cnes

11

FPGA EXPLORER

User Manual

ALGOSUP
International Software Development School

Table of Contents

Introduction

Welcome to **FPGA Explorer**, a web-based simulation tool for visualising and analysing an FPGA's internal chip flows. Designed for students, teachers, and engineers, this platform provides an interactive, user-friendly experience for exploring FPGA concepts through **SDF file uploads, clock speed adjustments, and step-by-step simulations**.

What You'll Learn in This Manual?

- **Navigating the Interface** – How to use the FPGA Explorer efficiently.
- **Uploading & Simulating FPGA Designs** – Step-by-step file uploads and visualisation.
- **Controlling Simulations** – Start, pause, reset, and step through processes.
- **Adjusting Clock Speeds** – Modify simulation speed in real time.
- **Troubleshooting** – Common issues and solutions.

Who Is This Manual For?

- **Teachers** – For interactive FPGA instruction.
- **Students** – To explore and understand FPGA behaviour.
- **Developers & Engineers** – For a web-based FPGA simulation tool.

Keep this guide handy for reference while using **FPGA Explorer**. For further assistance, check the **FAQ** or **contact** support.

Getting Started

The web interface is accessible via our GitHub repository and a hosted link.

Local Access

To run the web interface locally:

1. **Clone the repository**

   GitHub Repository *

2. **Open a terminal**

3. **Navigate to the directory**

   ```
   cd src
   ```

4. **Start the application**

   ```
   npm start
   ```

The web interface will open in your default browser.

Online Access

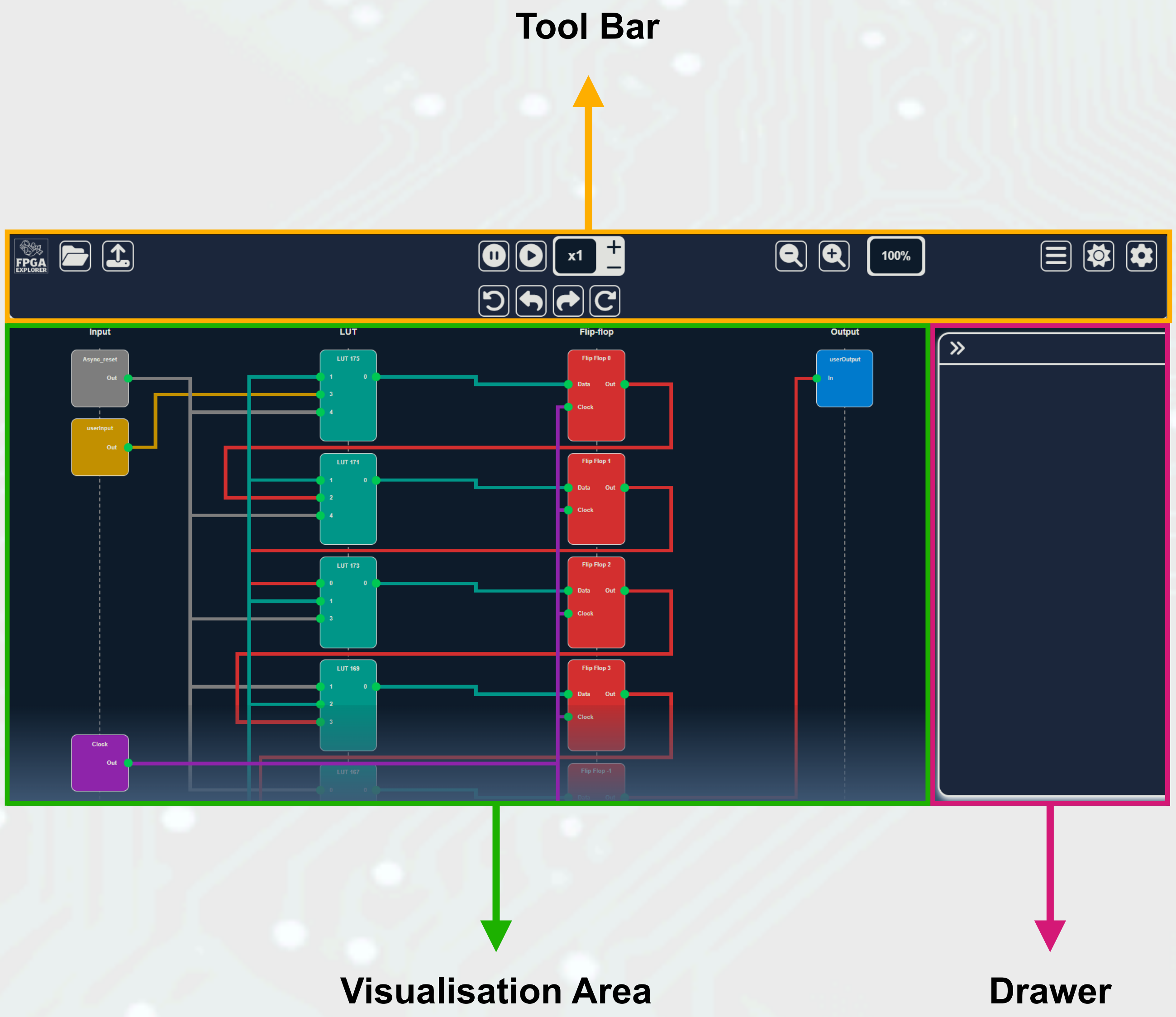
You can also access the interface online at:

