EDA Technology in Digital Circuit Design: A Study on Application Methodologies

Liu Ya

School of Electronic Information Engineering, Jiangxi University of Engineering, Xinyu 338000, Jiangxi, China

Abstract: ***Digital circuit design is a crucial foundational element of modern electronic engineering, and the application of EDA technology provides efficient, automated development methods, demonstrating significant advantages especially in FPGA design. This paper focuses on the practical application of EDA technology in digital circuit design, emphasizing optimization strategies such as standardizing design languages, strengthening timing control, streamlining resource structures, and refining simulation mechanisms. By integrating specific design cases, it analyzes the supporting role of EDA tools in modeling, synthesis, placement, and verification, promoting more efficient and reliable FPGA circuit development and comprehensively enhancing digital system performance.***

Keywords: Digital circuit design; EDA technology; FPGA; Automated design.

1. INTRODUCTION

The 14th Five-Year National Informatization Plan calls for accelerating breakthroughs in core foundational technologies such as integrated circuits, key electronic components, and EDA tools to build a self-controlled digital technology system. Under this guidance, the electronic information sector has placed higher demands on efficient and precise digital circuit design tools. Especially in FPGA development and application, improving design automation levels and optimizing development processes have become key directions for industry development. Driven by policy, the deep integration and engineering implementation of EDA technology have become critical issues in practice.

2. OVERVIEW OF EDA TECHNOLOGY

EDA technology, short for Electronic Design Automation, is an essential tool for modernizing and automating digital circuit design. Its core role is to leverage computer-aided design platforms to transform the traditionally manual drawing and verification of complex circuit design processes into streamlined, modular workflows, thereby greatly improving design efficiency. In digital circuit development, EDA technology covers multiple key stages, from early logic function modeling and hardware description language design, through mid-stage logic synthesis and timing analysis, to later placement and routing, functional simulation, and circuit testing. Through the collaborative action of various software tools, EDA technology can effectively manage the entire design flow, reduce design redundancy, avoid logic errors, and enhance circuit stability and resource utilization. Foundational progress in multimodal and 3D representation learning is exemplified by the work of Peng et al. (2025), who introduced 3D Vision-Language Gaussian Splatting[1]. In robotics and system architecture, Guo (2025) explored deterministic AI for optimal robotic trajectory control[2], while Zhou (2025) investigated performance monitoring and optimization within microservices architectures[3]. The healthcare sector has seen substantial AI-driven innovation, with Wei et al. (2025) developing intelligent health management systems for telemedicine[4], We et al. (2025) leveraging multimodal data for intelligent anesthesia depth monitoring[5], and Liu (2025) optimizing cardiac disease prediction models by integrating Adaboost with LSTM networks[11]. Concurrently, a strong research focus is on enhancing the capabilities of Large Language Models (LLMs). Zhang et al. (2024) proposed a multi-stage ensemble architecture with adaptive attention to boost logical reasoning[6], and Huang et al. (2025) enhanced document-level question answering through multi-hop retrieval-augmented generation[9]. Specialized applications of attention and fine-tuning mechanisms are also prominent, as seen in Zhang et al. (2025)'s use of dynamic cross-attention for fine-grained image captioning in advertising[7], Zhang et al. (2025)'s application of LLaMA-based meta-attention networks for automated essay assessment[8], and Wang and Bi (2025)'s hierarchical adaptive framework for multi-task learning in large-scale models[10]. Finally, from a public health informatics perspective, Su et al. (2025) conducted a structural assessment of familial and educational influences on student health behaviors[12].

3. ANALYSIS OF FPGA DIGITAL CIRCUIT DESIGN FLOW

The FPGA digital-circuit design flow is a highly systematic technical process, typically composed of several sequential stages that are tightly linked and indispensable. The entire flow generally begins with requirement analysis and circuit modeling of the logical functions, using a hardware description language to express the target functionality clearly and form an initial logic design. It then proceeds to functional simulation to verify whether the logic description is accurate and to ensure that the basic functions operate correctly under ideal conditions. After confirming the logic is correct, the design enters the synthesis stage, where tools automatically convert the logic description into a gate-level structure and optimize it according to the specific device resources. Next comes placement and routing, where the system performs the actual physical layout based on chip resources and timing requirements so that the logical functions can be realized in the target device. On this basis, static timing analysis is also required to evaluate whether the design meets the operating frequency and data-stability requirements, ultimately generating a configuration file that is downloaded into the FPGA chip and entering the actual testing phase.

