Democratizing Hardware Verification By Formalizing High-Level Abstractions

Luisa Cicolini

Hardware verification is a fundamental step of the hardware design process, preventing expensive and potentially critical mistakes. As the complexity of hardware designs increases, so does the need for more efficient verification strategies that can streamline the overall process. The CIRCT Electronic Design Automation (EDA) stack has recently shown that explicit representations of higher-level, domain-specific hardware enable the effective generation of optimized Verilog code. Such higher-level hardware representations can be a game changer in verifying the designs progressively as they are lowered and exploring novel, domain-specific verification approaches. In fact, taking advantage of these representations for verification purposes remains a challenge due to the lack of formalized semantics for high-level abstractions, which often rely on domain-specific structures such as Finite State Machines (FSMs). In my Ph.D. I will exploit modern multilevel EDA approaches to formally verify the semantics of high-level, domain-specific hardware abstractions, making the manipulation and verification of their semantics scalable and white-box. In particular, I aim to (1) formalize higher-level abstractions to lower-level verification efforts, (2) verify transformations from several high-level level abstractions to low-level ones (e.g. in the CIRCT ecosystem) (3) explore domain-specific decision procedures and reasoning frameworks for the verification of higher-level abstractions beyond SMT solvers, thus such that the verification evolves as the design is lowered. This work will make hardware verification accessible in the early stages of the design process, in a white-box framework that can take advantage of different domain-specific abstractions.

1 INTRODUCTION

Hardware design is inherently complex, requiring a thorough understanding of numerous components and tools. Verification is crucial in the design process as it can uncover expensive and potentially critical mistakes before the tape-out.

The CIRCT project [2, 13] introduces a novel concept of hardware compiler, comprising different domain-specific abstractions (*dialects*), frontends and backends that make the compilation of a design progressive, through numerous lowerings through different dialects. Exploiting these abstractions allows the designers to reason about different aspects of the design separately and introduce various optimizations at different levels of abstraction. In brief, CIRCT [2, 13] allows the progressive compilation of high-level abstractions, such as Finite State Machines, into lower-level representations, such as Verilog, via different lowerings. This approach proved successful in increasing the reusability of the code and introducing specific optimizations at different abstraction levels, by relying on the progressive lowering of the code. Being able to optimize and reason about a design at various abstraction levels is especially appealing from a hardware designer's perspective. Moreover, CIRCT's higher-level abstractions provide valuable information about the design's structure, which is available to perform verification tasks at a higher level, verify the compilation itself, and extract useful information that can serve lower-level verification tools [11, 15, 17].

Taking advantage of high-level abstractions is fundamental to building a verification toolchain that fully exploits the unique characteristics of the CIRCT ecosystem. This allows verification tools to leverage the expressive power of different representations, for example, dataflow and FSM. However, extracting useful high-level information remains a complex task due to the scarce formalization and the lack of precise semantics for CIRCT dialects. Overall, accurately formalizing the semantics of such diverse dialects and abstractions remains a significant challenge and is critical for verifying the correctness of the designs, optimizations and lowerings involved in the compilation.
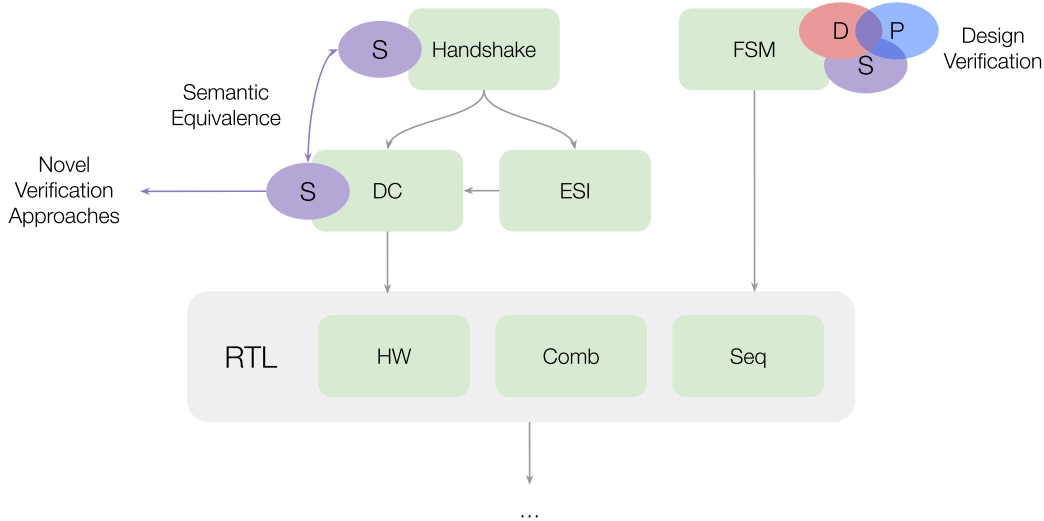
Figure 1: Formalizing the semantics (S) of CIRCT dialects in Lean4 allows us to (1) verify the compilation and equivalence between different abstractions, (2) explore novel verification approaches in a verified environment, (3) verify whether a property (P) holds for a design (D) at a certain level.

