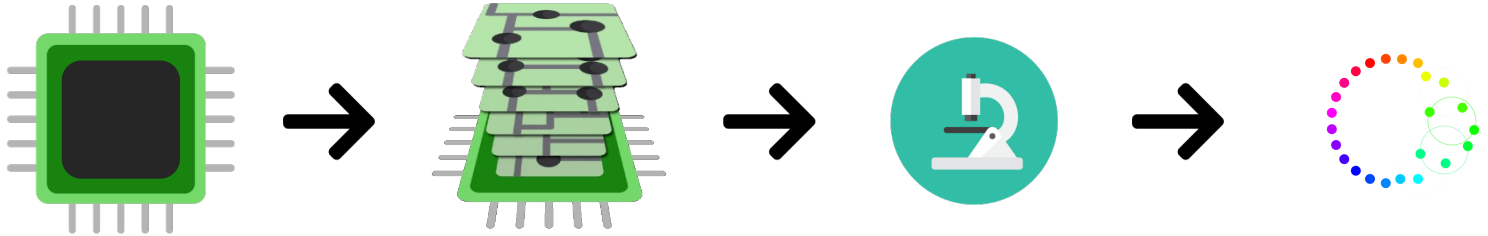
MAX PLANCK INSTITUTE
FOR SECURITY AND PRIVACY

HAL – A Modular Framework for Netlist Reverse Engineering

Julian Speith
MPI-SP, Bochum, Germany

Hardware Reverse Engineering

ASICs



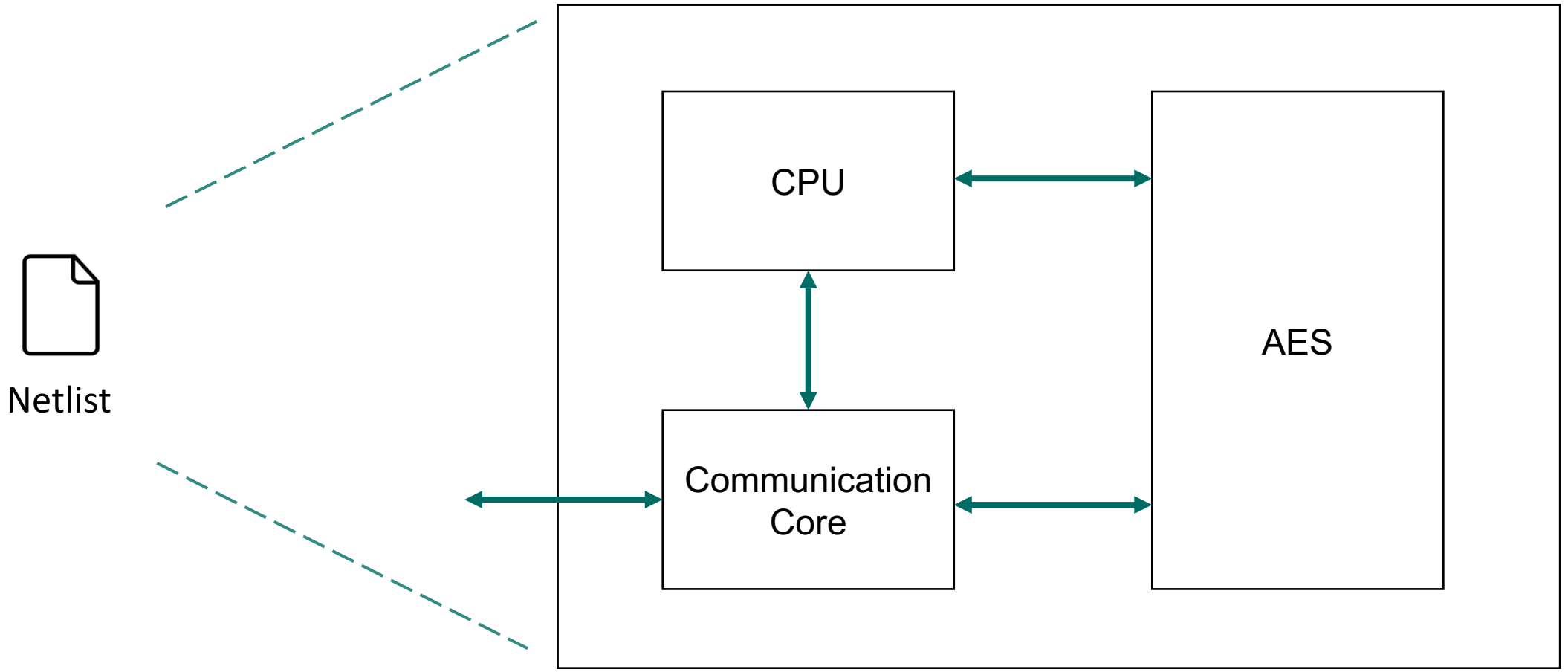
Decapsulation → Delayering → Imaging → Image Processing