4. FPGA DIGITAL-CIRCUIT DESIGN OPTIMIZATION STRATEGIES BASED ON EDA TOOLS

4.1 Standardize the Design Language to Improve Modeling Efficiency

The accuracy and standardization of the design language directly affect the efficiency of the FPGA digital-circuit modeling phase, making it especially critical at the project's outset. Adopting a hardware description language that is structurally clear and semantically explicit can establish a stable logic-model foundation early in system design, providing strong support for subsequent development stages [1]. Consistent syntax, standardized module invocation, and systematic naming help EDA tools recognize each functional module, thereby reducing analysis errors and redundant logic. At the same time, maintaining a reasonable language-level hierarchy during design effectively separates control logic from data paths, facilitating later debugging. In addition, good coding style can improve module reusability, lower the cost of repetitive work, and make the entire design process more efficient and clearer.

Using Verilog as an example, when creating an 8-bit counter module, the designer should adopt a clear structural partitioning, properly arrange input and output ports, and explicitly define the logical relationships of control signals such as clock and reset. At the beginning of the design, establish a unified naming convention—for instance, name all clock signals clk and all reset signals rst_n—so that EDA tools can accurately identify module functions during logic synthesis and place-and-route, reducing compilation errors caused by naming conflicts or ambiguities between modules. In module design, coding style is equally important. Consistent indentation and clear comments improve code readability, facilitating later maintenance and team collaboration. For example, when using an always block to control states, employ a case statement to clarify the transition logic between states and add concise comments after each state to quickly locate the logic flow during simulation. Once the design is complete, use the syntax check or preliminary synthesis features of EDA tools like Quartus or Vivado to immediately detect potential issues such as undefined signals, uninstantiated modules, or missing sensitivity lists, thereby avoiding functional failures at an early stage. When interconnecting modules later, if the interface naming is standardized and the structure is clear from the outset, the difficulty of writing the top-level file is significantly reduced, and interface-matching errors during module invocation are minimized. For instance, when connecting an SPI communication module to a main control module, because both follow the same naming format, the interface call becomes straightforward and clear, eliminating the need to repeatedly check port definitions and effectively shortening the development cycle. Moreover, as project scale grows and the number of logic modules increases, a disciplined language structure also enables EDA tools to quickly analyze the project hierarchy, automatically optimize resource allocation during synthesis and place-and-route, and reduce unnecessary logic duplication. Taking a timing control system composed of multiple counter modules as an example, if each module is developed following the same format, it can be instantiated and reused directly, greatly reducing code volume and improving overall design efficiency.

4.2 Strengthen Timing Control to Ensure Logic Stability

In FPGA digital-circuit development, the precision of timing control directly determines the stability of system logic operation, so it must be treated as a core concern. The timing-analysis features in the EDA tool platform

require accurate design constraints as their foundation; only then can they evaluate the delay of each signal path under the actual operating frequency [2]. To achieve the intended performance, the clock frequency should be set appropriately while maintaining a balance between setup time and hold time—an essential prerequisite for stable synchronous-logic operation. If timing details are overlooked during design, the circuit may suffer logic errors at edge-triggering moments, leading to overall system malfunction. To strengthen timing optimization effectively, designers should plan primary and secondary clock relationships in advance, avoiding resource contention and signal aliasing, thereby stabilizing data transfer among core modules.

Take the design of a pulse-width-modulation controller as an example: this module must precisely adjust the duty cycle at high frequency. During design, the first step is to set the primary clock constraint in an EDA tool such as Vivado—for instance, specifying the system clock as 50MHz and defining the clock period as 20 ns—so that the tool can apply rigorous timing analysis to all paths during synthesis and place-and-route. Next, in logic construction, the designer must partition the combinational logic between the counter and the comparator to avoid overly long paths. Pre-registering part of the logic or splitting it into multi-cycle operations can effectively shorten critical paths and increase timing slack. In one test, if the combinational-logic delay within a 20 ns period reaches 18 ns, the EDA tool will flag a timing violation. At this point, the designer can insert registers along the critical path or use pipelining to distribute the logic operations, reducing the delay of each stage to below 10 ns and successfully meeting timing requirements. Another key strategy is handling multi-clock-domain data exchange. In a system that includes ADC acquisition and DMA transfer, the ADC module operates under the 80MHz sampling clock, while the DMA module uses the 100MHz system clock. Because cross-clock-domain transfers exist, dual-flip-flop synchronizers or asynchronous FIFOs must be added to prevent data metastability or signal loss. The EDA tool can automatically identify cross-domain paths and mark synchronization risks in the timing report, alerting the designer to optimize the structure. Using the waveform-simulation tool ModelSim to analyze the transfer results, one can observe whether the edges of the synchronized signals are stable and verify the reliability of the synchronization scheme. Ultimately, the system can operate stably under the different module clocks, with data transferred accurately, thereby improving overall immunity to interference and logical rigor.

