A survey of RTL simulation acceleration

1st Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

2nd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

3rd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

4th Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

5th Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

6th Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

***Abstract*—This literature review provides a comprehensive survey of the state-of-the-art in Register-Transfer Level (RTL) simulation acceleration, a critical area in the digital design and verification process. With the increasing complexity of System-on-Chip (SoC) designs, the demand for faster and more efficient RTL simulation methods has become paramount. This survey explores a wide spectrum of acceleration techniques, from software optimizations and hardware-accelerated methods to high-level synthesis (HLS) and pre-RTL simulation frameworks. It also delves into innovative and hybrid approaches, such as symbolic simulation, fuzz testing, and compiler-driven optimizations, which offer potential solutions to the limitations of traditional simulation methods. By examining both foundational techniques and cutting-edge research, this review highlights the challenges, trends, and future directions in RTL simulation acceleration. The goal is to provide insights into how accelerated RTL simulation can reduce time-to-market, optimize design cycles, and enhance the reliability and efficiency of digital circuits, thereby supporting the advancement of digital design and verification processes.**

***Index Terms*—**

I. Introduction

Register-Transfer Level (RTL) simulation represents a cornerstone in the digital design and verification process, enabling engineers to validate the correctness and performance of hardware designs before physical fabrication. As the complexity of System-on-Chip (SoC) designs escalates, driven by the relentless pursuit of higher performance, lower power consumption, and greater functionality, the computational demands of RTL simulation have surged. This has led to a pressing need for accelerated RTL simulation techniques that can keep pace with the rapid development cycles and intricate design requirements of modern electronics.

The acceleration of RTL simulation is not merely a matter of convenience but a critical factor in reducing time-to-market, optimizing design cycles, and ensuring the reliability and efficiency of digital circuits. Traditional simulation methodologies, while accurate, are increasingly unable to meet the stringent time constraints and scalability requirements imposed by today's complex SoC designs. This has spurred a wealth of research and development efforts aimed at devising innovative approaches to accelerate RTL simulation, ranging from software optimizations and hardware-accelerated methods to high-level synthesis (HLS) and pre-RTL simulation frameworks.

Software optimizations seek to enhance the efficiency of simulation algorithms and better exploit the capabilities of modern processors, whereas hardware-accelerated simulation leverages the parallel processing power of GPUs and FPGAs to dramatically reduce simulation times. Meanwhile, HLS and pre-RTL simulation frameworks offer avenues for early-phase verification and rapid prototyping, potentially circumventing the bottlenecks of traditional RTL simulation.

Furthermore, the exploration of innovative and hybrid approaches, such as symbolic simulation, fuzz testing, and compiler-driven optimizations, reflects the dynamic and multifaceted nature of research in this area. These efforts not only aim to directly accelerate simulation but also to unlock auxiliary benefits, including improved debugging capabilities, enhanced performance evaluation, and more efficient test generation.

This literature survey delves into the state-of-the-art in RTL simulation acceleration, covering foundational techniques, direct approaches, hardware-specific acceleration methods, and the role of HLS and pre-RTL simulation frameworks. It also explores innovative and hybrid approaches that hold promise for overcoming the limitations of conventional methods. By examining the broader context and auxiliary benefits of accelerated RTL simulation, this survey aims to provide a comprehensive overview of current trends, challenges, and future directions in the field, ultimately contributing to the advancement of digital design and verification processes.

II. Foundations and Direct Approaches for RTL Simulation Acceleration

This section explores the foundational techniques and direct approaches employed to accelerate RTL simulation. It covers

Identify applicable funding agency here. If none, delete this.

both software optimizations and hardware-accelerated methods, including leveraging GPUs and FPGAs for simulation tasks. The acceleration of RTL simulation is crucial for the efficient design and verification of digital circuits, as it directly impacts the productivity and agility of hardware development processes.

A. *Software Optimizations for RTL Simulation*

Software-based RTL simulation acceleration focuses on optimizing the simulation algorithms and exploiting the capabilities of the host processor more efficiently. The work by Beamer et al. [1] highlights the potential for accelerating software RTL simulation by comparing open-source simulators to commercial ones. Their findings suggest that open-source simulators not only outperform some commercial simulators but also achieve higher instruction throughput, indicating significant room for software simulation acceleration. This is further supported by the exploration of low activity factors in digital designs by Beamer et al. [2], which presents a direct approach to improve simulation speed by leveraging the inherent signal activity characteristics of digital circuits.

