

GSoC 2025 Proposal - CircuitVerse

Project Name: Enhanced Verilog Support & Stability

Duration: 175 hours

Difficulty: Hard

Technologies: Verilog, Ruby on Rails, JavaScript, Canvas API, VueJS

Mentors: [Vedant Jain](#), [Niladri Adhikary](#), [Josh Varga](#), [Philip Abbey](#)(self-mentor)

The Verilog feature in CircuitVerse has been a great experimental addition till now. It has allowed users to write, experiment, and test their Verilog code along with graphical circuit design using CircuitVerse simulator. But currently, the Verilog feature is full of bugs, unstable, and unreliable, which leads to its failure and causes loss of trust in CircuitVerse among its userbase. The project aims to fix the bugs, make the Verilog feature stable, refine the Verilog interface, make it more intuitive, enable users to generate, view, edit, and test circuits comprehensively, and fix the testbench generation feature for Verilog code. Other goals include adding detailed documentation for Verilog and making improvements to assist users. Additional enhancements like adding play/pause functionality to the simulator and implementing a full-screen view for the Boolean Logic Table are also covered in this project.

Verilog is one of the most important parts of VLSI and digital circuit design which is used by professors, it is taught and used by everyone in the field. Hence, fixing it is one of the most important and urgent tasks.

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 experience in Digital circuit design, this makes a perfect choice 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 Smart India Hackathon, Hacktoberfest, Google Developers Club, and contributed extensively to them.

Commitment

1. No, I am not planning any vacations during the GSoC 2025 period (June 2 - November 17, 2025).
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 medium (175 hours) project size.
5. I plan to work 20 hours per week on the project. Although I am flexible about my work hours, I prefer to work from 9:00 a.m. to 2:00 p.m.
6. Currently, I don't think I need an extended timeline.

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 timing 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 server in local](#)
 - [Fixed the Language Filter](#)

- [Replaced offset-based pagination with cursor based pagination](#)
 - [SR FF Verilog Module](#)
- Issues and bugs found
 - Verilog Testbench not working
 - 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

Proposal

Overview: Deliverables and tools used

1. Fetch the upstream changes from Yosys and reapply the changes made in the CircuitVerse fork.
2. Add the Verilog module to the circuit elements that do not have the verillogModule.
3. Fix the bugs in Verilog modules of circuit elements that have bugs.
4. Adding the play/pause functionality to the simulator and full-screen view for the Boolean Logic Table.
5. Handling the cases where Yosys generates a very large [circuit](#), which makes the circuitverse simulator overload and stop.
6. Improving the UI/UX of the Verilog code editor similar to vs code.
7. Write tests for the Verilog feature.
8. Create documentation for the Verilog feature by designing a 2-bit processor in Circuitverse using the Verilog feature.
9. The project involves extensive use of Verilog, JavaScript, TypeScript, VueJS, CanvasAPI, Yosys and Ruby on Rails.

Detailed Description

Why?

The Verilog feature in Circuitverse has attracted a lot of interest from senior graduate students and professors who are now using Circuitverse's graphical interface to teach in their class, to build digital circuits and use the Verilog feature to either generate Verilog code for the circuits or generate the circuits by writing the corresponding Verilog code.

But currently Verilog feature is **unstable, unreliable and it generates incorrect Verilog code** for circuits. This erodes the trust of CircuitVerse users. The project aims to fix this and make the Verilog feature stable and reliable features of CircuitVerse, which can attract a whole lot of new hardcore electronics students towards CircuitVerse. Implementing the Verilog feature correctly can make CircuitVerse stand out among other EDA tools which usually don't provide Circuit -> Verilog & Verilog -> Circuit feature along with support for teachers.

