blackscholes, Facesim, Vips, and Raytraces are between 80% to 90% and for the rest of them is almost 70%. Although the differences between the mitigation rates of baseline and enhanced TCMA are not very much, but the result of enhanced method is always better than baseline as

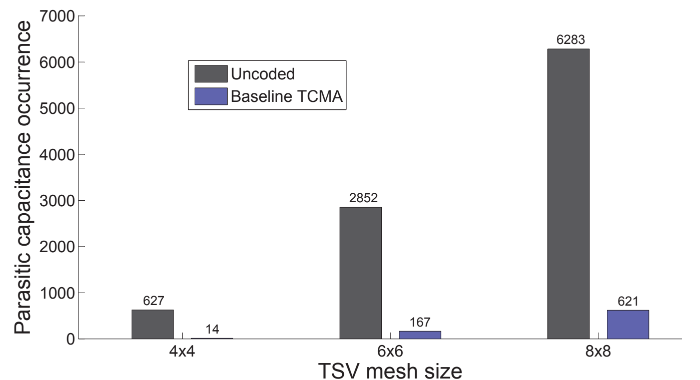
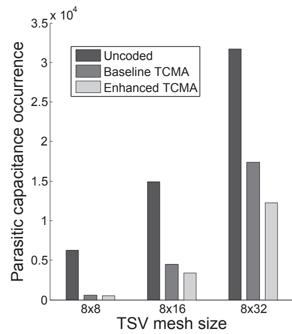
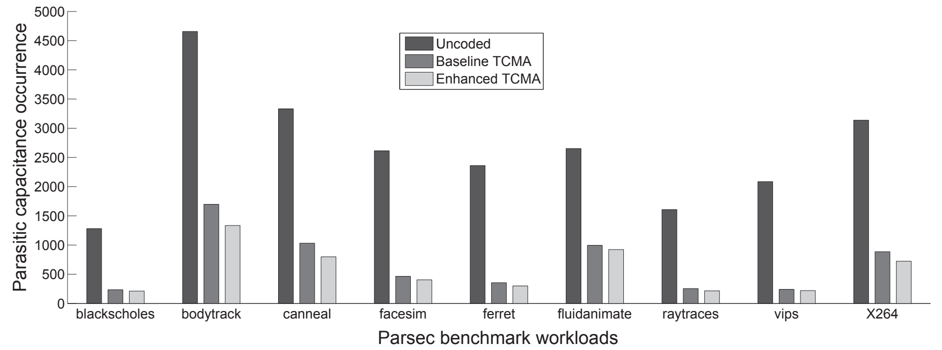


Fig. 6. Number of 7C/8C for random data bit patterns in small mesh of TSVs



(a) Random data bit patterns in larger mesh of TSVs



(b) PARSEC application data bit patterns in 8×32 mesh of TSVs

Fig. 7. 7C and 8C parasitic capacitance for random and PARSEC applications data with/without TCMA

it is expected. In other words, it is always guaranteed that by applying the enhanced TCMA the total number of 7C and 8C parasitic capacitance will never be worse off because of the bad configuration presence.

In order to evaluate the proposed coding methods, the baseline and enhanced TCMA encoders are implemented in Verilog and synthesized by Synopsys Design Compiler using 28nm TSMC library (1.05V, $25\,^{\circ}$C). Table III reports the synthesis results as power consumption and occupied area. The latency of the enhanced method is reported by the critical path including: registers latching the adjusted output data bits toward the feedback input for subsequent CMAT computation. In other words, it does not depend on the dimension of TSV arrays. According to the logic synthesis, the latency of the baseline and enhanced TCMA are reported as 69.5ps and 74.9ps for all given TSV dimensions in Table III. The feasibility of both proposed coding algorithms are confirmed by considering the gained coupled parasitic capacitance mitigation and its tangible footprint and power consumption. Decoder units are not implemented in this experiment since they are only composed of a comparator and a mix of inverter gates. They are much lighter than encoder components in terms of area, power consumption, and latency.