Moreover, the introduction of ESSENT, a high-performance RTL simulator [3], demonstrates the feasibility of achieving considerable simulation acceleration through software optimizations. ESSENT incorporates novel optimizations that typically render it faster than other software RTL simulators, serving as a foundation for further research in simulation acceleration.

B. *Hardware-Accelerated Simulation Approaches*

The limitations of software-only simulation methods in keeping pace with the increasing complexity of digital designs have led to the exploration of hardware-accelerated simulation approaches. ASH, presented by Elsabbagh et al. [4], introduces a hardware-software co-design approach specifically aimed at accelerating RTL simulation. ASH leverages fine-grained parallelism and selective event-driven execution to significantly reduce simulation time, showcasing the potential of tightly integrated hardware-software solutions.

Similarly, the Manticore project [5] utilizes static bulk-synchronous parallelism to address the challenges of fine-grain parallelism in RTL simulation. By eliminating fine-grain synchronization overhead and relying on static scheduling, Manticore achieves substantial speed improvements, demonstrating the effectiveness of hardware-accelerated approaches.

C. *Emerging Techniques and Novel Approaches*

Recent advancements have introduced innovative techniques aimed at further accelerating RTL simulation. SiFI-AI [6] combines AI inference with cycle-accurate simulation to accelerate RTL fault simulation for AI accelerators, specifically targeting DNNs. This approach not only speeds up the simulation process but also provides insights into the resilience of DNN layers to hardware faults.

On the other hand, symbolic simulation, as discussed by Kolbi et al. [7], offers a promising avenue for enhancing RTL simulation through the use of symbolic methods. This technique enables the simulation of a complete set of RT-level Verilog constructs with full delay support, potentially accelerating the verification process.

Furthermore, the exploration of loop-oriented code instrumentation by Mao et al. [8] introduces a novel method for reducing simulation time while maintaining accuracy. This approach exemplifies the continuous search for innovative solutions to accelerate RTL simulation, highlighting the dynamic nature of research in this area.

In conclusion, the acceleration of RTL simulation is a multifaceted challenge that encompasses both software optimizations and hardware-accelerated approaches. The ongoing research and development in this field are crucial for advancing digital design and verification processes, ultimately contributing to the faster realization of complex hardware systems.

III. Hardware-Specific Acceleration Techniques

Focusing on hardware-accelerated simulation, this section discusses specific techniques and frameworks that utilize FPGAs and GPUs to enhance RTL simulation speed, particularly for complex SoC designs and memory subsystems. The acceleration of RTL simulation is a critical area of research and development, aiming to reduce the time and resources required for the validation and testing of complex digital systems. This section delves into various approaches and methodologies that leverage the computational capabilities of FPGAs and GPUs to achieve significant improvements in simulation performance.

A. *FPGA-Based Acceleration Techniques*

FPGA-based acceleration techniques have shown promising results in enhancing the speed and efficiency of RTL simulation. The Chipyard framework, as presented by [9], exemplifies an integrated environment that facilitates the rapid validation and implementation of custom SoCs. By leveraging FPGA-accelerated simulation, Chipyard underscores the potential of FPGAs in streamlining the design and verification process of specialized compute systems.

Further emphasizing the role of FPGAs in simulation acceleration, the FASED project [10] introduces a parameterized generator of composable, high-fidelity FPGA-hosted last-level-cache and DRAM models. This approach not only enhances the accuracy of FPGA simulations but also demonstrates the feasibility of modeling complex memory organizations without the need for extensive RTL resynthesis.

The DESSERT methodology [11], [12] and the evaluation of RISC-V RTL with FPGA-accelerated simulation [13] further illustrate the effectiveness of FPGA-based techniques in accelerating RTL simulation. These works highlight the advantages of deterministic simulation on FPGAs for debugging and performance evaluation, offering insights into practical acceleration methods that can significantly reduce development cycles.