FPGA Explorer **

**https://github.com/algosup/2024-2025-project-4-web-fpga-team-4/tree/main*

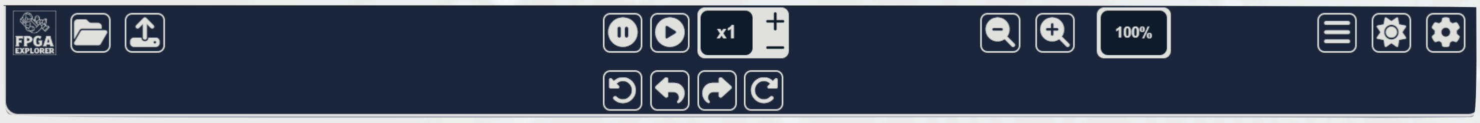
***https://two024-2025-project-4-web-fpga-team-4.onrender.com/client.html*

Interface Overview



Tool Bar

The **toolbar** in FPGA Explorer provides quick access to essential functions, including file management, simulation controls, zoom options, and display settings.



File Management

Load File – Opens an example file.
Download JSON – Downloads the pivot file.

Simulation Control

Play – Starts or resumes the simulation.
Stop – Stops the simulation temporarily.
Step Back – Moves the simulation one step backwards.
Step Forward – Advances the simulation by one step.
Reset – Resets the entire simulation.
Final Step – Moves to the end of the simulation.

Zoom & Navigation

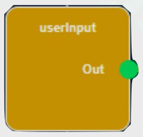
Zoom In – Enlarges the simulation view.
Zoom Out – Reduces the simulation view.

Options & Settings

Show/Hide – Expands or collapses the real-time clock signal data panel.
Light/Dark Mode – Toggles between light and dark themes.
Settings – Opens the configuration panel for additional options.

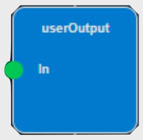
Component Color Code

The visualisation area is the main part of this interface. It shows all components and signal flows, each with a unique colour and shape for clarity.



