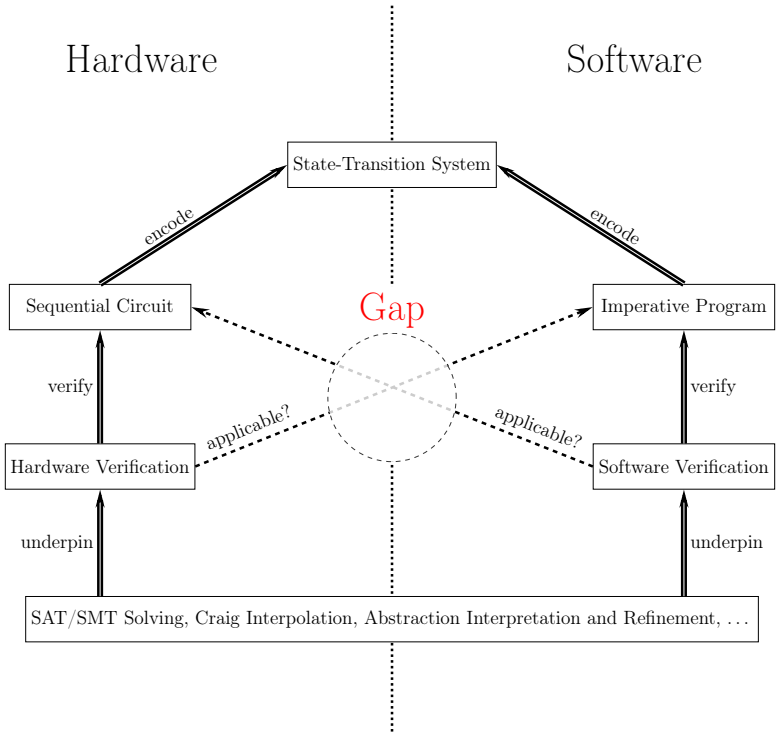
Nian-Ze Lee

Research Statement

Oettingenstr. 67, 80538 Munich
+49-176-677-030-45
nian-ze.lee@sosy.ifi.lmu.de
nianzelee.github.io
0000-0002-8096-5595
Passport spelling: Niann-Tzer Li

Vision: Computing systems, omnipresent in our daily lives, demand meticulous verification to ensure their correct functionality and security. Grounded in the foundational theories of logic, automata, and constraint solving, *formal methods* have been successfully applied in real-world scenarios and delivered correctness guarantees with mathematical rigor. The challenges to formal methods have escalated due to increasing interactions among diverse components, e.g., software programs, hardware circuits, and cyber-physical devices. Therefore, a collaborative and cross-disciplinary approach among traditionally separate research communities is necessary. Although the research communities for formal methods share similar concepts and techniques, their alignment with distinct computational models creates gaps between the communities. I am dedicated to bridging the gap between software and hardware research communities, as depicted in the figure below, by systematically cross-applying their techniques and extracting valuable insights into analyzing heterogeneous computing systems. Consolidating knowledge from both hardware and software domains, I aim to unify verification approaches for comprehensive system-level verification. Furthermore, I envision broader applicability of the results to emerging computational schemes, including neural networks and quantum computing. Through these endeavors, I strive to advance formal methods and ensure the continued reliability of computing systems in the face of evolving challenges. My vision on **Bridging Hardware and Software Analysis** has been funded by the German Research Foundation.



What Makes Me Unique? My experiences in formal methods encompass both hardware and software systems and span broadly from model checkers to constraint solvers. My scientific findings have been published in prestigious venues, such as the **IEEE Transactions on Computers** [17], **AAAI Conference on Artificial Intelligence** [19], **Journal of Automated Reasoning** [11], and **Proceedings of the ACM on**

Software Engineering [4], and received the **Best Paper Award** at SPIN 2024 [9] and the **Distinguished Artifact Award** at TACAS 2024 [1]. Notably, my recent focus on applying software verifiers to hardware circuits [8, 1] has yielded a remarkable improvement in bug detection compared to the best hardware model checkers, revealing the potential of cross-applying techniques. To enhance the reproducibility of my scientific results and widen the usage of formal methods, I prepare reproduction artifacts [15, 7, 10, 6, 2, 3] for my experiments and publish them on long-term archiving platforms. In summary, my unique profile, marked by experiences in both hardware and software, a commitment to reproducibility, and a track record of impactful results and awards, motivates me to pursue the interdisciplinary research vision above.