B. *GPU-Accelerated Simulation Approaches*

Parallel computing capabilities of GPUs offer another avenue for accelerating RTL simulation. The FAST-GP framework [14] introduces a GPU-based parallel simulation approach for RTL functional verification, leveraging fault injection and parallel automatic test pattern generation to expedite the verification process.

Expanding on the potential of GPUs, the work presented in [15] proposes RTLFlow, a GPU-accelerated RTL simulation flow that utilizes batch stimulus for high-throughput simulation. By transpiling RTL into CUDA kernels, RTLFlow achieves significant runtime speed-ups, demonstrating the scalability and efficiency of GPU-accelerated simulation for industrial-scale designs.

C. *Emerging Trends and Cross-Domain Applications*

The acceleration of RTL simulation is not limited to traditional digital circuit verification. The ALAMO compiler [16] showcases the application of FPGA acceleration in the domain of deep learning, providing a modularized and scalable solution for the efficient execution of convolutional neural networks. This cross-domain application underscores the versatility of FPGA-based acceleration techniques in addressing the computational demands of emerging technologies.

Moreover, the SimBSP tool [17] and the GPU-accelerated analog circuit simulation presented in [18] highlight the broader applicability of hardware-specific acceleration methods. These works illustrate the potential of leveraging FPGAs and GPUs to overcome simulation challenges across different domains, from digital RTL simulation to analog circuit analysis.

In conclusion, hardware-specific acceleration techniques, particularly those based on FPGAs and GPUs, play a pivotal role in advancing the state-of-the-art in RTL simulation. By harnessing the computational power of these platforms, researchers and practitioners can achieve significant improvements in simulation speed and efficiency, facilitating the rapid development and validation of complex digital systems. As technology continues to evolve, the exploration of innovative acceleration methodologies and their application across various domains will remain a critical area of research.

IV. High-Level Synthesis (HLS) and Pre-RTL Simulation Frameworks

This section covers the role of high-level synthesis and pre-RTL simulation frameworks in accelerating the RTL simulation process. It highlights how early phase verification and fast prototyping can contribute to overall simulation efficiency. The evolution of HLS tools and the introduction of pre-RTL simulation frameworks have significantly impacted the design and verification process, enabling faster prototyping and early detection of design issues. This section delves into the advancements in HLS and pre-RTL simulation frameworks, their contributions to simulation acceleration, and the challenges and opportunities that lie ahead.

A. *Advancements in HLS Tools*

High-Level Synthesis (HLS) tools have evolved significantly, bridging the gap between high-level design and RTL implementation. The FLASH simulator, as discussed by Choi et al. [19], exemplifies this evolution by proposing a new HLS simulation flow that accelerates the simulation process while improving performance estimation. FLASH extracts scheduling information from the HLS tool to construct an equivalent cycle-accurate simulation model, running orders of magnitude faster than traditional RTL simulation. This advancement underscores the potential of HLS tools to accelerate the RTL simulation process by providing faster and more accurate performance estimations.

Furthermore, the GAUT tool, dedicated to DSP applications, emphasizes the transition from C algorithms to RTL architecture [20]. By extracting potential parallelism and processing allocation, scheduling, and binding tasks, GAUT facilitates a streamlined design process that can indirectly impact RTL simulation acceleration. These advancements in HLS tools not only enhance simulation efficiency but also open up new opportunities for research and development in simulation acceleration.

B. *Pre-RTL Simulation Frameworks*

Pre-RTL simulation frameworks have emerged as powerful tools for early architectural evaluation and fast prototyping. SimuNN, introduced by Cao et al. [21], is a pre-RTL neural network simulator that enables early phase verification and fast prototyping for neural network hardware accelerators. By supporting inference in various data precision and compatibility with TensorFlow, SimuNN facilitates the design of both neural network models and hardware accelerators, thereby indirectly supporting RTL simulation acceleration.

Similarly, the gem5-accel toolchain, proposed by Vieira et al. [22], demonstrates the utility of pre-RTL simulation for early architectural evaluation of accelerators. By modeling complex accelerators and anticipating the results of hardware execution, gem5-accel accelerates the architecture validation process, contributing to faster RTL code development. These pre-RTL simulation frameworks exemplify how early phase verification and fast prototyping can significantly enhance the efficiency of the RTL simulation process.

