MCHPRS on FPGA

High-Performance Simulation of Virtual Logic Systems with Real-Time Interactive Control via FPGA-Based Hardware Acceleration

Paul C. Dötsch

Exposé for a Bachelor's thesis

Computer Science

Heinrich Heine University Düsseldorf, Germany

October 13, 2025

- Supervisor: John Witulski
- Zweitgutachter: Jens Bendisposto
- Desired starting date: 31.10.2025

1 Scientific Background and Motivation

Minecraft's redstone system provides digital logic components analogous to real electronic circuits: redstone dust acts as wires, repeaters function as buffers with programmable delays, and redstone torches implement NOT gates. These components enable construction of complex digital systems including ALUs, memory, and complete CPUs, making redstone Turing-complete for computation.

MCHPRS (Minecraft High-Performance Redstone Server) is a specialized server designed exclusively for computational redstone circuits. Unlike Minecraft's inefficient block-by-block simulation that operates at 20 Hz, MCHPRS transforms redstone circuits into directed weighted graphs and applies compiler-inspired optimizations through its ”Redpiler” system, achieving simulation rates of 100 kHz to 2 MHz.

Despite these improvements, MCHPRS users remain severely constrained by performance limitations that fundamentally restrict the complexity and functionality of achievable builds. Simulation times ranging from seconds to minutes, or even hours for complex circuits, create a significant barrier to iterative development and experimentation. This performance bottleneck forces builders to prioritize computational efficiency over functionality, leading to design compromises that limit innovation.

The single-threaded nature of current simulation particularly impacts advanced architectures: pipelined CPU designs, while theoretically superior, actually perform worse than simpler non-pipelined implementations due to increased total computation overhead. This counter-intuitive reality constrains the redstone community to suboptimal design patterns that prioritize minimal computation over architectural sophistication.

FPGA-based acceleration could fundamentally transform this landscape. The inherent parallelism of FPGAs would make previously impractical designs suddenly viable, allowing pipelined architectures to achieve their theoretical performance advantages. This shift would align Minecraft redstone circuits more closely with real-world digital systems, where design constraints focus on area, latency, throughput, and algorithmic efficiency rather than total computational burden. Such capabilities could unlock entirely new categories of complex builds and foster innovation in digital design education and experimentation.

2 Problem Statement

The thesis will aim to develop an FPGA-based backend for MCHPRS targeting 10-100 MHz simulation rates, representing a multiple orders of magnitude improvement over current software-based simulation limits. The primary goal is maximizing simulation throughput by compiling Redpiler's optimized graph representations to synthesizable HDL.

The research addresses three key questions:

- How can existing Redpiler graph representations be compiled to hardware description language for FPGA implementation?
- How can real-time interaction capabilities (debugging, simulation inputs, state observation) be maintained for FPGA-based logic simulation?
- What additional redstone feature limitations are required for efficient FPGA mapping within resource constraints?

Secondary challenges include implementing controlled stepping mechanisms for debugging and determining acceptable limitations beyond MCHPRS's existing constraints to achieve practical FPGA implementation.

3 Related Work

This work builds upon MCHPRS [3], an open-source project that has demonstrated significant performance improvements through software optimization. The Redpiler system's graph-based representation provides a foundation for hardware compilation.

Logisim Evolution [2], a graphical circuit design and simulation tool, demonstrates the feasibility of compiling graph-based logic representations to hardware. This tool generates VHDL from component-based designs, providing a reference for practical logic-to-hardware compilation approaches that could be adapted for MCHPRS's redstone circuits.

FPGA acceleration of CPU-based simulations has demonstrated significant speedups across various domains, particularly in areas requiring high parallelism, with FPGA systems in certain cases achieving up to 346× speedup compared to even GPU-based implementations [1, p. 14, Table 1].