FPGAs

Bitstream Extraction → Bitstream Conversion → Netlist → Netlist Analysis

NETLIST

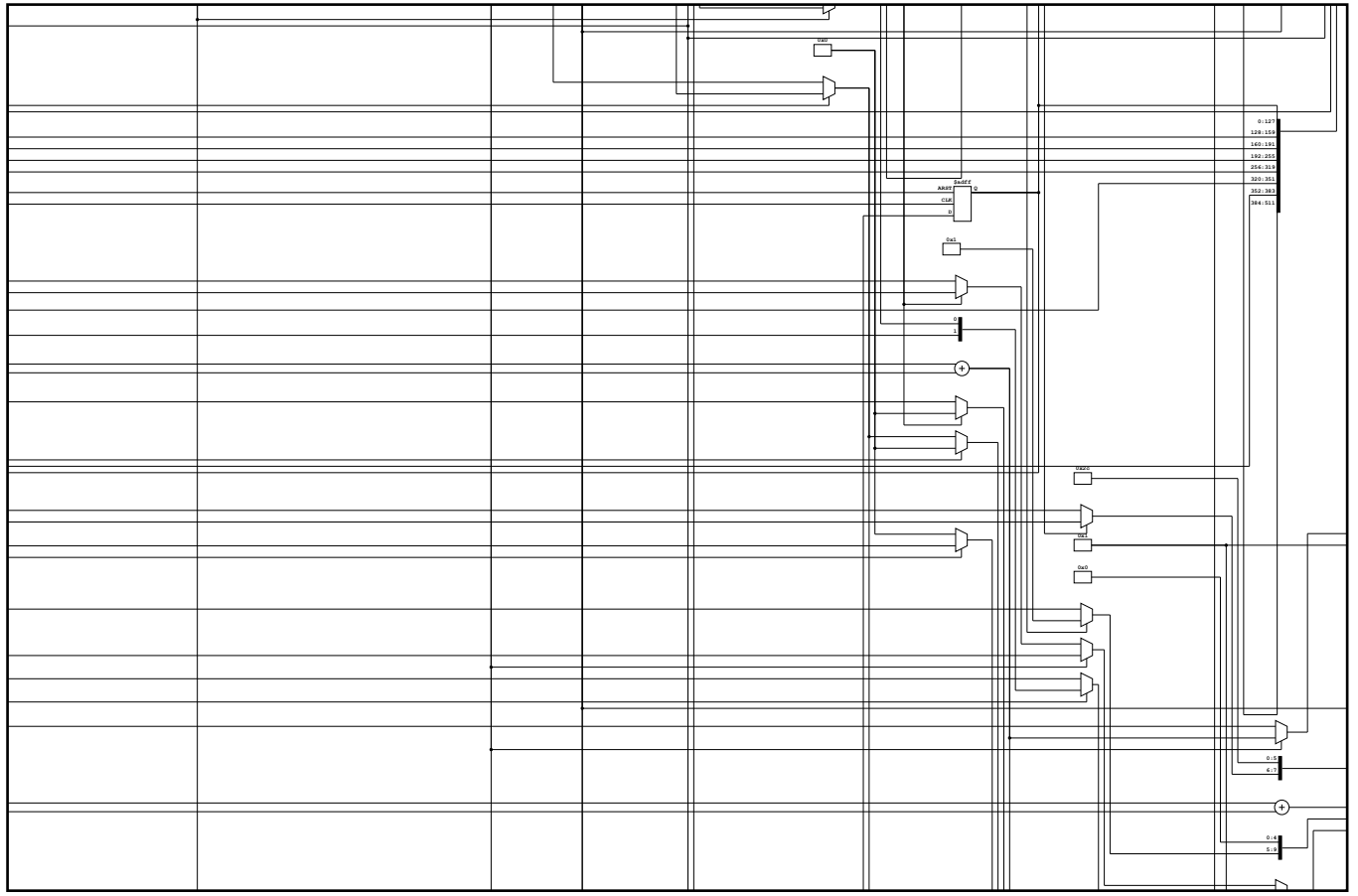
How you might think it looks like…



NETLIST

How it really looks like…

Netlist

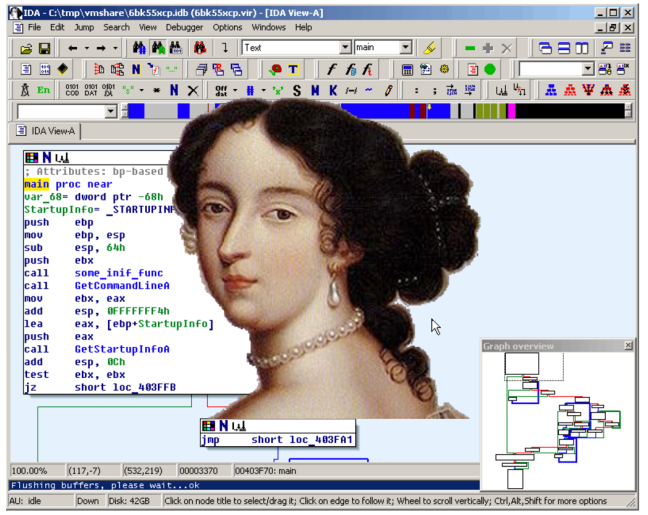


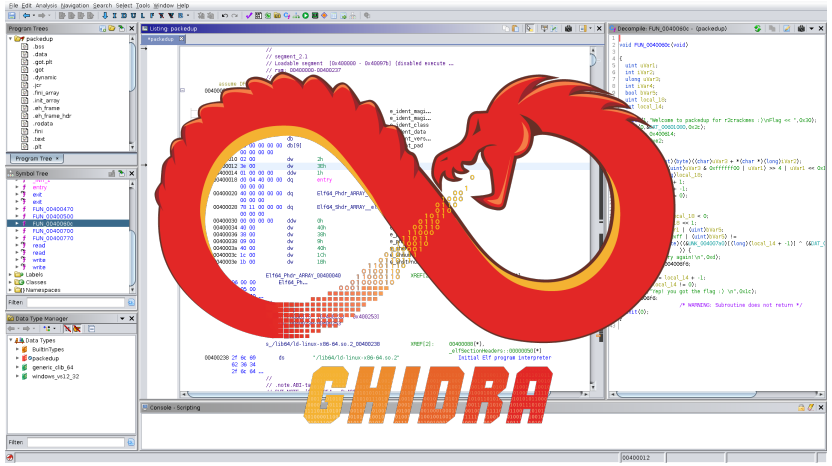
Software Reverse Engineering

Software RE has a very active community throughout industry and academia

- Popular tools: GHIDRA and IDA Pro
  - Advanced frameworks for binary analysis
  - Modularity through plugins
- Open-source software (GHIDRA) available