In the following, I will outline my important achievements, ongoing and future projects, and plans for third-party funding, Ph.D. supervision, and industrial collaboration.

Important Achievements

I have obtained impactful results in the following directions:

- **Formal Verification of Probabilistic Circuits and Stochastic Boolean Satisfiability** [17, 21, 22, 19]
- **Synthesis and Rectification of Hardware Models Using Formal Methods** [18, 20, 13, 14]

The findings in the first direction have driven further tool development for stochastic Boolean satisfiability [25] and found applications in verifying the fairness of machine-learning algorithms [26]. The approaches proposed in the second direction have been integrated into an industrial design-automation flow [20] and applied to high-performance commercial hardware.

Formal Verification of Probabilistic Circuits and Stochastic Boolean Satisfiability

In the nanometer regime of VLSI design fabrication, the variability and randomness of transistors are inevitable due to their quantum nature at the atomic level and impose serious challenges on manufacturing reliable hardware systems. To sustain the continuation of the Moore's law, formally analyzing VLSI circuits with probabilistic behaviors is imperative. Using randomized quantifiers to prescribe the probability distribution of Boolean variables, stochastic Boolean satisfiability (SSAT) [27] is suitable for modeling problems with uncertainty. I propose the first framework [17] to formally verify probabilistic VLSI circuits with SSAT and devise a novel circuit-based SSAT solver, which outperformed conventional clause-based solvers by orders of magnitude in the evaluation.
To achieve broader impact, I invent novel algorithms for SSAT solving [21, 22] and implement them in an open-source tool, providing a reference to the state of the art for the evaluation in follow-up publications [25]. To extend the expressiveness of SSAT, I formulate *dependency SSAT* and prove its complexity [19]. This theoretical work has shed light on solver development and modeling of distributed systems with uncertainty and partial information using SSAT.

Synthesis and Rectification of Hardware Models Using Formal Methods

Hardware synthesis is essential to reduce the manufacturing cost. Compared to heuristics, formal methods have the potential to explore the design space comprehensively. *Threshold logic gates* imitate the computation of neurons and offer a promising alternative to implement neural networks in hardware. I propose a synthesis technique based on constraint solving and two verification algorithms that cater to different application scenarios, making formal methods feasible for the implementations of neural networks as threshold logic circuits [18]. The implementations are available in an open-source tool.
Due to the growing complexity, last-minute changes before tape-out become common in the IC design industry. To alleviate the time-to-market pressure, automatically rectifying a heavily optimized circuit with minimal disruption is vital [13, 14]. During an internship at IBM Research, I augmented IBM's in-house design flow to accommodate circuits optimized by sequential synthesis. The method incurs negligible overhead and has been integrated into IBM's design flow [20]. The above achievements and industrial experiences have sharpen my skills to deal with real-world applications.

I have also worked on applying search-based testing to automotive controllers [16]. This project has broadened my research interests to cyber-physical systems.

Ongoing and Future Projects

Recently, I have expanded my research subjects to software and formulated a new direction on "**Bridging Hardware and Software Analysis**" to close the gap between the hardware and software domains. This gap emerges because formal methods and verification tools are usually aligned with specific computational models and input languages, hindering knowledge exchange and mutual learning across the communities. I am working on the following topics to systematically bridge the gap:

- **Cross-Application of Hardware and Software Techniques** [11, 8, 1, 12, 9]
- **Knowledge Consolidation of Formal Verification** [4]

