FULVIO CASTELLO, EUGENIO RESSA, UMBERTO TOPPINO

HARDWARE-BASED CTF

CAPTURE-THE-FLAG CHALLENGES: AN OVERVIEW ❖

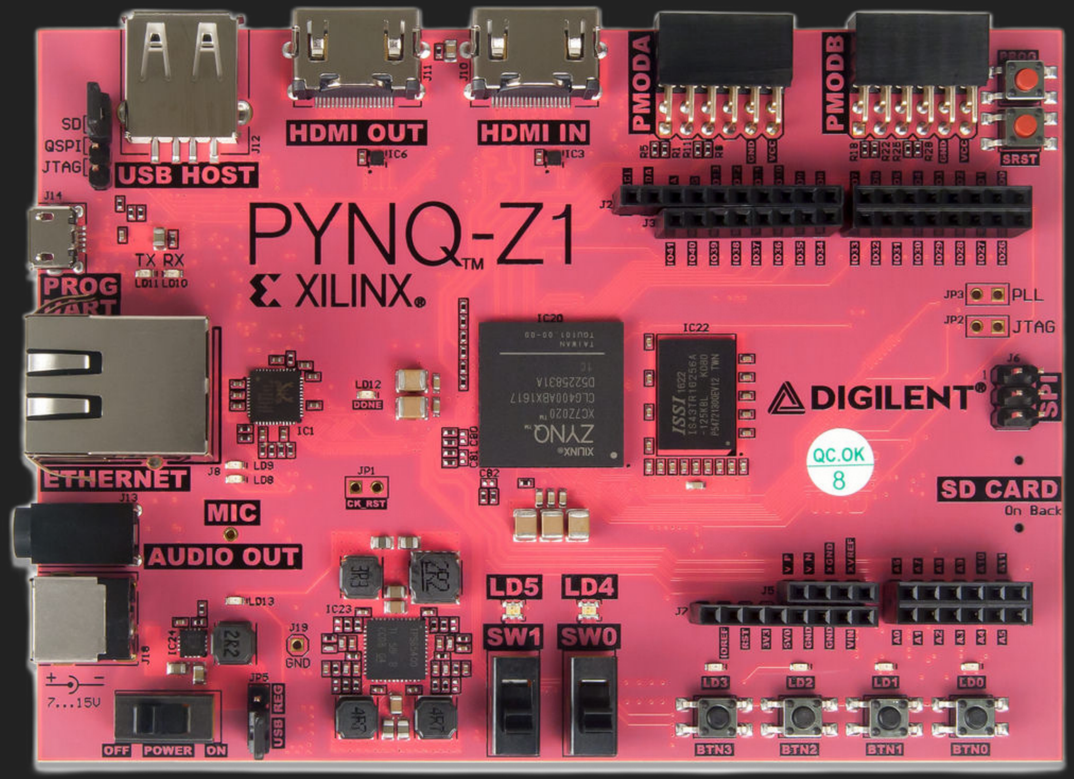
- **Purpose:** exploiting one or more vulnerabilities included inside the target platform by means of personal experience/skills, with the objective of capturing a “flag”
- **Flag:** a unique string that is formatted in a competition-specific manner. If acquired, it grants access to the desired asset, effectively certifying the success in the competition
- **Types** – three main modalities:
  - Attack/Defense ⚔
  - Jeopardy ⌖
  - King of the Hill ♔
- **Topics:** Binary, Crypto, Forensics, Hardware, Miscellaneous, Networking, Web…