C. *Challenges and Opportunities*

Despite the advancements in HLS tools and pre-RTL simulation frameworks, several challenges remain. The quality of results (QoRs) of HLS tools, as discussed by Lahti et al. [**?**], tends to lag behind manual RTL flows. However, HLS tools significantly reduce development time and increase productivity, indicating a trade-off between QoR and efficiency. Addressing this gap presents a significant opportunity for further research and development in HLS and simulation acceleration.

Moreover, the broader challenges and opportunities in HLS, highlighted by Cong et al. [23], such as achieving high clock frequency, coping with complex pragmas, and supporting domain-specific languages, indirectly affect RTL simulation acceleration. Addressing these challenges requires a concerted effort from the research community to advance HLS technology and simulation frameworks, ultimately contributing to the acceleration of the RTL simulation process.

In conclusion, HLS tools and pre-RTL simulation frameworks play a crucial role in accelerating the RTL simulation process through early phase verification and fast prototyping. While significant advancements have been made, challenges remain, presenting opportunities for further research and development in this field.

V. Innovative and Hybrid Approaches to Simulation Acceleration

Exploring novel methodologies in RTL simulation acceleration, this section delves into hybrid approaches that combine different techniques, such as symbolic simulation, fuzz testing, and compiler-driven optimizations, to achieve significant speed improvements. These innovative strategies not only aim to enhance the efficiency of simulation but also strive to address the growing complexity of modern System-on-a-Chip (SoC) designs. By integrating various acceleration techniques, researchers have proposed solutions that offer promising results in overcoming the limitations of traditional simulation methods.

A. Hybrid Simulation Techniques

One of the standout hybrid approaches in RTL simulation acceleration is the combination of symbolic simulation and fuzz testing. Li et al. [24] introduced a novel method that leverages the strengths of both symbolic simulation and mutation-based fuzz testing to enhance coverage-directed dynamic verification of RTL designs. Their approach, which utilizes Full Multiplexer Toggle Coverage (FMTC) for feedback, interleaves symbolic simulation and fuzz testing passes to achieve high coverage. Symbolic simulation generates tests that target untouched corners of the design, while fuzz testing handles test generation for large-scale designs. This hybrid method demonstrates the potential to significantly accelerate RTL simulation by efficiently exploring the design space.

B. Partitioning and Distributed Simulation

Another avenue for accelerating RTL simulation is through efficient partitioning algorithms for distributed simulation. Mahapatra et al. [25] explored partitioning Data Flow Graphs (DFG) for Coarse Grained Reconfigurable Array assisted Hardware Accelerators (CGRA-HA), proposing algorithms that minimize inter-processor communication. Their work highlights the importance of distributed simulation in handling the increasing complexity of circuit designs and presents partitioning as a critical step in optimizing simulation acceleration. By reducing the cut-edges in partitioning, their approach aims to enhance the efficiency of distributed RTL simulation, offering a pathway to address the bottleneck in the IC design process.

C. Advancements in Test Generation and Tandem Simulation

In addition to hybrid and distributed simulation techniques, advancements in test generation and tandem simulation also contribute to simulation acceleration. Pinto et al. [26] introduced CORT, a methodology based on factored concolic execution for RTL functional test generation. By transforming RTL source into a high-performance C++ compiled functional simulator and utilizing a novel Test Decision Tree (TDT) representation, CORT achieves high branch coverage with fewer input vectors. This method indirectly accelerates RTL simulation by optimizing test generation, showcasing the potential of auxiliary methods in enhancing simulation efficiency.

Furthermore, Xing et al. [27] generalized tandem simulation for both processors and accelerators, connecting high-level and RTL models to improve simulation efficiency. Their approach leverages Instruction-level Abstractions (ILAs) to automate the connection between models, demonstrating practical applicability in various case studies. This advancement in tandem simulation underscores the importance of cross-level simulation in accelerating the validation process.

Lastly, Zeng et al. [28] proposed an approach for automatically generating architecture-level models from RTL designs. By extracting architectural state variables and update functions, their method aims to reduce the time required for RTL simulation by providing higher abstraction levels. This automated approach represents a significant step towards accelerating the RTL simulation process through the generation of high-quality architecture-level models.

