Computer Architecture Lab Report

Experimental Purposes

- Understand the basic operation and architecture of FPGAs.
- Write and simulate Verilog code to design digital circuits (e.g., LED blinking).
- Utilize the high-level FPGA development environment provided by Gowin Semiconductor.
- Employ Git version control tools to manage and collaborate on FPGA projects.
- Interpret, compile, and deploy complex FPGA projects such as an NES emulator.
- Analyze and understand the implementation of the RISC-V instruction set on FPGA through the SparrowRV processor.
- Demonstrate successful integration of FPGA, peripheral hardware, and external devices.
- Produce comprehensive lab reports articulating design choices, experimental results, and lessons learned

Lab 0: Pre-Lab Setup: Unboxing and Environment Installation

Description:

We performed an initial test of the Tang Nano 20K FPGA board by unboxing it and verifying its functionality. We also installed and set up the necessary FPGA development tools.

Objectives:

- Ensure proper hardware functionality through unboxing procedures.
- Install and validate the Gowin FPGA IDE on their systems.
- Set up Git for version control as a collaborative tool for the lab course.

Instruction:

The Tang Nano 20K is a core board based on the GigaDevice GW2AR-18 QN88, featuring 20,736 logic look-up tables (LUT4) and 15,552 flip-flops (FF) inside the chip. It includes two PLLs and multiple DSP units supporting 18-bit x 18-bit multiplication operations to accelerate digital computations. The onboard BL616 chip serves as the FPGA JTAG downloader and a serial port for communication with the FPGA. By default, the board uses a 27MHz crystal oscillator to generate the clock required for HDMI display, and it also incorporates the MS5351 clock generation chip to produce various clocks as needed.

First Power-On:

After powering on, you can see the six LEDs on the board running in a running light pattern.The video of first power-on can be seen in the “Appendix” section-“Lab0.mp4”.

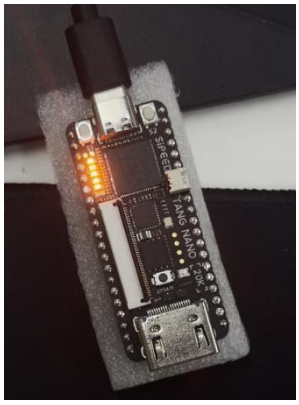


Figure 1. First power-on.

After powering on, Windows 10 and Windows 11 will automatically load the drivers, and see at least one serial port device in the computer's Device Manager. At this point, use serial port-supported software like Mobaxterm to open the serial port on the development board. Set the baud rate to 115200, then open the serial port on the development board to enter the default litex firmware terminal.

Hardware inspection:

By calling the clock and LED, test whether functions of the hardware normally.

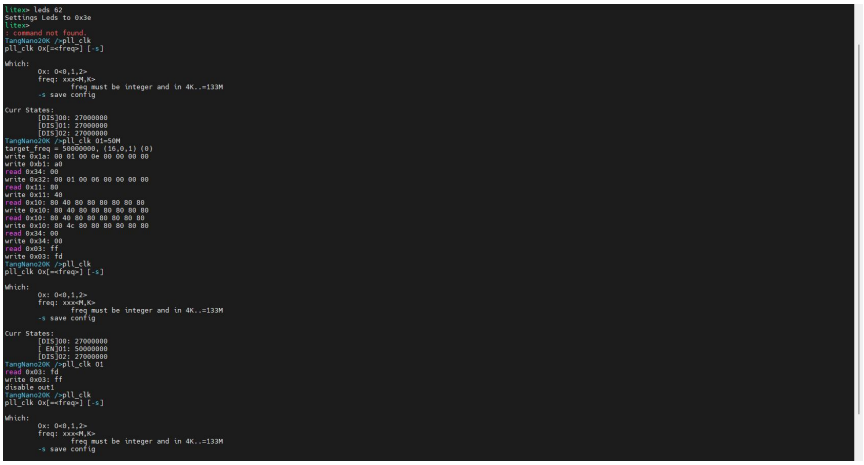


Figure 2. Hardware inspection.

