GSoC 2025 Proposal - CircuitVerse

Project Name: Open-Hardware Component Library
Duration: 90 hours
Difficulty: Hard
Technologies: Ruby on Rails, JavaScript, Canvas API, VueJS, Verilog
Mentors: Smriti Garg, Aman Asrani, Philip Abbey(self-mentor)

Circuitverse offers a wide range of components, but there is a need to expand its library to include more advanced and specialized components such as Comparators, Shift Registers, SRAM, DRAM, Ring Counters, Control Units, Sensors, etc. These components allow the users to design advanced circuits like processors. This project aims to design, implement, and integrate these components into CircuitVerse, enhancing its functionality and making it a more comprehensive tool for digital logic design and education. Several components, such as the splitter, ALU subtraction, and flip-flops, have implemented their logic or their Verilog module incorrectly, the project aims to fix them. The project also aims to implement serial hardware integration capabilities using protocols like I2C and UART to enable serial device connectivity using SerialPort Library and facilitate interaction with physical hardware through the Circuitverse Desktop application.

Personal Details:

Name: Vivek Kumar
Email: vk092kumar@gmail.com
GitHub: https://github.com/092vk
LinkedIn: https://www.linkedin.com/in/vk092/
Phone: 9596133638
Country: India

Portfolio Website:Portfolio Website
Resume/CV:Resume

About Yourself:

1. Please describe yourself, including your development background and specific expertise.

My name is Vivek Kumar Ray, I am a 3rd-year B.Tech student in the branch of Electronics and Communication. I have extensive knowledge of HDLs like Verilog, experience with Machine Learning, Web development, software development, and electronics. I have worked on several EDA tools similar to CircuitVerse and have contributed to open-source EDA tools and software in the past. I have contributed to eSim which is an EDA desktop-based analog circuit simulator. Also, I have done research work in the fields of Sensor Fusion, Battery life prediction, Software-defined radio, and other fields that combine the knowledge of both software and hardware. I have also worked with IOT, Cloud, and Desktop applications using PYQT and done software packaging for various LINUX distributions and versions. I have knowledge of a range of tools and technologies which include JavaScript, TypeScript, Node, Ruby on Rails, Vue, React, CSS, Bootstrap, Tailwind, HTML, Python, C, C++, AWS, SQL, MongoDB, Postgres, and other software development languages, libraries, framework, and tools.

2. Why are you interested in the CircuitVerse project(s) you stated above?

I have been using CircuitVerse for the past 3 years for digital circuit design and have learned digital design using it, which inspired me to choose CircuitVerse as my GSOC project. CircuitVerse is a perfect blend of electronics and computer science coming together to allow students, teachers, and hobbyists to design digital circuits and learn from them. I have worked in this field extensively, both on EDA tools and hardware-software applications. Together with my experience in Digital circuit design, Circuitverse becomes a perfect choice for me that aligns with my skill sets and passion hence the choice. I have contributed to CircuitVerse extensively which also makes it an obvious choice.

3. Have you participated in an open-source project before?

Yes, I have worked on open-source tools before, namely I have worked on eSim which is an EDA tool similar to CircuitVerse funded by FOSSEE but while eSim focuses on analog circuits CircuitVerse deals with digital circuits. My work at eSim involved:

1. Removing bugs from the eSim 2.4 Windows installer.
2. Updating the eSim version.
3. Building a toolchain for eSim to manage its resources.
4. Enabling dual plotting using Matplotlib and NgSpice
5. Repackaging of eSim for LINUX using FlatPack.

I have also participated in Open-Source hackathons like Hacktoberfest and Google Developers Club and contributed extensively to them.

Commitment

1. No, I am not planning any vacations during the GSoC 2025 period (June 2 - September 8, 2025, or June 2 - November 17, 2025, for extended projects).
2. I will be taking one remote class of ML during the GSOC period.
3. No, I don't have any employment during the GSOC period.
4. I am applying for a small (90 hours) project size.
5. I am planning to work 16-20 hours per week on the project. I am flexible with my work hours but I prefer to work from 9:00 am to 2:00 pm.
6. Currently, I don't think I need an extended timeline. However, hardware integration with the CircuitVerse Desktop application can extend the timeline from 90 hours to 175 hours. More details are given in the end.