My endeavors on these topics have obtained promising findings, part of which have been published or accepted at the **Journal of Automated Reasoning** [11], **International Conference on Tools and Algorithms for the Construction and Analysis of Systems** (TACAS) [8, 1], and **Proceedings of the ACM on Software Engineering** [4], and won the **Best Paper Award** [9] at SPIN 2024 and the **Distinguished Artifact Award** [2] at TACAS 2024. Based on these findings, I have obtained funding from the German Research Foundation for a three-year Ph.D. position. My goal is to obtain insights into analyzing heterogeneous computing systems and achieve comprehensive system-level verification in practice.
In future, I aim to apply formal methods to emerging computational paradigms, such as neural networks and quantum computing, and invent new approaches for the evolving challenges of modeling and specification:

- **Emerging Challenges and Applications of Formal Methods**

Cross-Application of Hardware and Software Techniques

Current Results

I have studied two schemes to cross-apply hardware and software verification: transferring verification algorithms and translating verification tasks. For algorithm transfer, I adopt *interpolation-based model checking* [29], a state-of-the-art approach for hardware, to analyze programs [11]. While the algorithm has inspired numerous interpolation-based methods for software, it had not been used to verify programs, leaving a knowledge gap in its performance characteristics. I implemented this algorithm in CPAchecker and evaluated it against mature methods for software verification, closing this significant gap in knowledge. In the experiment, it solved more tasks than all interpolation-based approaches in CPAchecker. The implementation has contributed to the success of CPAchecker in the Competitions on Software Verification. For task translation, I design a translation framework [8] from Btor2 [31], a hardware-description language, to the programming language C. Verifiers for C programs have been actively developed for two decades and can analyze industrial-scale code. However, their strengths had not been evaluated on hardware models, and whether hardware circuits benefit from software verifiers remained unclear. With the translator Btor2C [8], a hardware-verification task is translated into a software-verification task, and any off-the-shelf software verifier can be applied to the translated program. The first large-scale experiment comparing state-of-the-art software verifiers and hardware model checkers was conducted [8]. Remarkably, software verifiers detected more bugs in the hardware circuits than the best hardware verifiers, indicating the importance to systematically cross-apply approaches in the two communities. Our reproduction package [7] was evaluated at TACAS and awarded available and reusable badges.

I have conceived the following projects to further extend this direction.

Certificate Generation for Hardware Models from Software Verifiers

It is hard to construct formal verifiers, and even the best ones suffer from wrong results. Verifiers can improve the reliability of their analysis results by generating *certificates*. Continuing the work on translating hardware to software [8], I transfer the information in *verification witnesses* [24], produced by software verifiers as certificates for their verification results, back to the hardware domain as test vectors or state invariants for users' inspection [1]. Such a *certifying* verification flow will increase circuit designers' trust in software tools, making the advancements of software verification more easily accessible to different communities.

Circuit-Based Program Verification

Verifying programs is challenging, and numerous intermediate representations have been proposed to facilitate the construction of verifiers. Sequential circuits compactly describe state-transition systems, and abundant endeavors have been invested in their synthesis and formal verification. Therefore, they should be considered as an alternative foundation to build software verifiers. I am working on translating programs to circuits and applying hardware model checkers to solve the verification tasks of programs. Our verifier CPV [12] participated in the 2024 Competition on Software Verification and competed well against software verifiers using control-flow graphs as the intermediate representation.

Knowledge Consolidation of Formal Verification

Current Results

The knowledge of formal verification is spread across different research communities, hindering a unified understanding. Systematic knowledge exchange between hardware and software verification is pivotal for complete system-level verification. Transferability studies, a crucial method to consolidate knowledge by inspecting to which extent the conclusions in a publication could plausibly apply to other circumstances, are unfortunately scarce in formal verification. To address the gap, I conducted a transferability study [4] on two interpolation-based algorithms [33, 34], initially designed for hardware model checking, and applied them to software verification. Our experiments reveal that specific algorithmic properties persist across diverse verification tasks as either circuits or programs [4]. These properties can be leveraged in hardware-software co-verification thanks to their agnostic nature to the modeling languages.

I have conceived the following topics to further pursue this direction.