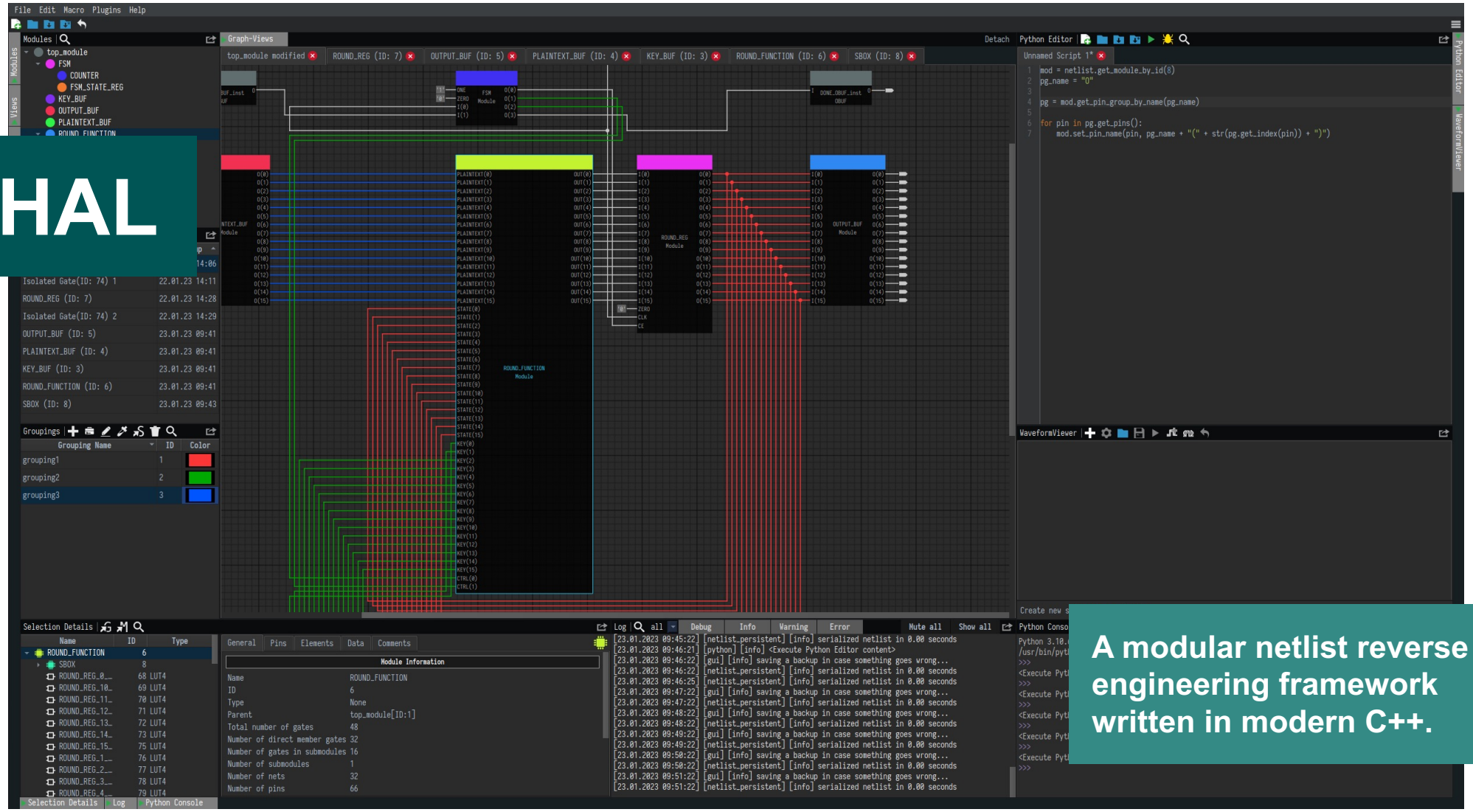
The situation for Hardware RE is different …





