Work-in-Progress: An SMT-Based, Correct-by-Construction Place-and-Route Framework

PHD FORUM SUBMISSION

Edward Wang
Dept. of EECS
MIT
Cambridge, MA, USA
edwardw@csail.mit.edu

***Abstract*—Place-and-route (P&R) plays a key bottleneck in integrated circuit design, and existing approaches for P&R fall short in correctness and methodology. We introduce an SMT-based approach which addresses these concerns and introduces some new possibilities for improved co-optimization of designs. We introduce the system design and problem encoding and show some preliminary results.**

***Index Terms*—place-and-route, P&R, EDA, formal, SMT, solvers, synthesis, correct-by-construction**

I. Introduction

Integrated circuit design costs remain unsatisfactorily high [1] [2]. Promising novel design methodologies such as agile hardware design [3] are hindered by a bottleneck in physical design. [4] [5] [2] Place-and-route (P&R) (shown in Figure 1) is the key problem in physical design. Two significant factors in the difficulty of P&R are the lack of correct-by-construction approaches and limited adoption of improved software engineering methodologies [2].

In this work, we introduce an SMT-based compiler framework for P&R. With this approach, we no longer have to manually write tedious search/matching/routing algorithms. Instead, we leverage advanced modern SMT solver capabilities for ameliorated correctness, security, and effort. [2] Moreover, the correct-by-construction nature of SMT as formal methods eliminates the time-consuming step of LVS [2]. Our approach also reduces software engineering efforts. We replace separate traditional EDA tools (i.e., placer, router) with a single tool for enhanced security and lower debugging time.

We also open up possibilities for co-optimization between placement and routing by providing the solver with the freedom to co-adjust and optimize variables traditionally fixed and decided by separate tools. Additionally, our approach also allows for the trade-off between optimality and design time, as well as the use of soft constraints and designer insight in the pre-placement stage. [2] SMT has shown promise in other domains such as computer graphics [6] and web layout rendering. [7]

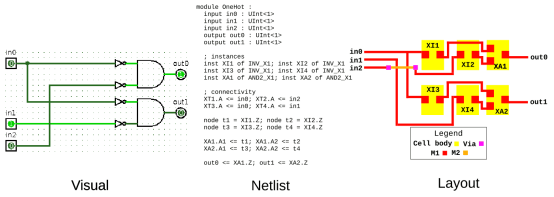

Fig. 1. The key problem in physical design is place-and-route which converts a netlist of cells into a physical layout.

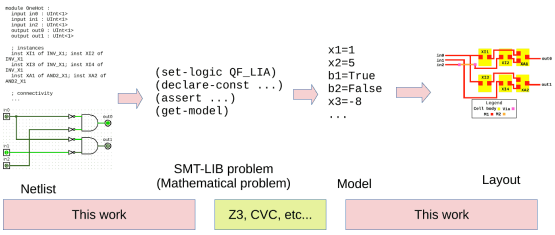

Fig. 2. A summary of our system as a whole.

II. System Design

Our compiler takes in an input netlist and encodes the place-and-route problem as an SMT-LIB formulation, as shown in Figure 2.

Figure 3 illustrates our compiler flow. The input netlist

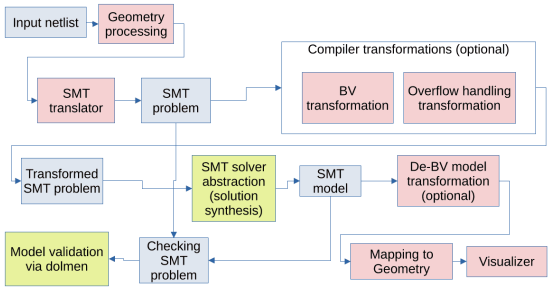

Fig. 3. Overview of our compiler flow.

is first parsed and converted into geometry, which includes cells, wires/routes, placement and routing grids, and ports. This is followed by an initial translation into an SMT problem. We also incorporate several compiler transformations, such as bitvector and overflow handling. These transformations are optional and can be enabled or disabled to enhance performance. Using a solver abstraction layer, we then make one or more calls to solves like [8] [9]. The resulting model is then reverse transformed to match the original SMT problem and validated using dolmen [10]. Finally, the solution is mapped back to geometry, producing a layout and visualization.