In conclusion, the exploration of innovative and hybrid approaches to simulation acceleration reveals a multifaceted landscape where symbolic simulation, fuzz testing, partitioning algorithms, test generation methodologies, and tandem simulation converge to address the challenges of RTL simulation. These advancements not only demonstrate the potential for significant speed improvements but also highlight the ongoing efforts to tackle the increasing complexity of SoC designs. As the field continues to evolve, further research and development in these areas are essential for realizing the full potential of simulation acceleration in the era of complex circuit designs.

VI. Broader Context and Auxiliary Benefits of Accelerated RTL Simulation

Beyond direct acceleration techniques, this section discusses the broader implications of accelerated RTL simulation, including improvements in debugging, test generation, and the auxiliary benefits such as enhanced power and energy modeling capabilities. The evolution of System on Chip (SoC) designs and the increasing complexity of electronic design automation (EDA) tasks necessitate a reevaluation of traditional RTL simulation approaches. This reevaluation not only focuses on accelerating the simulation process but also on leveraging the simulation environment for broader benefits such as improved performance evaluation, debugging, and system-level integration testing.

A. Enhanced Performance Evaluation and Debugging

The work by Chusov et al. [29] introduces a configurable test environment for cycle-accurate Network on Chip (NoC) simulation within SoCs. This environment aids in evaluating NoC performance, showcasing how tailored simulation environments can provide detailed insights into system behavior, which is crucial for optimizing communication between computational units in SoCs. Such environments extend the utility of RTL simulation beyond mere acceleration, enabling precise performance evaluation at early development stages.

Similarly, the integration of RTL models within full-system simulators, as explored by Lopez-Paradis et al. [30], presents a significant advancement in debugging capabilities. By allowing RTL models to operate within a full-system context, developers can identify and rectify functional and performance issues more efficiently, thereby reducing the overall development cycle time.

B. Cloud-Based Acceleration and Parallel Computing

The characterization and optimization of EDA flows for cloud deployment by Hosny et al. [31] highlight the potential of cloud computing in accelerating RTL simulation tasks. By understanding the specific requirements of EDA jobs in cloud environments, it becomes possible to optimize resource allocation, thereby reducing costs and improving simulation speed. This approach opens up new avenues for accelerating RTL simulation through cloud-based resources.

Furthermore, the general-purpose parallel and heterogeneous task programming system introduced by Huang et al. [32] addresses the challenge of implementing parallel CAD algorithms. By leveraging parallelism and heterogeneous computing, this system demonstrates the potential for significant performance improvements in RTL simulation tasks, highlighting the importance of parallel and heterogeneous computing approaches in the broader context of CAD and RTL simulation acceleration.

C. Future Perspectives on Simulation Acceleration

Emerging technologies such as large language models for zero-shot RTL code generation, as discussed by Sandal et al. [33], represent a promising direction for further accelerating the RTL simulation process. By streamlining the design and code generation phases, these technologies have the potential to significantly reduce iteration cycles and facilitate the exploration of complex design spaces. This approach underscores the expanding role of artificial intelligence and machine learning in shaping the future of hardware design automation and simulation acceleration.

In conclusion, accelerated RTL simulation not only offers direct benefits in terms of reduced simulation times but also facilitates a range of auxiliary benefits including enhanced performance evaluation, improved debugging capabilities, and the potential for leveraging cloud-based resources and parallel computing. As the complexity of SoC designs continues to increase, these broader implications and auxiliary benefits will play a crucial role in meeting the challenges of modern electronic design automation.

References

[1] S. Beamer, "A Case for Accelerating Software RTL Simulation," vol. 40, no. 4, pp. 112–119. [Online]. Available: https://ieeexplore.ieee.org/document/9099598/

[2] S. Beamer and D. Donofrio, "Efficiently Exploiting Low Activity Factors to Accelerate RTL Simulation," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/9218632/

[3] S. Beamer, T. Nijssen, K. Pandian, and K. Zhang, "ESSENT: A High-Performance RTL Simulator."

