Timing Analysis and Optimization for 3D Stacked Multi-Core Microprocessors

Young-Joon Lee and Sung Kyu Lim
Electrical and Computer Engineering, Georgia Institute of Technology
email: yjlee@gatech.edu

***Abstract*— In this paper we demonstrate the methodology for designing and optimizing the LEON3 multi-core microprocessor in 3D stacked ICs. Based on GDSII-level details, we compare the 3D IC implementations as well as the traditional 2D IC implementation. For 3D IC implementation, we compare three partitioning styles: core-level, block-level, and gate-level. These partitioning styles represent three most relevant 3D implementation choices. The design methodology for such partitioning styles and their implications on the physical layout are discussed. Then we propose two methods to perform timing optimizations for 3D stacked ICs: timing scaling and timing budgeting. By analyzing the timing constraints from each method and the effects on the timing results and the layout, we show that each method has different impacts on the overall design quality. Lastly, we discuss additional 3D optimization opportunities.**

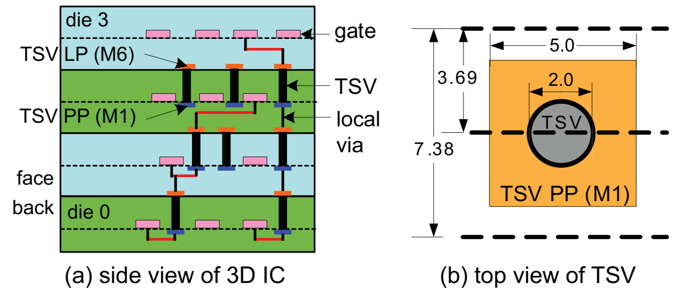

Fig. 1. Target 3D structure. (a) Dies are flipped over and facing down. TSV pin pad (PP) and landing pad (LP) are shown. (b) Our TSV occupies two standard cell rows. Unit is μm.

I. Introduction

As the complexity and cost for continuing Moore's law in 2D ICs increases rapidly, 3D ICs attract more and more attention from both academia and industry. To make 3D ICs practical and profitable, much research has been done in various fields—material, chemical, fabrication, integration [1], etc—not to mention the electronic design automation (EDA). In the EDA field, various algorithms for design steps such as circuit partitioning, placement [2], routing [3], and timing optimization have been proposed, yet many of them neglected the impact of the through-silicon-vias (TSVs) on the physical layout. Depending on fabrication technology, TSVs can be so large that the aforementioned algorithms may not work as intended.

Today's 3D IC market is mostly encompassed by memory chips [4] and image sensor chips [5], which are designed in a full-custom fashion. In the near future, we expect to see many-core processors on a 3D stacked IC or core-memory stacked ICs. However, currently there is no fully-integrated 3D IC design EDA software that can perform the full design flow from the register transfer level code to the GDSII layout. Thus, if we want to design a complex digital system in a 3D IC, we need to combine the existing tools with several tweaks to make them work for 3D designs. In this work, we show how we designed and optimized a multi-core microprocessor in 3D ICs with different design options using existing 2D design tools and our in-house tools. The major contributions of this work are as follows:

- To the best of our knowledge, this is the first work to compare 2D and 3D IC designs of a commercial-grade multi-core processor based on GDSII layout designs. We discuss the design methodology for each design option and the implications on the layout.
- We show the impact of TSVs on the 3D designs in terms of chip area, wirelength, and performance. For this purpose, we run 3D analysis using existing 2D design tools with some modifications.
- We propose two methods to perform timing optimizations for 3D stacked ICs: timing scaling and timing budgeting. Timing scaling method is to scale the input/output delay timing constraints at each boundary point, while timing budgeting method is to distribute the timing slack of a path to each net on the path.

The remainder of the paper is organized as follows. First, we present the target system in Section II. In Section III, we discuss the three design options. Then, in Section IV we present two kinds of 3D timing optimizations that are based on different timing constraints. Experimental results are presented in Section V, followed by discussions in Section VI. Finally, we conclude in Section VII.

II. Target System

A. 3D Structure

Our target 3D structure is illustrated in Fig. 1. The overall stack structure is shown on the left, where all four dies are bonded in a face-to-back fashion. We assume via-first TSVs, which occupy the device layer and Metal 1 and 6 (M1 and M6). As shown on the right side in the stack diagram, when a net spans more than two dies, it is routed through TSVs

This material is based upon work supported by the National Science Foundation under CAREER Grant No. CCF-0546382, the SRC Interconnect Focus Center (IFC), and Intel Corporation.