Contributions So Far

- PRs merged
  - Removed the conf files for GitPod and its refrence
  - Fixed the Assignment UI
  - Fixed the Testbench
  - Fixed security vulnerability
  - Fixed the timming diagram
  - Removed third-party services duplicate information
  - Added testbench description in feature section
  - Added Docs for Verilog
- PRs unmerged
  - Improved Teachers section and added a new UI teachers_component
  - Added warning for special condition in FlipFlops

    - Changed the SETUP.md description to setup Yosys serer in local
    - Fixed the Language Filter
    - Replaced offset-based pagination with cursor based pagination
    - SR FF Verilog Module
- Issues and bugs found
    - Verilog Testbench
    - Verilog module undefined for JK FF, SR FF, T-latch etc
    - Flip Flop invalid conditions not handled
    - Testbench object not detected issue
    - Timing Diagram overlapping
    - Mobile view for testbench
    - Inconsistent style for testbench window
- Documentation contributions
    - Added Docs for Verilog
- Other contributions to CircuitVerse
    - Updated Wiki notes regarding GitPod and removed its reference
    - Updated Wiki notes for Verilog
- Prototypes Built
    - Real-time communication and shared editing
    - Finite State Machine Simulator
    - Comparator Circuit Element

Proposal

Overview: Goals and tools

1. The project aims to expand the components library of CircuitVerse by adding new components such as **Comparators, Shift Registers, SRAM, DRAM, Ring Counters, Control Units, Sensors**, etc.
2. Fix the incorrect logic implementation of circuit elements and do their versioning.
3. The project aims to introduce hardware integration capabilities within the CircuitVerse desktop application giving it capabilities to serialy communicate with hardware.
4. The project execution involves extensive use of Canvas API, JavaScript, BootStrap CSS, Ruby on Rails, VueJS, Verilog, and SVG graphic designing.
5. For serial communication use of **UART protocol** will be enabled using the **SerialPort Library.**

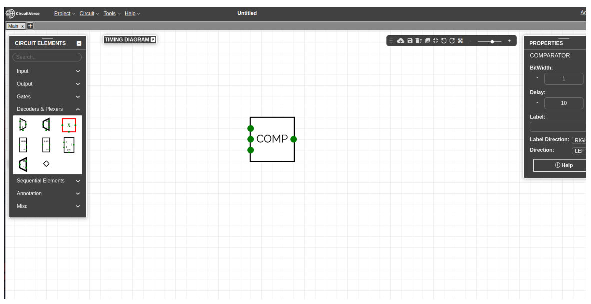
Detailed Description

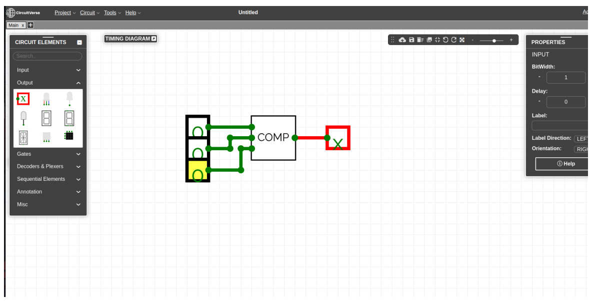
Why ?

Expanding the components library allows the users of CircuitVerse to build complex circuits like Processors using ALU, CU, comparators, shift registers, etc. Due to the expanded library users don't have to build everything from scratch using gates but simply use the components from the library and focus their energy on the process itself. This will allow the users of CircuitVerse to implement things like code generators, processors, and other advanced digital circuits. The project aims to correct the wrong implementation of certain components, which either have incorrect logic or Verilog implementation and do versioning of components to avoid conflicts, which was introduced in an earlier versioning project by Aryan Dwivedi.

In phase 2 the project will focus on allowing Serial communication of hardware with CircuitVerse desktop application using protocols like UART which allows users to write and upload code in development boards like Arduino. This allows the users to build practical digital systems on real hardware and learn from it.