Below is the change of LED of the board after entering leds 62 in the serial terminal.

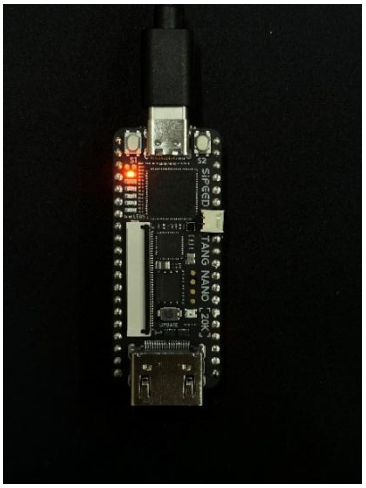


Figure 3. After entering leds 62 in the serial terminal.

Install the Gowin FPGA IDE:

After installing the software, the interface that opens is as follows.

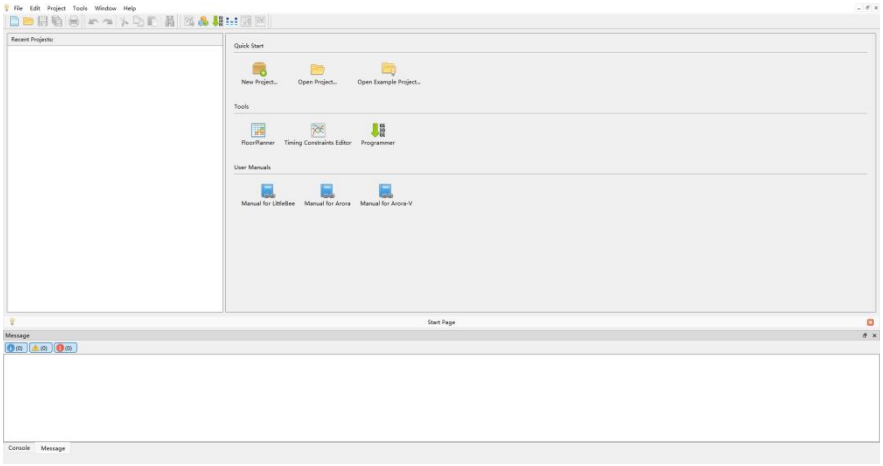


Figure 4. Install the Gowin FPGA IDE.

Lab 1: Verilog Programming & LED Blinking Experiment

Description:

This lab introduces Verilog fundamentals and the FPGA development process. We wrote Verilog code to create an LED blinking experiment on the Tang Nano 20K board.

Objectives:

- Learn basic Verilog syntax and coding styles.
- Understand and implement a simple counter-based LED blinking design.
- Use the Gowin FPGA IDE to compile, synthesize, and program the FPGA.
- Gain proficiency in using Git to manage source code and track changes.

Creating a new project:

File -> New -> FPGA Design -> Project. Then, from the chip models below, select GW2AR-LV18QN88C8/I7.

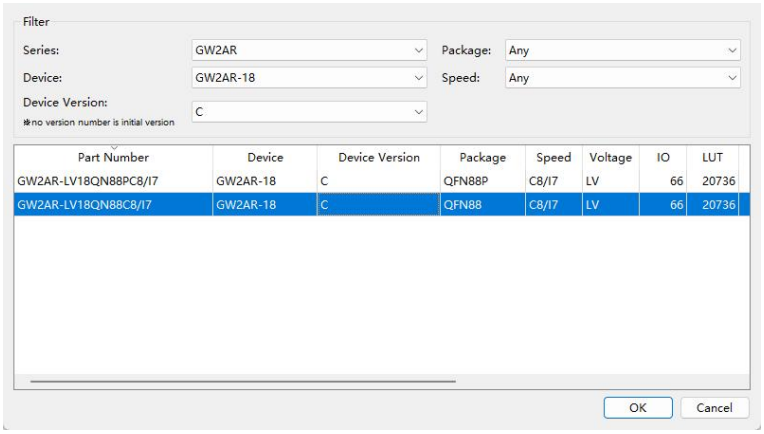


Figure 5. Create a new project.