TABLE I

ARCHITECTURE CONFIGURATION OF THE LEON3 DESIGN.

Instruction cache	16 KB, 2 way
Data cache	16 KB, 2 way
Register file	32 32-bit registers, 8 windows
Multiplier	32 x 32bits
Divider	iterative

TABLE II

SUMMARY OF THE SYNTHESIS OF THE QUAD-CORE LEON3 DESIGN.

Technology	130nm
# memory blocks	44
# standard cells	82,461
# nets	87,451
Average fanout	2.46
Total cell area (um^2)	6,101,542
Target clock period (ns)	3.333
Slack (ns)	0

as well as local vias. Note that there is no TSV on Die 3. The top view of a TSV is shown on the right. A TSV pin pad (PP) on M1 or a landing pad (LP) on M6 occupies two standard cell rows (denoted by the dotted lines), which is not negligible considering that the area corresponds to about 16 minimum-size inverters.

Depending on the TSV dimensions, the capacitance (C_{TSV}) and the resistance (R_{TSV}) of TSVs varies. Since the timing values through these TSVs depend on the parasitic values, we varied the values to see the impact on timing. TSV resistance is dependent on ohmic resistance and contact resistance, the contact resistance being more dominant. In this paper, we used C_{TSV}=25, 50fF and R_{TSV}=1Ω for experiments.

B. Architecture

We use the LEON3 processor [6] to demonstrate our 3D timing analysis and optimization. The LEON3 processor is a 32-bit processor compliant with the SPARC V8 architecture. It contains an advanced 7-stage pipeline with a hardware multiplier and divider. The LEON3 design also has configurable caches and local instruction and data scratch memories. It is configurable as a multi-core processor on AMBA bus. For this project we configured a quad-core processor with a single configuration for all the cores, which is described in Table I.

Table II summarizes the synthesis results. We used Synopsys Design Compiler with the physical libraries for the target technology. Memory macro blocks were generated using a memory compiler for the target technology. The original HDL source code was modified to include the memory blocks. The synthesized circuit met the timing goal, excluding interconnect delay.

The memory macro blocks in the core are summarized in Table III. Total 11 memory macro blocks are used per each core, which are as follows in decreasing size: two banks per instruction and data caches, two dual port memory blocks for a three-port register file, two banks per instruction/data cache tags, and an address translation table. Since these macro blocks are large, the location and orientation of them affect the overall design quality much. Thus they should be placed carefully, considering connections to other parts.

TABLE III

SUMMARY OF THE MEMORY MACRO BLOCKS.

Name	Capacity (bits)	Dimension (um)	I/O pins
Instr./Data cache	2048x32	427.205x544.295	78
Register file	256x32	401.29x193.265	84
Instr./Data cache tag	256x40	269.035x178.805	91
Address translation	32x32	131.915x108.805	72

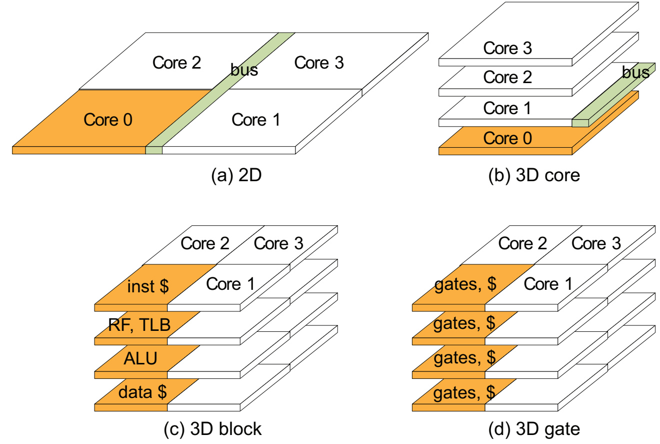

Fig. 2. Four design options. Blocks highlighted in orange denote Core 0. inst $ and data $ denote instruction and data cache, while RF and TLB represent register file and address translation buffer.

