A Survey of AI-Aided FPGA Development Tools and Considerations for COTS Software Dedication

Yoona Heo[a], Younggyu Lee[a], Dae-won Kim[a], Eunkyoung Jee[b], Jiyoung Song[c], Junbeom Yoo[a,*]
[a]*Konkuk University, 120 Neungdong-ro, Gwangjin-gu, Seoul, Korea,*
[b]*Korea Advanced Institute of Science and Technology, 291, Daehak-ro, Yuseong-gu, Daejeon, Republic of Korea,*
[c]*Hannam University, 70 Hannamro, Daedeok-Gu, Daejeon, Republic of Korea*
[*]*Corresponding author: jbyoo@konkuk.ac.kr*

***Keywords : FPGA, Nuclear I&C Safety Systems, COTS Software, AI, COTS Software Dedication**

1. Introduction

Traditionally, Programmable Logic Controllers (PLCs) have been widely used to implement digital controllers in nuclear safety Instrumentation and Controls (I&C) systems [1]. Field-programmable gate arrays (FPGAs) have also emerged as an alternative implementation platform, including for safety I&C systems in Small Modular Reactors (SMRs) [2,3]. Accordingly, relevant standards and regulatory guidance have increasingly addressed the use of FPGA-based controllers in nuclear safety applications.

FPGA development relies heavily on Electronic Design Automation (EDA) tools provided by commercial vendors. Because these tools do not themselves perform safety functions but generate design artifacts that may affect the implementation of such functions, NUREG/CR-6421 [4] classifies them as indirect Commercial Off-the-Shelf (COTS) software.

In recent years, EDAs incorporating Artificial Intelligence (AI) technologies have been introduced. However, the use of AI techniques for the development of FPGA-based controllers in nuclear safety I&C systems was not in the scope of existing COTS software dedication approaches. This paper surveys commercial tools and the recent studies on AI-enabled EDA, including ASICs, due to the limitations of FPGA-dedicated commercial tools and research. Based on the survey, we briefly discuss the implications of applying an existing COTS software dedication approach to AI-aided FPGA development tools and identify areas that may require additional considerations.

2. Background

2.1 Electronic Design Automation (EDA)

EDA comprises the software tools and related technologies used to design and verify integrated circuits, including FPGA-based designs [5,6]. Figure 1 illustrates a typical FPGA EDA flow, in which a high-level design is transformed into Register-Transfer Level (RTL), a synthesized netlist, and a placed-and-routed design before bitstream generation and FPGA configuration [7]. The main EDA processes, design artifacts, and simulation activities are shown in orange, green, and gray boxes, respectively.

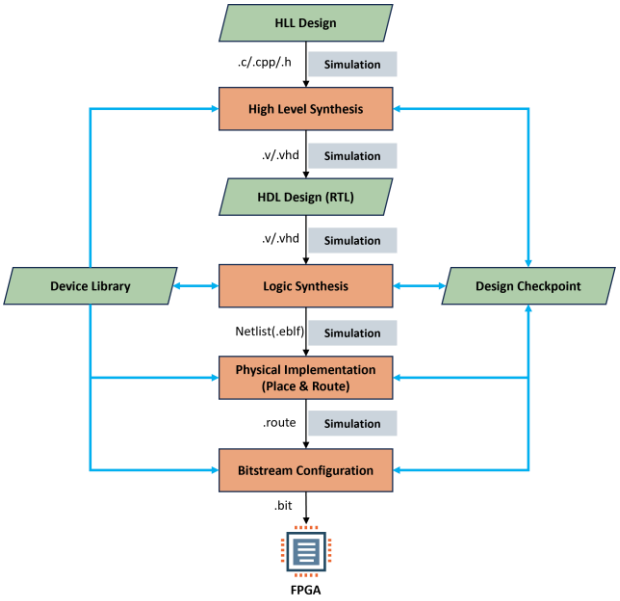


Figure 1. FPGA application design EDA flow [7]

2.2 NUREG/CR-6421-Based COTS Software Dedication Approach

In nuclear safety I&C systems, dedication of COTS software indirectly used for the development of FPGA-based controller was challenging. Jung et al. [8] proposed a four-step process for the dedication of both direct and indirect COTS software based on NUREG/CR-6421 [4].

