A High Level Approach to Co-Designing 3D ICs

Daniel Xing
dxing97@umd.edu
University of Maryland
College Park, Maryland, USA

Ankur Srivastava
ankurs@umd.edu
University of Maryland
College Park, Maryland, USA

Abstract

3D ICs promise increased logic density and reduced routing congestion over conventional monolithic 2D ICs. High level synthesis (HLS) tools promise reduced design complexity by approaching the design from a higher abstraction level and allow for more optimization flexibility. We propose improving timing closure of 3D ICs by co-designing the architecture and physical design by integrating HLS and 3D IC macro placement into the same holistic loop. On average our method is able to reduce estimated total negative slack (TNS) by 62% and 92% when compared to a traditional binding and placement technique for 2D and 3D ICs respectively.

CCS Concepts

• **Hardware → 3D integrated circuits**; **Resource binding and sharing**; **Partitioning and floorplanning**.

Keywords

3D IC, high level synthesis, co-design, binding, physical design

ACM Reference Format:
Daniel Xing and Ankur Srivastava. 2024. A High Level Approach to Co-Designing 3D ICs. In *61st ACM/IEEE Design Automation Conference (DAC '24), June 23–27, 2024, San Francisco, CA, USA*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3649329.3658268

1 Introduction

3D circuit integration presents a new dimension in the design and packaging of ICs by stacking conventionally fabricated 2D monolithic chips in the vertical dimension, with connections facilitated by through-silicon vias (TSVs). 3D ICs promise improved density, shortened wirelength, and improved performance [27]. However, 3D chips present additional design complexity as each chip still needs to be separately fabricated, and TSV routing overheads need to be considered.

Many other works have focused on improving the 3D IC RTL-to-GDS flow. One method is to "trick" a conventional 2D tool-flow into generating a valid 3D design [15, 23]. Others propose extending 2D design methods to 3D (i.e. so-called "true-3D" methods) [8, 12, 19]. However, these works focus their attention on the physical design of a 3D IC after the architecture has been synthesized.

A few previous works have explored integrating high level synthesis (HLS) and physical design. [21] explores a HLS-floorplanning methodology based on reinforcement learning. [26] proposes a physically-aware HLS methodology for reducing routing congestion that suggests what parts of a high level description are causing routing congestion for an engineer in-the-loop to adjust. [11] proposes a complete physically-aware HLS tool flow that optimizes binding and scheduling using a distributed-register architecture. [6] integrates HLS binding and floorplanning into the same loop for power optimization by using a probabilistic approach to estimate power from floorplan-derived net lengths. To our knowledge, no works have explored integrating HLS with 3D ICs targeting timing.

Our proposed method optimizes the datapath for timing in a 3D design. We integrate operation binding with 3D module-level placement in order to reduce TNS violations by directly optimizing binding for timing at each iteration of the binding-placement loop. We treat resources as fixed-outline 2D modules to be placed on a 3D stacked chip design, and integrate module-module interconnect information as constraints on our binding methods. Our contributions are

(1) A novel physically-aware binding method based on mixed-integer linear programming (MILP) for optimizing datapath synthesis for minimizing TNS.
(2) A dynamic weighting approach compatible with both force-directed 3D analytic placers and simulated annealing that biases placement into timing-valid module placements.
(3) We show that our integrated binding-placement co-design flow on average reduces total negative slack (TNS) by 62% and 92% when compared to a conventional placement-unaware binding and unweighted module placement method for 2D and 3D floorplans respectively.

2 Preliminaries

2.1 High Level Synthesis

HLS is a process that takes a high level behavioral description of some functionality (typically given as C, SystemC, or MATLAB code) and transforms it to RTL. There are three main steps to HLS (not necessarily performed in this order): resource allocation, operation scheduling, and operation binding. The allocation step determines how many functional modules are available for operations to use. Operation scheduling takes a dataflow graph (DFG) derived from the high level description and constrains which clock cycle each operation takes place in while meeting constraints such as timing, data dependency, and resource allocation. The output from these two steps will be a scheduled DFG, a directed acyclic graph with nodes representing operations and edges representing data dependencies. The binding step takes these scheduled operations and maps them to functional modules from the allocation step. Common binding optimization schemes including optimizing for low power [3, 24] and low area [5].