III. DESIGN OPTIONS

There are three 3D partitioning options that we examine and compare to a traditional 2D design. Fig. 2 shows the four design options. The following list summarizes them:

- 2D (**2D**): This is the traditional design on a single die. The bus controller in the center connects the cores. No TSV exists in this design option.
- 3D core-level partition (**3D-core**): Each core is placed on each die, and the bus controller is placed on Die 1. TSVs are used to connect cores on Die 0/2/3 to the bus controller. Minimal number of TSVs is used.
- 3D block-level partition (**3D-block**): The processor core is partitioned in core+memory style and stacked. On Die 1, all the core logics and the bus controller are placed, while on Die 0/2/3 all the memory blocks are placed. A moderate number of TSVs are used to connect the blocks to the core logics.
- 3D gate-level partition (**3D-gate**): The whole circuit is partitioned into four parts, and mapped onto four dies. The memory blocks are also placed on four dies. This design uses the largest number of TSVs.

In the following subsections we describe design flows of the design options in detail.

A. 2D

For 2D IC design style, we follow a conventional 2D design flow. Starting with the synthesized netlist, we perform floorplanning, placement, routing and timing optimization.

Clock tree is out of our scope; an ideal clock is assumed. In the floorplan step, we divide the whole chip area into four core regions and a bus region, and decide the memory macro block locations inside core regions. An identical macro block placement is used for all cores, with proper rotations to face cores towards the bus. In the center region the AMBA bus controller logic is placed.

B. 3D-core

The main idea of this design option is to reuse existing 2D core design and expand in 3D with minimum effort. In this option, we place one core on each die. We use the same macro block placement per core as in 2D case. The bus controller logic is placed on Die 1 which connects the core on Die 1 as well as the cores on Die 0, 2, and 3 using TSVs. All the TSVs are manually placed outside the core region, in a clustered fashion.

C. 3D-block

We divide a single core into logic cells and memory blocks. Then all the logic cells are placed on Die 1, while the memory blocks are placed on Die 0, 2, and 3. The reason for placing logic cells on the Die 1 is to minimize the total distance to memory blocks. Note that the ordering of logic cell die and memory die is important, because the relationship is asymmetric. For instance, if the memory blocks are on Die 0 and the logic cells are on Die 1, the TSVs that connects memory blocks to logic cells are on Die 0, which means the active device space on Die 0 is consumed by TSVs while no active device space is needed on Die 1. On the other hand, if the logic cells are on Die 0 and the memory blocks are on Die 1, the TSVs will consume the active device space on the logic cell die. Thus, with our configuration, TSV connections to memory blocks on Die 2 and 3 occupy the active device space on logic cell die (Die 1).

Due to the shape of the biggest memory blocks (instruction and data cache banks), the core region is rectangular. All TSVs are manually placed around the pins of memory blocks. Since the pin pitch of memory blocks is smaller than the minimum pitch of our TSV, we put the TSVs in multiple rows, trying to reduce the distances between memory pins and TSVs. Since connections between the core logics and the memory blocks on Die 3 has to go through Die 2, we placed pass-through TSVs on Die 2, avoiding contact to the memory blocks on Die 2. Four of this four die stack are put together on x-y plane to form the quad-core processor.

D. 3D-gate

Our last design option is based on gate-level partitioning. First, the input netlist is partitioned into four parts. The memory blocks in the netlist are very large compared to the standard cells, thus we take care of them first. Per a core, each die has a bank of either the instruction cache or the data cache, and its cache tag. In addition, Die 0 has the address translation table, whereas Die 1 and 2 have a bank of the register file each. The location of these memory blocks are manually determined considering pin locations as well as net connectivity. Then the standard cells are placed in 3D by the recursive partitioning technique [7].

IV. 3D Timing Analysis and Optimization

In this section we discuss the 3D timing analysis and optimization techniques.

A. 3D Timing Analysis

Our 3D static timing analysis (STA) is performed using Synopsys PrimeTime. First, we prepare the Verilog netlist files of all four dies and the SPEF files containing extracted parasitic values for all the nets of the dies. Then, we create a top-level Verilog netlist that instantiates each die design and connects the 3D nets using TSV connections. We also create a top-level SPEF file that has parasitic models of the TSVs. After that, we run PrimeTime to get the 3D timing analysis results. The worst negative slack (WNS) and the total negative slack (TNS) are reported to demonstrate the timing quality of the design. Meanwhile, we generate timing constraints from the timing analysis results to perform 3D timing optimization.

B. 3D Timing Optimization

Considering that each die design is a subdesign of the entire design, 3D IC designs are essentially hierarchical. Thus we perform the 3D timing optimization in a hierarchical manner. Compared to a non-hierarchical design flow, in a hierarchical design flow the timing constraints on the boundary is important because it is the key information that the timing optimization engine uses. We work on the timing optimization of each die with timing constraints on the die boundaries (TSV PP and LP ports). We demonstrate two methods to generate the timing constraints: timing scaling and timing budgeting.