2 RELATED WORK

Given the paramount importance of verification in hardware design, numerous works tackled the formalization and definition of semantics in hardware description languages, considering different levels of abstraction [18]. Koika [9] directly derives from Bluespec [22] and features novel and deterministic operational semantics, including a verified compiler to circuits. In particular, Koika highlights the importance of cycle-accurate description (and semantics) when dealing with the performance of circuits without compromising on the straightforward description of their functional properties. In particular, the authors highlight how traditional HDLs and HLS compilers lack precise semantics, making it especially hard to describe complex interactions between different modules. ReWire [23] introduces a functional programming language with a compiler that translates the high-level description of a design into working hardware relying on the language's semantics to describe its behaviour. Chisel [4] is another powerful structural language embedded in Scala, also used as a frontend for CIRCT. Finally, Bernstein et al. [6] suggest exploiting abstract interpretation to formalize and represent different hardware abstractions, which require focusing on different aspects of the semantics. The authors use abstract interpretation to describe the semantics at different levels of abstraction, mainly focusing on the representations used in the early phase of the design when reasoning at the latency-insensitive level of the hardware representation. In particular, the semantics of latency-insensitive abstractions and their relationship to cycle-based representations are incredibly challenging. In this context, Carloni et al. [10] introduce a theory to describe and represent latency-insensitive circuits and systems. The idea is to represent complex systems considering their single functional components and communication according to a specific protocol. The theory introduced allows for the composition of such single components into a complex system to satisfy synchronization and communication properties.

Overall, these works focus on precise abstraction levels - either low or high. Instead, the CIRCT compilation toolchain exposes different abstraction at different levels: reasoning about their semantics allows **taking advantage of CIRCT's flexible approach to hardware design**, complementing it with solid correctness guarantees concerning the designs and, ultimately, the compilation itself.

Due to the complexity of its environment, correctness guarantees in CIRCT require careful reflection on the semantics of the various Intermediate Representations (IRs) involved and the lowerings between

them. To tackle this problem, CIRCT features different lowerings to standard hardware verification formats, such as `btor2` [20, 21], `aig` [8] and `SMT-LIB` [5], that contribute to integrating verification within the hardware design process. In particular, the SMT dialect represents one means of formalizing dialects' semantics at different levels. In fact, this dialect comprises most SMT-LIB [5] operators and is the endpoint of various lowerings from different dialects, meaning that their semantics is encoded in SMT-LIB and can be exported to SMT-LIB for verification. The dialect is part of the compiler verification and optimization efforts currently under development in a research group at the University of Cambridge, with whom I have worked for the past year. However, the underlying complexity of SMT makes it hard to embed such domain-specific information into the actual SMT model [14]. `circt-lec` and `circt-bmc` are other tools that perform logical equivalence checking and bounded model checking, respectively, at the Register Transfer Level (RTL) level. Another recent work [25] provided a preliminary formalization of CIRCT dialects, only focusing on lower-level IRs. While the current CIRCT infrastructure features various efforts toward the verification of the toolchain and its design, none of these take full advantage of CIRCT's high-level abstractions and eventually always fall back to lower-level representations.

