Advances in Computer, Signals and Systems (2024)
Clausius Scientific Press, Canada

DOI: 10.23977/acss.2024.080604
ISSN 2371-8838 Vol. 8 Num. 6

Research on the Principle and Architecture of Icarus Verilog System

Jianxin Wang[1], Xiangze Chang[1,*], Chaoen Xiao[1], Lei Zhang[1]

[1]Beijing Electronic Science and Technology Institute, Beijing, 100070, China
[*]Corresponding author: changxiangze@gmail.com

Keywords: Open-source EDA tools; Icarus Verilog; System principles; Internal architecture; ZUC cryptographic algorithm

Abstract: As global technological competition intensifies, the challenges in chip design are becoming increasingly complex, highlighting the urgent need for open-source EDA (Electronic Design Automation) platforms. The introduction of open-source EDA tools can lower the barriers to chip design, foster scientific research, and promote talent development. However, issues such as a limited user base and insufficient contributions need to be addressed. This study investigates the 2022 version of Icarus Verilog, providing a detailed introduction to its system principles and analyzing its internal architecture and module composition. Additionally, we validate its preprocessing, compilation, and simulation functionalities by testing the ZUC-128 cryptographic algorithm on the LicheePi 4A, a high-performance RISC-V Linux development board based on the Lichee Module 4A and powered by the TH1520 core. Experimental results indicate that Icarus Verilog offers flexible open-source characteristics and a wide range of applications, reducing R&D costs and providing high utility. This research fills a gap in the domestic study of Icarus Verilog and offers valuable insights for the future development and optimization of open-source EDA tools.

1. Introduction

The increasing complexity of digital circuit design necessitates advanced Hardware Description Language (HDL) simulators and Electronic Design Automation (EDA) tools. These tools aid in design, simulation, synthesis, and layout, ensuring effective chip design verification and manufacturing [1]. Traditional commercial EDA tools are expensive and have restrictive licenses, limiting access and flexibility [2]. Open-source EDA tools offer a solution, reducing barriers and providing greater customization and scalability. However, challenges such as a lack of users and contributors, and limited foundational knowledge persist.

Icarus Verilog is a free, open-source, cross-platform Verilog HDL simulator supporting Verilog 1995, 2001, and IEEE Std 1364-2005 standards. It is cost-effective, lightweight, and integrates well with other tools like Vivado, Quartus, and ModelSim. It is widely used in educational settings due to its comprehensive toolset and robust community support.

Previous research has laid a rich foundation for the application and study of Icarus. Williams et al. (2002) introduced the principles and implementation of Icarus Verilog, evaluating its compatibility

and performance [3]. Wang et al. (2003) researched the parallelization of its compiler [4], while Li et al. (2004) implemented a distributed virtual simulator based on it [5]. Further development includes Shu's FPGA functional simulation platform (2008) [6,7], Liu's parallel distributed simulation environment ODPSim (2009) [8,9], and Wang's Verilog integrated development tool (2018) [10]. Despite these contributions, recent research on the latest versions of Icarus remains scarce. This paper addresses this gap by exploring the system principles, internal architecture, and new features of the latest version, providing updated insights and practical applications for EDA tool development. The study focuses on the 2022 version of Icarus Verilog, validating its preprocessing, compilation, and simulation functionalities through testing the ZUC cryptographic algorithm, thereby filling gaps in domestic studies and offering valuable insights for open-source EDA tool development.

2. Icarus Verilog System Architecture

The Icarus Verilog system architecture comprises compiler components and runtime components (**Figure 1**). The compiler components include the compiler driver, preprocessor, core compiler, and code generator. The compiler driver coordinates these components to complete the compilation process [11]. The core compiler performs syntax analysis and optimization, with the code generator producing the target code. The runtime components, consisting of the vvp simulator and its key elements, support the simulation process and user interaction, ensuring effective result presentation [12].