1) Timing Scaling: Timing scaling method is to scale the input/output delay timing constraints at each boundary point. Consider a *3D path* from a source F/F in a die through a die boundary to a sink F/F in a neighbor die. After the 3D timing analysis is done, we get the longest path delay from the source to the sink ($= T_{LPD}$) as well as the delay up to the die boundary ($= T_{boundary}$). To achieve the target clock period T_{CLK}, ideally we need to make T_{LPD} the same as T_{CLK}. Thus, we set the scaling factor $SF = T_{CLK}/T_{LPD}$. Then we calculate the scaled boundary constraints as follows:

$$T_{boundary,scaled} = T_{boundary} \times SF$$

The updated timing constraint file is used in the timing optimization. By this method, all the 3D paths are constrained so as to meet the target clock period. We implemented this method in PrimeTime Tcl and Perl.

2) Timing Budgeting: Timing budgeting [8] is to distribute the timing slack of a path to each net on the path. This method analyzes the timing graph of the entire circuit to find out where the critical paths are. Nets on non-critical paths can be given a positive timing budget which can be used for other circuit optimizations such as area and power minimization. On the other hand, nets on critical paths are given negative timing

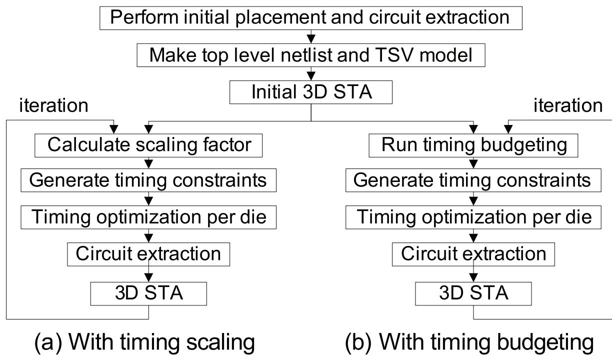

Fig. 3. Design flow with timing scaling and timing budgeting.

TABLE IV
EXPERIMENTAL SETTINGS.

	2D	3D-core	3D-block	3D-gate
Die size (μm^2)	2900x2600	1500x1300	1709x1151	1500x1400
Total area (μm^2)	7.54E6	7.80E6	7.87E6	8.40E6
Footprint (μm^2)	7.54E6	1.95E6	1.97E6	2.10E6

budgets, which means the delays of the nets should be reduced by timing optimization. We use Synopsys Design Compiler to perform timing budgeting.

The overall design flow is shown in Fig. 3. With the generated timing constraints, timing optimization is performed by Encounter. We iterate the optimization loop several times.

V. EXPERIMENTAL RESULTS

Experimental settings are shown in Table IV. All 3D cases use four dies. Target clock period was set to $3.333ns$. For 2D case, the chip area was chosen so that the initial utilization is around 80%, whereas for 3D cases chip area was expanded considering the TSV impact. The default R_{TSV} and C_{TSV} were 1Ω and $25fF$, respectively.

A. Initial Design Results

Fig. 4 shows the Cadence Encounter images of top-dies (Die 0) in the four design styles we study in this paper. Fig. 5 shows zoom-in shots of the GDSII images using Cadence Virtuoso. Table V shows the initial design results of the design options before timing optimization. Due to the pre-place optimization, the total number of placed instances differs for each design. In 3D-gate case the utilization is set to a lower value than other cases, because when the utilization is set at the same level as in other cases, after timing optimization the design had severe congestion and too high utilization. The number of TSVs is the smallest in 3D-core case, while 3D-block uses about 9.6 times more TSVs than 3D-core case. 3D-gate case uses about 82% more TSVs than 3D-block case. As the designs use more TSVs, the total wirelength decreases. In particular, 3D-gate case shows 22.6% shorter total wirelength than 2D case. However, shorter total wirelengths do not always lead to better timing results as we will see in Subsection V-B.

