LOGICLAB: A COMPREHENSIVE WEB-BASED FRAMEWORK FOR DIGITAL LOGIC CIRCUIT VIRTUALIZATION AND REAL-TIME SIMULATION

Ergashev Adizbek Kamol ugli
Muhammad al-Khwarizmi Tashkent University of Information Technologies, PhD
E-mail: adizbek@tuit.uz
Phone: +99893-656-86-89
https://doi.org/10.5281/zenodo.17838254

Thesis Overview

This thesis presents LogicLab, a web-based framework for virtualizing and simulating digital logic circuits in real time, spanning basic combinational logic through complex sequential circuits and finite state machines. The work targets limitations in existing digital logic education tools by delivering a zero-installation, multi-language platform that leverages modern reactive programming for immediate visual feedback and interactive learning, publicly available at logiclab.uz.

Keywords: Digital Logic Simulation, Web-Based Virtualization, Sequential Circuits, Reactive Programming, Educational Technology, State Machines, Boolean Algebra, Circuit Visualization, Robotic Control Systems

Motivation and Problem Statement

Digital logic design underpins modern computing, embedded systems, and robotic control, yet traditional learning relies on physical breadboards or desktop simulators such as Logisim, Proteus, and ModelSim. These approaches impose accessibility constraints through installation requirements, operating system compatibility, and license costs, while physical circuits demand laboratory access and are unsuitable for remote learning. Language support is frequently limited to English, creating cognitive overhead for non-native speakers. Most web tools remain weak in modeling feedback loops, clocked logic, and state machines, and few provide progressive templates or visual signal-flow feedback. The challenge is to combine real-time interactive simulation, visual programming, multilingual access, and extensibility for complex systems within the constraints of a browser environment.

Research Objectives and Contributions

The research formalizes digital circuits as executable graphs that support both acyclic and cyclic structures, builds a reactive simulation engine for real-time signal propagation, extends the framework to handle feedback via clock-driven evaluation, and ships an extensible component architecture that already includes

thirteen gates, flip-flops, and state machine support. A template library with nineteen categorized circuits enables progressive learning, while feedback loop detection and temporal separation prevent circular dependency failures. Educational effectiveness is evaluated through controlled user studies and the applicability to robotic control system virtualization is demonstrated.

Key contributions include a Vue 3 reactivity-based architecture for automatic signal propagation, formal graph representations with Boolean algebra extended to temporal models, an (O(|V|+|E|)) Tarjan strongly connected components approach to detect feedback combined with clock-driven updates to resolve circular dependencies, sub-millisecond response times up to fifty components with event-driven optimizations reducing computation by 73%, and state transition accuracy of 99.7% against ModelSim for sequential circuits. The framework introduces multi-language, template-based learning for Central Asian learners and formalizes sequential circuits as finite automata to enable robotic control design.

Theoretical Foundation

Mathematical Models

Combinational circuits are represented as the tuple

$$C=(V,E,F,S)$$

where

$$V=\{v_1,v_2,\ldots,v_n\}, \quad E\subseteq V\times V, \quad F:V\to\{f_1,f_2,\ldots,f_m\}, \quad S:V\to\{0,1\}.$$

Boolean algebra operations such as AND, OR, XOR, and NOT are implemented with constant-time JavaScript bitwise operators to maintain (O(1)) execution per gate.

Sequential circuits extend the model to

$$SC=(V,E,F,S,C,T)$$

where cycles may exist, (C V) identifies clocked components, and (T:) represents discrete steps. Logical cycles are broken through temporal separation, replacing erroneous direct dependencies

$$Q=\text{NOR}(R,\bar{Q})$$

with time-indexed updates

$$Q(t+1)=\text{NOR}\,(R(t),\bar{Q}(t)).$$

Reactive Programming Integration

LogicLab relies on Vue 3 Composition API with computed properties to track dependencies automatically. Combinational outputs update reactively from their inputs, while sequential logic introduces explicit state variables and clock-driven watchers that sidestep Vue's circular dependency detection, enabling SR latches,

D flip-flops, and other feedback-based components to simulate correctly in the browser.

System Architecture and Implementation

The framework is organized into presentation, application, simulation, and data layers. The presentation layer uses Vue 3 with a drag-and-drop component palette and Vue Flow canvas visualization. Application logic relies on Pinia stores for state management, circuit templates, and multilingual support. The simulation engine provides Boolean operations, reactive signal propagation, and a component registry. Data handling covers JSON export and import, localStorage persistence, and the template library.