Implementation plan:

1. Updating the Yosys Repo

Yosysdigitaljs-server is the technology behind the feature that allows users to convert **Verilog code into circuits** in the simulator. The Yosys server converts the Verilog constructs like 'for loop' into basic circuit components like Gates and Mux to be implemented in the CircuitVerse Simulator. **It takes .v files and convert them into JSON files** of digitalJS format which can be used by Circuitverse to parse the circuit into simulator. **Below is a given example. Note** - this is done using the newest version of Yosys0.9

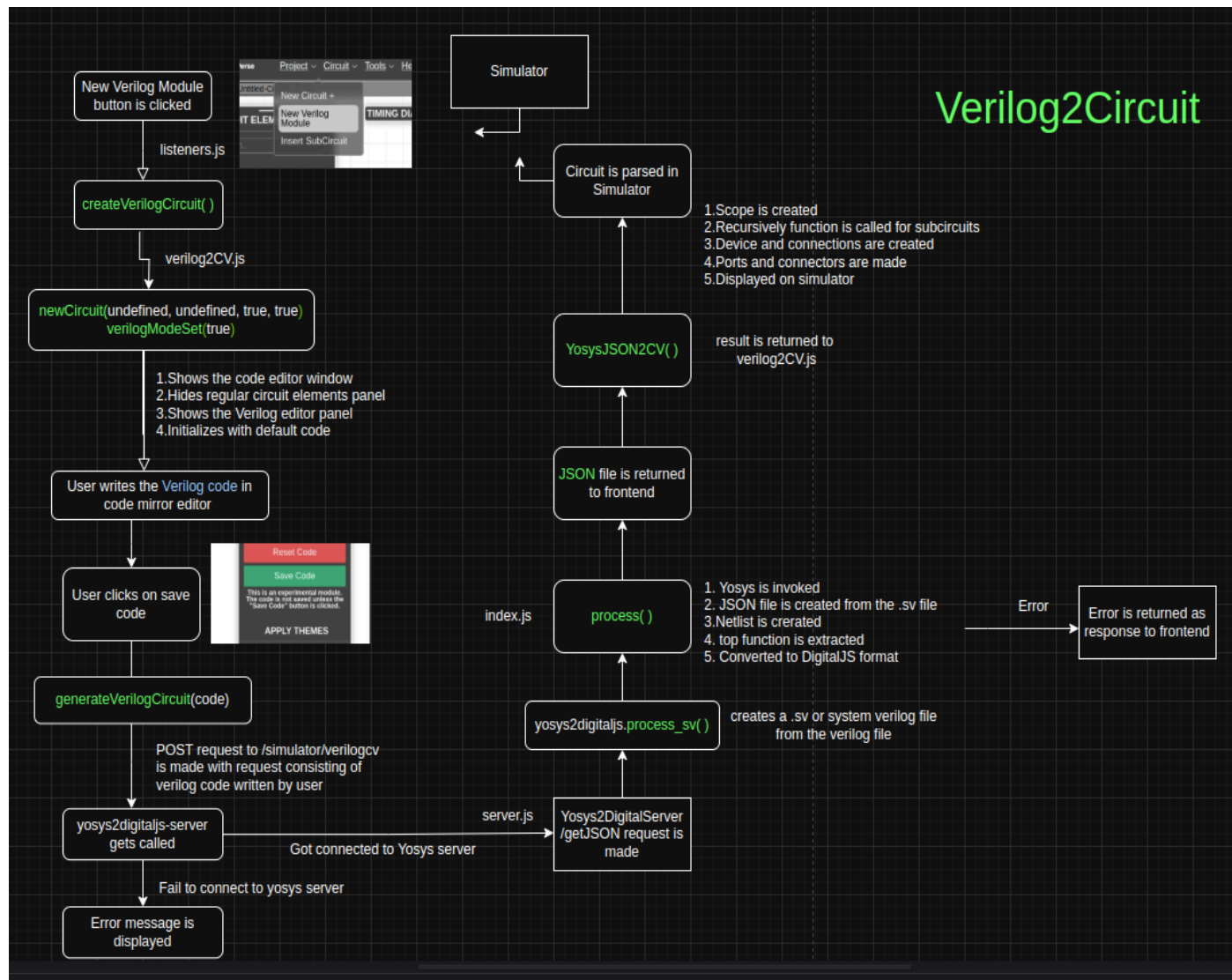
```
yosysdigitaljs-server > 1 and_gate.v
1 module top(input a, input b, output y);
2     assign y = a & b;
3 endmodule
4
```