The process first identifies the COTS software to dedicate, then determine the safety category of the COTS software by identifying the safety category of the target system and the usage and safety category of the COTS software. The acceptance criteria applied according to the safety category of COTS software are followed by the table related to the acceptance criteria for indirect COTS software in [8].

Subsequently, the process refines each acceptance criterion, selects and identifies appropriate acceptance method(s) and techniques based on four acceptance methods presented in EPRI NP-5652 [9]. The final step is to apply the selected acceptance methods and techniques. This approach can support appropriate selection of acceptance methods and verification and

Table I: Commercial AI-assisted EDA tools

Tool	Manufacturer	AI techniques	AI-assisted functions	Primary target
Vivado ML Edition [10] (Vivado Design Suite)	Xilinx, AMD	Machine learning (ML) [13]	Congestion and delay estimation	FPGA
Synopsys.ai Copilot [11]	Synopsys	Large language model (LLM) [14], Natural language processing (NLP) [15], Generative AI [16]	1) RTL code generation from natural language inputs 2) Formal test bench generation from specifications 3) Automated lint violation fixing	Mainly ASIC
Aprisa AI [12]	Siemens	Generative AI	RTL-to-GDS transformation	General semiconductor

Table II: Representative studies on AI-assisted EDA, adapted in part from [17]

EDA task	AI techniques	Representative studies	Year
RTL/HDL design generation	LLM	ChipGPT [18], AutoChip [19], RTLLM [20], ChipNeMo [21], ScaleRTL [22], REvolution [23], Spec2RTL-Agent [24], OpenLLM-RTL [25]	2023-2025
Verilog generation	Multi-agent AI [26]	VerilogCoder [27]	2025
HLS optimization	LLM, RAG [28], agentic workflows	TimelyHLS [29], LAAFD [30]	2025-2026
Logic synthesis and technology mapping	Reinforcement Learning (RL) [31], Attention-based regression [32]	ESE [33], AiMap+ [34]	2023-2024
CAD parameter tuning	RL	InTime [35]	2015
Placement	RL, GNN [36], CNN [37], Neural network training	DREAMPlace [38], MaskPlace [39], GraphPlanner [40], AlphaChip [41], TransPlace [42], Lee et al. [43], Mirhoseini et al. [44], Agnesina et al. [45]	2020-2025
Routing and congestion prediction	CNN, FCN [46]	CongestNN [47], Maarouf et al. [48], Al-Hyari et al. [49]	2018-2021
Hardware architecture exploration	NAS [50], Bayesian optimization [51]	AutoDNNchip [52]	2020
RTL bug verification	Formal verification [53], LLM	FVDebug [54], FLAG [55]	2025
Circuit logic generation	Boolean function approximation [56]	Cheng et al. [57]	2024
Overall flow orchestration	Agentic AI [58]	AiEDA [59], ShortCircuit [60], VTR-LLM [61]	2023-2026

validation (V&V) techniques.

The approach proposed by Jung et al. [8] provides the basis for considering the dedication of AI-aided FPGA development tools in this study. The following section surveys commercial AI-assisted EDA tools and related research trends, after which Section 4 briefly discusses AI-related assurance considerations that may need to be addressed when applying the existing approach.

3. Survey of AI-Assisted EDA for FPGA Development

This section deals with commercial AI-aided EDA tools and studies on AI-assisted EDAs. However, due to the limitations of current FPGA-dedicated commercial EDA tools and studies, the scope of analysis has been expanded to include the ASIC field.