DAC '24, June 23–27, 2024, San Francisco, CA, USA

ACM ISBN 979-8-4007-0601-1/24/06
https://doi.org/10.1145/3649329.3658268

2.2 Analytical Global Placement

The global placement step during physical design concerns itself with the general placement of macros and standard cells, although in this work we will focus on the placement of fixed-size macros. Given a collection U of n nodes representing placement objects and a set of hyperedges E representing nets connecting them, the problem can be represented as a constrained optimization problem. The objective is to find a placement of nodes $\mathbf{x} = (x_1, x_2, ..., x_n, y_1, y_2, ..., y_n)^T$ such that the total wirelength $W(\mathbf{x})$ and placement density $D(\mathbf{x})$ is minimized:

$$\min_{\mathbf{x}} W(\mathbf{x}) + \lambda D(\mathbf{x}) \tag{1}$$

Where λ adjusts the tradeoff point between wirelength and density. Multiple density functions have been proposed [4, 7, 13, 14, 18]. The current state of the art uses density functions based on the electrostatic model used in the ePlace[18]/RePlAce[4] family of placers, which models each placement object as a electrostatically charged object. The density function $D(\mathbf{x})$ can then be thought of as the total electric potential given a particular placement $\mathbf{x}$, which the optimizer seeks to minimize by finding its gradient.

The wirelength of each net is traditionally modeled by the half-perimeter wirelength (HPWL) model at the global placement stage [10]. However, since analytic global placers use gradient-based optimization and the HPWL is not differentiable everywhere, various approximation methods have been proposed, including log-sum-exp and weighted-average (WA). We use the WA model in our work as it has been shown to yield lower estimation errors compared to LSE [8].

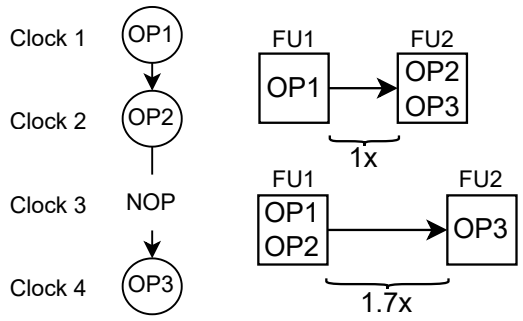
3 Module Placement at the System Design Level

Our key insight is that module binding and module placement can be tightly coupled together into a single optimization loop to improve timing. Timing delays between operations derived from the scheduled DFG can be used to inform binding about timing constraints between physically placed modules, and therefore search for resource bindings that minimize timing violations. Timing information can then be passed along to a force-directed macro placer, which we utilize using our dynamic weighting approach. We will start by illustrating the above in the following example, followed by a description of our binding method and our modification to the electrostatics-based 3D placer ePlace-3D.

3.1 An Illustrative Example

We will use Fig. 1 to illustrate how binding can affect timing closure between modules. Fig. 1a shows a scheduled DFG with three operations sequenced in a chain. OP2 is scheduled in the next clock after OP1, and OP3 is scheduled two clocks after OP2. In the first binding shown in Fig. 1b, OP1 is bound to FU1 and OP2, OP3 are bound to FU2. Since OP2 occurs directly after OP1, data needs to be transferred from FU1 to FU2 within one clock cycle. In the second binding, both OP1 and OP2 are bound to FU1, and OP3 is bound to FU2, therefore the data transfer can take up to two clock cycles.

We can then represent the two modules as floorplan blocks, as shown in Fig. 1b. Suppose T is the clock period, and each module has a propagation delay estimate of $T/2$. In the first binding, the available timing budget between FU1 and FU2 is only one clock period T, $T/2$ of which is taken up by the delay of FU1, leaving $T/2$ left for the interconnect delay between FU1 and FU2. However, the second binding leaves $3T/2$ for the interconnect delay, a



(a) Scheduled DFG **(b) Possible bindings and timing-valid module placements**

Figure 1: 3-operation linear scheduled DFG with two possible binding solutions

3x increase in available timing budget. Translated to placement, that means a 1.73x allowable increase in wirelength (due to the quadratic relationship between wire length and wire delay). If the two modules were already placed close to each other, then either binding may work. However, if placed further apart, then only the second binding may be able to meet timing closure. Conversely, if the binding is fixed to the first one shown, then the module placer has less freedom to move FU1 and FU2 if timing is to be met. This motivates the need for a binding algorithm that is aware of these physical realities.

3.2 Physically-Aware Binding

Finding a timing-valid binding based on a particular resource placement can be expressed as mixed-integer linear program (MILP). To make our formulation easy to follow, Table 1 lists the constants and variables that we use.

The first few constraints ensure that any bindings in the feasible solution space are valid bindings. The first constraint limits the number of operations that can be bound to a module in any clock to at most one:

$$\sum_o b_{o,u} S_{o,t} \leq 1 \quad \forall u \in U, t \in T \tag{2}$$

Additionally, every operation must be bound to a module:

$$\sum_u b_{o,u} = 1 \quad \forall o \in O \tag{3}$$

From there we can calculate c, which captures every pair of operation bindings:

$$b_{x,i} b_{y,j} = c_{x,i,y,j} \quad \forall x, y \in O \quad i, j \in U \tag{4}$$

Note that the above constraint is not linear, but because b is a binary value, it can be rewritten as a set of linear constraints.