Key Technical Achievements

Component architecture isolates each gate as a Vue component with a shared interface, maximizing modularity and extensibility. The useGate composable leverages Vue reactivity to propagate signals without manual graph traversal, while useSequentialGate separates next-state evaluation from clocked commits to avoid circular updates. A two-phase clock synchronization model evaluates combinational logic, commits sequential state on clock edges, and re-propagates changes to ensure temporal consistency. Event-driven optimization monitors changed nodes and propagates only through affected subgraphs, delivering a 3.9x speedup and maintaining sub-millisecond performance for educational circuits.

Experimental Validation

Benchmarks show 0.12 ms update time for five-component circuits, 0.45 ms for twenty components, 1.23 ms for fifty components, and a stable 60 FPS for complex 120-component layouts, all while sustaining 99.7% state-transition accuracy against ModelSim. Event-driven scheduling reduces computational overhead by 73% and preserves linear scaling with circuit complexity. User studies with fifty-five computer engineering students report post-test gains of 10–13% over a Logisim control group, 23% faster circuit completion, 30% fewer wiring errors, 23% better understanding of clock behavior, 21% improved state machine accuracy, and a 4.7/5 satisfaction rating that highlights animated signal flow, templates, and native-language support.

Applications to Robotic Systems

The sequential circuit framework supports robotic control virtualization by modeling line-following controllers with four-state finite state machines, abstracting sensors and actuators through analog inputs, PWM motor drivers, and servo components, and enabling hardware-in-the-loop evaluation via WebSerial connections to microcontrollers such as Arduino or ESP32.

Impact and Future Directions

Educational impact arises from zero-installation access, multilingual interfaces critical for Central Asia, visual feedback that clarifies timing and state transitions, and template-based scaffolding. Research impact includes transferable models for reactive systems with feedback, performance baselines that validate browser-based real-time simulation, and architectural extensibility toward control systems, neural networks, and workflow automation. Planned extensions include completing the component library with T flip-flops and shift registers, adding timing diagram visualization and critical path analysis, enabling HDL export with Verilog and VHDL, incorporating hardware-in-the-loop via WebSerial, and exploring formal verification with temporal logic. Longer-term goals cover CPU microarchitecture simulation, embedded systems IDE capabilities, cyber-physical system modeling, ROS integration, and ultimately a unified platform spanning education to deployment on FPGA or microcontroller targets for robotic control.

Conclusion

LogicLab delivers a comprehensive framework for digital logic virtualization that combines formal mathematical models, reactive programming, and pedagogical design. Demonstrated learning gains of 10–13%, sub-millisecond simulation performance, and 99.7% accuracy validate the approach, while the pathway to robotic control systems positions the platform as a foundation for broader research in mathematical-software methods for robotic system virtualization.

Literature:

1.M. Morris Mano and Michael D. Ciletti, Digital Design: With an Introduction to the Verilog HDL, VHDL, and SystemVerilog, 6th ed., Pearson, 2023.
2.Charles H. Roth Jr. and Larry Kinney, Fundamentals of Logic Design, 8th ed., Cengage, 2019.
3.David Money Harris and Sarah L. Harris, Digital Design and Computer Architecture, 3rd ed., Morgan Kaufmann, 2021.
4.Randy H. Katz and Gaetano Borriello, Contemporary Logic Design, 3rd ed., Pearson, 2014.
5.Stephen Brown and Zvonko Vranesic, Fundamentals of Digital Logic with VHDL Design, 4th ed., McGraw-Hill, 2019.
6.A.Bainomugisha, A. Carreton, T. Van Cutsem, S. Mostinckx, and W. De Meuter, A Survey on Reactive Programming, ACM Computing Surveys, 45(4):52, 2013.
7.Robert Tarjan, Depth-First Search and Linear Graph Algorithms, SIAM Journal on Computing, 1(2):146–160, 1972.

8.Mentor Graphics, ModelSim Advanced Simulation User's Manual, Siemens EDA, 2024.
9.IEEE Std 1364-2005, IEEE Standard for Verilog Hardware Description Language, IEEE, 2005.
10.IEEE Std 1076-2019, IEEE Standard VHDL Language Reference Manual, IEEE, 2019.
11.Frank Vahid and Roman Lysecky, Digital Design: An Embedded Systems Approach Using Verilog, 2nd ed., Elsevier, 2016.