GOAL OF THE PROJECT ✎

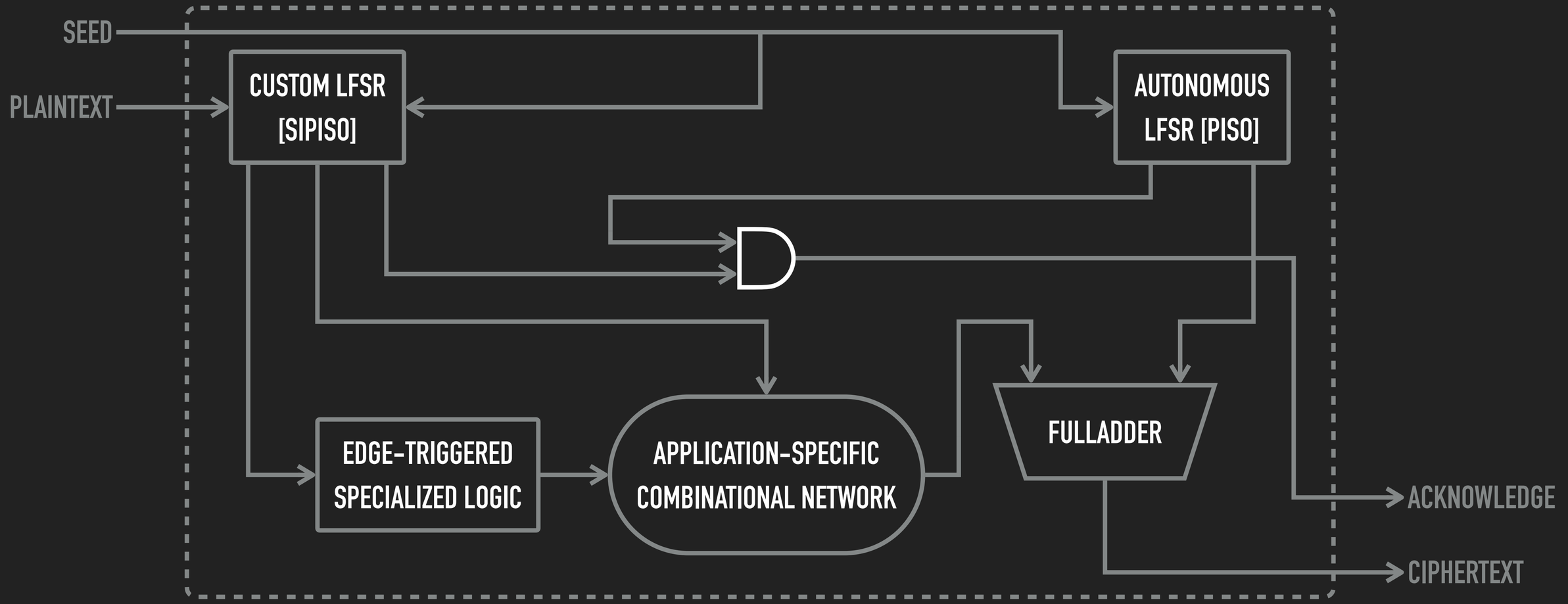
- Creating a brand new CTF challenge based on a target embedded platform, featuring the following characteristics:
  - Style – jeopardy
  - Theme – hardware
  - Design – 256-bit block cipher
  - Connectivity – public IP (+ port)
  - Interaction – online webpage.

CHOSEN HARDWARE ARCHITECTURE ⌘

- PYNQ-Z1 Python Productivity board equipped with a Zynq-7000 XC7Z020-1CLG400C SoC:
  - 650 MHz dual-core Cortex-A9 processor
  - 512MB DDR3 with 16-bit bus @ 1050 Mbps
  - High-bandwidth peripheral controllers: 1 Gigabit Ethernet, USB 2.0, SDIO
  - Low-bandwidth peripheral controllers: SPI, UART, CAN, I2C
  - Remotely programmable via JTAG/Quad-SPI flash/microSD.



TOP-LEVEL DESIGN ▸ BLOCK CIPHER ▣



LINKING FPGA ⇔ MCU ⚀

- Data exchange with the FPGA itself is done through the AXI4 peripheral, which includes a flexible amount of registers directly accessible from the external world
- The top-level component has to be instantiated inside said device in order to create and package an application-specific IP
- The result of this process is a bitstream – a collection of configuration files used to describe all the logic needed to actually implement the VHDL design into hardware
- The Overlay Python library subsequently handles the interface with the MCU, exposing all the desired functionalities in the form of easily accessible APIs.