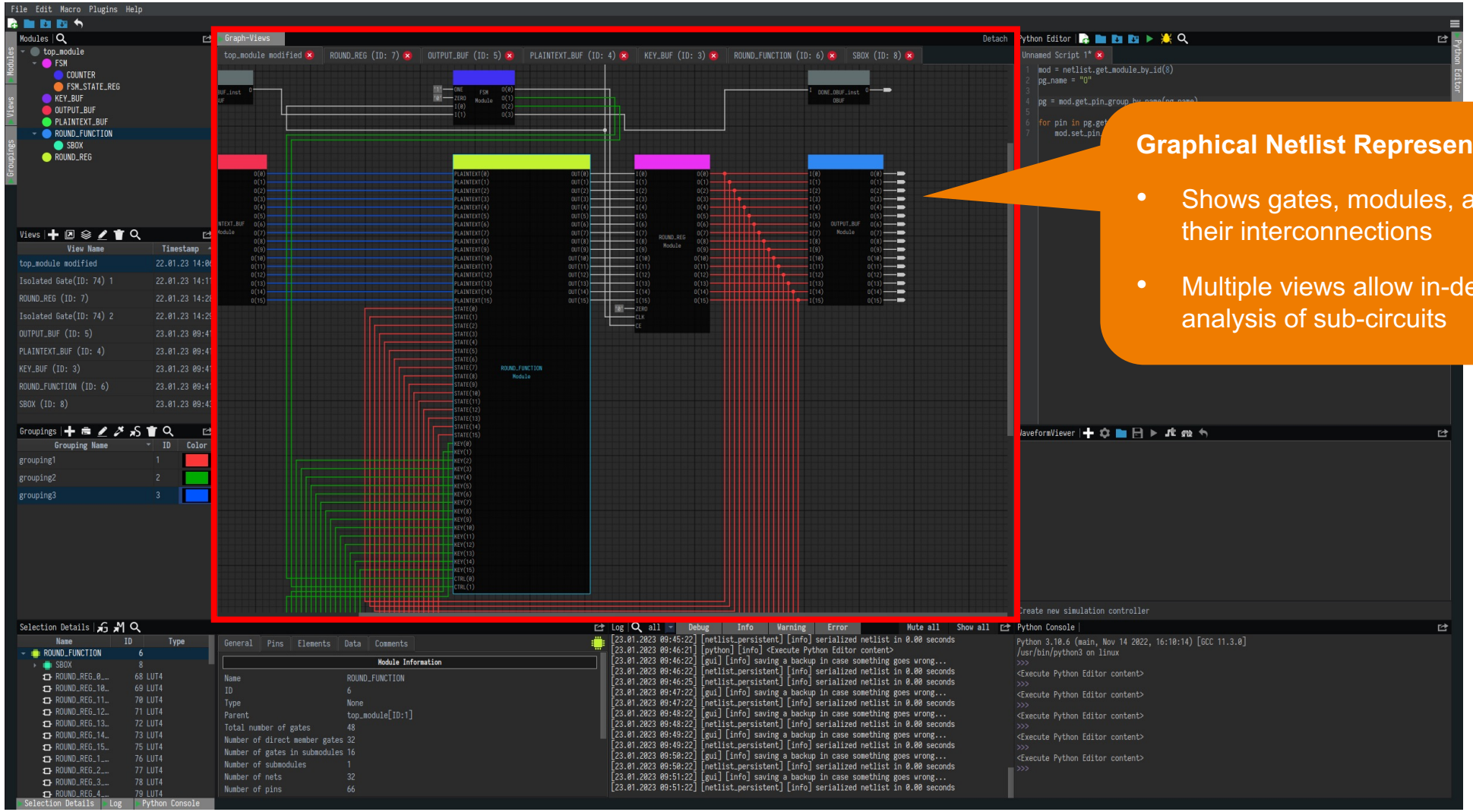

WANTED
DEAD OR ALIVE
IDA PRO
FOR
HARDWARE RE

HAL



A modular netlist reverse engineering framework written in modern C++.