The wirelength distributions of the design options are shown in Fig. 6. Compared to 2D case, 3D-core shows less number

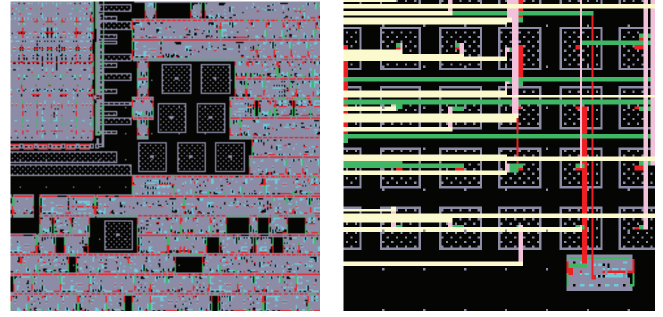

Fig. 5. Left: TSVs and gates. Right: routing to TSVs. (Cadence Virtuoso GDSII shots)

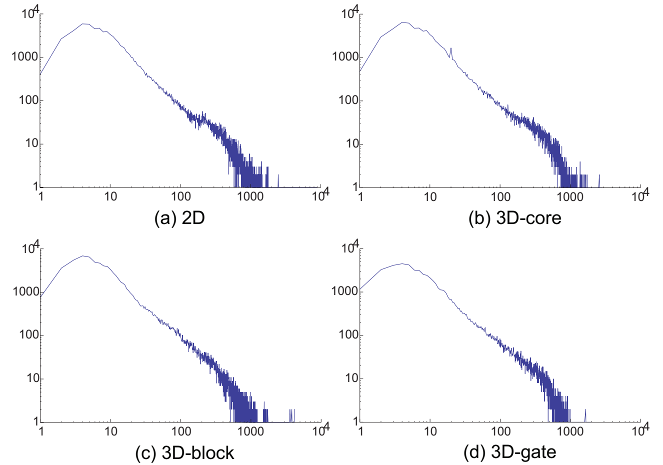

Fig. 6. Wirelength distribution of design options before timing optimization. The x-axis is wirelength in μm and the y-axis is net count.

of nets around $1,000\mu m$ because the distances between cores and the bus controller have been reduced by TSVs. Compared to 3D-core case, in 3D-block case there are more nets with very short wirelengths ($< 4\mu m$), yet there are still several nets with long wirelengths. Compared to other cases, the overall distribution of 3D-gate has been shifted towards left, and there are no net with a very long wirelength. The nets in 3D-gate case generally have shorter wirelengths than other cases.

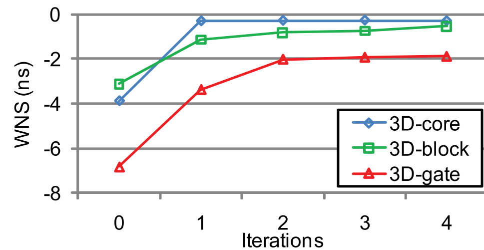

Fig. 7. WNS values of 3D-core, 3D-block, and 3D-gate cases with timing budgeting when $C_{TSV} = 25fF$.