[4] F. Elsabbagh, S. Sheikhha, V. A. Ying, Q. M. Nguyen, J. S. Emer, and D. Sanchez, "Accelerating RTL Simulation with Hardware-Software Co-Design," in *56th Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, pp. 153–166. [Online]. Available: https://dl.acm.org/doi/10.1145/3613424.3614257

[5] M. Emami, S. Kashani, K. Kamahori, M. S. Pourghannad, R. Raj, and J. R. Larus, "Manticore: Hardware-Accelerated RTL Simulation with Static Bulk-Synchronous Parallelism," in *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 4*, pp. 219–237. [Online]. Available: http://arxiv.org/abs/2301.09413

[6] J. Hoefer, F. Kempf, T. Hotfilter, F. Kreß, T. Harbaum, and J. Becker, "SiFI-AI: A Fast and Flexible RTL Fault Simulation Framework Tailored for AI Models and Accelerators," in *Proceedings of the Great Lakes Symposium on VLSI 2023*. ACM, pp. 287–292. [Online]. Available: https://dl.acm.org/doi/10.1145/3583781.3590226

[7] A. Kolbi, J. Kukula, and R. Damiano, "Symbolic RTL simulation."

[8] F. Mao, Y. Guo, X. Liao, H. Jin, W. Zhang, H. Liu, L. Zheng, X. Liu, Z. Jiang, and X. Zheng, "Accelerating Loop-Oriented RTL Simulation With Code Instrumentation," vol. 42, no. 12, pp. 4985–4998. [Online]. Available: https://ieeexplore.ieee.org/document/10128151/

[9] A. Amid, D. Biancolin, A. Gonzalez, D. Grubb, S. Karandikar, H. Liew, A. Magyar, H. Mao, A. Ou, N. Pemberton, P. Rigge, C. Schmidt, J. Wright, J. Zhao, Y. S. Shao, K. Asanovic, and B. Nikolic, "Chipyard: Integrated Design, Simulation, and Implementation Framework for Custom SoCs," vol. 40, no. 4, pp. 10–21. [Online]. Available: https://ieeexplore.ieee.org/document/9099108/

[10] D. Biancolin, S. Karandikar, D. Kim, J. Koenig, A. Waterman, J. Bachrach, and K. Asanovic, "FASED: FPGA-Accelerated Simulation and Evaluation of DRAM," in *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. ACM, pp. 330–339. [Online]. Available: https://dl.acm.org/doi/10.1145/3289602.3293894

[11] D. Kim, C. Celio, S. Karandikar, D. Biancolin, J. Bachrach, and K. Asanovic, "Debugging RISC-V Processors with FPGA-Accelerated RTL Simulation in the FPGA Cloud."

[12] ——, "DESSERT: Debugging RTL Effectively with State Snapshotting for Error Replays across Trillions of Cycles," in *2018 28th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, pp. 76–764. [Online]. Available: https://ieeexplore.ieee.org/document/8533471/

[13] D. Kim, C. Celio, D. Biancolin, J. Bachrach, and K. Asanovic, "Evaluation of RISC-V RTL with FPGA-Accelerated Simulation."

[14] N. Bombieri, F. Fummi, and V. Guarnieri, "FAST-GP: An RTL functional verification framework based on fault simulation on GP-GPUs," in *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, pp. 562–565. [Online]. Available: http://ieeexplore.ieee.org/document/6176532/

[15] D.-L. Lin, H. Ren, Y. Zhang, B. Khailany, and T.-W. Huang, "From RTL to CUDA: A GPU Acceleration Flow for RTL Simulation with Batch Stimulus," in *Proceedings of the 51st International Conference on Parallel Processing*. ACM, pp. 1–12. [Online]. Available: https://dl.acm.org/doi/10.1145/3545008.3545091

[16] Y. Ma, N. Suda, Y. Cao, S. Vrudhula, and J.-s. Seo, "ALAMO: FPGA acceleration of deep learning algorithms with a modularized RTL compiler," vol. 62, pp. 14–23. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0167926017304777

[17] A. Sanaullah, C. Yang, D. Crawley, and M. C. Herbordt, "SimBSP: Enabling RTL Simulation for Intel FPGA OpenCL Kernels."

[18] C. Zhao, Z. Zhou, D. Wu, and Z. Z. Empyrean, “Empyrean ALPS-GT: GPU-accelerated Analog Circuit Simulation (Invited Talk).”