Profiling Hardware and Software Verifiers

Benchmarking is essential to understand the performance characteristics of a tool. Thanks to the task translators invented to bridge the gap between hardware and software verification, it becomes possible to compare verifiers from both domains on the same set of verification tasks. Therefore, verifier developers are equipped with new reference baselines to assess and improve their tools, which were unavailable due to the barrier caused by distinct modeling languages. I plan to profile the performance differences between hardware model checkers and software verifiers on large benchmark suites and use data mining to extract insights, which will also help users to select and configure verifiers based on their applications.

Algorithm Selection Based on Machine Learning

Built on top of the profiling results, I will apply machine learning to construct robust portfolios of off-the-shelf verifiers by algorithm selection [32]. Different from previous studies on portfolios, which only consider tools with the same modeling language, I will employ the translation methods as frontend and enlarge the set of candidate verifiers with tools for other computational models. Therefore, the resulting portfolios will reflect the best efforts of both hardware and software verification techniques. To assess the practical impact of the portfolios, I will evaluate them on real-world applications, such as the open-source silicon IP cores on OpenCores, device drivers in Linux kernel modules, and common libraries of AWS software-development kits.

Emerging Challenges and Applications of Formal Methods

To ensure the correctness of emerging computing systems in the face of evolving challenges, I will look into these topics in future.

Verification of Embedded Software

The increasing usage of software in safety-critical applications, such as automotive and aviation electronics, necessitates the verification of programs running on embedded systems with constrained resources. Nevertheless, existing software verifiers usually target common computer architectures and assume unlimited hardware resources, such as memory. I will apply the results from bridging hardware and software analysis to invent more scalable approaches for embedded software, whose verification requires intensive hardware modeling. While standard approaches for hardware/software co-verification encode both software and hardware to low-level verification conditions [30], we can flexibly transform problem representations with our task translators instead of sticking to a specific encoding as investigated in the literature.

Lightweight and Incremental Verification
To make formal methods more widely adopted in practice, I have also worked on lightweight static analysis. We propose a data-flow analyzer [5] based on expressions over variable intervals, which runs efficiently and complements other competitive tools by solving extra verification tasks. I aim to continue this line of work by a "guess-and-check" approach to compute invariants, inspired by the observation that checking a candidate invariant is usually faster than synthesizing one from scratch. Massive parallel computing facilitates checking many candidates at once. I will also investigate incremental verification, which uses information from previous executions for acceleration.

Novel Aspects: Neural Networks, Quantum Computing, and Information Security
Emerging computational paradigms and evolving requirements of computing systems unveil new challenges and opportunities for formal methods. Software verifiers have recently been shown to struggle with the C implementations of neural networks [28]. I aim to develop solutions and make formal verification feasible for neural networks. With the growing applicability of quantum computers, quantum software testing has started to receive attention [23]. I plan to extend my research scope to the verification of quantum programs in future. Our privacy data are processed by software programs, and it is vital to ensure no leakage of sensitive information. Formal methods offer abundant techniques to tackle this problem. I will investigate building provably secure systems and complement existing algebraic approaches with formal methods.

Plans for Funding, Ph.D. Supervision, and Industrial Collaboration

Built on top of the results about cross-application and unification of hardware and software verification, my grant proposal on **Bridging Hardware and Software Analysis**, submitted together with my postdoctoral advisor Prof. Dirk Beyer, has been approved by the German Research Foundation (DFG) for a three-year Ph.D. position. The student will work on performance profiling and portfolio construction of hardware and software verifiers, case studies with real-world applications, and new approaches for hardware/software co-verification. Moreover, I have submitted another grant proposal for industrial collaboration on automatically verifying security properties of firmware modules used in trusted and confidential cloud computing. The proposal is under review and has received positive feedback from our collaborator.

Own Publications

[1] Z. Ádám, D. Beyer, P.-C. Chien, N.-Z. Lee, and N. Sirrenberg. Btor2-Cert: A certifying hardware-verification framework using software analyzers. In *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, LNCS 14572, pages 129–149. Springer, 2024. doi: 10.1007/978-3-031-57256-2_7.

[2] Z. Ádám, D. Beyer, P.-C. Chien, N.-Z. Lee, and N. Sirrenberg. Reproduction package for TACAS 2024 article 'Btor2-Cert: A certifying hardware-verification framework using software analyzers'. Zenodo, 2024. doi: 10.5281/zenodo.10548597.