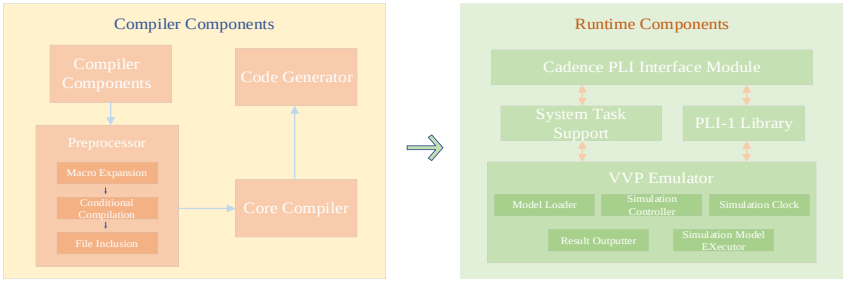

Figure 1: Internal Architecture of Icarus Verilog

The core architecture includes three main components: the preprocessor, the Iverilog compiler, and the vvp simulator:

(1) Preprocessor: Initiates the compilation process by handling macro expansion, conditional compilation, and file inclusion for the input Verilog code. The preprocessed code is a character stream that simplifies and optimizes subsequent compilation and simulation.

(2) Iverilog Compiler: Compiles and optimizes the preprocessed code into vvp assembly code, detailing timing and logical changes of signals. It includes a syntax parser, expander, and code generator. The syntax parser creates an abstract syntax tree, the expander checks for semantic errors, and the code generator produces executable vvp assembly. This stage involves syntax and semantic analysis, ultimately generating a simulation target file (VVP) containing logical circuit netlist information.

(3) vvp Simulator: Loads and executes the compiled vvp assembly files. Its architecture includes a model loader, simulation controller, simulation clock, simulation model executor, and results output. Using an event-driven approach, it simulates signal value changes and logical operations, generating signal values and waveform diagrams per clock cycle to verify circuit designs.

These components form an efficient compilation and simulation workflow. The preprocessor converts source code into intermediate code, the Iverilog compiler optimizes and generates executable simulation files, and the vvp simulator executes these files, producing simulation results and waveform diagrams. This architecture ensures efficient and accurate HDL code processing,

supporting robust digital circuit design and verification.

3. Principles of the Icarus Verilog System

3.1. Operating Principles of Icarus Verilog

The Icarus Verilog system operates based on the Verilog HDL standard, divided into preprocessing, compilation, and simulation stages (**Figure 2**). During preprocessing, the system collects files, performs macro substitutions, and generates intermediate code. Compilation involves syntax parsing, code expansion, and generating executable simulation files. Simulation uses an event-driven approach to execute these files, simulating signal changes and logic operations to produce signal values and waveform diagrams per clock cycle, validating the circuit design [13].

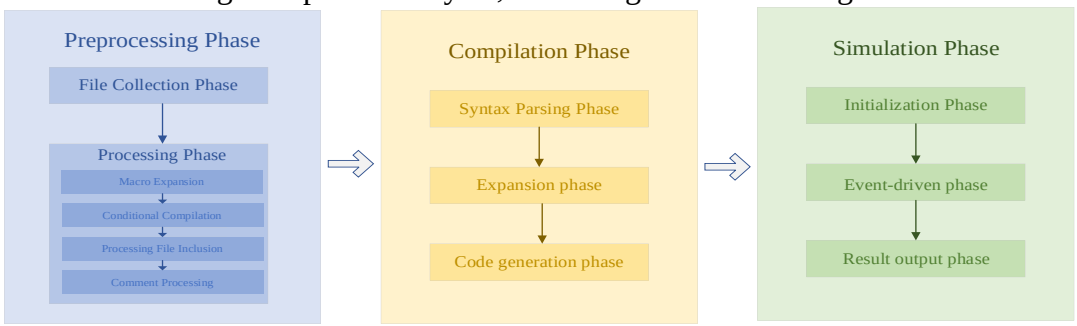

Figure 2: Icarus Verilog System operation principle

3.2. Simulation Principles of the Icarus Verilog

Icarus Verilog enhances efficiency through event-driven simulation, which involves:

(1) Initialization: Sets the simulation time, initializes signal values and module states, and configures the simulation clock frequency and period to control timing behavior.