Works such as Lean-MLIR [7] have already proved the benefits of embedding dialects' semantics in Lean4, an open-source programming language and theorem prover providing a flexible environment to write verified code. In fact, when using typical verification methods, such as SMT and SAT solvers, we need to put a lot of trust in their correctness. As an example, consider z3 [12]: it consists of a huge C++ codebase, it is not verified, and these characteristics make the extension of its tactics and logics incredibly hard. As a consequence, optimizing the job of the SMT solver, for example by introducing tactics that exploit domain-specific knowledge available at higher abstraction levels is not a practically viable solution. Instead, Lean4 provides a flexible and verified environment to expand CIRCT's verification tooling, making it possible to write theorems that encode certain domain-specific behaviours and successfully guide the verification, with a significantly higher degree of trust. This characteristics also makes Lean4 the perfect environment to experiment with further,

verified verification methods such as abstract interpretation. Lean4 already proved very effective in building white-box automation techniques for bitvector manipulation (to which I contributed) and verification. In particular the `bv_decide` tactic [1] represents a first effort in this space, implementing a verified bitblaster and LRAT checker. This fact demonstrates the potential of Lean4 in the field of verification and the interest of the community in working towards this direction.

3 RESEARCH DIRECTION AND CHALLENGES

During my PhD, I aim to enhance verification efforts for the CIRCT infrastructure by formalizing the semantics of its dialects to enable the verification of lowerings and optimizations, the adoption of novel verification techniques, and the progressive verification of designs' properties. Following the direction indicated by Lean-MLIR [7], I intend to formalize more high-level abstractions and dialects, starting from the Finite State Machine (FSM) dialect, and also considering those relying on radically different design paradigms from traditional imperative or functional IRs, such as the Dynamic Control (DC) dialect. This aspect will require careful mathematical modelling of complex, dataflow-like behaviour to fill gaps in the informal model, e.g. where latency-insensitive and circuit-based semantics meet. Besides introducing a powerful means to reason about dialects' semantics and the overall correctness of the compilation, this work will also lay the foundations for including higher-level verification techniques into CIRCT, for example, those leveraging automata-theoretic and abstract-interpretation-based approaches.