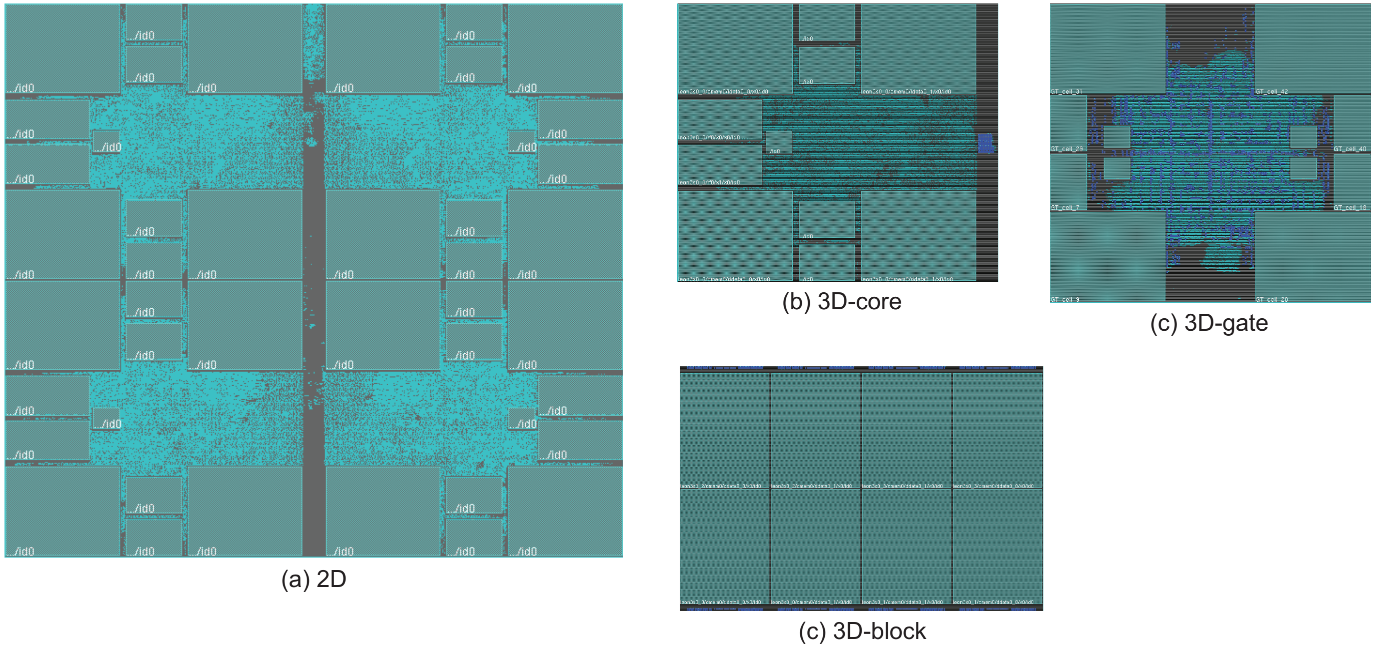

Fig. 4. Top-die layouts of the four design styles drawn in scale. (Cadence Encounter shots)

TABLE V

INITIAL LAYOUT RESULTS FOR THE DESIGN OPTIONS. UTILIZATION MEANS AREA UTILIZATION INCLUDING STANDARD CELLS AND MEMORY BLOCKS, AND WIRELENGTH MEANS TOTAL WIRELENGTH.

	2D	3D-core					3D-block					3D-gate				
		Die 0	Die 1	Die 2	Die 3	total	Die 0	Die 1	Die 2	Die 3	total	Die 0	Die 1	Die 2	Die 3	total
# instances	84,562	21,093	24,368	21,672	21,048	88,181	8	86,618	28	8	86,662	26,179	18,630	20,711	28,271	93,791
Utilization (%)	80.97	78.15	79.85	78.39	78.15	78.64	94.61	48.46	73.58	94.61	77.81	70.29	78.32	78.11	67.62	73.59
# TSVs	0	112	222	111	0	445	624	3,040	624	0	4,288	2,345	2,388	3,089	0	7,822
Wirelength (m)	6.405	1.476	1.587	1.462	1.483	6.008	0.027	5.087	0.074	0.027	5.215	1.352	0.963	0.981	1.663	4.959

B. Timing Optimization

Fig. 7 shows how the WNS value of 3D-core, 3D-block, and 3D-gate changes during the timing optimization iterations. The biggest reduction of WNS was observed in the first optimization. In 3D-core case, WNS converged fast after the first iteration, while in 3D-block and 3D-gate cases WNS kept decreasing during the four iterations.

Table VI shows the results of our 3D timing optimization. Comparing the results to the ones in Table V, we can see that the total wirelengths increased by 2.4%, 12.9%, and 47.7% in 3D-core, 3D-block, and 3D-gate cases with timing budgeting. Compared to the increase of 1.6% in 2D case, the wirelength increase is severe in 3D-gate case. Also the utilization values increased after the optimization, due to the gate sizing and buffer insertion by the optimization engine to meet the timing goal. From the WNS values, we can see that 3D-core design can operate 13% faster than 2D design. Other designs are slower than 2D, especially 3D-gate case. In terms of TNS, 3D-core is better than 3D-block. In 3D-gate case, although average utilization was around 80%, the designs had very densely packed placement regions around center, which prevented further timing optimizations. In sum, 3D-core with timing budgeting resulted in the best quality circuit in terms of timing.

Fig. 8 shows the timing critical path after timing optimization for the 3D-gate design. The path starts on Die 2, goes down to Die 1, comes back to Die 2, goes down deeper to Die 1 and 0 and comes back to Die 1 and 2, then goes up to Die 3 and comes back to Die 2, goes down to Die 1 and comes back to Die 2, then goes up to Die 3 where the path ends. This path snakes through the entire stack, involving many TSVs. Looking at the path from (1) to (8), we can see that the path goes through the dies back and forth. And the path from (8) to (10) as well as the path from (22) to (24) could be shorter if the gates at (9) and (23) are placed closer to (8) and (22), although it may affect other nets that are connected to this path. Since we know that this path is the critical path, we could move these gates and TSVs to make the entire path shorter. Also we may decrease the delay of the entire path by making the path encompass less number of dies thus using less number of TSVs. This demonstrates the need for a real 3D-aware placer for TSVs.