-> .v file

```
1 {
2     "creator": "Yosys 0.9 (git sha1 1979e0b)",
3     "modules": {
4         "top": {
5             "attributes": {
6                 "top": 1,
7                 "src": "and_gate.v:1"
8             },
9             "ports": {
10                 "a": {
11                     "direction": "input",
12                     "bits": [ 2 ]
13                 },
14                 "b": {
15                     "direction": "input",
16                     "bits": [ 3 ]
17                 },
18                 "y": {
19                     "direction": "output",
20                     "bits": [ 4 ]
21                 }
22             },
23             "cells": {
24                 "$abc$47$auto$blifparse.cc:371:parse_blif$48": {
25                     "hide_name": 1,
26                     "type": "$_AND_",
27                     "parameters": {
28                     },
29                     "attributes": {
30                     },
31                     "port_directions": {
32                         "A": "input",
33                         "B": "input",
34                         "Y": "output"
35                     },
36                     "connections": {
37                         "A": [ 3 ],
38                         "B": [ 2 ],
39                         "Y": [ 4 ]
40                     }
41                 }
42             },
43             "netnames": {
44                 "a": {
45                     "hide_name": 0,
46                     "bits": [ 2 ],
47                     "attributes": {
48                         "src": "and_gate.v:1"
49                     }
50                 },
51                 "b": {
52                     "hide_name": 0,
53                     "bits": [ 3 ],
54                     "attributes": {
55                         "src": "and_gate.v:1"
56                     }
57                 },
58                 "y": {
```

-> JSON file generated by Yosys

Below is a flow chart explaining how the Verilog2Circuit feature works:



Problems

1. Yosys is a fork of the [YosysHQ/Yosys](#) repo with changes over it to implement it into the [CircuitVerse simulator fork](#).
2. But the CV fork is **behind the upstream repo**, and the upstream repo has migrated from JavaScript to TypeScript.
3. Yosys fork contains bugs and is not updated which results in faulty Verilog code generation.
4. For larger circuits the JSON file becomes very large creating a **nested circuit** which makes the simulator stop/crash for example in case of : [7SeqDecoder](#) demonstrated.

Project goals include:

- Fetch all new upstream changes from [YosysHQ/Yosys](#).
- Migrate the Yosys server from JavaScript to TypeScript.
- Reapply earlier **custom changes** made for the CircuitVerse fork.

- Resolve conflicts that arise with the forked repo.
- Ensure the Yosys server remains stable and efficient.
- Write tests to validate functionality.
- Write proper error handling to let the user know what is wrong.

2. Adding the VerilogModule of elements that are currently missing

The `moduleVerilog()` method, which is part of the `CircuitElement class()`, that contains the Verilog code to be implemented, is missing in some of the elements. This causes the below-given condition where the `D_latch` method is called, but implementation is missing; the aim is to have this method defined in their class as `moduleVerilog()` properly to produce complete Verilog code as it is already done with other elements. [Video Explanation of Issue](#)

```

59
60 */
61
62 module Main(out_0, out_1, clk_0, inp_0);
63     output out_0, out_1;
64     input inp_0, clk_0;
65     wire Dlatch_0_Q, Dlatch_0_Q_inv;
66     Dlatch Dlatch_0(Dlatch_0_Q, Dlatch_0_Q_inv, clk_0, inp_0);
67     assign out_1 = Dlatch_0_Q_inv;
68     assign out_0 = Dlatch_0_Q;
69 endmodule
70

```

Here on line 66, the D-Latch function or module is called, but it is not defined anywhere, causing the error.