This work takes advantage of CIRCT's high-level information at different abstraction levels, and combines it with Lean4's flexibility and minimal trusted codebase. Until now, encoding high-level information in standard lower-level hardware verification techniques (such as assertion-based verification or SMT solvers) has been very complex [3, 24], due to the scarce flexibility of the tools involved (e.g. z3 [12]) and the limited number of abstractions they support. With this work, the high-level information we extract by accurately formalizing dialects' semantics can contribute to guiding the effort of lower-level, classical verification approaches. In fact, Lean4 makes it possible to write verified theorems that extract and exploit the domain-specific information
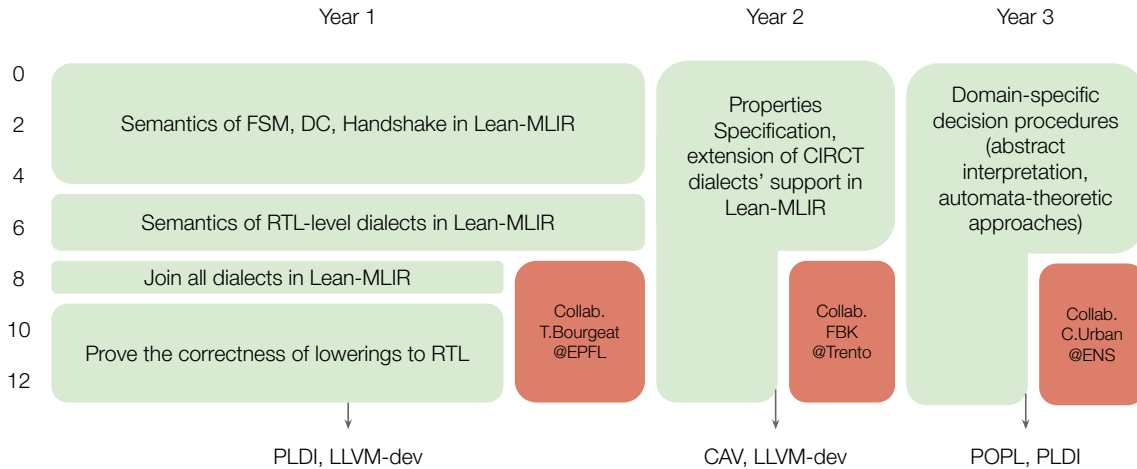
Figure 2: I will spend the first four months of the PhD implementing in Lean4 the semantics of three high-level dialects (FSM, Handshake, DC). The next three months will be devoted to the semantics of the CIRCT dialects necessary for the RTL-level representation (Hardware, Comb, Seq). One more month will be necessary to tackle the engineering involved in combining all these dialects in the Lean-MLIR framework and ensuring they work flexibly and their integration is functional. During the the last four months of the first year I will the correctness of the lowerings from the higher-level abstractions to RTL, collaborating with Thomas Bourgeat's research group at EPFL. The second year will be dedicated to implementing the property specifications in Lean-MLIR. I also aim to visit Fondazione Bruno Kessler (FBK) and University of Trento during this time. During the last year I will investigate alternative decision procedure and verification frameworks, collaborating with Caterina Urban's research group at ENS.

available at higher abstraction levels. The formalization of dialects within Lean4 also enables the verification of compiler passes, offering further guarantees for the correctness of CIRCT itself.

Moreover, formalizing the semantics of CIRCT dialects allows semantics manipulation up to a point where the application of different verification approaches is possible and beneficial. Exploring different approaches in the context of hardware verification is currently a challenge [16, 19], requiring significant effort to bridge the hardware level with a suitable abstraction that can be digested by alternative verification methodologies, such as abstract interpretation frameworks. Nevertheless, preliminary studies suggested the effectiveness of these methodologies, especially at specific abstraction levels [6]. Formalizing the semantics of CIRCT dialects in an interactive theorem prover is a game-changer in this context, as it proves a stable foundation to explore many verification strategies.

Lean4 is a fertile, open-source environment to work on verification frameworks thanks to its flexibility, versatility and trust level. Moreover, working in an open-source environment is an incredibly valuable aspect of this work, as it significantly increases the quality of the output - thanks to peer-review processes - and creates impact, being available to numerous users.

Overall, formalizing the semantics of a large subset of CIRCT's dialects is a first step towards integrating the EDA toolchain with automatic verification, which is simultaneous and progressive with the design procedure and comprises different verification techniques. While this aspect can increase the compilation time for a single design iteration, its correctness guarantees can significantly reduce the number of iterations required to generate the desired design, eventually improving the overall design pipeline. This work paves the way for filling and reasoning about subtle semantics gaps in hardware abstractions, by introducing formally verified semantics in the CIRCT ecosystem.