4.3 Streamline Resource Structure and Optimize Device Placement

Rational allocation and scheduling of FPGA internal resources is a key way to boost circuit-design performance, so it must be planned holistically from the very start. When using EDA tools to analyze the architecture, special attention should be paid to eliminating unnecessary logic blocks and control units, thereby preventing routing congestion or sluggish operation caused by redundant resources. At the same time, structural streamlining covers not only the control of gate count but also the judicious arrangement of hardware resources such as LUTs, registers, and memory blocks [3]. Moreover, replacing duplicated units with versatile multi-function modules helps reduce device load and further increases layout compactness. During the actual placement phase, EDA tools map devices according to resource distribution and logical relationships; if the logic structure is chaotic, critical modules may end up scattered, raising interconnect complexity and signal delay. On the other hand, efficient resource planning not only lowers system power consumption but also markedly improves overall chip utilization, making the system run more compactly.

Take a four-channel data acquisition module as an example. In the initial design, each channel was equipped with its own buffer, controller, and memory interface, causing severe duplication of logic blocks. The EDA synthesis report showed that LUT and register utilization was near the upper limit; during placement and routing, unreasonable module distribution led to routing congestion and timing violations. Using the resource-analysis feature of the EDA tool, the designer discovered that several sub-modules were not active simultaneously within the same clock cycle. Consequently, identical functional logic structures were unified into a configurable multiplexer module, employing shared control logic to create time-division access for the different channels. After the change, logic-cell usage dropped by 30 %, routing resources were freed, and system frequency rose by 20 %, markedly improving design performance. In the development of an image-recognition system, to perform sliding-window processing, the designer built a large number of redundant shift-register chains. The EDA report indicated extremely high register usage and highly similar functional logic across multiple modules. During optimization, the sliding window was redesigned as a parameterized module, its structure unified with for-generate statements, while multiple independent serial/parallel register structures were merged into a shared buffer area, eliminating duplicated logic. After the adjustment, the EDA tool relocated the originally scattered logic modules to adjacent locations, greatly shortening wire length, stabilizing signal transmission, and maintaining data-output accuracy in simulation; system power consumption also decreased. Moreover, in the design of a serial-communication controller, the state machine and baud-rate generator were initially placed far apart, and the

EDA placement report flagged a cross-region critical path. To improve layout compactness, the designer employed the area-constraint feature provided by the EDA tool to confine related logic within a specified region, achieving logic concentration and interconnect optimization. In the final design, critical-path delay was reduced by 15 %, effectively eliminating timing risks caused by dispersed placement.

4.4 Refine the Simulation Mechanism to Improve Verification Accuracy

In FPGA digital-circuit development, simulation verification is the key step for ensuring functional correctness; therefore, the design flow should prioritize building a complete verification framework. The simulation platform provided by EDA tools must be highly compatible with the chosen design language so that every signal transition can be tracked accurately and the validity of logical judgments can be enhanced [4]. From a holistic perspective, a well-rounded simulation mechanism should cover functional simulation, timing simulation, boundary testing, and more, so that the system's stability under diverse operating conditions can be examined. At the initial design stage, establishing a unified, standardized testbench and clearly defining the relationships between input and output signals facilitates consistent comparison in later verification. When further optimizing the simulation flow, test phases and verification cadence should be set rationally, advancing layer by layer to detect potential logic defects effectively. Moreover, by means of waveform recording and timing tracing, abnormal data exchanges between modules can be located quickly, thereby reducing the time spent on repeated tests.

Take the design of a UART serial-communication module as an example. The module includes sub-modules such as a receiver, a transmitter, parity checking, and state control. At the beginning of the design, the EDA tool ModelSim is used to build the simulation platform; a top-level testbench is written, clock signals, reset logic, and input stimuli are defined, and observation points are set on the output signals. During functional simulation, a set of standard test vectors is applied, and the output waveforms are checked against expectations. In one test, the receiver exhibited a data-bit shift; analysis revealed that the state-transition decision was delayed. After adjusting the relevant logic and re-simulating, the waveform returned to normal, the received data matched the transmitted data exactly, and functional verification passed. In the subsequent timing-simulation phase, the designer generated a gate-level netlist with the EDA tool and combined it with a delay model for verification. In another SPI master module design, simulation showed that data output lagged behind the clock rising edge, preventing the receiver from correctly recognizing the data. By shifting the data-latch position and outputting the data one clock cycle earlier, the timing alignment was restored, data transmission completed correctly, and the verification result met the specification. To further improve simulation completeness, assertions were added; when a data-write anomaly occurs, the simulation halts automatically and records the error point, allowing rapid problem location. During system integration, to avoid test omissions, the coverage-analysis feature of the EDA tool is used to count how many functional paths have been triggered. For example, when designing a digital-filter module, the initial simulation only verified intermediate input values, leaving boundary cases uncovered. The designer supplemented tests with maximum and minimum inputs; after re-simulation, the filter output remained stable, confirming that the module operates correctly across the entire input range. In addition, waveform-comparison tools can be used to contrast logic changes before and after optimization, ensuring that functionality is not broken after structural modifications and thereby improving verification efficiency.