3.1 Commercial AI-Aided EDA Tools

Several commercial EDA tools have recently incorporated AI capabilities. Table I summarizes selected tools that either directly support FPGA development or provide AI-assisted capabilities potentially relevant to future FPGA development environments. These include Vivado ML Edition, introduced in 2021 and subsequently incorporated into the Vivado Design Suite [10] by AMD and Xilinx, Synopsys.ai Copilot [11] by Synopsys, and Aprisa AI [12] by Siemens.

Vivado directly targets FPGA implementation, whereas Synopsys.ai Copilot and Aprisa AI primarily support ASIC or broader semiconductor design flows. Nevertheless, their capabilities in RTL generation, verification, and optimization indicate how similar AI functions could be incorporated into FPGA

development environments. Commercial AI-assisted tools specifically targeting FPGA development remain limited.

3.2 Research Trends in AI-Assisted EDA

The application of AI to EDA has gradually evolved from optimizing individual design tasks to supporting and automating broader design workflows. Zang et al. [17] surveyed 26 studies on agentic EDA published between 2021 and 2025. Drawing partly on this survey, Table II summarizes representative studies according to the EDA tasks addressed and the AI techniques employed [18-61]. The following discussion traces this evolution from task-specific optimization to agentic orchestration of the overall EDA flow.

Early AI-for-EDA approaches primarily applied machine learning (ML) techniques to individual tasks, such as placement [38-45], congestion prediction [47-49], and design parameter optimization [35]. More recently, advances in large language models (LLMs) and generative AI have extended the application of AI to generative and reasoning-intensive tasks, including RTL/Verilog generation [18-25,27], verification [54,55], and High-Level Synthesis (HLS) optimization [29,30].

AI is also increasingly being used to coordinate multiple EDA tasks and optimize design outcomes across broader workflows. More recent studies have explored agentic EDA [59-61], in which AI agents reason and plan, interact with existing EDA tools, and iteratively refine designs based on tool feedback. This trend suggests that the role of AI in EDA is expanding from task-specific prediction and assistance toward more active participation in the design workflow.

Overall, the survey reveals two notable trends. First, conventional ML techniques are used primarily for task-specific prediction and optimization, particularly in backend activities such as parameter tuning, placement, and congestion prediction [35,38-45,47-49], whereas LLMs and generative AI are increasingly applied to frontend activities such as RTL/Verilog generation, HLS optimization, and verification [18-25,27,29,30,54, 55]. Second, recent research is expanding toward agentic orchestration across multiple EDA tasks [59-61], while commercial AI-assisted support specifically targeting FPGA development remains limited [10-12]. Most generative and agentic capabilities are currently demonstrated in research prototypes or tools primarily targeting ASIC and broader semiconductor design flows.

This evolution also has implications for COTS software dedication. As AI functionality expands from isolated prediction and optimization to artifact generation and autonomous design modification, AI may exert more direct and extensive influence on safety-related FPGA design outputs. At the same time, establishing the provenance, reproducibility, and validity of AI-generated or AI-modified outputs may become more challenging.

4. Considerations for COTS Software Dedication

Previous work [8] addressed both direct and indirect COTS software but did not explicitly consider AI-aided development tools. Its categorization approach can still be used to assess the safety significance of such tools based on the safety category of the target system and their intended use. However, the associated acceptance criteria were developed for conventional COTS software and do not explicitly address the following AI-related assurance concerns:

1) AI behavior and performance — AI-enabled functions may exhibit variable or uncertain performance depending on input characteristics and operational conditions.
2) Data and model assurance — The quality, representativeness, provenance, and versioning of the data and models may affect the reliability of tool outputs.
3) AI-specific V&V — Additional V&V may be needed to evaluate robustness, output variability, uncertainty, and performance under previously unseen or boundary conditions.
4) Configuration and change management — Changes to models, training data, prompts, dependencies, or tool configurations may alter tool behavior and outputs.
5) Operating experience — Relevant operating experience may be limited or may not remain applicable as AI models and tool versions evolve.
6) Problem reporting and traceability — Problems and generated outputs should be traceable to the corresponding model, data, configuration, and tool version.