The following elements need VerilogModule to be added:

1. SR Flip Flops
2. JK Flip Flops
3. ALU
4. T-Latch
5. ForceGate
6. Others like LSB, MSB etc

A prototype correction -> for JK Flip Flop

```

static moduleVerilog() {
    return `
module JKflipFlop(q, q_inv, clk, j, k, a_rst, pre, en);
    parameter WIDTH = 1;
    output reg [WIDTH-1:0] q, q_inv;
    input clk, a_rst, pre, en;
    input [WIDTH-1:0] j, k;

    always @ (posedge clk or posedge a_rst)
    if (a_rst) begin
        q <= 'b0;
        q_inv <= 'b1;
    end else if (en == 0) ;
    else begin
        case ({j,k})
            2'b00: q <= q;          // No change
            2'b01: q <= 1'b0;      // Reset
            2'b10: q <= 1'b1;      // Set
            2'b11: q <= ~q;        // Toggle
        endcase
        q_inv <= ~q;
    end
endmodule
}

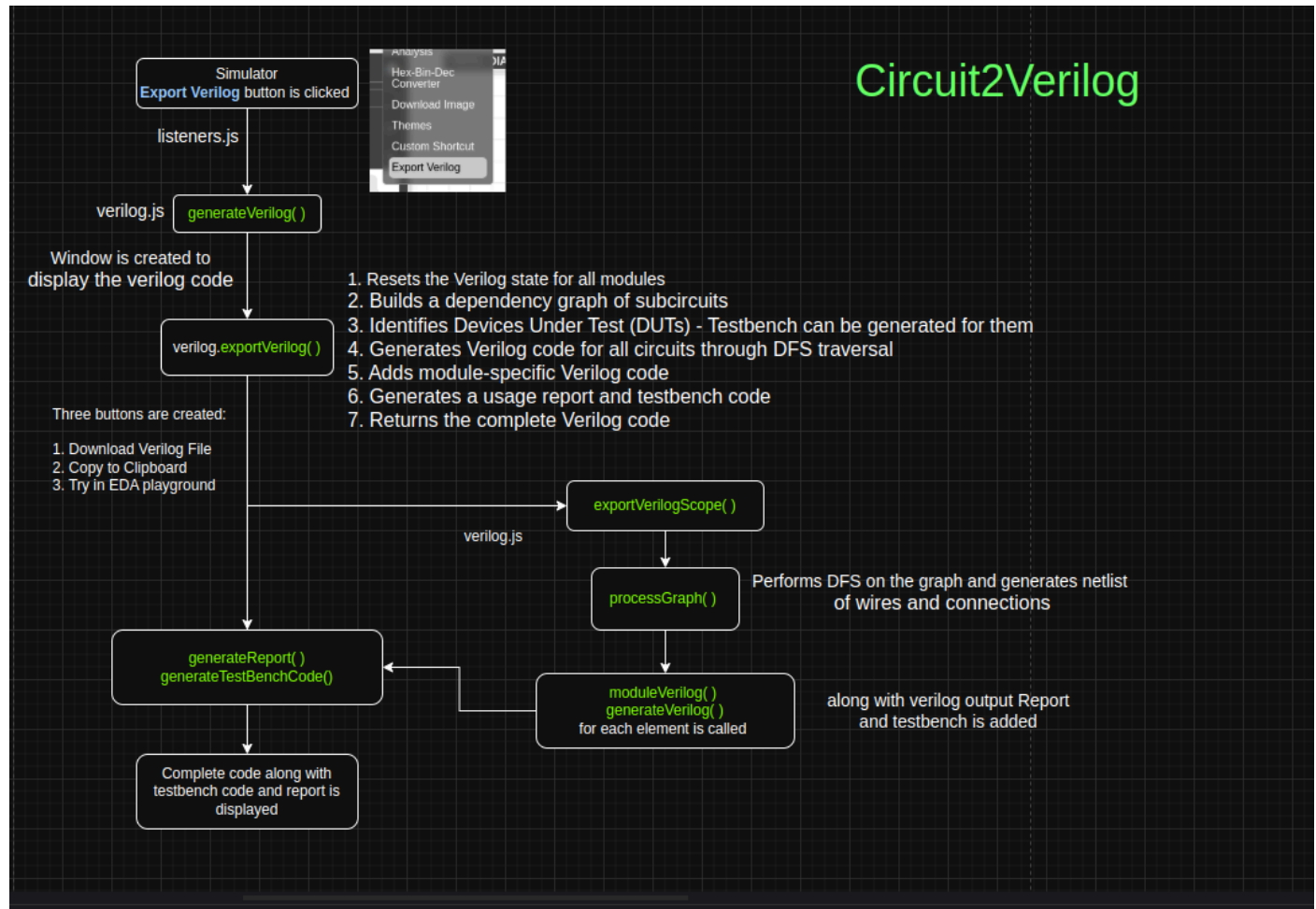
```