(2) Event Queue Creation and Management: Creates and maintains an event queue, organizing events by timestamp and operation task to ensure time-sequenced processing. This method efficiently controls and predicts event handling for system behavior and performance evaluation.

(3) Simulation Main Loop: Extracts the earliest event from the queue, updates signal values, calculates output signals, and changes states of modules like flip-flops. New events generated during this process are added back to the queue. Concurrent operations may be required for events with the same timestamp or when updating the simulation clock state, utilizing parallel computing resources to accelerate simulation speed.

(4) Simulation End and Result Recording: Monitors the event queue until the simulation ends or reaches the preset time, recording significant results, including signal changes and module updates. Outputs simulation results and waveform diagrams for analysis and verification of circuit behavior.

The event-driven simulation method's flexibility and efficiency allow accurate simulation of asynchronous and concurrent systems, dynamically adjusting based on event occurrences. However, challenges include event scheduling impacts and higher storage space requirements.

4. Functional Testing and Simulation

The ZUC-128 algorithm, designed by China, is used in LTE wireless communication systems for data encryption and integrity protection [14,15]. This study uses the Icarus Verilog system with the TH1520 core on the LicheePi 4A RISC-V Linux development board to simulate and test the ZUC-128 algorithm, verifying its functionality and correctness.

4.1. Preprocessing

The design file "zuc.v" and testbench file "zuc_tb.v" are placed in the same directory. The testbench includes the clock generator, enable signals, and VCD file settings. Using the command iverilog -E zuc.v zuc_tb.v, preprocessing generates the intermediate file "a.out" (**Figure 3**).

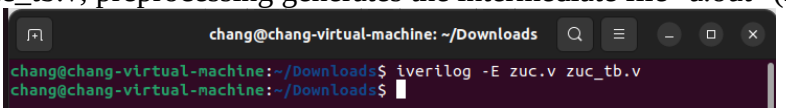

Figure 3: Reiwa ichiban uncle

4.2. Compilation

The command iverilog -o _test zuc.v zuc_tb.v compiles the files, producing an executable file "_test". Verify the presence of required files using ls. A successful compilation results in the _test file appearing in the directory (**Figure 4**).

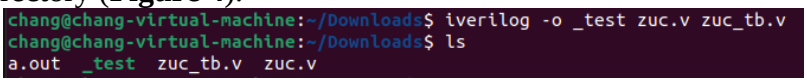

Figure 4: Compilation process

4.3. Simulation

In the directory with _test and zuc.vcd, the command vvp -n _test generates the VCD file (**Figure 5**). Use gtkwave zuc.vcd to view the simulation waveform.

(1) vvp -n _test: Runs the executable file _test in non-interactive mode.

(2) gtkwave zuc.vcd: Opens the VCD file zuc.vcd using the gtkwave tool.

Figure 5: Simulation process

4.4. Simulation Results Analysis

The simulation results (**Figure 6**) show a clock (clk) set to 100 MHz. The reset signal (reset) starts at 1'b1, initializing the module. The address (addr) starts at 8'b0 and increments every 20 clock cycles. The write data (wdata) is set to 32'b0, indicating the initial state of the encryption process. The write enable signal (wr) controls data writing. The initialization ready signal (init_rdy) goes high after initialization. The work done signal (work_done) indicates task completion. Each clock cycle generates an output key word (z_out), producing expected results like Z1=27BEDE74 and Z2=018082DA, confirming the accuracy of Icarus Verilog in simulating the ZUC-128 algorithm.

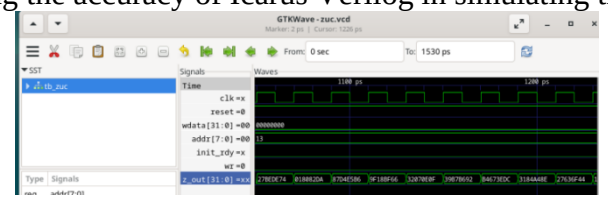

Figure 6: Simulation results

5. Conclusions

This study deeply examined Icarus Verilog, focusing on its principles, architecture, and applications in digital circuit design and verification, particularly the 2022 version. It explored the system principles, including simulation mechanisms like event types and event queue management, and provided a detailed analysis of its hierarchical and modular architecture. Functionality testing demonstrated the reliability of Icarus Verilog in compilation, simulation, and Verilog-to-VHDL conversion. This research underscores the importance and potential of Icarus Verilog in digital circuit design and provides a solid reference for further exploration and optimization. Future work may delve into system principles, implementation details, and cross-platform performance validation.