These areas may therefore require additional considerations when the approach is applied to AI-aided FPGA development tools.

5. Conclusion

This study surveyed commercial tools and recent research on AI-assisted EDA relevant to FPGA development. The survey shows that the role of AI is expanding from task-specific prediction and optimization to artifact generation, design modification, and workflow orchestration. Although the existing COTS software categorization approach remains applicable, its acceptance criteria do not explicitly address several assurance concerns arising from these expanded AI capabilities. Future work will examine these concerns in relation to individual acceptance criteria and develop a graded dedication approach reflecting the role and influence of AI in FPGA development.

ACKNOWLEDGMENTS

This work was supported by the Nuclear Safety Research Program through the Regulatory Research Management Agency for SMRS (RMAS) and the Nuclear Safety and Security Commission (NSSC) of the Republic of Korea (No. RS-2024-00509643).

REFERENCES

[1] K. C. Kwon, M. Lee, "Technical review on the localized digital instrumentation and control systems", Nuclear Engineering and Technology. vol.41, no.4, pp.447-454, 2009.
[2] U.S. Nuclear Regulatory Commission, "DSRS 7.0: Instrumentation and Controls—Introduction and Overview of Review Process", Technical Report, U.S. NRC, 2016.
[3] NuScale Power LLC, "NuScale Standard Plant Design Certification Application—Chapter 7 Instrumentation and Controls - Part 02 – Tier 2", Technical Report, NuScale Power LLC, 2020.
[4] U.S. Nuclear Regulatory Commission, "NUREG/CR-6421: A Proposed Acceptance Process for Commercial Off-the-Shelf (COTS) Software in Reactor Applications", Technical Report, U.S. NRC, 1996.
[5] L.-T. Wang, Y.-W. Chang, K.-T. T. Cheng, "Electronic design automation: synthesis, verification, and test", Morgan Kaufmann, 2009.
[6] Cadence, "What is Electronic Design Automation (EDA)?", [Available Online] https://www.cadence.com/en_US/home/explore/what-is-electronic-design-automation.html, Latest access: 19 Aug 26.
[7] K. Tu, X. Tang, C. Yu, L. Josipović, Z. Chu, "FPGA EDA", Springer, 2024.
[8] S. Jung, E.-S. Kim, J. Yoo, J.-Y. Kim, J. G. Choi, "An Evaluation and Acceptance of COTS Software for FPGA-based Controllers in NPPS", Annals of Nuclear Energy, vol.94, pp.338-349, 2016.
[9] Electric Power Research Institute, "Plant Engineering: Guideline for the Acceptance of Commercial-Grade Items in Nuclear Safety-Related Applications (NP-5652)", Technical Report, EPRI, 2014.
[10] Xilinx, "ML-Based Routing Congestion and Delay Estimation in Vivado ML Edition", White Paper, WP532 (v1.0), Xilinx, 2021.
[11] Synopsys, "New Synopsys.ai Copilots Deliver 2–5× Faster Chip Design Productivity", [Available Online] https://www.synopsys.com/blogs/chip-design/synopsys-ai-copilots-chip-design.html, Latest access: 19 Aug 26.
[12] Siemens, "Aprisa AI", [Available Online] https://www.siemens.com/en-us/products/ic/ic-design/aprisa/ai/, 2025, Latest access: 19 Aug 26.
[13] T. M. Mitchell, "Machine Learning", vol.1, no.9, McGraw-Hill, 1997.
[14] H. Naveed, A. U. Khan, S. Qiu, M. Saqib, S. Anwar, M. Usman, N. Akhtar, N. Barnes, A. Mian, "A Comprehensive Overview of Large Language Models", ACM Transactions on Intelligent Systems and Technology, vol.16, no.5, pp.1-72, 2025.
[15] J. Eisenstein, "Introduction to natural language processing.", MIT press, 2019.