From these prerequisite constraints we can begin to incorporate physical timing information. If we only want to find a binding that meets timing constraints with no negative slack, the following constraint could be used:

$$\sum_{i,j} c_{x,i,y,j} D_{i,j} \leq P_{x,y} \quad \forall x, y \in O \tag{5}$$

This constraint finds the relevant physical delay $D_{i,j}$ between modules i and j based on what physical modules x and y are bound to via $c_{x,i,y,j}$ and ensures its less than the schedule-defined delay constraint $P_{x,y}$. Note that we assume every operation's data dependencies happen in previous clocks. If multiple operations can

Table 1: MILP variables, constants, and definitions

Notation	Definition
U	The set of functional modules to be placed
O	The set of operations in the DFG
T	The number of clock cycles in the schedule
$D_{i,j}$	Delay matrix representing the estimated minimum achievable delay between modules i and j extracted from a module-level placement, with the intrinsic delay of module i subtracted.
$P_{x,y}$	Maximum allowed delay between operations x and y, derived from the schedule and target clock frequency. Only specified for operations with direct data dependencies on each other.
$S_{o,t}$	1 if operation o is scheduled for clock t, 0 otherwise
$b_{o,u}$	Binary variable indicating operation binding. 1 if operation o is bound to resource u, 0 otherwise. Only defined for valid o, u mappings.
$c_{x,i,y,j}$	Binary variable capturing how operation binding affects module-to-module data movement. 1 if operation x is bound to module i and operation y is bound to module j.
$s_{i,j}$	Non-negative variable capturing the negative slack between modules i and j. A value of 0 indicates positive slack, any positive values of s indicates the amount of negative slack on the path between i and j
ws	Non-negative variable representing the worst negative slack

happen in the same clock, 5 can be modified by considering a sequence of nodes that occur in the same clock instead of a single node x.

3.2.1 Objective Functions The MILP as-written so far with a objective function of 0 would force the solver to find a binding that meets timing as-is, with any bindings with negative slack being infeasible solutions. However, if the physical placement is especially bad or the target clock frequency too constrained, the feasible set may end up being empty. Additionally, we would like some kind of cost function to guide our modified placement tool. A few choice objective functions are available to us:

3.2.2 Total Negative Slack Total negative slack captures the total slacks of all failing timing paths, which can be a better indicator of overall placement quality [17]. The negative slack of each module-module path $s_{i,j}$ can be found by modifying 5 to be:

$$\sum_{i,j} c_{x,i,y,j} D_{i,j} - P_{x,y} <= s_{i,j} \quad \forall x, y \in O \quad \forall i, j \in U \tag{6}$$

Note that we also constrain $s_{i,j}$ to be non-negative to prevent positive slacks from being captured. The optimization objective then is

$$\min_{b,c,s} \sum_{i,j} s_{i,j} \tag{7}$$

subject to eqs. (2) to (4) and (6).

3.2.3 Worst Negative Slack A similar objective function can be used for worst negative slack. First we add a new constraint to find the worst negative slack value ws:

$$s_{i,j} <= ws \quad \forall i, j \in U \tag{8}$$

The objective function is simply

$$\min_{b,s,ws} ws \tag{9}$$

subject to eqs. (2) to (4), (6) and (8).

3.2.4 Number of Failing Timing Paths The number of failing timing paths can be a useful global metric for determining how many paths need fixing. Finding the number of failing timing paths (i.e. paths between modules with negative slack) is equivalent to minimizing the zero "norm" of s, which is not convex. A common relaxation of the zero norm is to use the 1-norm, however the 1-norm s is equivalent to TNS, so TNS should be used instead.

3.3 Timing-Weighted Wirelength Cost Function

Our MILP formulation generates a valid datapath architecture along with timing constraints at connections between modules. Based on these timing constraints, we can estimate the longest wire that can reasonably accommodate a timing constraint for each net. For each net $e \in E$, we have a target net length $target_e$. Total wirelength is the most common metric used for determining the quality of a global placement. The most commonly used wirelength metric for global placers $W(\mathbf{x})$ is the HPWL:

$$W(\mathbf{x}) = \sum_{e \in E} \text{HPWL}_e(\mathbf{x}) \tag{10}$$

$$= \sum_{e} \text{HPWL}_{e_x}(\mathbf{x}) + \text{HPWL}_{e_y}(\mathbf{x}) + \text{HPWL}_{e_z}(\mathbf{x}) \tag{11}$$

We propose a flexible dynamic weighting method that increases the weight of timing-constrained nets as the wirelength approaches the timing-derived wirelength target by using an exponential function:

$$W(\mathbf{x}) = \sum_{e} HPWL_e(\mathbf{x})(1 + \exp((HPWL_e(\mathbf{x}) - target_e)/\gamma) \tag{12}$$

where γ controls the penalty of exceeding a particular timing-derived wirelength target $target_e$. For 3D ICs, we account for the delay contribution of TSVs by weighting the Z dimension by the expected RC delay of a single TSV. The advantage of using this weighting method over previous timing-driven placement methods such as [17] is that we can directly incorporate a timing budget into the placer's cost function without needing to go through a costly STA step.