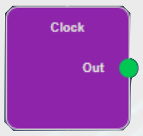
Input

- Inputs *send data* into the circuit (like buttons or lights)



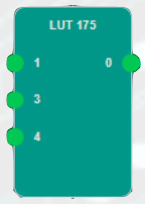
Output

- Outputs *display results*



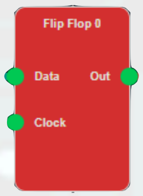
Clock

- Drives the simulation timing
- Feeds pulses to *synchronise Flip-Flops*



Look-Up Table (LUT)

- Executes logical operations
- Used for *combining and transforming input* values

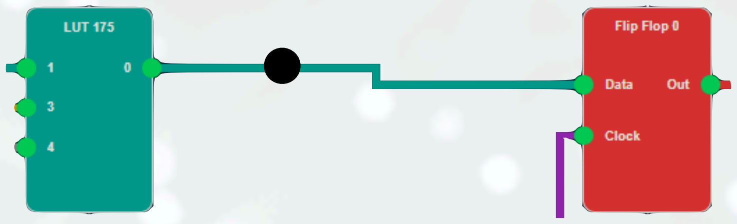


Flip-Flop

- Stores data and *introduces a delay in circuits*
- Controlled by clock edges

Wire

- T*ransfers logic signals* between components
- Matches the component it originates from



A **white or black circle** moves along the wire to represent real-time signal flow.

Running a Simulation

To run a simulation in the FPGA Explorer, follow the steps below.

1. Upload a File

Use the Upload button at the top left.
You can upload:

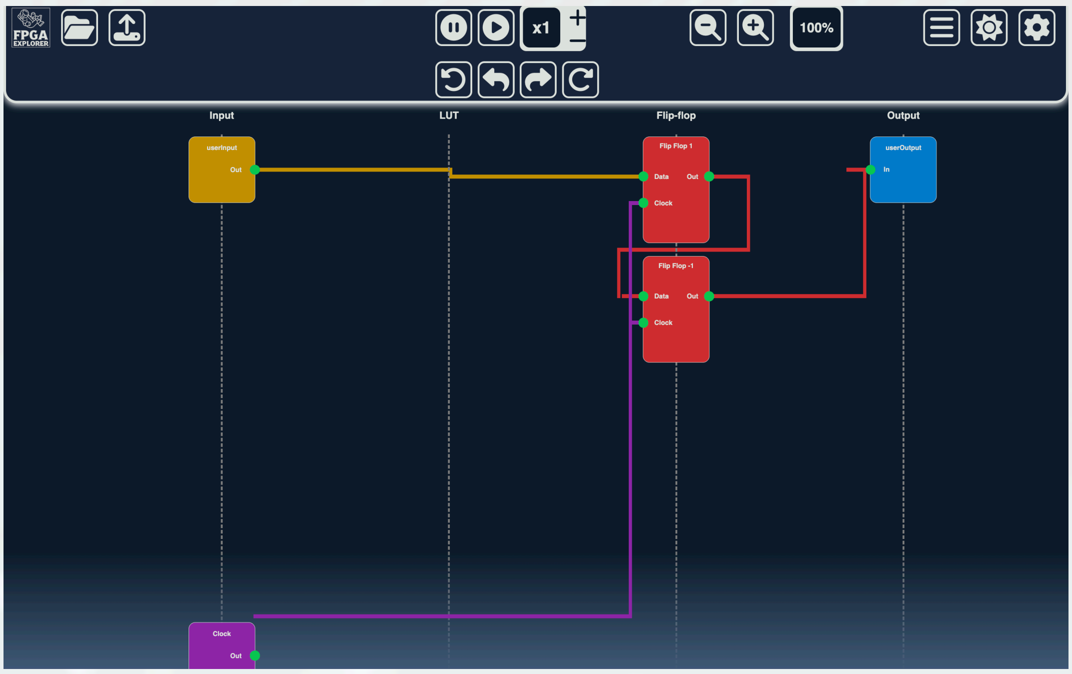
- .SDF (Standard Delay Format)
- .JSON (already formatted for the visualiser)

When you upload an `.SDF`, the system will convert it automatically to a `.JSON` structure.

2. View the Circuit

Once the file is uploaded:

- The circuit is automatically displayed in the interface
- Components appear organised by type (Input, Flip-Flop, Output)
- Wires show signal flow between them
- Each wire's colour matches the component it originates from



3. Run the Simulation

Use the toolbar to control the simulation:

- ▶️ Play – Starts or resumes the simulation
- ⏸️ Pause – Temporarily stops the simulation
- ⏮️ / ⏭️ Step Back / Step Forward – Move frame-by-frame
- 🔁 Reset – Restarts the simulation from the beginning

Adjust the speed using the selector to slow down or speed up the playback.