Acknowledgments

We acknowledge the administrative and technical support provided by our institution.

Funding: This research was funded by the College-level Characteristic Teaching Material Project, grant number 20220119Z0221; the College Teaching Incubation Project, grant number 20220120Z0220; the Ministry of Education Industry-University Cooperation Collaborative Education Project, grant number 20220163H0211; the Central Universities Basic Scientific Research Fund, grant numbers 3282024009, 20230051Z0114, 20230050Z0114; the Beijing Higher Education "Undergraduate Teaching Reform and Innovation Project", grant numbers 20220121Z0208, 202110018002; and the College Discipline Construction Project, grant numbers 20230007Z0452, 20230010Z0452.

References

[1] Liu Weiping, Wang Zongyuan. Development status and development trend of EDA industry and IP nuclear industry [J]. Foresight Technology, 2022, 1 (03): 90-100.
[2] Liu Xing, Xia Fan. Analysis on the development prospect of digital integrated circuit EDA software industry [J]. China Informatization, 2021 (09): 117-120.
[3] Williams S, Baxter M. Icarus Verilog: open-source Verilog more than a year later[J]. Linux Journal, 2002, (99): 3.
[4] C. Tropper, H. Huang and L. Li, "DVS: An Object-Oriented Framework for Distributed Verilog Simulation," in 2012 ACM/IEEE/SCS 26th Workshop on Principles of Advanced and Distributed Simulation, San Diego, California, 2003:173.
[5] C. Tropper and J. Wang, "Nicarus: A Distributed Verilog Compiler," in Workshops on Mobile and Wireless Networking / High Performance Scientific, Engineering Computing/Network Design and Architecture/Optical Networks Control and Management/Ad Hoc and Sensor Networks/Compil, Montreal, QC, Canada, 2004:514-519.
[6] Wang Shi. Construction of functional simulation platform and static timing analysis based on FPGA chip [D]. Xidian University, 2008.
[7] Weng Yanling. Study of RTL to gate-level design [D]. Zhejiang University, 2008.
[8] Liu Chao. Research and Implementation of Parallel Distribution Simulation Environment based on open source simulator Icarus Verilog [D]. National University of Defense Technology, 2009.
[9] Liu Chao, Dou Qiang, and Zheng Yi. Simulation technical analysis of the open-source simulator "Icarus Verilog" [J]. Science and Technology Information, 2009 (04): 169-170.
[10] Wang Jianxin, Xiao Chao'en, Zhang Lei, Sun Meng, Han Ying, Xu Hongke. Design of FPGA simulation tool for experimental teaching based on Domestic hardware and software systems [J]. Laboratory Research and Exploration 2022, 41 (11): 124-128 + 180.
[11] Liu Sanxian. Analysis and design of Gaussian word method analyzer and grammar analyzer based on ANTLR [D]. Lanzhou University, 2009.
[12] Lee Y C, Chan Y K, Koo V C. Design and Implementation of Fpga-Based Fft Co-Processor Using Verilog Hardware Description Language [J]. Progress in Electromagnetics Research B, 2021. DOI:10.2528/PIERB20122806.
[13] Kwon Y S, Lee J G, Kyung C M. Performance-driven event-based design mapping in multi-FPGA simulation accelerator[J]. ISOCC, 2004: 328-331.
[14] Feng Xiutao. The sequence graphic algorithm [J]. Information Security Research, 2016,2 (11): 1028-1041.
[15] Ding Qun, Yu Longfei, Li Xiaoyou, et al. An IP core construction method of ZUC encryption system based on FPGA. CN202011089725.4 [2023-07-22].