Because we are working with wire delays and not path delays, we can determine values of $target_e$ before we begin module placement, and the fully differentiable nature of our weight function means the modified wirelength term is flexible enough to be applied to any placer that uses wirelength in its cost function. The next two sections demonstrate how we apply this weight to the two major 3D components of ePlace-3D applicable to modules: the simulated annealing-based macro legalization step and the force-directed 3D global placement step.

3.3.1 Timing-Weighted Wirelength for Simulated Annealing The cost function used for low temperature simulated annealing in ePlace is similar to that of the analytic global placer's cost function (without the overlap term):

$$\text{HPWL}(\mathbf{x}) + \mu_D D(\mathbf{x}) \tag{13}$$

where $D(\mathbf{x})$ is the electrostatics-based density function and μ_D is statically set to $\text{HPWL}(\mathbf{x})/D(\mathbf{x})$. Incorporating our timing-based

Algorithm 1 Timing-directed co-design algorithm

Input: SDFG, resource allocation, clock period in
Output: placement out
Initialization :
1: Initialize placement delays to 0
Optimization Loop:
2: **while** iteration limit not reached **do**
3: binding = bind(calculate_delays(placement)
4: macro_netlist = generate_datapath(binding)
5: placement = ePlace-3D(macro_netlist)
6: **if** TNS(placement) == 0 **then**
7: **return** placement
8: **end if**
9: **end while**
10: **return** placement

wirelength weight is as simple as modifying the HPWL term to include the exponential weight for timing-constrained nets:

$$\sum_e \text{HPWL}_e(\mathbf{x})(1+\exp((\text{HPWL}_e(\mathbf{x})-target_e)/\gamma)+\mu_D D(\mathbf{x}) \quad (14)$$

Since the simulated annealing step directly calculates the HPWL for determining the total improvement at every movement step, no further modifications are necessary, greatly simplifying implementation.

3.3.2 Timing-Weighted Wirelength for Force-Directed Placement ePlace-based global placers determine the direction to move each placement object by finding the gradient of the cost function. Since HPWL is not differentiable everywhere, approximations of it have been proposed, the leading of which is the weighted-average $WA(\mathbf{x}$ approximation [8]. We can apply our weighting term to the cost function (1) by substituting HPWL with its WA approximation:

$$\sum_{e\in E} \text{WA}_e(\mathbf{x})(1+\exp((\text{WA}_e(\mathbf{x})-target_e)/\gamma)+\lambda D(\mathbf{x}) \quad (15)$$

Since our weight is continuous everywhere when applied to the WA approximation, a closed form expression for the gradient of our weighted wirelength term $W(\mathbf{x})$ can be found:

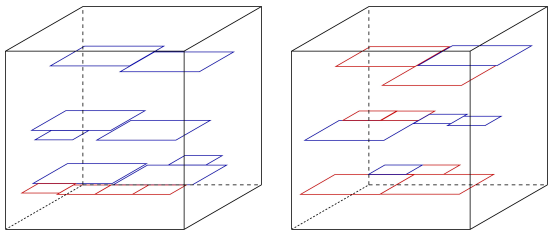
$$\nabla\text{WA}(\mathbf{x})(1+\exp((\text{WA}_e(\mathbf{x})-target_e)/\gamma)(1+\text{WA}(\mathbf{x})/\gamma) \quad (16)$$

making implementation relatively straightforward. Additionally, the gradual change in gradient as WA approaches $target_e$ means wirelengths cannot "rubber-band" and oscillate around $target_e$.

4 The Overall Co-Design Flow

Algorithm 1 describes our entire 3D IC co-design flow. We first initialize module-module delays to 0 (line 1), so that the MILP solver can see if the target clock period is even feasible with just the known module propagation delays. In the loop, we first call the binder (line 3) and find a binding that minimizes TNS. After binding, we generate values for $target_e$ based on the length of a wire with the same RC delay as the timing budget for each net. We pass the generated datapath and target wirelengths to the 3D force-directed placer (line 5). If the placement returned by the placer has 0 TNS at this point, we can return this placement (lines 6, 7). Otherwise the loop continues until we reach an iteration limit (line 2).