Key contributions include a geometry engine for processing and handling rectilinear shapes and basic operations and constraints. We also introduce encodings for VLSI geometry objects such as cells, wires/routes, placement and routing grids, and ports. We have also incorporated optimization strategies and transformational techniques to improve the performance. Finally, we feature an iterative CEGAR/counterexample-style meta-solver for efficient solving.

III. Encoding

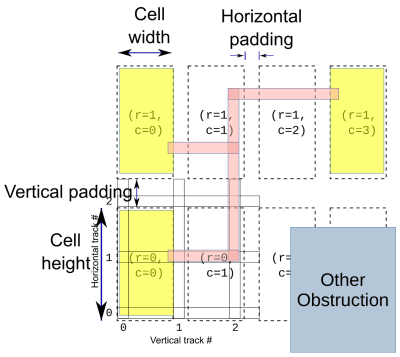

Fig. 4. Placement and routing constraints are simultaneously considered, a significant feature enabled by SMT.

To being with, we follow two conventions for encoding which are commonly used [11]: 1) the Manhattan grid using integer variables, and 2) rectilinear layout. We also use grid-based placement and a channel routing grid.

The first phase of our encoding process involves converting from a netlist to a geometry-based representation. This is necessary because the standard RTL/netlist AST is inconvenient for physical design because in physical design, we are primarily concerned with the geometry of components rather than their logical relationships. Thus, in the second phase of our encoding, we convert from geometry to rectangles.

Rectangles are defined as follows:

- width $\in \mathbb{Z}$
- height $\in \mathbb{Z}$
- ll $\in (\mathbb{Z}, \mathbb{Z})$

This allows us to then very easily define rectangle non-intersection using four disjunctions. Rectangle intersection is a key constraint in this problem.

The channel-based routing grid is parameterised by `grid_width` which represents the width of each routing track, and `grid_pitch`, representing the repeating width. Each routing segment, as defined below, can be mapped 1:1 to a rectangle.

- Track number - $t_n \in \mathbb{Z}$
- Span - $s \in (\mathbb{Z}, \mathbb{Z})$
- Orientation - $o \in \mathbb{B}$

Routes are sequences of connected routing segments. Multi-point routes are represented as a sequence of routes which all join at a central point. Standard cells can have complex layouts with areas connected to different nets as well as carveouts. To handle this complexity, we decompose standard cells into basic rectangles as well. We use a placement grid where the placement of cells is restricted by ”snapping” to certain locations. In digital circuits, cells must be placed in an orderly manner for many considerations including power straps.

Our placement and routing are overlaid and done simultaneously, as shown in Figure 4. This is radically different from traditional EDA. Placement and routing grids co-exist on the same layer, and all objects, including cells and routes, exist simultaneously as well. This has implications for object consistency, as well. For example, if a placement location is occupied by a wire or obstruction, it becomes unavailable. Similarly, if a routing track is occupied by another object, it is also unavailable.

We ensure layer consistency and object consistency as fundamental constraints. We do not use uninterpreted functions and instead opt for a fixed number of variables without quantifiers to improve performance. This use of a fixed model size may limit the complexity of designs or overapproximate them. Our compiler provides meta-parameters, such as the number of segments allowed per linear route, in order to accommodate for this. Finally, our approach also considers the future incorporation of clock trees and power grids but leaves their implementation to future work.

IV. Preliminary Results

benchmarks unsolved (**timeout >=45s**) summary
Lower is better

Configuration	# unsolved
Z3 integer (naive)	12
CVC4 integer	6

Configuration	# unsolved
CVC5 int	4
CVC5 int cold-iter	18
CVC5 hot incr	4
CVC5 bitvec eager bb	12

Configuration	# unsolved
Yices integer	0
Yices cold iterative	3
Yices hot incr	TODO

Configuration	# unsolved
MathSAT integer	5
MathSAT cold iterative	28(all)
MathSAT hot incr	TODO

Configuration	# unsolved
Bitwuzla	2
Bitwuzla local search	28(all)
Bitwuzla cold iterative	26
Bitwuzla hot incr	TODO

Fig. 5. Overview of our compiler flow.