V. Conclusion

Two baseline and enhanced algorithms have been proposed in order to minimize the TSV-to-TSV capacitive coupling issue. Baseline algorithm is proposed for small mesh of TSVs which are considered in 3D NoC applications, while enhanced method is suggested for large mesh of TSVs which are more applied in 3D memory applications. The enhanced method guarantees that the encoding process prevents generating undesirable parasitic capacitance values by recognizing all susceptible configurations. According to experimental results, the baseline method's mitigation rate is more than 90% for TSV meshes smaller than 10×10. The enhanced algorithm mitigates the TSV-to-TSV capacitive coupling more than 70% for 8×32 mesh of TSVs.

Table III
Hardware synthesize results

Mesh size	Baseline		Enhanced	
	Area (μm^2)	Power (μW)	Area (μm^2)	Power (μW)
8×8	918	2340	1096	3000
8×16	1818	4520	2173	5900
16×8	2094	5260	2165	5880
8×32	3321	8840	4331	11700
32×8	4086	11000	4323	11700

References

[1] J. Burns, "Tsv-based 3d integration," in *Three Dimensional System Integration*, A. Papanikolaou, D. Soudris, and R. Radojcic, Eds. Springer US, 2011, pp. 13–32.

[2] D. H. Kim and S. K. Lim, "Design quality trade-off studies for 3-d ics built with sub-micron tsvs and future devices," *Emerging and Selected Topics in Circuits and Systems, IEEE Journal on*, vol. 2, no. 2, pp. 240–248, 2012.

[3] A. Eghbal, P. M.Yaghini, N. Bagherzadeh, and M. Khayambashi, "Analytical fault tolerance assessment and metrics for tsv-based 3d network-on-chip," *Computers, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2015.

[4] C. Liu and S. K. Lim, "A study of signal integrity issues in through-silicon-via-based 3d ics," in *Interconnect Technology Conference (IITC), 2010 International*, June 2010, pp. 1–3.

[5] A. Eghbal, P. M. Yaghini, and N. Bagherzadeh, "Tsv-to-tsv inductive coupling-aware coding scheme for 3d network-on-chip," in *Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2014 IEEE International Symposium on*, Oct 2014, pp. 92–97.

[6] P. Yaghini, A. Eghbal, M. Khayambashi, and N. Bagherzadeh, "Coupling mitigation in 3-d multiple-stacked devices," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2015.

[7] 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, no. 4, pp. 551–560, April 2009.

[8] K. N. Patel and I. L. Markov, "Error-correction and crosstalk avoidance in dsm busses," in *Proceedings of the 2003 International Workshop on System-level Interconnect Prediction*, ser. SLIP '03. ACM, 2003, pp. 9–14.

[9] C. Duan, B. J. LaMeres, and S. P. Khatri, "Preliminaries to on-chip crosstalk," in *On and Off-Chip Crosstalk Avoidance in VLSI Design*. Springer US, 2010, pp. 13–26.

[10] 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*, June 2011, pp. 783–788.

[11] R. Weerasekera, M. Grange, D. Pamunuwa, and H. Tenhunen, "On signalling over through-silicon via (tsv) interconnects in 3-d integrated circuits," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2010*, March 2010, pp. 1325–1328.

[12] K. Salah, A. El Rouby, H. Ragai, and Y. Ismail, "Tsv impact on circuit performance and recommended design methodologies," in *Microelectronics (ICM), 2012 24th International Conference on*, Dec 2012, pp. 1–4.

[13] R. Kumar and S. P. Khatri, "Crosstalk avoidance codes for 3d vlsi," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2013*, 2013, pp. 1673–1678.

[14] Q. Zou, D. Niu, Y. Cao, and Y. Xie, "3dlat: Tsv-based 3d ics crosstalk minimization utilizing less adjacent transition code," in *Design Automation Conference, 2014 19th Asia and South Pacific*, 2014, pp. 762–767.

[15] PTM, "Predictive Technology Model," ptm.asu.edu.

[16] S. Itr, "ITRS 2012 Executive Summary," ITRS.

[17] C. Bienia and K. Li, "Parsec 2.0: A new benchmark suite for chip-multiprocessors," in *Proceedings of the 5th Annual Workshop on Modeling, Benchmarking and Simulation*, June 2009.

[18] Intel-cooperation, "Pin-A Dynamic Binary Instrumentation Tool," Intel.