The designer would need to provide a scheduled DFG, a resource allocation, and module outlines and delay estimates for each allocated resource. The output will be a timing-optimized placement



(a) Baseline 3D Placement **(b) Co-designed 3D Placement**

Figure 2: 3D Module Placements for the `motion2` benchmark. An aggressive clock period of 3ns was set for the co-design flow. Large modules are 32-bit multipliers, small modules are 32 bit adders.

along with estimated slack values along each net connecting modules in the floorplan. As we operate at a higher abstraction level than standard cells, we assume TSVs have zero physical width and do not take up any placement area. The placement can then be passed along to more detailed physical design tools.

5 Results

To evaluate our binding and module placement co-design flow, we extract functions from the MediaBench benchmark suite [16] for use with HLS. The DFG of each benchmark was extracted with SUIF and scheduled with a path-based scheduler [22]. As the DFGs extracted from these benchmarks (and hence the required module allocation) were small, we increased their size by duplicating the base DFG and appending them together to create a larger benchmark to stress test our method. We used a security-aware binding algorithm as our baseline reference binding method [28]. We limited the number of iterations of our co-design flow to 10 to keep runtimes manageable.

We used ePlace-3D [19] as our true 3D placer, although we note that any placer that utilizes HPWL (including RePlAce [4]) in its cost function can be used. As ePlace-3D is designed for mixed-size 3D placement, we modify its placement flow to tailor it for module-level placement by skipping the 3D and 2D global standard cell placement steps. Data for the conventional method used our module-optimized ePlace-3D implementation with no wirelength targets set. We empirically found that a γ value of 20000 yielded the best results for our benchmarks without causing numerical issues. As we are performing high level module placement and not a more detailed global placement step, we set the pin locations of all module-module nets to be at the center of each module's macro, and the macro's length/width is added to HPWL targets accordingly. In a similar vein, we model TSVs as having zero volume.

Gurobi was used to implement the MILP binding optimization program with a timeout limit of 20 minutes. Cadence Genus and Innovus were used to generate module macro blocks from RTL synthesized with Nangate45 [25] with an aspect ratio of 1. OpenSTA [1] was used to characterize each functional module's timing delay. OpenROAD [2] was used to initialize the die outline with a placement density of 70%, and RosettaStone [9] was used to translate LEF/DEF files from OpenROAD to the academic Bookshelf format used by ePlace-3D.

Figure 3 compares how our co-design technique compares against a conventional datapath synthesis and placement flow over all clock

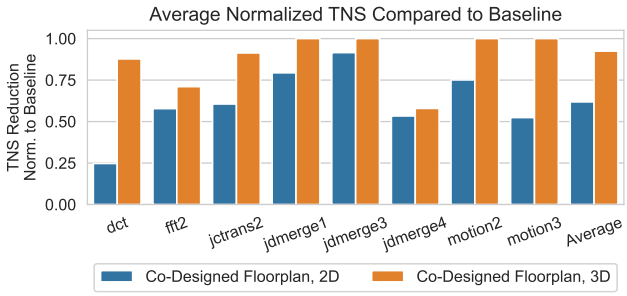


Figure 3: Reduction in total negative slack using a co-designed datapath and physical placement normalized to the corresponding 2D or 3D conventional binding and module placement approach. Data shown are averaged over all clock periods tested.

period targets tested. Data are for 3D floorplans are normalized against the TNS obtained from the conventional approach in 3D, and similarly for 2D. Our co-design flow was able to significantly reduce TNS in all cases, especially in 3D floorplanned test cases, while TNS reduction in the 2D case was less. This is due to the lesser degree of placement freedom in 2D, causing higher congestion and longer wirelengths which has been well documented in previous works [20].

Figures 4a and 4b show how TNS changes with tighter clock period constraints for 2D and 3D respectively. In all cases except one, our method was able to meet or beat the TNS of a conventional flow. Moreover, our co-design flow was able to reach 0 TNS in some cases where the conventional approach could not. The only case where the co-design flow did worse than the baseline was the `dct` benchmark at a clock period of 6ns for a 2D design, which upon further investigation was due to the iteration limit of 10 we used. By increasing the iteration limit to 20 we were able to find a module placement that yielded a TNS of 133ns, marginally better than the baseline TNS of 135ns. The small difference in TNS between the conventional binding and placement and our method for the other target clock periods in 2D `dct` tests also indicates that the conventional method was already close to timing optimal.

6 Conclusion

In this work we present a HLS physical placement co-design flow for use with 3D ICs. Our method improves TNS significantly by proposing a physically-aware timing-optimized binding method in conjunction with timing-informed module placement. We apply the co-design flow on DFGs extracted from MediaBench and compare TNS values for a conventional flow where HLS and placement take place separately and our flow which integrates physical placement with HLS. Overall our flow was able to reduce TNS by 62% and 92% on average when compared to the conventional approach for 2D and 3D floorplans respectively.