[16] J. Euchner, "Generative AI", Research-Technology Management, vol.66, no.3, pp.71-74, 2023.
[17] Z. Zang, Y. Song, A. Wang, B. W.-K. Ling, Q. Sun, Z. Lei, F. Yang, C. Zhuo, J. Luo, "The Dawn of Agentic EDA: A Survey of Autonomous Digital Chip Design", v2, arXiv, 2026.
[18] K. Chang, Y. Wang, H. Ren, M. Wang, S. Liang, Y. Han, H. Li, X. Li, "ChipGPT: How far are we from natural language hardware design", v4, arXiv, 2025.
[19] S. Thakur, J. Blocklove, H. Pearce, B. Tan, S. Garg, R. Karri, "AutoChip: Automating HDL Generation Using LLM Feedback", v2, arXiv, 2024.
[20] Y. Lu, S. Liu, Q. Zhang, Z. Xie, "RTLLM: An Open-Source Benchmark for Design RTL Generation with Large Language Model", 2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC), pp.722-727, 2024.
[21] M. Liu, T.-D. Ene, R. Kirby, C. Cheng, N. Pinckney, R. Liang, J. Alben, H. Anand, S. Banerjee, I. Bayraktaroglu, B. Bhaskaran, B. Catanzaro, A. Chaudhuri, S. Clay, B. Dally, L. Dang, P. Deshpande, S. Dhodhi, S. Halepete, E. Hill, J. Hu, S. Jain, A. Jindal, B. Khailany, G. Kokai, K. Kunal, X. Li, C. Lind, H. Liu, S. Oberman, S. Omar, G. Pasandi, S. Pratty, J. Raiman, A. Sarkar, Z. Shao, H. Sun, P. P. Suthar, V. Tej, W. Turner, K. Xu, H. Ren, "ChipNeMo: Domain-Adapted LLMs for Chip Design", v5, arXiv, 2024.
[22] C. Deng, Y.-D. Tsai, G.-T. Liu, Z. Yu, H. Ren, "ScaleRTL: Scaling LLMs with Reasoning Data and Test-Time Compute for Accurate RTL Code Generation", v2, arXiv, 2025.
[23] K. Min, K. Cho, J. Jang, S. Kang, "REvolution: An Evolutionary Framework for RTL Generation driven by Large Language Models", 2026 31st Asia and South Pacific Design Automation Conference (ASP-DAC), pp.282-288, 2026.
[24] Z. Yu, M. Liu, M. Zimmer, Y. Celine, Y. Liu, H. Ren, "Spec2RTL-Agent: Automated Hardware Code Generation from Complex Specifications Using LLM Agent Systems", 2025 IEEE International Conference on LLM-Aided Design (ICLAD), pp.37-43, 2025.
[25] S. Liu, Y. Lu, W. Fang, M. Li, Z. Xie, Z. (2024, October). "OpenLLM-RTL: Open Dataset and Benchmark for LLM-Aided Design RTL Generation", Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design, pp.1-9, 2024.
[26] S. Allmendinger, N. Kühl, L. Bonenberger, K. Endres, Dominik Fetzer, Henner Gimpel, "Multi-agent AI", Electron Markets, vol.36, no.18, 2026.
[27] C.-T. Ho, H. Ren, B. Khailany, "VerilogCoder: Autonomous Verilog Coding Agents with Graph-based Planning and Abstract Syntax Tree (AST)-based Waveform Tracing Tool", Proceedings of the AAAI Conference on Artificial Intelligence, vol.39, no.1, pp.300-307, 2025.
[28] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-T. Yih, T. Rocktäschel, S. Riedel, D. Kiela, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks", Advances in Neural Information Processing Systems 33 (NeurIPS 2020), 2020.
[29] N. Mashnoor, M. Akyash, H. Kamali, K. Azar, "TimelyHLS: LLM-Based Timing-Aware and Architecture-Specific FPGA HLS Optimization", v1, arXiv, 2025.
[30] M. Moraru, K. Kamalakkannan, J.Dominguez-Trujillo, P. Diehl, A. Barai, J. Loiseau, Z. K. Baker, H. Pritchard, G. M. Shipman, "LAAFD: LLM-based Agents for Accelerated FPGA Design", v1, arXiv, 2026.
[31] R. S. Sutton, A. G. Barto, "Reinforcement learning: An introduction", MIT press, 1998.
[32] Y. Chen, W. Guan, R. Azzam. "Residual bayesian attention networks for uncertainty quantification in regression tasks." Scientific reports (Nature), vol.15, 2025.