An existing hobby project, MCHPRS-RoC [4], has explored redstone-to-hardware compilation over the past 8 months, targeting Cyclone V FPGAs using Intel's Quartus toolchain. While this project demonstrates the feasibility of hardware-based redstone simulation, it remains in early development stages and imposes significant fundamental constraints beyond MCHPRS's existing limitations. This thesis will take a different approach by targeting Xilinx FPGAs via the Vivado toolchain while preserving most of MCHPRS's current feature set and applying rigorous scientific evaluation methodologies to this domain.

4 Methodology

The implementation will roughly follow this approach:

HDL Compiler Development: Transform optimized Redpiler graphs into synthesizable Verilog/VHDL, mapping redstone components to corresponding hardware logic blocks.

Component Library Implementation: Develop hardware models for redstone components ensuring proper timing behavior and signal propagation characteristics that match Minecraft / MCHPRS semantics.

Host-FPGA Interface: Create communication protocols between host software and FPGA hardware for circuit control, input stimulation, and output monitoring. Embed debugging components into HDL generation, adding stepping controls and state observation points.

Synthesis Integration: Integrate synthesis workflows and FPGA programming toolchains into MCHPRS, enabling automated compilation from redstone circuits to programmed hardware.

Evaluation: Compare simulation performance, resource utilization, and behavioral compatibility against existing software simulation using representative redstone circuits of varying complexity.

The Synthesis Integration step will likely need to be done incrementally during the first three steps.

5 Resources

The thesis will use a Basys 3 FPGA development board (32k logic cells) provided by the supervisor, along with the Xilinx Vivado toolchain for HDL synthesis. While this constrains the maximum circuit complexity compared to larger FPGAs, it provides sufficient resources for proof-of-concept implementation and evaluation of the hardware acceleration approach.

6 Obstacles and Difficulties

Key challenges include:

- Mapping redstone timing semantics to synchronous hardware while preserving correctness
- FPGA resource limitations constraining implementable circuit size and complexity
- Real-time bidirectional communication between host software and FPGA hardware
- Limited prior experience with FPGA development tools and HDL synthesis workflows

7 Rough Time Schedule

Month 1: Literature review, FPGA toolchain setup, and basic HDL generation for simple redstone components.

Month 2: Implementation of HDL compiler for complete redstone component set and timing model development.

Month 3: FPGA backend integration with MCHPRS, stepping system implementation, host-FPGA communication protocols, comprehensive testing and evaluation.

8 Supervision

Weekly 30-minute meetings with supervisor to discuss progress, technical challenges, and design decisions. Meetings to be scheduled and cancelled with 24-hour advance notice.

The supervisor will attempt to read and grade the thesis within the winter semester 2025/26, though this cannot be guaranteed.

9 Literature

Beyond the references cited in related work, the thesis will draw upon:

- Hardware description languages (Verilog, VHDL) and synthesis documentation
- Xilinx Vivado toolchain documentation
- MCHPRS source code and Redstone mechanics documentation
- Compiler design principles (e.g., ”Compilers: Principles, Techniques, and Tools”)

References

[1] Zhihong You et al. *Acceleration for Deep Reinforcement Learning using Parallel and Distributed Computing: A Survey.* 2024. arXiv: `2411.05614 [cs.DC]`. URL: `https://arxiv.org/abs/2411.05614`.

[2] Carl Burch et al. *Logisim-evolution.* Version 3.9.0. Open-source digital circuit simulator with VHDL code generation. 2024. URL: `https://github.com/logisim-evolution/logisim-evolution`.

[3] StackDoubleFlow and MCHPR Team. *MCHPRS: Minecraft High-Performance Redstone Server.* Version 0.5.1. 2025. URL: `https://github.com/MCHPR/MCHPRS`.

[4] Zachary Zollers. *MCHPRS-RoC: Redstone on Chip.* Redstone compiler targeting Cyclone V FPGAs with Quartus toolchain. 2025. URL: `https://github.com/zwzollers/MCHPRS-RoC`.