References

[1] 2023. Parallax Static Timing Analyzer. https://github.com/The-OpenROAD-Project/OpenSTA original-date: 2018-09-28T15:45:57Z.

[2] T Ajayi, D Blaauw, R Dreslinski, Mateus Fogaca, S Hashemi, et al. 2019. OpenROAD: Toward a Self-Driving, Open-Source Digital Layout Implementation Tool Chain. *Proceedings of Government Microcircuit Applications and Critical Technology Conference* (2019).

[3] D. Chen and J. Cong. 2004. Register binding and port assignment for multiplexer optimization. In *ASP-DAC 2004: Asia and South Pacific Design Automation Conference 2004 (IEEE Cat. No.04EX753)*. 68–73. https://doi.org/10.1109/ASPDAC.2004.1337542

[4] Chung-Kuan Cheng, Andrew B. Kahng, Ilgweon Kang, and Lutong Wang. 2019. RePlAce: Advancing Solution Quality and Routability Validation in Global Placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 38, 9 (Sept. 2019), 1717–1730. https://doi.org/10.1109/TCAD.2018.2859220 Conference Name: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.

[5] Chu-Yi Huang, Yen-Shen Chen, Yan-Long Lin, and Yu-Chin Hsu. 1990. Data path allocation based on bipartite weighted matching. In *27th ACM/IEEE Design Automation Conference*. IEEE, Orlando, FL, USA, 499–504. https://doi.org/10.1109/DAC.1990.114907

[6] A. Davoodi and A. Srivastava. 2005. Power-driven simultaneous resource binding and floorplanning: a probabilistic approach. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 13, 8 (Aug. 2005), 934–942. https://doi.org/10.1109/TVLSI.2005.853618 Conference Name: IEEE Transactions on Very Large Scale Integration (VLSI) Systems.

[7] Hans Eisenmann and Frank M. Johannes. 1998. Generic global placement and floorplanning. In *Proceedings of the 35th annual conference on Design automation conference - DAC '98*. ACM Press, San Francisco, California, United States, 269–274. https://doi.org/10.1145/277044.277119

[8] Meng-Kai Hsu, Valeriy Balabanov, and Yao-Wen Chang. 2013. TSV-Aware Analytical Placement for 3-D IC Designs Based on a Novel Weighted-Average Wirelength Model. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 32, 4 (April 2013), 497–509. https://doi.org/10.1109/TCAD.2012.2226584 Conference Name: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.

[9] Andrew B. Kahng, Minsoo Kim, Seungwon Kim, and Mingyu Woo. 2022. RosettaStone: Connecting the Past, Present, and Future of Physical Design Research. *IEEE Design & Test* 39, 5 (Oct. 2022), 70–78. https://doi.org/10.1109/MDAT.2022.3179247 Conference Name: IEEE Design & Test.

[10] A. Kennings and I. Markov. 2000. Analytical minimization of half-perimeter wirelength. In *Proceedings 2000. Design Automation Conference. (IEEE Cat. No.00CH37106)*. 179–184. https://doi.org/10.1109/ASPDAC.2000.835093

[11] Daehong Kim, Jinyong Jung, Sunghyun Lee, Jinhwan Jeon, and Kiyoung Choi. 2001. Behavior-to-placed RTL synthesis with performance-driven placement. In *IEEE/ACM International Conference on Computer Aided Design. ICCAD 2001. IEEE/ACM Digest of Technical Papers (Cat. No.01CH37281)*. 320–325. https://doi.org/10.1109/ICCAD.2001.968640 ISSN: 1092-3152.

[12] Dae Hyun Kim, Krit Athikulwongse, and Sung Kyu Lim. 2013. Study of Through-Silicon-Via Impact on the 3-D Stacked IC Layout. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 21, 5 (May 2013), 862–874. https://doi.org/10.1109/TVLSI.2012.2201760 Conference Name: IEEE Transactions on Very Large Scale Integration (VLSI) Systems.