[33] Y. Qian, X. Zhou, H. Zhou, L. Wang, "An Efficient Reinforcement Learning Based Framework for Exploring Logic Synthesis", ACM Transactions on Design Automation of Electronic Systems, vol.29, no.2, pp.1-33, 2024.
[34] J. Liu, Q. Zhao, "AiMap+: Guiding Technology Mapping for ASICs via Learning Delay Prediction", Electronics, vol.13, no.18, 2024.
[35] N. Kapre, H. Ng, K. Teo, J. Naude, "InTime: A Machine Learning Approach for Efficient Selection of FPGA CAD Tool Parameters", 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA '15), 2015.
[36] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, G. Monfardini, "The graph neural network model", IEEE transactions on neural networks, vol.20, no.1, pp.61-80, 2008.
[37] N. Ketkar, J. Moolayil, "Convolutional Neural Networks", In: Deep Learning with Python: Learn Best Practices of Deep Learning Models with PyTorch, pp.197-242, Apress, 2021.
[38] Y. Lin, Z. Jiang, J. Gu, W. Li, S. Dhar, H. Ren, B. Khailany, D. Z. Pan, "DREAMPlace: Deep Learning Toolkit-Enabled GPU Acceleration for Modern VLSI Placement", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol.40, no.4, pp.748-761, 2021.
[39] Y. Lai, Y. Mu, P. Luo, "MaskPlace: Fast Chip Placement via Reinforced Visual Representation Learning", v1, arXiv, 2022.
[40] Y. Liu, Z. Ju, Z. Li, M. Dong, H. Zhou, J. Wang, F. Yang, X. Zeng, L. Shang, "GraphPlanner: Floorplanning with Graph Neural Network," ACM Transactions on Design Automation of Electronic Systems, vol.28, no.2, Article 21 pp.1-24, 2022.
[41] A. Mirhoseini, A. Goldie, M. Yazgan, J. W. Jiang, E. Songhori, S. Wang, Y.-J. Lee, E. Johnson, O. Pathak, A. Nova, J. Pak, A. Tong, K. Srinivasa, W. Hang, E. Tuncer, Q. V. Le, J. Laudon, R. Ho, R. Carpenter, J. Dean, "A graph placement methodology for fast chip design", Nature, vol.594, no.7862, pp.207-212, 2021.
[42] Y. Hou, H. Ye, S. Yang, Y. Zhang, S. Xu, G. Song, "TransPlace: Transferable Circuit Global Placement via Graph Neural Network", Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1, pp.484-495, 2025.
[43] V. Lee, M. Nguyen, L. Elzeiny, C. Deng, P. Abbeel, J. Wawrzynek, "Chip Placement with Diffusion Models", v3, arXiv, 2025.
[44] A. Mirhoseini, A. Goldie, M. Yazgan, J. Jiang, E. Songhori, S. Wang, Y.-J. Lee, E. Johnson, O. Pathak, S. Bae, A. Nazi, J. Pak, A. Tong, K. Srinivasa, W. Hang, E. Tuncer, A. Babu, Q. V. Le, J. Laudon, R. Ho, R. Carpenter, J. Dean, "Chip Placement with Deep Reinforcement Learning", v1, arXiv, 2020.
[45] A. Agnesina, K. Chang, S. K. Lim, "Parameter Optimization of VLSI Placement Through Deep Reinforcement Learning", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol.42, no.4, pp.1295-1308, 2023.
[46] J. Long, E. Shelhamer, T. Darrell, "Fully Convolutional Networks for Semantic Segmentation", Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3431-3440, 2015.