REMOTE COMMUNICATION WITH THE BOARD ⏲

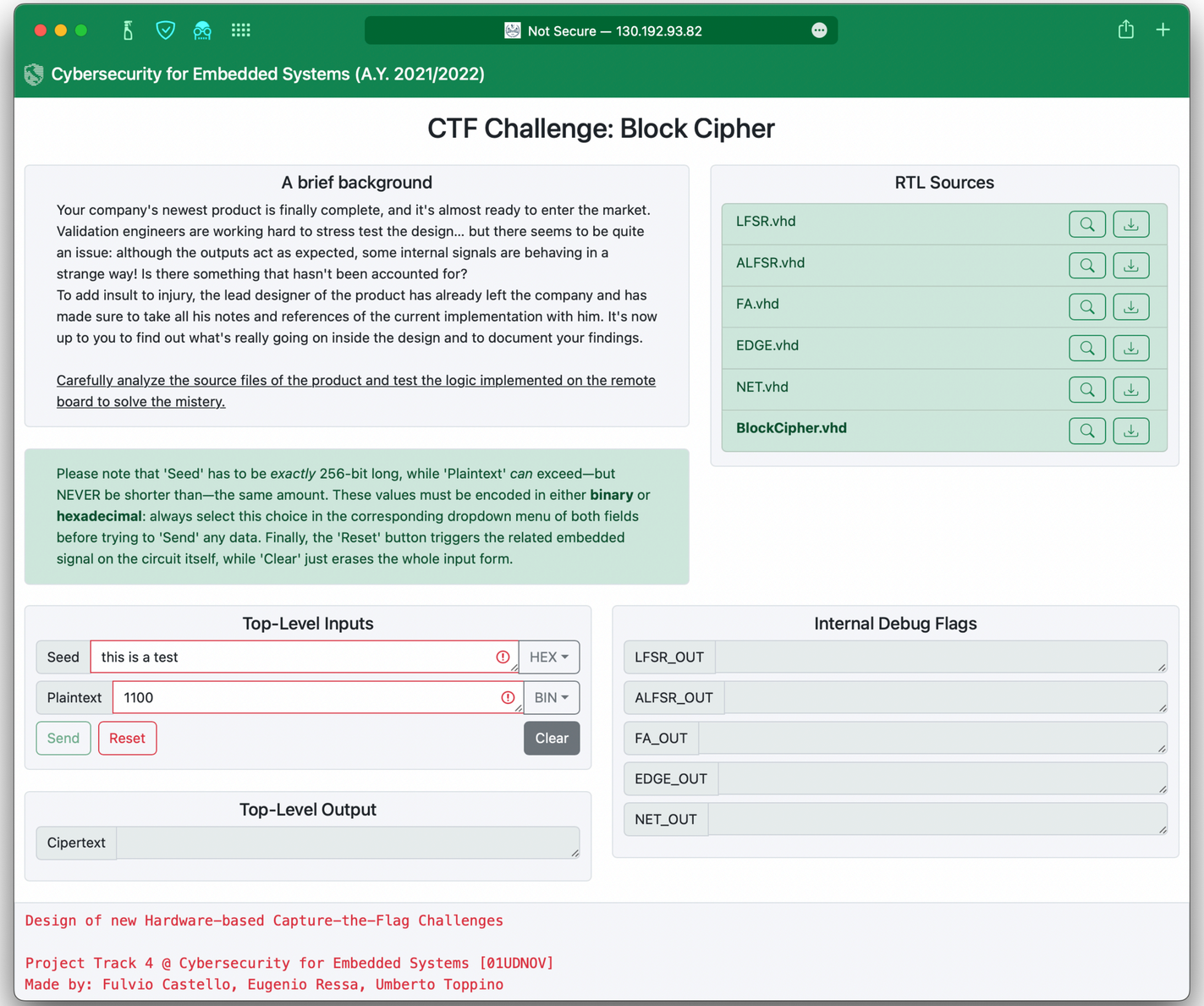
- The server side of the challenge is powered by CherryPy, an object-oriented Python web framework
- All the methods used to link FPGA & MCU are thereby exposed as publicly accessible static URLs
- The contestant can thus send/receive data by means of standard HTTP requests, directly interfacing with the AXI4 peripheral registers from remote
- Several front-end packets made of HTML/CSS/JS code are then returned, so that it becomes possible to interact with the challenge itself by means of an intuitive GUI.

CLIENT-SIDE GUI

- The Graphical User Interface (GUI) is a webpage accessible to the competitors by means of any popular browser. It includes four main sections:
  - Challenge Description – a brief narrative that serves as a background for the CTF challenge itself, for the sake of player immersion
  - Source Files – a list of links used to preview and/or download all the components that make up the inspected VHDL design
  - Input Form – a number of text fields that allow the player to send data to different input ports of the top-level entity fitted onto the board
  - Output Forms – a group of read-only dynamic labels containing both intermediate and final results of the functionality implemented into the FPGA.