[13] Myung-Chul Kim and Igor L. Markov. 2012. ComPLx: A Competitive Primal-dual Lagrange Optimization for Global Placement. In *Proceedings of the 49th Annual Design Automation Conference*. ACM, San Francisco California, 747–752. https://doi.org/10.1145/2228360.2228496

[14] Myung-Chul Kim, Natarajan Viswanathan, Charles J. Alpert, Igor L. Markov, and Shyam Ramji. 2012. MAPLE: multilevel adaptive placement for mixed-size designs. In *Proceedings of the 2012 ACM international symposium on International Symposium on Physical Design*. ACM, Napa California USA, 193–200. https://doi.org/10.1145/2160916.2160958

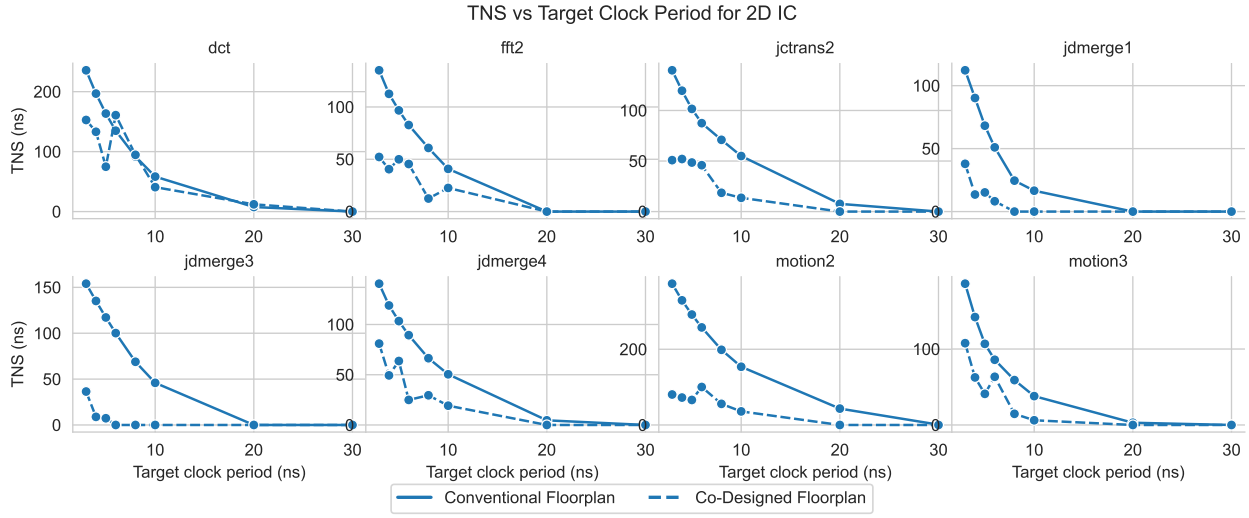
[15] Bon Woong Ku, Kyungwook Chang, and Sung Kyu Lim. 2018. Compact-2D: A Physical Design Methodology to Build Commercial-Quality Face-to-Face-Bonded 3D ICs. In *Proceedings of the 2018 International Symposium on Physical Design (ISPD '18)*. Association for Computing Machinery, New York, NY, USA, 90–97. https://doi.org/10.1145/3177540.3178244

[16] Chunho Lee, M. Potkonjak, and W.H. Mangione-Smith. 1997. MediaBench: a tool for evaluating and synthesizing multimedia and communications systems. In *Proceedings of 30th Annual International Symposium on Microarchitecture*. 330–335. https://doi.org/10.1109/MICRO.1997.645830 ISSN: 1072-4451.

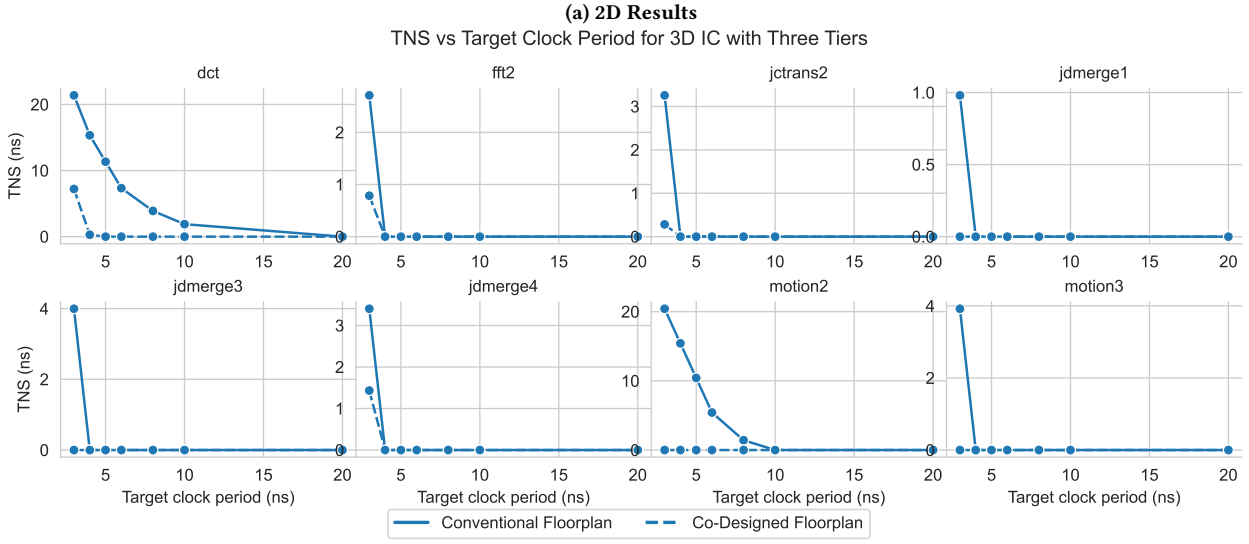
[17] Peiyu Liao, Siting Liu, Zhitang Chen, Wenlong Lv, Yibo Lin, et al. 2022. DREAMPlace 4.0: Timing-driven Global Placement with Momentum-based Net Weighting. In *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 939–944. https://doi.org/10.23919/DATE54114.2022.9774725 ISSN: 1558-1101.