C. Impact of TSV parasitics

To see the impact of TSV parasitics on timing, we optimized the 3D designs with different C_{TSV}. For all three design options, the utilization and the total wirelength slightly

TABLE VI

TIMING OPTIMIZATION RESULTS WHEN $C_{TSV} = 25fF$.

	2D	3D-core		3D-block		3D-gate	
		scaling	budgeting	scaling	budgeting	scaling	budgeting
# inserted buffers	9,177	8,758	8,516	11,310	11,223	13,557	13,528
Utilization (%)	85.59	83.22	83.14	82.88	82.98	80.07	79.56
Wirelength (m)	6.51	6.176	6.15	5.728	5.89	7.346	7.323
WNS (ns)	-0.357	-0.478	-0.310	-0.659	-0.543	-1.694	-1.884
TNS (ns)	-344	-169	-182	-237	-194	-2522	-2691

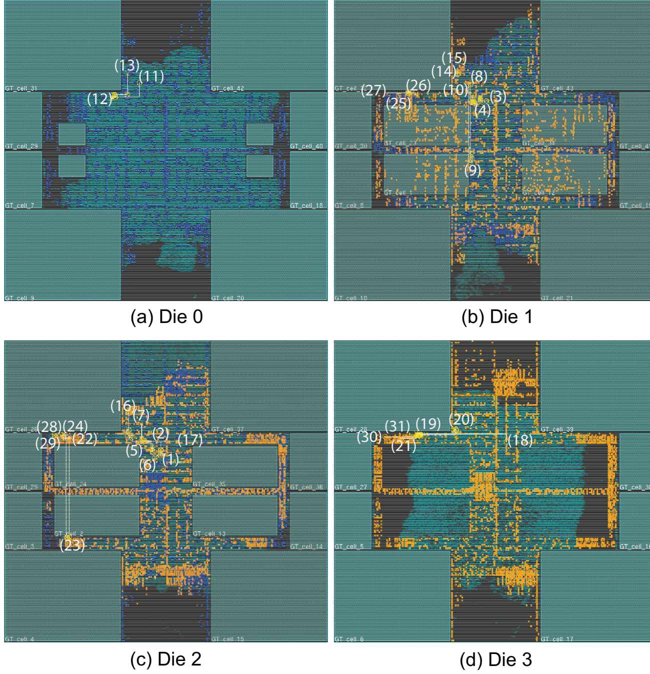

Fig. 8. Layout snapshots of dies for 3D-gate, with timing critical path highlighted in white. Numbers in bright yellow represent the path sequence. Small blue squares are TSV PPs on M1, and orange squares are TSV LPs on M6.

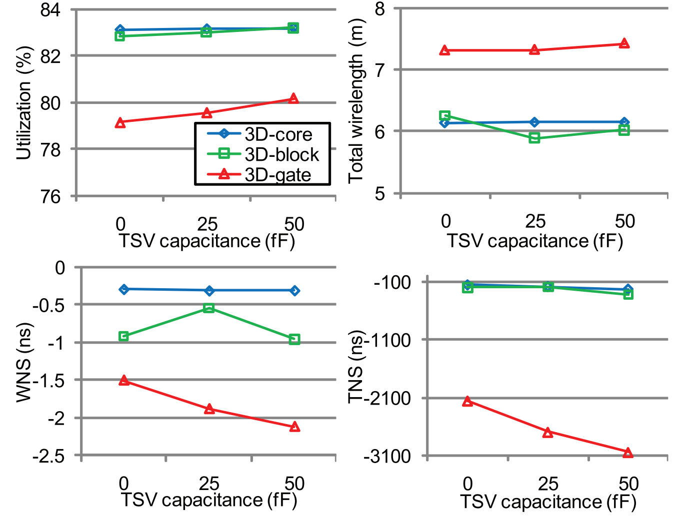

Fig. 9. The impact of TSV parasitics on various metrics. $C_{TSV} = 0fF$ means ignoring the parasitics of TSVs. Timing budgeting was used for optimization.

increased with higher C_{TSV}. This is because the optimization engine tends to insert more buffers and upsize gates with higher C_{TSV}. The WNS of 3D-block when C_{TSV} is $25fF$ was lower than when C_{TSV} is $0fF$. Checking on the timing constraints, we found that when C_{TSV} is $0fF$, the timing constraints were not tight enough, thus the optimization engine did not perform enough optimization. The WNS and the TNS of 3D-gate degraded quickly with increased C_{TSV}. That is because the TSV count is rather high in 3D-gate case, thus more timing paths are affected by increased C_{TSV}. Also the TNS of 3D-block degrades with higher C_{TSV}. In contrast, 3D-core case was not so much affected by C_{TSV} variation. We may conclude that when the TSV count is high, the overall timing quality is more likely to be affected by TSV parasitics.