Implementation plan:
1. Component Library





[Prototype implementation of Comparator]()

The implementation involves writing logic for each component in the **simulator/src/modules** folder. This file includes the element class which contains **resolve( )** method, which is responsible for the circuit element logic, **customDraw( )** method which is responsible for drawing the element on the simulator using canvas api, **customSave( )** method to save the data and **moduleVerilog( )** and **generateVerilog( )** which is responsible for generating the verilog code for the circuit element.

Other files which will be affected due to the addition of new components include **moduleSetup.js, metadata.json, verilogClasses.js**. The implementation will also involve designing custom SVG icons for each new circuit element to be included into the components library. Detailed documentation will be added for each circuit element and testing will be done.

Following components will be added:
1. Comparators
2. Shift Registers
3. SRAM
4. DRAM
5. Ring Counters
6. Control Units
7. Sensors

Note: Implementation of Comparator element is given above.

Implementation of the Shift-Register consists of implementing its various versions:

1. Serial In – Serial Out (SISO)
2. Serial In – Parallel Out (SIPO)
3. Parallel In – Serial Out (PISO)
4. Parallel In – Parallel Out (PIPO)
5. Bidirectional Shift Register
6. Universal Shift Register

[All types of Shift Register implemented in current Simulator]()

1. The solution involves creating a single shift-register to represent all the simulators
2. This will be done by how ALU is implemented, by using 2 selection lines, which together can form 4 selection lines options for various Shift registers
3. Depending on what inputs are given to the selection lines, the type of shift register will be selected.
4. Special type of input and output elements needs to be constructed to give parallel input and output.
5. Shift-Register Prototype Implementation

Handling of Edge cases in Circuits:

1. Edge cases for each circuit will be handled as in the case of Shift-Register and Flip-Flops
2. Like **Pre and Reset** in Flip flops all the sequential circuits will have async pins which will handle the cases of preset and reset.
3. Other kind of edge cases like conflicts will also be handles like in case of flip-flops when both reset and pre are 1.

2. Fixing of Circuit element logic

Several circuit elements in the existing library suffer from either flawed or partially implemented logic, resulting in unreliable simulations. I will remove the incomplete circuit element logic and correct their implementation by doing versioning of circuit elements. By doing versioning no present circuit implementation will be affected and any change to circuit elements will only be reflected on that version safeguarding against any conflict to already implemented circuit logic. The versioning system is already developed by Aryan Diwedi which will be used.
The elements which need corrections include:

1. Flip Flops preset and reset conditions correction
2. Flip Flops invalid cases handling
3. ALU subtraction carry correction
4. Splitter element bug correction
5. Other minor corrections in circuit elements