Writing code:

Here, directly use the shortcut Ctrl + N to create a new file, In the pop-up window, select Verilog File.

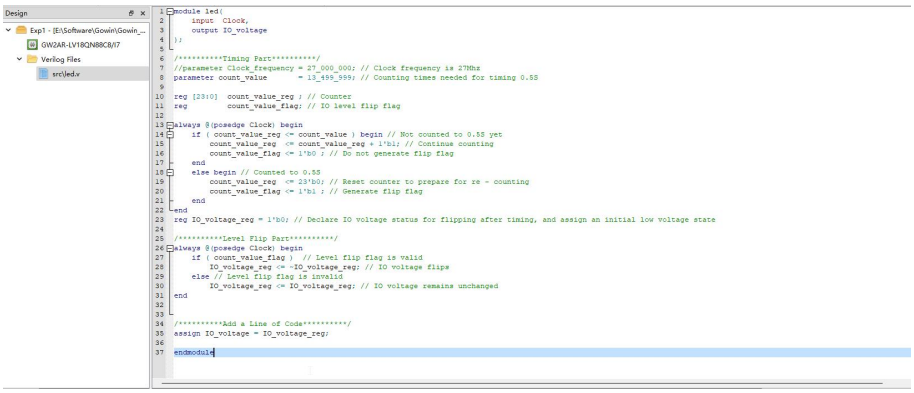


Figure 6. Write code.

Comprehensive, constraints, layout and routing:

After saving the code, double-click on Process -> Synthesize within the IDE to perform code synthesis, converting the Verilog code content into a synthesis netlist.

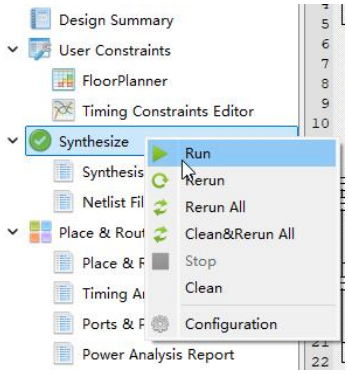


Figure 7. Synthesis.

After synthesis, perform pin constraints to map the ports of the written modules to the FPGA pins and implement the module's functionality.

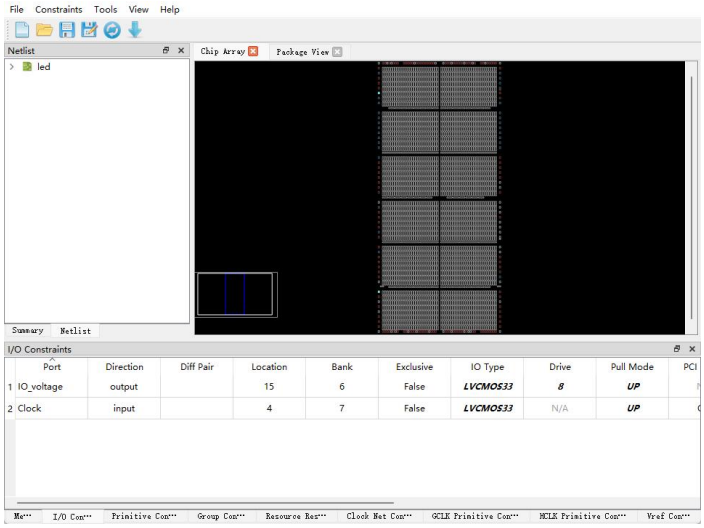
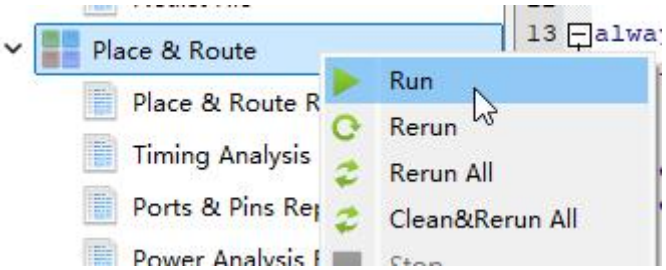


Figure 8. Perform pin constraints and module's functionality.