We experiment with a 1-layer 5-stage inverter chain. At the moment, our focus has been on answering the first research question, ”RQ1: What is the best solver configuration for this problem?”. Results are shown in Figure 5. However, we are continuously working on answering both ”RQ2: How does the performance scale?” and ”RQ3: Comparison with existing tools.”

REFERENCES

[1] B. Bailey, "What will that chip cost?" http://web.archive.org/web/20240719072950/https://semiengineering.com/what-will-that-chip-cost/, Oct. 2023. [Online]. Available: http://web.archive.org/web/20240719072950/https://semiengineering.com/what-will-that-chip-cost/

[2] E. Wang, L. Daniel, Y. Zohar, and C. Barrett, "How I learned to stop worrying and love physical design," in *2024 Workshop on Languages, Tools, and Techniques for Accelerator Design (LATTE)*, 2024. [Online]. Available: http://capra.cs.cornell.edu/latte24/paper/5.pdf

[3] Y. Lee, A. Waterman, H. Cook, B. Zimmer, B. Keller, A. Puggelli, J. Kwak, R. Jevtic, S. Bailey, M. Blagojevic, P.-F. Chiu, R. Avizienis, B. Richards, J. Bachrach, D. Patterson, E. Alon, B. Nikolic, and K. Asanovic, "An Agile Approach to Building RISC-V Microprocessors," *IEEE Micro*, vol. 36, no. 2, pp. 8–20, Mar. 2016.

[4] E. Wang, C. Schmidt, A. Izraelevitz, J. Wright, B. Nikolić, E. Alon, and J. Bachrach, "A methodology for reusable physical design," in *2020 21st International Symposium on Quality Electronic Design (ISQED)*, 2020, pp. 243–249.

[5] H. Liew, D. Grubb, J. Wright, C. Schmidt, N. Krzysztofowicz, A. Izraelevitz, E. Wang, K. Asanović, J. Bachrach, and B. Nikolić, "Hammer: a modular and reusable physical design flow tool: invited," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, ser. DAC '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 1335–1338. [Online]. Available: https://doi.org/10.1145/3489517.3530672

[6] J. Whitehead, "Spatial layout of procedural dungeons using linear constraints and SMT solvers," in *Proceedings of the 15th International Conference on the Foundations of Digital Games*, ser. FDG '20. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: https://doi.org/10.1145/3402942.3409603

[7] J. Liu, Y. Chen, E. Atkinson, Y. Feng, and R. Bodik, "Conflict-driven synthesis for layout engines," *Proc. ACM Program. Lang.*, vol. 7, no. PLDI, jun 2023. [Online]. Available: https://doi.org/10.1145/3591246

[8] H. Barbosa, C. Barrett, M. Brain, G. Kremer, H. Lachnitt, M. Mann, A. Mohamed, M. Mohamed, A. Niemetz, A. Nötzli, A. Ozdemir, M. Preiner, A. Reynolds, Y. Sheng, C. Tinelli, and Y. Zohar, "cvc5: A versatile and industrial-strength smt solver," in *Tools and Algorithms for the Construction and Analysis of Systems*, D. Fisman and G. Rosu, Eds. Cham: Springer International Publishing, 2022, pp. 415–442.

[9] A. Niemetz and M. Preiner, "Bitwuzla," in *Computer Aided Verification*, C. Enea and A. Lal, Eds. Cham: Springer Nature Switzerland, 2023, pp. 3–17.

[10] G. Bury, "Dolmen: A validator for SMT-LIB and much more." in *2021 SMT Workshop Proceedings*, 2021, pp. 32–39.

[11] A. B. Kahng, J. Lienig, I. L. Markov, and J. Hu, *VLSI physical design: from graph partitioning to timing closure*. Springer, 2011, vol. 312.