3. Serial communication with Hardware using Circuitverse Desktop application

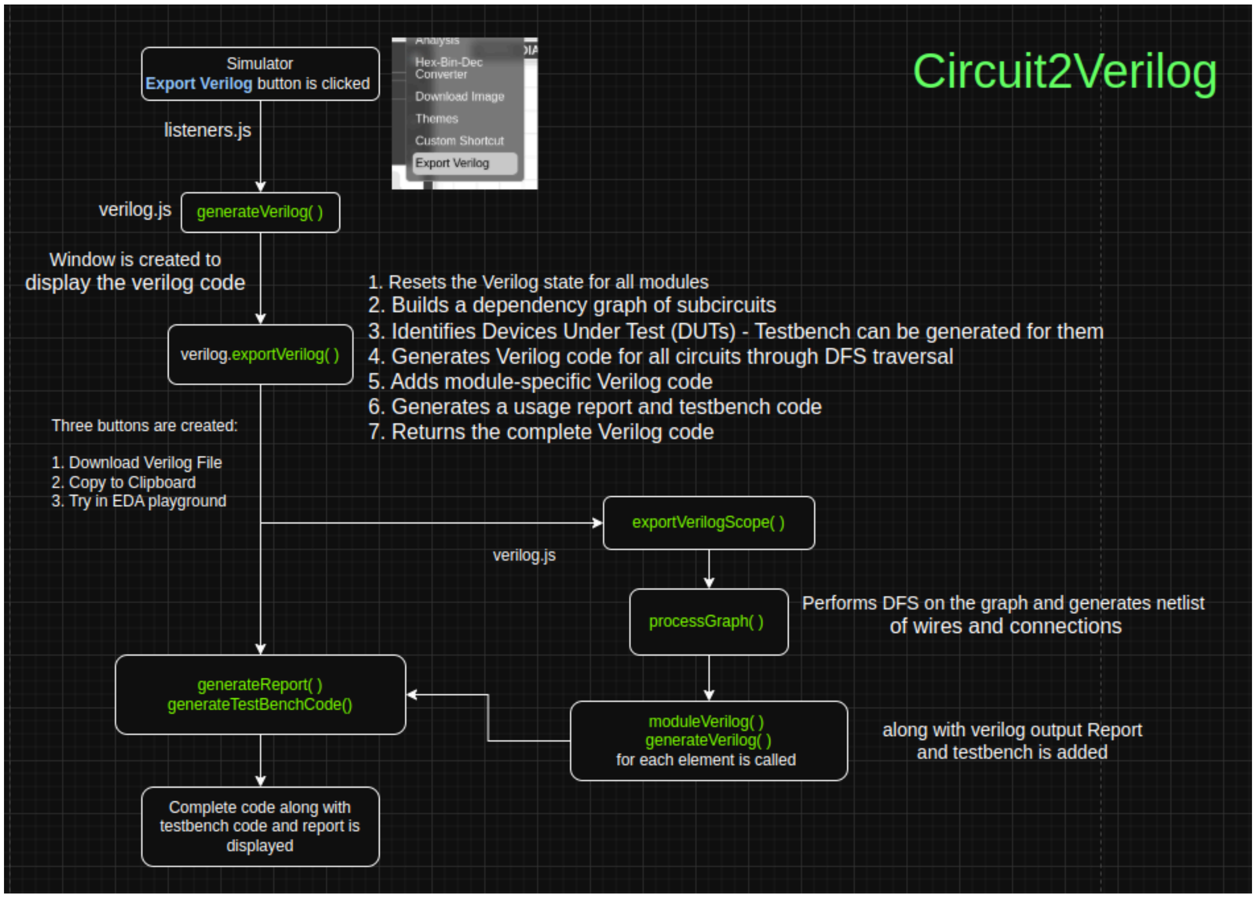
Hardware integration and communication capabilities will be added to the circuitverse desktop application. For this purpose, **UART, SPI or I2C protocols** will be used depending upon the hardware. These protocols will be applied either as a microservice or implemented in the same CV backend server using libraries like **SerialPort Library**, i2c-bus Library, and spi-device Library.

1. **Interface:** The interface in the Circuitverse Desktop application will be created to handle serial communication with the device. The interface will allow the user to change the baud rate, data bits, parity, and stop bits.
2. Backend Server logic to configure the **Protocol library** and handle its implementation, and synchronize it with the current background, ensuring the backend does not get affected due to the implementation of the protocols.
3. An interface and logic for enabling the user to write code and convert the written code into **bitstreams using libraries like avr-gcc and avrdude**.

4. Documentation: Detailed documentation will be added on how the user can use Circuitverse to connect hardware and **upload the bitstream to Arduino** and other hardware.

4. Verilog Module for new elements to generate Verilog code

1. Verilog code each element that is implemented will be written in **moduleVerilog( )**
2. Along with the **generateVerilog( )** function to generate the correct Verilog code for each element.
3. Below is flowchart of how the circuit to Verilog feature works.