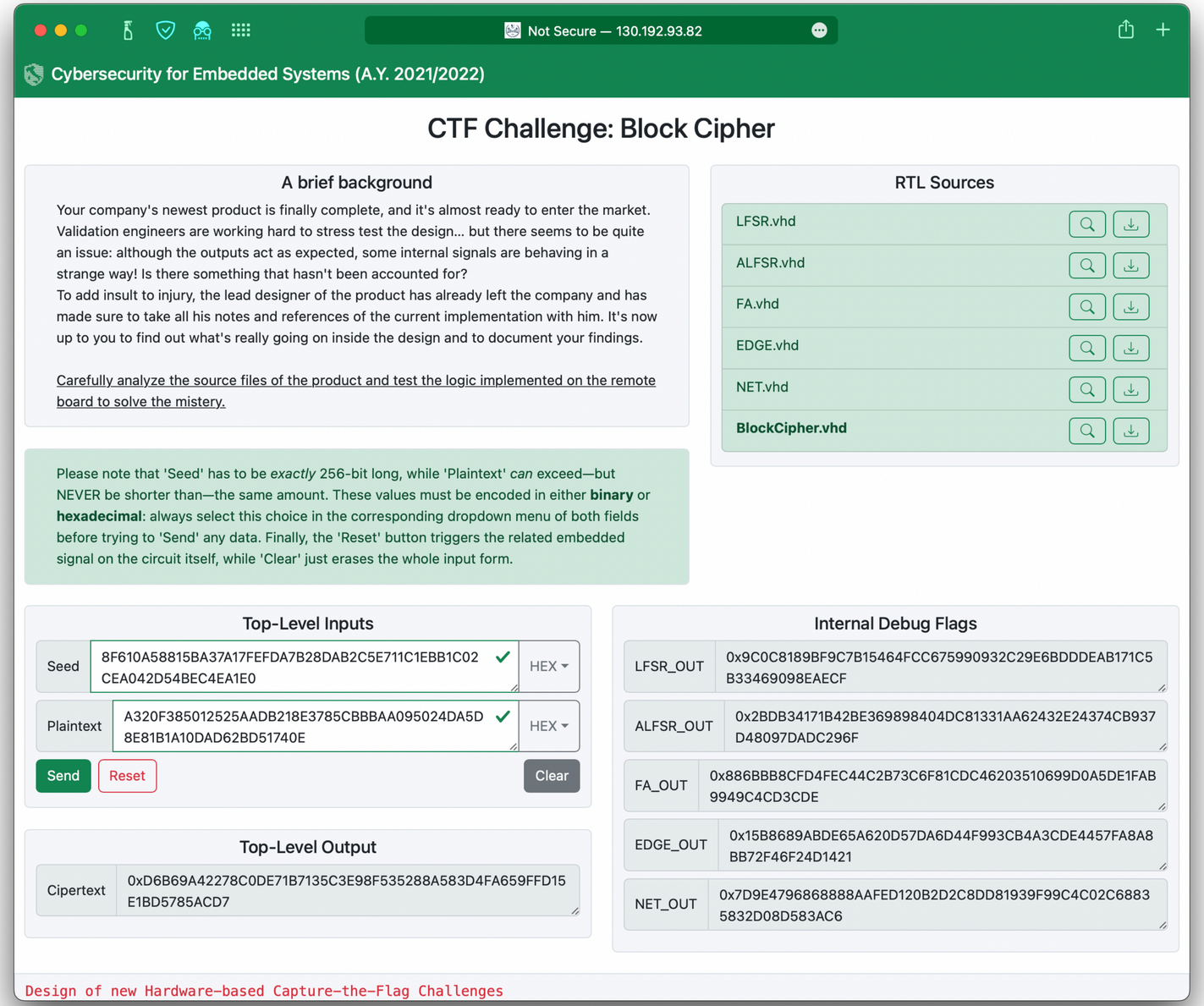
DEMO ☑

- Seed == 256 bit
- Plaintext ≥ 256 bit
- Ciphertext length is a consequence of the previous field
- Binary & HEX are both supported.