[47] C. Ma, Y. Xiao, S. Wang, J. Yu, J. Chen, "CongestNN: An Bi-Directional Congestion Prediction Framework for Large-Scale Heterogeneous FPGAs", 2021 IEEE 14th International Conference on ASIC (ASICON), pp.1-4, 2021.
[48] D. Maarouf, A. Alhyari, Z. Abuowaimer, T. Martin, A. Gunter, G. Grewal, "Machine-Learning Based Congestion Estimation for Modern FPGAs", 2018 28th International Conference on Field Programmable Logic and Applications (FPL), pp.427-4277, 2018.
[49] A. Alhyari, A. Shamli, Z. Abuwaimer, S. Areibi, G. Grewal, "A Deep Learning Framework to Predict Routability for FPGA Circuit Placement", 2019 29th International Conference on Field Programmable Logic and Applications (FPL), pp.334-341, 2019.
[50] T. Elsken, J. H. Metzen, F. Hutter, "Neural architecture search: A survey", Journal of Machine Learning Research, vol.20, no.55, pp.1-21, 2019.
[51] M. Pelikan, "Bayesian Optimization Algorithm", In: Hierarchical Bayesian Optimization Algorithm: Studies in Fuzziness and Soft Computing, pp.31-48, Springer, 2005.
[52] P. Xu, X. Zhang, C. Hao, Y. Zhao, Y. Zhang, Y. Wang, C. Li, Z. Guan, D. Chen, Y. Lin, "AutoDNNchip: An Automated DNN Chip Predictor and Builder for Both FPGAs and ASICs", v4, arXiv, 2020.
[53] O. Hasan, S. Tahar, "Formal Verification Methods", In: Encyclopedia of Information Science and Technology, 3rd Edition, pp.7162-7170, 2015.
[54] Y. Bai, G. B. Hamad, C.-T. Ho, S. Suhaib, H. Ren, "FVDebug: An LLM-Driven Debugging Assistant for Automated Root Cause Analysis of Formal Verification Failures", v1, arXiv, 2025.
[55] B. Ahmad, J. Ah-Kiow, B. Tan, R. Karri, H. Pearce, "FLAG: Finding Line Anomalies (in RTL code) with Generative AI", ACM Transactions on Design Automation of Electronic Systems, vol.30, no.6, pp.1-30, 2025.
[56] M. Sauerhoff, "Approximation of boolean functions by combinatorial rectangles", Theoretical Computer Science, vol.301, no.1-3, pp.45-78, 2003.
[57] S. Cheng, P. Jin, Q. Guo, Z. Du, R Zhang, X. Hu, Y. Zhao, Y. Hao, X. Guan, H. Han, Z. Zhao, X. Liu, X. Zhang, Y. Chu, W. Mao, T. Chen, Y. Chen, "Automated CPU Design by Learning from Input-Output Examples", Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI-24), pp.3843-3853, 2024.
[58] D. B. Acharya, K. Kuppan, B. Divya, "Agentic AI: Autonomous Intelligence for Complex Goals—A Comprehensive Survey", IEEE Access, vol.13, pp.18912-18936, 2025.
[59] A. Patra, S. Rout, A. Ravindran, "AiEDA: Agentic AI Design Framework for Digital ASIC System Design", v1, arXiv, 2024.
[60] M. E. Yanık, İ. Çiçek, E. Afacan, "ShortCircuit: An Open-Source ChatGPT Driven Digital Integrated Circuit Front-End Design Automation Tool", 2023 30th IEEE International Conference on Electronics, Circuits and Systems (ICECS), pp.1-4, 2023.
[61] M. A. Elgammal, J. Wu, L. Liu, T. Kim, V. Betz, "VTR-LLM: Multi-Agent LLM Framework for Automated Debugging of FPGA CAD Flows", ACM Transactions on Reconfigurable Technology and Systems, Accepted, 2026.