PR's Which fixes this issue:

[JK FF](#) , other similar fixes are needed for all the elements which are listed above

This flow chart explains how the Circuit2Verilog feature works:

→ The correction is needed where DFS is performed on the graph and `moduleVerilog()` is called for different elements



3. Fixing the bugs in Verilog modules of circuit elements that are not consistent with their simulation models :

[Link of Issue highlighting this issue](#)

1. Many circuit elements in CircuitVerse have **bugs in their Verilog module definitions** and are not consistent with their simulation model.
2. For example, **all flip-flop modules include pre (preset) in their parameter list**, but:
 - a. Pre is **not implemented in the Verilog code**,

- b. Even though it **exists in the actual circuit element's logic implementation**.
3. Other circuit components show **similar inconsistencies and bugs** between:
 - a. The **Verilog module code**, and
 - b. Their **logic implementation within the CircuitVerse simulator**.

The project aims to:

1. Identify and fix these inconsistencies
2. Ensure the Verilog module matches the **actual circuit behavior**,
3. And **achieve consistency** between the Verilog code and the simulation logic of each element.

Below is a error demonstrated :

```

70
71 module DflipFlop(q, q_inv, clk, d, a_rst, pre, en);
72     parameter WIDTH = 1;
73     output reg [WIDTH-1:0] q, q_inv;
74     input clk, a_rst, pre, en;
75     input [WIDTH-1:0] d;
76
77     always @(posedge clk or posedge a_rst)
78     if (a_rst) begin
79         q <= 'b0;
80         q_inv <= 'b1;
81     end else if (en == 0) ;
82     else begin
83         q <= d;
84         q_inv <= ~d;
85     end
86 endmodule
87

```

In the above code, you can see **Pre** is listed in the parameter list but not used even though pre is a **valid pin in simulation model**.

The **verilogModule()** and **generateVerilog()** methods will be evaluated and for each circuitElement, and it will be fixed.

4. Improving the UI/UX of the Verilog code editor

1. Improve the UI/UX of the Verilog code editor.
2. Remove unnecessary **double scroll bars** for a cleaner interface.
3. Enhance syntax highlighting for better readability.
4. Redesign the buttons and panels to improve usability.
5. Take inspiration from VS Code for a modern look.
6. Enable **autosave** for the code editor.
7. Improve the **Verilog output window** for better clarity.
8. Highlight different sections of the generated Verilog code for a better user experience.

5. For certain Verilog code, CircuitVerse becomes unresponsive

[Link of Issue highlighting this issue](#)

Certain [Verilog Code](#) when inserted in Simulator it makes the whole website non-responding. This happens due to the large layers of mux or other circuit elements applied on each other which makes the simulator slow and thus non-responding. The project aims to fix this by streamlining the logic of certain components like Mux and Demux to improve their efficiency and avoid scenarios like one which is mentioned in the link.