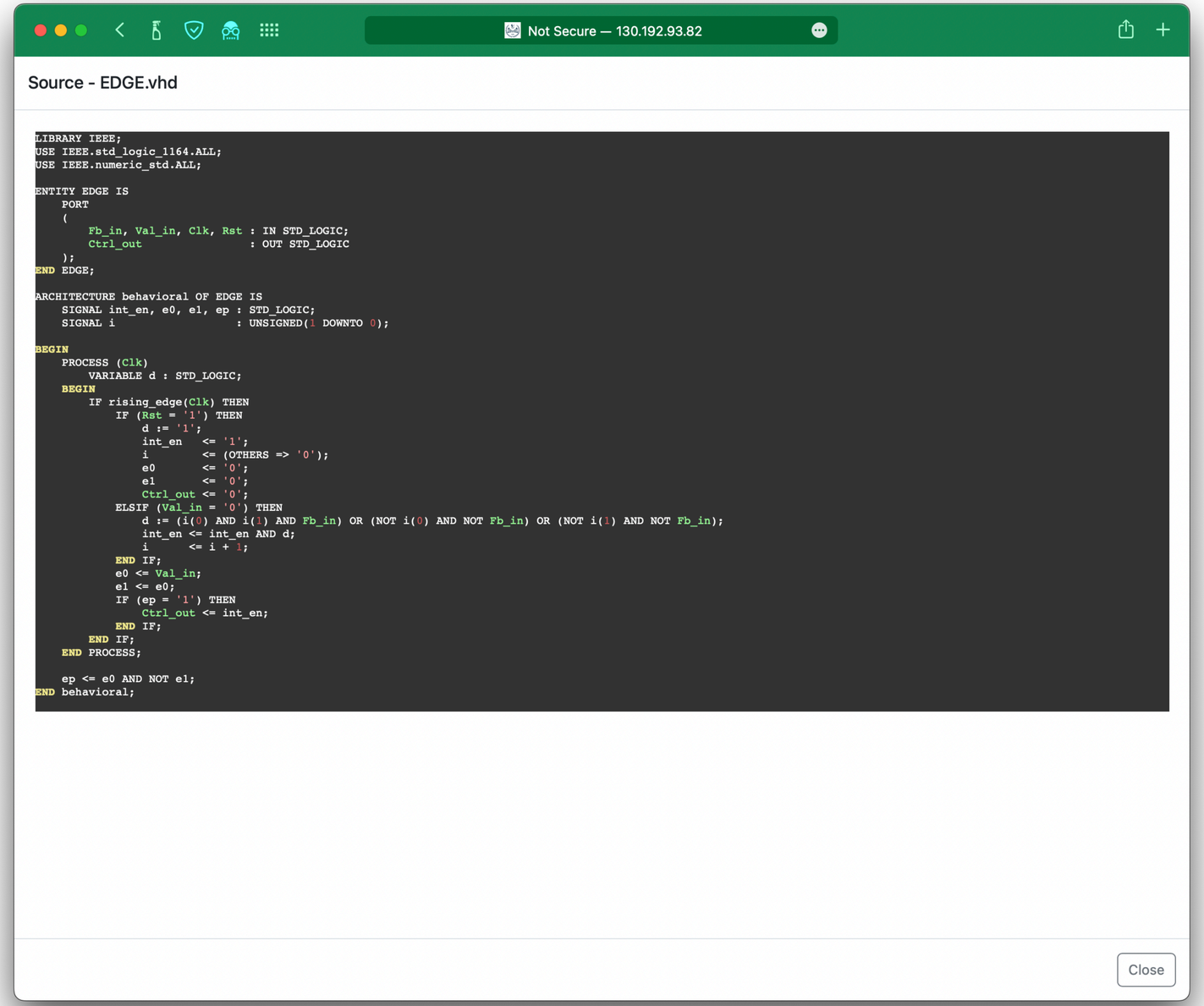
DEMO ☑

- Connect to 130.192.93.82 @ port 8080
- ‘Send’: issue, await, fetch information
- ‘Reset’: trigger the related signal on the board
- ‘Clear’: erase all available user inputs.



DEMO ☑

- Preview VHDL netlists inside the browser
- Download source files to the local system
- Well-formatted, easy to read code…
- …but (hopefully) hard to unriddle!



“AND THAT’S A WRAP! THANK YOU ALL FOR YOUR ATTENTION.”

Project Track 4