After completing the constraints, it's time to start the layout and routing process. The goal is to use the IDE to calculate the optimal solution by matching the netlist generated during synthesis with the constraints we defined, and then allocate resources reasonably on the FPGA chip.



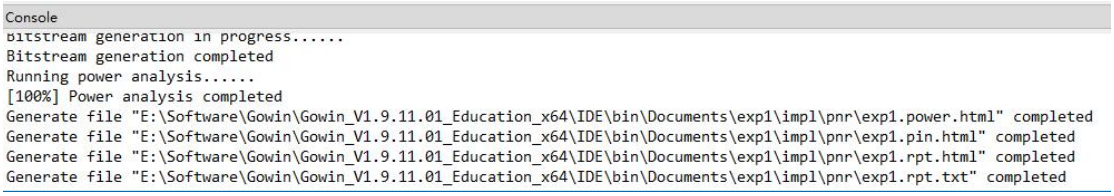


Figure 9. Place and route.

Firmware burning:

The drivers have already been installed when installing the IDE. Therefore, simply connect the board to our computer.

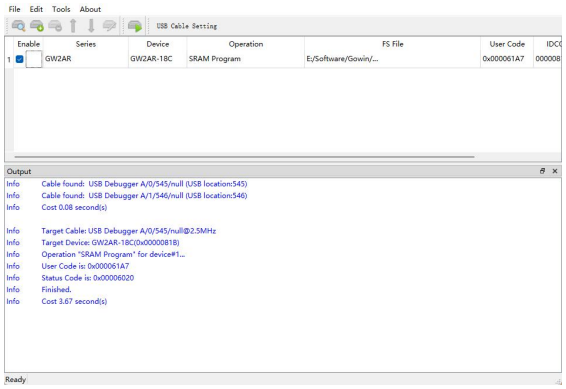


Figure 10. Burn firmware

The final result: A counter toggles an LED every 0.5 seconds.The video of the demonstration of the LED blinking functionality can be seen in the “Appendix” section-“Lab1.mp4”

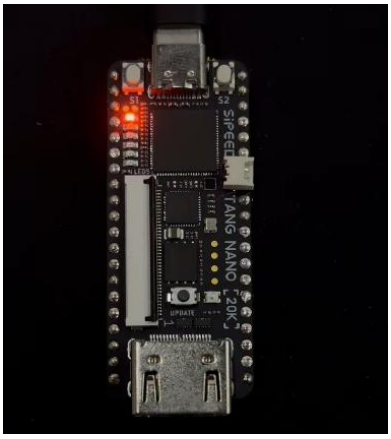


Figure11. A counter toggles an LED every 0.5 seconds.

Upload our project to Github:

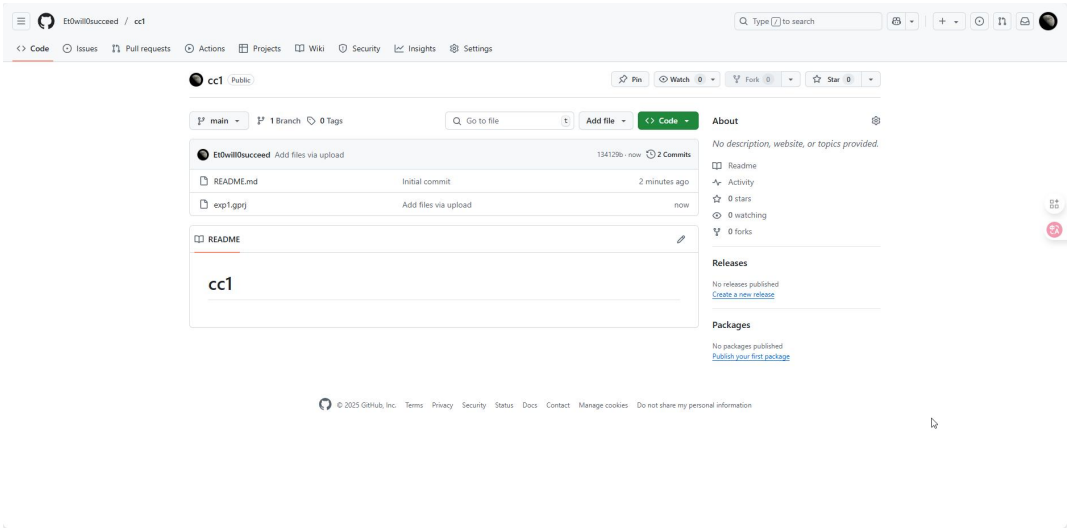


Figure 12. Upload project to Github.

Lab 2: NES Emulator on FPGA

Description:

We worked with an existing NES emulator project for the Tang Nano 20K board.We read and understood the source code, setting up peripherals(joysticks, HDMI monitor, TF card), and achieved full compilation and execution.

Objectives:

- Understand the interaction between the FPGA and external devices (display, input controllers, storage).
- Analyze the NES architecture and instruction set principles.
- Compile and deploy an NES emulator on the FPGA board.
- Evaluate system performance and troubleshoot hardware–software interfacing issues.

Create images of the selected game:

Select the image from FC Game Network and extract the downloaded images into the same folder as the firmware dependencies.

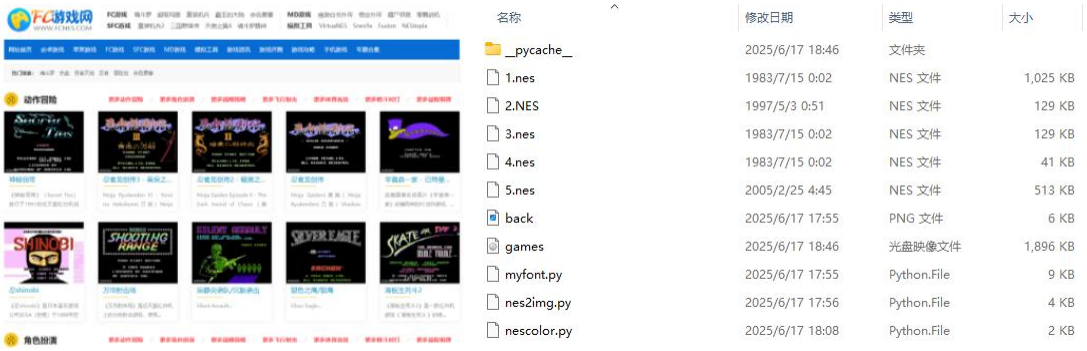


Figure 13. Select games and extract the images.

Burning FPGA firmware:

Use the Gowin Programmer to burn this firmware into the FPGA's Flash.

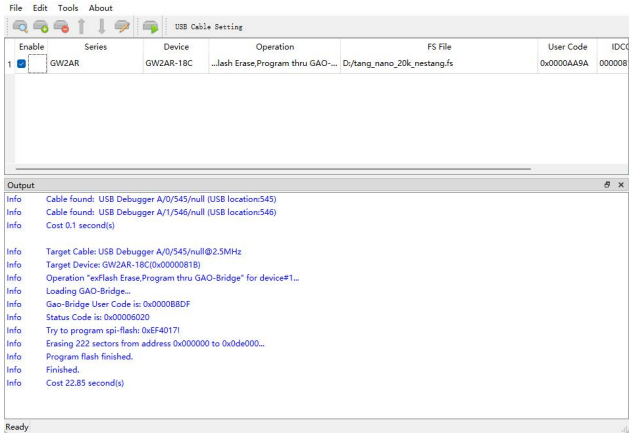


Figure 14. Burn firmware to flash.