This involves streamlining the logic of certain elements such as MUX.

6. Adding the play/pause functionality to the simulator and full-screen view for the Boolean Logic Table

Currently, the simulator keeps on running all the time, even in cases where the user is still designing the circuit or is performing some other functions. This wastes lots of computing. This project introduces a play/pause button for the simulator.

1. A **play/pause button** will be introduced to stop and start the simulator as needed by user.
2. The implementation involves stopping the clock function, which sends the clock signal to all elements in the simulator.
3. Files which will be changed are **/simulator/src/simulationArea.js, engine.js, plotArea.js, and plotArea.js**.
4. A full-screen view for the **Boolean logic table** will be added for better usability.
5. This will allow users to properly enter and evaluate Boolean algebra and the corresponding table.

7. Documenting the Verilog feature

Initial PR which is merged regarding Docs

1. All changes will be backed by proper documentation and examples for users.
2. A 2-bit processor will be designed using the Verilog feature as an example for the documentation.
3. The entire process of designing the processor will be recorded for users to refer to.
4. This helps users understand how to use the Verilog feature correctly.
5. Tests will be written for files to ensure reliable implementation.

Project Plan

Project Size: Medium 175-hours

Project Timeline (12 weeks)

May 8 - June 1 - Community Bonding Period

Week	Dates	Tasks to be Completed
Week 1	June 2 - June 8, 2025	Implement the Verilog module for circuit elements that do not have them.
Week 2	June 9 - June 15, 2025	Check and fix the errors in the Verilog module of the circuit elements and fix the bugs found as listed above.
Week 3	June 16 - June 22, 2025	Fetching the upstream changes from the Yosys repo and applying them over the CircuitVerse fork.
Week 4	June 23 - June 29, 2025	Removing any conflict that arises due to the fetching of upstream changes.

Week 5	June 30 - July 6, 2025	Migration of files in Yosys repo from Javascript to TypeScript
Week 6	July 7 - July 13, 2025	Testing of Yosys server, documentation of Verilog modules created, and of Yosys server itself.
Midterm Evaluation	July 14 - July 18, 2025	Midterm Evaluation Milestone
Week 7	July 14 - July 20, 2025	Improving the UI/UX of the Verilog code editor.
Week 8	July 21 - July 27, 2025	Testing of the Verilog Code editor and other Verilog features.
Week 9	July 28 - August 3, 2025	Adding the play/pause button to the simulator.
Week 10	August 4 - August 10, 2025	Implementing full-screen view for the Boolean logic table.
Week 11	August 11 - August 17, 2025	Documenting the whole Verilog feature and changes made to it. Finding bugs and fixing them
Week 12	August 18 - August 24, 2025	Creation of a 2-bit processor using the Verilog and documenting it. Writing tests for the Verilog feature and the code editor.
Final Week	August 25 - September 1, 2025	Final testing, documentation, and code cleanup
Final Submission	September 1, 2025	Project submission deadline

Major Milestones:

1. June 9: All the bugs in C2V feature will be removed and the feature will be tested.
2. July 7: Completion of fetching of Upstream changes and migration of Yosys server.
3. July 21: Completion of changes in the UI/UX of the Verilog code editor and its testing.
4. July 28: Completion of the addition of the play/pause feature in the simulator.
5. August 4: Completion of implementation of full-screen view for the Boolean logic table.
6. August 18: Completion of documentation and testing of Verilog feature.
7. September 1: Final report submission.

Additional Information

Why me?

This project requires expertise in EDA tools, electronic devices, Verilog, and web development. My proficiency in these areas, combined with my experience of 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 the necessary skills to 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 fix the Verilog issue 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 worked on the simulator, testbench, and Verilog feature, I can reduce the time taken to understand the codebase and directly jump into the coding part. I have selected my project size to be 175 hours, which I think will be more than enough to complete my project.

Ending Note

By working on the CircuitVerse codebase 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.