VI. DISCUSSIONS

From this study, we found the following noteworthy points.

- Even though the speed of convergence with timing budgeting method was slower than timing scaling, it led to better timing results. When the 3D timing path goes through many gates, we need more accurate ways of generating timing constraints than merely scaling the timing constraints obtained from 3D STA.
- The impact of the TSV parasitics on design quality is significant; TSV parasitics affected the overall quality of the design in terms of area utilization, wirelength, and timing.
- It is possible to perform timing optimization with existing 2D CAD tools by providing timing constraints on the boundaries of design hierarchy, however we cannot exploit the full benefit of 3D structure. The 2D EDA tool does not see the whole 3D picture, thus various powerful optimization techniques such as net transformation and restructuring cannot be performed. This shortcoming could get worse when more dies are stacked together, thus true 3D EDA tool development is required to enable higher level of integration.
- The placement of gates and TSVs should be 3D-timing-aware. In this study we performed the standard cell placement for 3D-gate case using non-timing-aware, wirelength driven 3D placer. However, due to the TSV parasitics the optimization engine inserted many buffers as well as increased gate sizes. Thus the placement had too densely packed regions, leading to premature gate sizing and buffer insertion.

VII. CONCLUSIONS

In this paper, we presented the timing analysis and optimization of a quad-core microprocessor in 3D ICs. Three different partitioning options for 3D ICs were explored in layout level and timing results were analyzed. Current commercial 2D EDA tools cannot fully utilize benefits of 3D, which calls for the development of 3D-aware design tools. Our timing optimization did not lead to an optimal design, because the partitioner and placer were not 3D-timing-aware, and optimization was not aggressive enough.

TSV parasitics affected the overall quality of the design in terms of utilization, wirelength, and timing. With high TSV parasitics, it is better not to use too many TSVs, because of buffering cost and timing degradation by TSVs. Furthermore, the target circuit size also correlates to the benefit of 3D IC, because the relative size of the capacitance of a TSV and a metal wire matters. In large circuits, we have better chance of improving the delay along timing paths with long wires by adopting 3D connections. Conversely, with small and simple circuits, it would not be beneficial to implement a 3D design. Our future works include further timing optimizations with buffer insertion and wire/driver sizing for timing critical 3D nets.

REFERENCES

[1] S. J. K. et al., "Wafer-level 3D integration technology," *IBM Journal of Research and Development*, vol. 52, no. 6, pp. 583–597, 2008.

[2] J. Minz, E. Wong, M. Pathak, and S. K. Lim, "Placement and Routing for 3D System-On-Package Designs," *IEEE Transactions on Components and Packaging Technologies*, vol. 29, no. 3, pp. 644–657, 2006.

[3] M. Pathak and S. K. Lim, "Thermal-aware Steiner Routing for 3D Stacked ICs," in *Proc. IEEE Int. Conf. on Computer-Aided Design*, 2007, pp. 205–211.

[4] S.-M. J. et al., "Three Dimensionally Stacked NAND Flash Memory Technology Using Stacking Single Crystal Si Layers on ILD and TANOS Structure for Beyond 30nm Node," in *International Electron Device Meeting*, 2006, pp. 37–40.

[5] V. S. et al., "Megapixel CMOS Image Sensor Fabricated in Three-Dimensional Integrated Circuit Technology," in *IEEE International Solid-States Circuits Conf.*, 2005.

[6] A. G. AB., "Leon3 Processor." [Online]. Available: http://www.gaisler.com/

[7] M. Pathak, Y.-J. Lee, T. Moon, and S. K. Lim, "Through Silicon Via Management during 3D Physical Design: When to Add and How Many?" in *Proc. IEEE Int. Conf. on Computer-Aided Design*, 2010, to appear.

[8] R. Nair, C. L. Berman, P. S. Hauge, and E. J. Yoffa, "Generation of performance constraints for layout," *IEEE Transactions on Computer-Aided Design*, vol. 8, pp. 860–874, Aug. 1989.