[19] Y.-K. Choi, Y. Chi, J. Wang, and J. Cong, “FLASH: Fast, Parallel, and Accurate Simulator for HLS,” vol. 39, no. 12, pp. 4828–4841. [Online]. Available: https://ieeexplore.ieee.org/document/8976247/

[20] P. Coussy, C. Chavet, P. Bomel, D. Heller, E. Senn, and E. Martin, “GAUT: A High-Level Synthesis Tool for DSP Applications: From C Algorithm to RTL Architecture,” in *High-Level Synthesis*, P. Coussy and A. Morawiec, Eds. Springer Netherlands, pp. 147–169. [Online]. Available: http://link.springer.com/10.1007/978-1-4020-8588-8_9

[21] S. Cao, W. Deng, Z. Bao, C. Xue, S. Xu, and S. Zhang, “SimuNN: A Pre-RTL Inference, Simulation and Evaluation Framework for Neural Networks,” vol. 10, no. 2, pp. 217–230. [Online]. Available: https://ieeexplore.ieee.org/document/9091143/

[22] J. Vieira, N. Roma, G. Falcao, and P. Tomás, “Gem5-accel: A Pre-RTL Simulation Toolchain for Accelerator Architecture Validation,” vol. 23, no. 1, pp. 1–4. [Online]. Available: https://ieeexplore.ieee.org/document/10304264/

[23] J. Cong, J. Lau, G. Liu, S. Neuendorffer, P. Pan, K. Vissers, and Z. Zhang, “FPGA HLS Today: Successes, Challenges, and Opportunities,” vol. 15, no. 4, pp. 1–42. [Online]. Available: https://dl.acm.org/doi/10.1145/3530775

[24] T. Li, H. Zou, D. Luo, and W. Qu, “Symbolic Simulation Enhanced Coverage-Directed Fuzz Testing of RTL Design,” in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, pp. 1–5. [Online]. Available: https://ieeexplore.ieee.org/document/9401267/

[25] I. B. Mahapatra, U. Agarwal, and S. K. Nandy, “DFG Partitioning Algorithms for Coarse Grained Reconfigurable Array Assisted RTL Simulation Accelerators,” in *2018 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*. IEEE, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/8482367/

[26] S. Pinto and M. S. Hsiao, “RTL functional test generation using factored concolic execution,” in *2017 IEEE International Test Conference (ITC)*. IEEE, pp. 1–10. [Online]. Available: http://ieeexplore.ieee.org/document/8242038/

[27] Y. Xing, A. Gupta, and S. Malik, “Generalizing Tandem Simulation: Connecting High-level and RTL Simulation Models,” in *2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, pp. 154–159. [Online]. Available: https://ieeexplore.ieee.org/document/9712564/

[28] Y. Zeng, A. Gupta, and S. Malik, “Automatic Generation of Architecture-Level Models from RTL Designs for Processors and Accelerators,” in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, pp. 460–465. [Online]. Available: https://ieeexplore.ieee.org/document/9774527/

[29] S. A. Chusov, E. V. Primakov, Y. V. Savchenko, A. L. Pereverzev, and E. S. Barkov, “Configurable Test Environment for RTL Simulation and Performance Evaluation of Network on Chip as Part of SoC,” in *2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus)*. IEEE, pp. 1969–1974. [Online]. Available: https://ieeexplore.ieee.org/document/9396634/

[30] G. López-Paradís, A. Armejach, and M. Moretó, “Gem5 + rtl: A Framework to Enable RTL Models Inside a Full-System Simulator,” in *50th International Conference on Parallel Processing*. ACM, pp. 1–11. [Online]. Available: https://dl.acm.org/doi/10.1145/3472456.3472461

[31] A. Hosny and S. Reda. Characterizing and Optimizing EDA Flows for the Cloud. [Online]. Available: http://arxiv.org/abs/2102.10800

[32] T.-W. Huang, “A general-purpose parallel and heterogeneous task programming system for VLSI CAD,” in *Proceedings of the 39th International Conference on Computer-Aided Design*. ACM, pp. 1–2. [Online]. Available: https://dl.acm.org/doi/10.1145/3400302.3415750

[33] S. Sandal and I. Akturk. Zero-Shot RTL Code Generation with Attention Sink Augmented Large Language Models. [Online]. Available: http://arxiv.org/abs/2401.08683