4 RESEARCH ORGANIZATION

Figure 2 summarizes the organization of my PhD. This research proposal comprises three milestones, which I plan to distribute in the three years of the PhD:

(1) The goal of the first year is to **formalize the semantics of a first subset of high-level CIRCT dialects** (namely FSM, Handshake and DC) and **verify their lowering to RTL level within Lean-MLIR**. This phase will require careful study of existing works, especially concerning the formalization of latency-insensitive behaviours in DC and their interface with cycle-accurate representation. Then, I will implement the semantics of the CIRCT dialects necessary for the RTL-level representation (Hardware, Comb, Seq) and spend some time on the engineering aspects involved in combining all these dialects in Lean-MLIR. As a final task for the first year, I will prove the correctness of the lowerings from the higher-level abstractions to RTL, including some of the optimizations involved. I will aim to publish the formally verified semantics resulting from this work at PLDI and present it at LLVM-dev meetings. Thomas Bourgeat and his research group represent an excellent collaboration opportunity at this stage of the Ph.D., because of their expertise in the area of hardware semantics definition and verification.

(2) **Bringing designs' verification at a higher abstraction level, exploiting different dialects and their formalized semantics**. At this point, a crucial step is understanding how to correctly specify relevant properties, which will require thorough readings in the state of the art. Moreover, I plan to add further dialects to the Lean-MLIR framework during the second year. During the second year I intend to also collaborate with Stefano Tonetta's research group at Fondazione Bruno Kessler (FBK), due to their expertise in the development of verification techniques and solvers. CAV is the ideal conference to publish the output of this second year.

(3) **Explore different verification strategies**, such as automata-theoretic approaches and abstract interpretation frameworks, to fully take advantage of the formalized semantics. During the last year, I plan to take advantage of the effort put into effectively formalizing CIRCT dialects. I will investigate how other verification techniques can further take advantage of the formalized semantics to improve the guarantees concerning the design's behaviour, aiming to reduce the number of necessary iterations. To work on these topics I intend to collaborate with Caterina Urban's research group at ENS, given the high-impact work they carried out in the field of abstract interpretation and semantics analysis. PLDI and POPL are the conference targets to publish the output of the last year.

(4) I plan to spend 6 additional months writing the Ph.D. thesis, which will focus on how CIRCT and Lean4 are a unique combination of tools to make hardware verification flexible and extensible, bending its tools to fit different designs and techniques.

Since I intend to continue my path in academia after the Ph.D., the work of the next 3.5 years is a starting point to become an expert in this field, build valuable connections and lay the foundations for the research work to come.

REFERENCES

[1] bv_decide. https://github.com/leanprover/lean4/, Online.

[2] Circt. https://circt.llvm.org/, Online.

[3] Symbiyosys (sby) – front-end for yosys-based formal verification flows. Accessed 24/11/21.

[4] Bachrach, J., Vo, H., Richards, B., Lee, Y., Waterman, A., Avižienis, R., Wawrzynek, J., and Asanović, K. Chisel: constructing hardware in a scala embedded language. In *Proceedings of the 49th Annual Design Automation Conference* (New York, NY, USA, 2012), DAC '12, Association for Computing Machinery, p. 1216–1225.

[5] Barrett, C., Stump, A., Tinelli, C., et al. The smt-lib standard: Version 2.0. In *Proceedings of the 8th international workshop on satisfiability modulo theories (Edinburgh, UK)* (2010), vol. 13, p. 14.

[6] Bernstein, G. L., and Ragan-Kelley, J. What are the semantics of hardware. In *Workshop on Languages, Tools, and Techniques for Accelerator Design (LATTE)* (2021).

[7] Bhat, S., Keizer, A., Hughes, C., Goens, A., and Grosser, T. Verifying peephole rewriting in ssa compiler irs. *arXiv preprint arXiv:2407.03685* (2024).

[8] Biere, A., Heljanko, K., and Wieringa, S. AIGER 1.9 and beyond. Tech. Rep. 11/2, Institute for Formal Models and Verification, Johannes Kepler University, Altenbergerstr. 69, 4040 Linz, Austria, 2011.