Burning the game image

On the Tang Nano 20K, a TF card is required to store games, so the game image file needs to be burned onto the TF card. Here, the software used is balenaEtcher. Open this software, select Flash from file , and select the previously generated game image file games.img

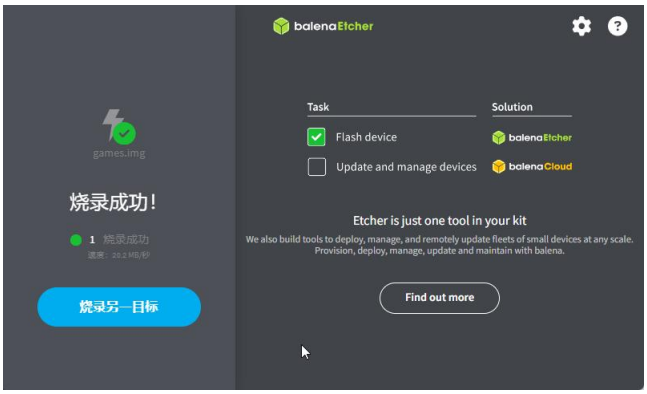


Figure 15. Successfully burning.

Start playing game:

Power on the Tang Nano 20K, and it will display the game menu, with the number and names of the games generated based on the NES game files and names used when creating the game image file.The video of game demonstrating can be seen in the “Appendix” section-“Lab2.mp4”

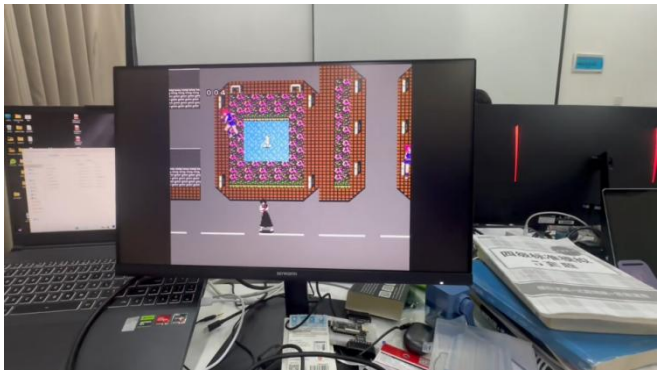


Figure 16.Demonstration of the game.

Lab 3: SparrowRV RISC-V Processor Experiment

Description:

In this lab,we explored the implementation of the RISC-V instruction set on FPGA by working with the SparrowRV processor project. We read the source code, compiling the processor design, and run a simple “Hello World” program.

Objectives:

• Understand the principles behind implementing a RISC-V compatible processor on an FPGA.

• Analyze the SparrowRV processor’s source code to grasp the core concepts of pipelined processing and instruction set architecture.

• Successfully compile and program the FPGA with the SparrowRV design.

• Execute a “Hello World” test on the running processor.

Get codes from Github:

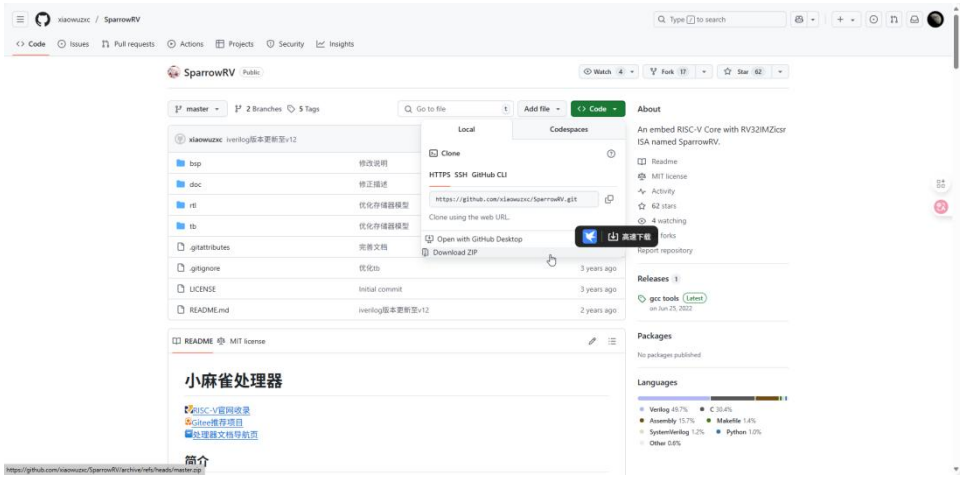


Figure 17. Get codes.

After the same steps as Lab1:

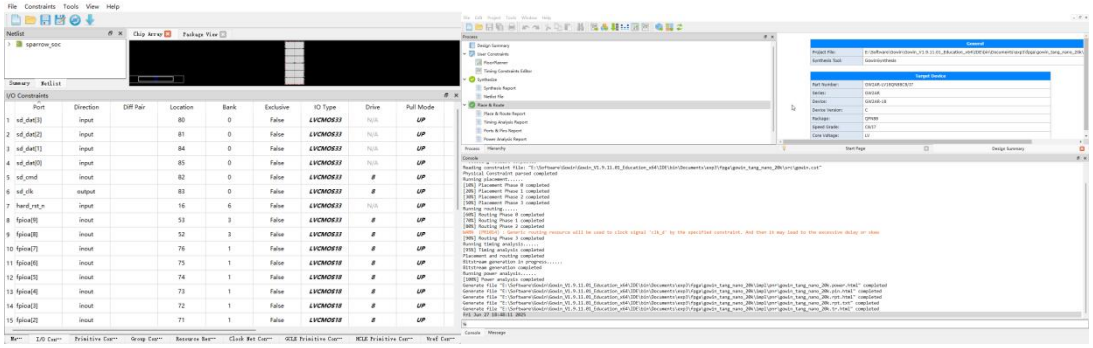


Figure 18. Constraint, layout and routing.

We finally get the result (The video of demonstration of the “Hello World” program running can be seen in the “Appendix” section-“Lab3.mp4”):

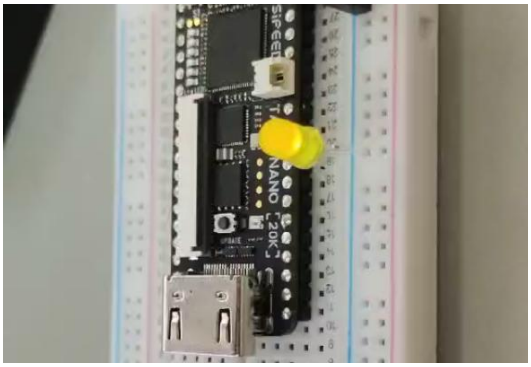


Figure 19. Result of lab3

Experiment Summary and Gains

FPGA Development Basics:

1. Mastered setting up the FPGA development environment, including installing tools and configuring version control with Git.
2. Gained proficiency in Verilog programming through implementing an LED blinking experiment, understanding basic syntax and hardware design concepts.
3. Became familiar with the FPGA design workflow, from project creation to programming the device.

NES Emulator Project:

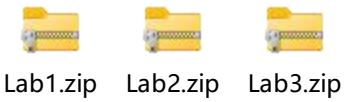
1. Learned to interface FPGA with external devices like HDMI and joysticks.
2. Analyzed NES architecture and successfully ran an emulator on the FPGA board.
3. Gained experience in compiling and debugging complex FPGA projects.

SparrowRV Processor Experiment:

1. Deepened understanding of RISC-V architecture by studying the SparrowRV processor code.
2. Successfully compiled and tested the processor design, running a "Hello World" program.
3. Enhanced problem-solving skills by addressing issues encountered during the experiment.

Appendix

1. **Codes and models:**



2. **Demonstrations of the results:**