[3] D. Beyer, P.-C. Chien, M. Jankola, and N.-Z. Lee. Reproduction package for FSE 2024 article 'A transferability study of interpolation-based hardware model checking for software verification'. Zenodo, 2024. doi: 10.5281/zenodo.11070973.

[4] D. Beyer, P.-C. Chien, M. Jankola, and N.-Z. Lee. **A Transferability Study of Interpolation-Based Hardware Model Checking to Software Verification**. *Proceedings of the ACM on Software Engineering*, Issue FSE, 2024. Accepted.

[5] D. Beyer, P.-C. Chien, and N.-Z. Lee. CPA-DF: A tool for configurable interval analysis to boost program verification. In *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering*, pages 2050–2053. IEEE, 2023. doi: 10.1109/ASE56229.2023.00213.

[6] D. Beyer, P.-C. Chien, and N.-Z. Lee. Reproduction package for ASE 2023 article 'CPA-DF: A tool for configurable interval analysis to boost program verification'. Zenodo, 2023. doi: 10.5281/zenodo.8245821.

[7] D. Beyer, P.-C. Chien, and N.-Z. Lee. Reproduction package for TACAS 2023 article 'Bridging hardware and software analysis with Btor2C: A word-level-circuit-to-C translator'. Zenodo, 2023. doi: 10.5281/zenodo.7551707.

[8] D. Beyer, P.-C. Chien, and N.-Z. Lee. **Bridging Hardware and Software Analysis with Btor2C: A Word-Level-Circuit-to-C Translator**. In *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, LNCS 13994, pages 152–172. Springer, 2023. doi: 10.1007/978-3-031-30820-8_12.

[9] D. Beyer, P.-C. Chien, and N.-Z. Lee. Augmenting interpolation-based model checking with auxiliary invariants. In *Proceedings of the International Symposium on Model Checking Software*. Springer, 2024. Accepted, extended version available via https://arxiv.org/abs/2403.07821.

[10] D. Beyer, N.-Z. Lee, and P. Wendler. Reproduction package for JAR article 'Interpolation and SAT-based model checking revisited'. Zenodo, 2023. doi: 10.5281/zenodo.8245824.

[11] D. Beyer, N.-Z. Lee, and P. Wendler. **Interpolation and SAT-Based Model Checking Revisited: Adoption to Software Verification**. *Journal of Automated Reasoning*, 2024. Accepted, preprint available via https://doi.org/10.48550/arXiv.2208.05046.

[12] P.-C. Chien and N.-Z. Lee. CPV: A circuit-based program verifier (competition contribution). In *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, LNCS 14572, pages 365–370. Springer, 2024. doi: 10.1007/978-3-031-57256-2_22.

[13] A. Q. Dao, N.-Z. Lee, L.-C. Chen, M. P.-H. Lin, J.-H. R. Jiang, A. Mishchenko, and R. K. Brayton. Efficient computation of ECO patch functions. In *Proceedings of the Annual Design Automation Conference*, pages 51:1–51:6. ACM, 2018. doi: 10.1145/3195970.3196039.

[14] J.-H. R. Jiang, V. N. Kravets, and N.-Z. Lee. Engineering change order for combinational and sequential design rectification. In *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition*, pages 726–731. IEEE, 2020. doi: 10.23919/DATE48585.2020.9116504.

[15] N.-Z. Lee. Reproduction package for doctoral dissertation 'Stochastic Boolean satisfiability: Decision procedures, generalization, and applications'. Zenodo, 2021. doi: 10.5281/zenodo.5084146.

[16] N.-Z. Lee, P. Arcaini, S. Ali, and F. Ishikawa. Stability analysis for safety of automotive multi-product lines: A search-based approach. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1241–1249. ACM, 2019. doi: 10.1145/3321707.3321755.

[17] N.-Z. Lee and J.-H. R. Jiang. **Towards Formal Evaluation and Verification of Probabilistic Design**. *IEEE Transactions on Computers*, 67(8):1202–1216, 2018. doi: 10.1109/TC.2018.2807431.