[9] Bourgeat, T., Pit-Claudel, C., Chlipala, A., and Arvind. The essence of bluespec: a core language for rule-based hardware design. In *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation* (2020), pp. 243–257.

[10] Carloni, L. P., McMillan, K. L., and Sangiovanni-Vincentelli, A. L. Theory of latency-insensitive design. *IEEE Transactions on computer-aided design of integrated circuits and systems 20*, 9 (2001), 1059–1076.

[11] Chen, J., and He, F. Leveraging control flow knowledge in smt solving of program verification. *ACM Transactions on Software Engineering and Methodology (TOSEM) 30*, 4 (2021), 1–26.

[12] De Moura, L., and Bjørner, N. Z3: An efficient smt solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems* (2008), Springer, pp. 337–340.

[13] Eldridge, S., Barua, P., Chapyzhenka, A., Izraelevitz, A., Koenig, J., Lattner, C., Lenharth, A., Leontiev, G., Schuiki, F., Sunder, R., Young, A., and Xia, R. MLIR as Hardware Compiler Infrastructure. In *WOSET '21: Workshop on Open Source EDA Technology* (2021), SiFive.

[14] Gurfinkel, A. Program verification with constrained horn clauses. In *International Conference on Computer Aided Verification* (2022), Springer, pp. 19–29.

[15] Huang, B.-Y., Zhang, H., Subramanyan, P., Vizel, Y., Gupta, A., and Malik, S. Instruction-level abstraction (ila) a uniform specification for system-on-chip (soc) verification. *ACM Transactions on Design Automation of Electronic Systems (TODAES) 24*, 1 (2018), 1–24.

[16] Malik, S. Hardware verification: Techniques, methodology and solutions. In *Tools and Algorithms for the Construction and Analysis of Systems: 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings 14* (2008), Springer, pp. 1–1.

[17] Mattarei, C., Mann, M., Barrett, C., Daly, R. G., Huff, D., and Hanrahan, P. Cosa: Integrated verification for agile hardware design. In *2018 Formal Methods in Computer Aided Design (FMCAD)* (2018), IEEE, pp. 1–5.

[18] Melham, T. F. Abstraction mechanisms for hardware verification. In *VLSI Specification, Verification and Synthesis*. Springer, 1988, pp. 267–291.

[19] Mukherjee, R., Kroening, D., and Melham, T. Hardware verification using software analyzers. In *2015 IEEE Computer Society Annual Symposium on VLSI* (2015), IEEE, pp. 7–12.

[20] Niemetz, A., Preiner, M., Wolf, C., and Biere, A. Btor2, btormc and boolector 3.0. In *International Conference on Computer Aided Verification* (2018).

[21] Niemetz, A., Preiner, M., Wolf, C., and Biere, A. Btor2, btormc and boolector 3.0. In *Computer Aided Verification* (Cham, 2018), H. Chockler and G. Weissenbacher, Eds., Springer International Publishing, pp. 587–595.

[22] Nikhil, R. Bluespec system verilog: efficient, correct rtl from high level specifications. In *Proceedings. Second ACM and IEEE International Conference on Formal Methods and Models for*

Co-Design, 2004. MEMOCODE '04. (2004), pp. 69–70.

[23] Procter, A., Harrison, W. L., Graves, I., Becchi, M., and Allwein, G. Semantics driven hardware design, implementation, and verification with rewire. In *Proceedings of the 16th ACM SIGPLAN/SIGBED Conference on Languages, Compilers and Tools for Embedded Systems 2015 CD-ROM* (2015), pp. 1–10.

[24] Witharana, H., Lyu, Y., Charles, S., and Mishra, P. A survey on assertion-based hardware verification. *ACM Computing Surveys (CSUR) 54*, 11s (2022), 1–33.

[25] Zhao, J., Kang, J., and Zhao, Y. K-circt: A layered, composable, and executable formal semantics for circt hardware irs. *arXiv preprint arXiv:2404.18756* (2024).