5. CONCLUSION

In summary, the application of EDA technology in FPGA digital circuit design has become a key means of improving design efficiency, optimizing system performance, and ensuring functional reliability. In the future, as circuit scales continue to expand and application scenarios become increasingly complex, EDA tools will further evolve toward greater intelligence and integration. Designers should continuously master advanced tool capabilities, refine design strategies, and drive digital circuit design toward higher performance, greater precision, and even stronger stability, providing solid technical support for the development of electronic information engineering.

FUNDING:

Provincial Science and Technology Project: No. 171179, titled "Design and Implementation of Photoelectric Switches in Automatic Monitoring Systems"; University Teaching Reform Project: No. 2020-JGJG-27, titled "Exploration and Implementation of Online + Offline Blended Teaching Based on Network Teaching Platforms".

REFERENCES

[1] Peng, Q., Planche, B., Gao, Z., Zheng, M., Choudhuri, A., Chen, T., Chen, C. and Wu, Z., 3D Vision-Language Gaussian Splatting. In The Thirteenth International Conference on Learning Representations.

[2] Guo, Y. (2025). The Optimal Trajectory Control Using Deterministic Artifi cial Intelligence for Robotic Manipulator. Industrial Technology Research, 2(3).

[3] Zhou, Z. (2025). Research on Software Performance Monitoring and Optimization Strategies in Microservices Architecture. Artificial Intelligence Technology Research, 2(9).

[4] Wei, Xiangang, et al. "AI driven intelligent health management systems in telemedicine: An applied research study." Journal of Computer Science and Frontier Technologies 1.2 (2025): 78-86.

[5] We, X., Lin, S., Pruś, K., Zhu, X., Jia, X., & Du, R. (2025). Towards Intelligent Monitoring of Anesthesia Depth by Leveraging Multimodal Physiological Data. International Journal of Advance in Clinical Science Research, 4, 26–37. Retrieved from https://www.h-tsp.com/index.php/ijacsr/article/view/158

[6] Zhang, Wenqing, et al. "Enhancing Logical Reasoning in Large Language Models via Multi-Stage Ensemble Architecture with Adaptive Attention and Decision Voting." Proceedings of the 2024 5th International Conference on Big Data Economy and Information Management. 2024.

[7] Zhang, W., Shih, K., Jin, Y., Chen, Z., Liu, L., & Zhang, Z. (2025, January). Dynamic Cross-Attention and Multi-Level Feature Fusion for Fine-Grained Image Captioning in Advertising. In 2025 5th International Conference on Neural Networks, Information and Communication Engineering (NNICE) (pp. 282-286). IEEE.

[8] Zhang, D., Fu, J., Zheng, J., Deng, Z., & Yang, Z. (2025). Maximizing Scoring Divergence in Automated Essay Assessment with LLaMA-Based Meta-Attention Networks.

[9] Huang, X., Lin, Z., Sun, F., Zhang, W., Tong, K., & Liu, Y. (2025). Enhancing Document-Level Question Answering via Multi-Hop Retrieval-Augmented Generation with LLaMA 3. arXiv preprint arXiv:2506.16037.

[10] Wang, Y., & Bi, X. (2025, January). Hierarchical Adaptive Fine-Tuning Framework for Enhancing Multi-Task Learning in Large-Scale Models. In 2025 5th International Conference on Neural Networks, Information and Communication Engineering (NNICE) (pp. 1582-1586). IEEE.

[11] Liu, C. (2025, January). Optimization of Adaboost cardiac disease prediction and classification based on long and short term memory network. In 5th International Conference on Signal Processing and Machine Learning (CONF SPML 2025) (Vol. 2025, pp. 196-200). IET.

[12] Su, Z., Yang, D., Wang, C., Xiao, Z., & Cai, S. (2025). Structural assessment of family and educational influences on student health behaviours: Insights from a public health perspective. Plos one, 20(9), e0333086.