[18] N.-Z. Lee and J.-H. R. Jiang. Constraint solving for synthesis and verification of threshold logic circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 40(5):904–917, 2021. doi: 10.1109/TCAD.2020.3015441.

[19] N.-Z. Lee and J.-H. R. Jiang. **Dependency Stochastic Boolean Satisfiability: A Logical Formalism for NEXPTIME Decision Problems with Uncertainty**. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3877–3885. AAAI Press, 2021. doi: 10.1609/aaai.v35i5.16506.

[20] N.-Z. Lee, V. N. Kravets, and J.-H. R. Jiang. Sequential engineering change order under retiming and resynthesis. In *Proceedings of the International Conference on Computer-Aided Design*, pages 109–116. IEEE, 2017. doi: 10.1109/ICCAD.2017.8203767.

[21] N.-Z. Lee, Y.-S. Wang, and J.-H. R. Jiang. Solving stochastic Boolean satisfiability under random-exist quantification. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 688–694. IJCAI Organization, 2017. doi: 10.24963/ijcai.2017/96.

[22] N.-Z. Lee, Y.-S. Wang, and J.-H. R. Jiang. Solving exist-random quantified stochastic Boolean satisfiability via clause selection. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1339–1345. IJCAI Organization, 2018. doi: 10.24963/ijcai.2018/186.

Other References

[23] S. Ali and T. Yue. Quantum software testing: A brief introduction. In *Proc. ICSE Companion*, pages 332–333. IEEE, 2023.

[24] D. Beyer, M. Dangl, D. Dietsch, M. Heizmann, T. Lemberger, and M. Tautschnig. Verification witnesses. *ACM Trans. Softw. Eng. Methodol.*, 31(4):57:1–57:69, 2022.

[25] P.-W. Chen, Y.-C. Huang, and J.-H. R. Jiang. A sharp leap from quantified Boolean formula to stochastic Boolean satisfiability solving. In *Proceedings of the AAAI Conference on Artificial Intelligence, AAAI*, pages 3697–3706. AAAI Press, 2021. doi: 10.1609/AAAI.V35I5.16486.

[26] B. Ghosh, D. Basu, and K. S. Meel. Justicia: A stochastic SAT approach to formally verify fairness. In *Proceedings of the AAAI Conference on Artificial Intelligence, AAAI*, pages 7554–7563. AAAI Press, 2021. doi: 10.1609/AAAI.V35I9.16925.

[27] S. M. Majercik. Stochastic Boolean satisfiability. In *Handbook of Satisfiability*, pages 887–925. IOS Press, 2009. doi: 10.3233/978-1-58603-929-5-887.

[28] E. Manino, R. S. Menezes, F. Shmarov, and L. C. Cordeiro. NeuroCodeBench: a plain C neural network benchmark for software verification. *arXiv/CoRR*, 2309(03617), September 2023.

[29] K. L. McMillan. Interpolation and SAT-based model checking. In *Proc. CAV*, LNCS 2725, pages 1–13. Springer, 2003.

[30] R. Mukherjee, M. Purandare, R. Polig, and D. Kroening. Formal techniques for effective co-verification of hardware/software co-designs. In *Proc. DAC*, pages 1–6. ACM, 2017.

[31] A. Niemetz, M. Preiner, C. Wolf, and A. Biere. BTOR2, BTORMC, and BOOLECTOR 3.0. In *Proc. CAV*, LNCS 10981, pages 587–595. Springer, 2018.

[32] C. Richter and H. Wehrheim. Attend and represent: a novel view on algorithm selection for software verification. In *Proc. ASE*, pages 1016–1028, 2020.

[33] Y. Vizel and O. Grumberg. Interpolation-sequence based model checking. In *Proc. FMCAD*, pages 1–8. IEEE, 2009.

[34] Y. Vizel, O. Grumberg, and S. Shoham. Intertwined forward-backward reachability analysis using interpolants. In *Proc. TACAS*, LNCS 7795, pages 308–323. Springer, 2013.