Circuit2Verilog
Simulator
Export Verilog button is clicked
Analysis
Hex-Bin-Dec Converter
Download Image
Themes
Custom Shortcut
Export Verilog
listeners.js
verilog.js
generateVerilog( )
Window is created to display the verilog code
verilog.exportVerilog( )
1. Resets the Verilog state for all modules
2. Builds a dependency graph of subcircuits
3. Identifies Devices Under Test (DUTs) - Testbench can be generated for them
4. Generates Verilog code for all circuits through DFS traversal
5. Adds module-specific Verilog code
6. Generates a usage report and testbench code
7. Returns the complete Verilog code
Three buttons are created:
1. Download Verilog File
2. Copy to Clipboard
3. Try in EDA playground
exportVerilogScope( )
verilog.js
processGraph( )
Performs DFS on the graph and generates netlist of wires and connections
generateReport( )
generateTestBenchCode()
moduleVerilog( )
generateVerilog( )
for each element is called
along with verilog output Report and testbench is added
Complete code along with testbench code and report is displayed

Project Plan

Project Size: Small 90-hours
Project Timeline (8 weeks)

May 8 - June 1 - Community Bonding Period

Week	Dates	Tasks to be Completed
Week 1	June 2 - June 8, 2025	Implemention of Comparator and SRAM. SVG creation for both the elements.
Week 2	June 9 - June 15, 2025	Implementation of DRAM and Shift registers(all the 4 types). SVG creation for both the elements and testing of SRAM and Comparators implemented last week.
Week 3	June 16 - June 22, 2025	Implementation of Ring Counters and Control Units, creation of their SVGs and testing of DRAM and Shift Registers.
Week 4	June 23 - June 29, 2025	Implmentation of Sensors and documentation of all the implemented circuit elements till now together with testing of each element.
Midpoint Check	**June 30, 2025**	**Midpoint Progress Review**
Week 5	June 30 - July 6, 2025	Corrections in Circuit elements and their versioning.
Week 6	July 7 - July 13, 2025	Configuration and Implementation of serial communication protocols in the CircuitVerse desktop application.
Week 7	July 14 - July 20, 2025	Building of interface for serial communication and writing of hardware code.
Week 8	July 21 - July 27, 2025	Development of logic to parse the code and convert it into bitstreams to feed to Hardware using the serial communication protocol.
Final Submission	**July 28, 2025**	**Project submission deadline: Final report creation and review by the mentors.**

Major Milestones:

1. June 30: Completion of the addition of circuit elements, their testing, and review.
2. July 7: Completion of corrections to the already implemented circuit elements.

3. July 14: Completion of implementation of serial communication protocol in the Desktop application.
4. July 28: Completion and testing of serial device connectivity, bit stream generation, and upload to the hardware.

Additional Information

Why me?

This project requires expertise in EDA tools, electronic devices, Verilog, web development, and desktop application development. My proficiency in these areas, combined with my experienceof working on the CircuitVerse codebase, gives me a strong advantage in implementing these features within the given timeframe. In the past, I have contributed to CircuitVerse by enhancing the timing diagram, implementing Verilog and testbench features, improving simulator logic, fixing bugs, and introducing new functionalities. This hands-on experience gives me with the necessary skills to successfully complete this project on a tight schedule. Additionally, my prior experience in developing the eSim desktop application and my research projects have provided me with the knowledge to integrate serial communication features effectively, thoroughly test them, and create comprehensive documentation to ensure a seamless and user-friendly experience for CircuitVerse users.

Yes, I am applying to other projects in Circuitverse, namely:

Project 6: Open Hardware Component Library and
Project 4: Assignment Suite Enhancement

Project Size and Timeline Selection

Since I have already implemented one of the circuit elements, namely the comparator, building other components won't take much time, as all the circuit elements share the same class and method design. I have selected my project size to be 90 hours, which I think will be more than enough to complete my project.

However, due to the complexity of the serial communication feature and the desktop application of Circuitverse, which is still in the development phase, I might have to increase the project size to 175 hours. This condition can arise if the serial communication library is not compatible with CircuitVerse codebase and we have to introduce our own changes to it to make it compatible or the CircuitVerse desktop application itself has many bugs that hinder the integration of serial communication protocols.

Ending Note

By working on the CircuitVerse code base and this proposal I have learned a lot of things. I have learned a lot about how big projects are managed, how open-source projects are changing the world for the better, and how much I love to code and build stuff.

References:

All the contents written in this proposal or the images used belong to me.