[18] Jingwei Lu, Hao Zhuang, Pengwen Chen, Hongliang Chang, Chin-Chih Chang, et al. 2015. ePlace-MS: Electrostatics-Based Placement for Mixed-Size Circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 34, 5 (May 2015), 685–698. https://doi.org/10.1109/TCAD.2015.2391263 Conference Name: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.

[19] Jingwei Lu, Hao Zhuang, Ilgweon Kang, Pengwen Chen, and Chung-Kuan Cheng. 2016. ePlace-3D: Electrostatics based Placement for 3D-ICs. In *Proceedings of the 2016 on International Symposium on Physical Design*. ACM, Santa Rosa California USA, 11–18. https://doi.org/10.1145/2872334.2872361



(a) 2D Results



(b) 3D Results

Figure 4: Breakdown of how TNS increases as the target clock period decreases for both our co-design approach and a conventional binding and module placement toolflow when applied to a 2D chip. All units are in nanoseconds.

[20] Yi-Chen Lu, Sai Surya Kiran Pentapati, Lingjun Zhu, Kambiz Samadi, and Sung Kyu Lim. 2020. TP-GNN: A Graph Neural Network Framework for Tier Partitioning in Monolithic 3D ICs. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*. 1–6. https://doi.org/10.1109/DAC18072.2020.9218582 ISSN: 0738-100X.

[21] Marie-Claire Melhem. [n. d.]. Layout Driven Binding In High Level Synthesis Using Reinforcement Learning. ([n. d.]).

[22] S. Ogrenci Memik, E. Bozorgzadeh, R. Kastner, and M. Sarrafzadeh. 2001. A super-scheduler for embedded reconfigurable systems. In *IEEE/ACM International Conference on Computer Aided Design. ICCAD 2001. IEEE/ACM Digest of Technical Papers (Cat. No.01CH37281)*. 391–394. https://doi.org/10.1109/ICCAD.2001.968653 ISSN: 1092-3152.

[23] Shreepad Panth, Kambiz Samadi, Yang Du, and Sung Kyu Lim. 2017. Shrunk-2-D: A Physical Design Methodology to Build Commercial-Quality Monolithic 3-D ICs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 36, 10 (Oct. 2017), 1716–1724. https://doi.org/10.1109/TCAD.2017.2648839 Conference Name: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.

[24] A. Stammermann, D. Helms, M. Schulte, A. Schulz, and W. Nebel. 2003. Binding allocation and floorplanning in low power high-level synthesis. In *ICCAD-2003. International Conference on Computer Aided Design (IEEE Cat. No.03CH37486)*. 544–550. https://doi.org/10.1109/ICCAD.2003.159736

[25] James E. Stine, Ivan Castellanos, Michael Wood, Jeff Henson, Fred Love, et al. 2007. FreePDK: An Open-Source Variation-Aware Design Kit. In *2007 IEEE International Conference on Microelectronic Systems Education (MSE'07)*. 173–174. https://doi.org/10.1109/MSE.2007.44

[26] Masato Tatsuoka, Ryosuke Watanabe, Tatsushi Otsuka, Takashi Hasegawa, Qiang Zhu, et al. 2015. Physically aware High Level Synthesis design flow. In *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*. 1–6. https://doi.org/10.1145/2744769.2744893 ISSN: 0738-100X.

[27] Yuan Xie. 2010. Processor Architecture Design Using 3D Integration Technology. In *2010 23rd International Conference on VLSI Design*. 446–451. https://doi.org/10.1109/VLSI.Design.2010.60 ISSN: 2380-6923.

[28] Michael Zuzak, Yuntao Liu, and Ankur Srivastava. 2021. A Resource Binding Approach to Logic Obfuscation. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*. 235–240. https://doi.org/10.1109/DAC18074.2021.9586179 ISSN: 0738-100X.