4. Simulation Behavior

As the simulation runs:

- Data flows from input through Flip-Flops to the output
- Flip-Flops update their output only on clock edges
- Each signal respects the delay defined between components

In the example shown:

- `userInput` feeds into Flip-Flop 1
- Flip-Flop 1 sends its output to Flip-Flop 2
- Flip-Flop 2 loops back to Flip-Flop 1 and also drives `userOutput`

This setup simulates a simple delay chain with feedback.

How to Read Delays

Delays represent the time it takes for a signal to travel from one component to another. Understanding how to read and interpret these delays helps to analyse timing behaviour in digital circuits.

Visual Indicators of Delay

- Each wire represents a connection between two components.
- A moving dot travels along the wire to show signal propagation.
- The speed of the dot reflects the length of the delay:
    - *Longer delays = slower animation*.

Delay Behavior in Simulation

- Flip-Flops only update on clock edges and only after input signals arrive.
- Outputs change state only after all delays along the signal path are completed.
- Delays accumulate along multiple connections:
    - *input → Flip-Flop 1 → Flip-Flop 2 → output*.

Tips for Observing Delay

- Use x1 speed to see the full delay effect.
- Use Step Forward to monitor changes clock cycle by clock cycle.
- Watch when output blocks change: it will never be instantaneous.

Troubleshooting

The simulation doesn't start
The file may be missing components or incorrectly formatted.

- Use a `.sdf` or `.json` file
- Include at least one LUT or Flip-Flop
- Try uploading a known working example

No wires between components
Connections might be missing or malformed in the `.json` file.

- Each connection should include `input`, `output`, and `delay`
- Check that port numbers and types match

Play/Pause/Step buttons don't respond
The simulation may not be properly loaded.

- Upload a valid file
- Refresh the page

Layout looks broken or components overlap
Rendering or browser zoom issue.

- Set browser zoom to **100%**
- Use Chrome, Firefox, Safari, or Edge
- Avoid mobile devices

Signal animation isn't visible
Simulation may be paused or the delay is too short to notice.

- Press Play
- Lower speed to x1
- Check that delay values exist in your connections

Error when uploading a file
The file is not readable or exceeds size limits.

- Make sure the file extension is .sdf or .json
- The file must be under 10MB
- If the error persists, try another file or refer to the example files

FAQ

Can I write Verilog or VHDL inside the tool?

No — the tool is for visualisation only. You must upload `.SDF` or `.JSON` files.

What files are supported?

Only `.SDF` and `.JSON` are supported at the moment. Verilog must be compiled externally.

How do I control the simulation speed?

Use the speed control buttons (x1, x2, x4) in the toolbar.

👉 *See "Running a Simulation" for more.*

What do the different block colours mean?

Each component has a colour (LUT = green, Flip-Flop = red, etc.)

👉 *See "Component Color Code" for the full legend.*

What is the `.json` file supposed to contain?

The `.JSON` file must describe the FPGA circuit in a structured way, including:

- Components (LUTs, Flip-Flops, I/Os) with their IDs and ports
- Connections between components, including delays
- A list of I/Os with their types (input/output)

If you're unsure, you can:

- Upload a `.SDF` file and download the generated `.JSON` to use as a template.
- Refer to the Technical Specifications or example files included in the repository.

Contact

If you would like to get in touch with the development team — whether to report a bug, ask a question, or share feedback — please use our GitHub repository. We manage all inquiries directly through GitHub Issues to ensure transparency and traceability.

How to Contact Us

1. Navigate to the issue tab of the GitHub repository*.
2. Open a new issue.
3. In the title, include the word contact.

💡 **Tip:** Please be as clear and specific as possible when describing your request so we can assist you efficiently.

We monitor GitHub regularly and will get back to you as soon as possible.

**https://github.com/algosup/2024-2025-project-4-web-fpga-team-4/issues*