HAL – The Hardware Analyzer



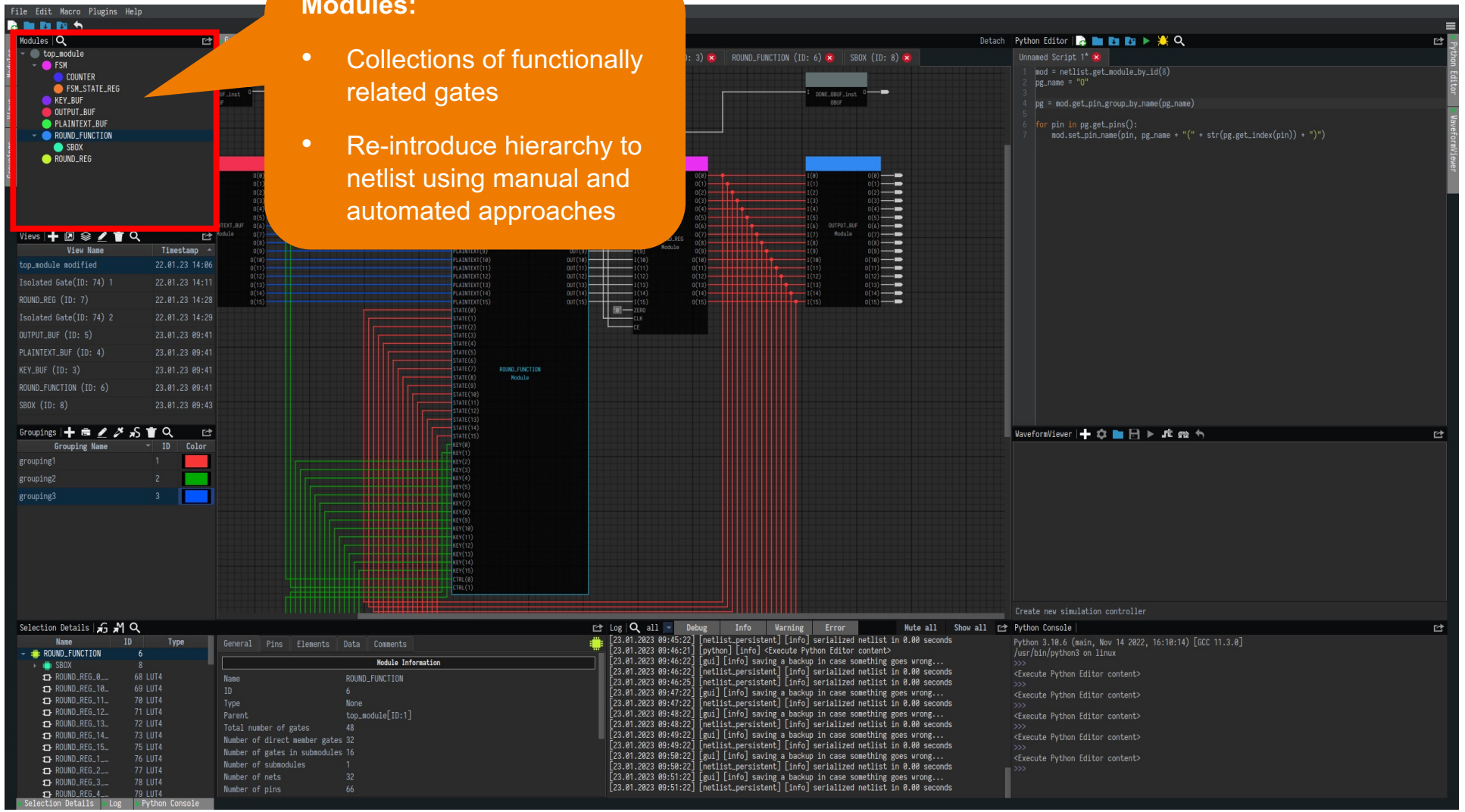
Graphical Netlist Representation:

- Shows gates, modules, and their interconnections
- Multiple views allow in-depth analysis of sub-circuits

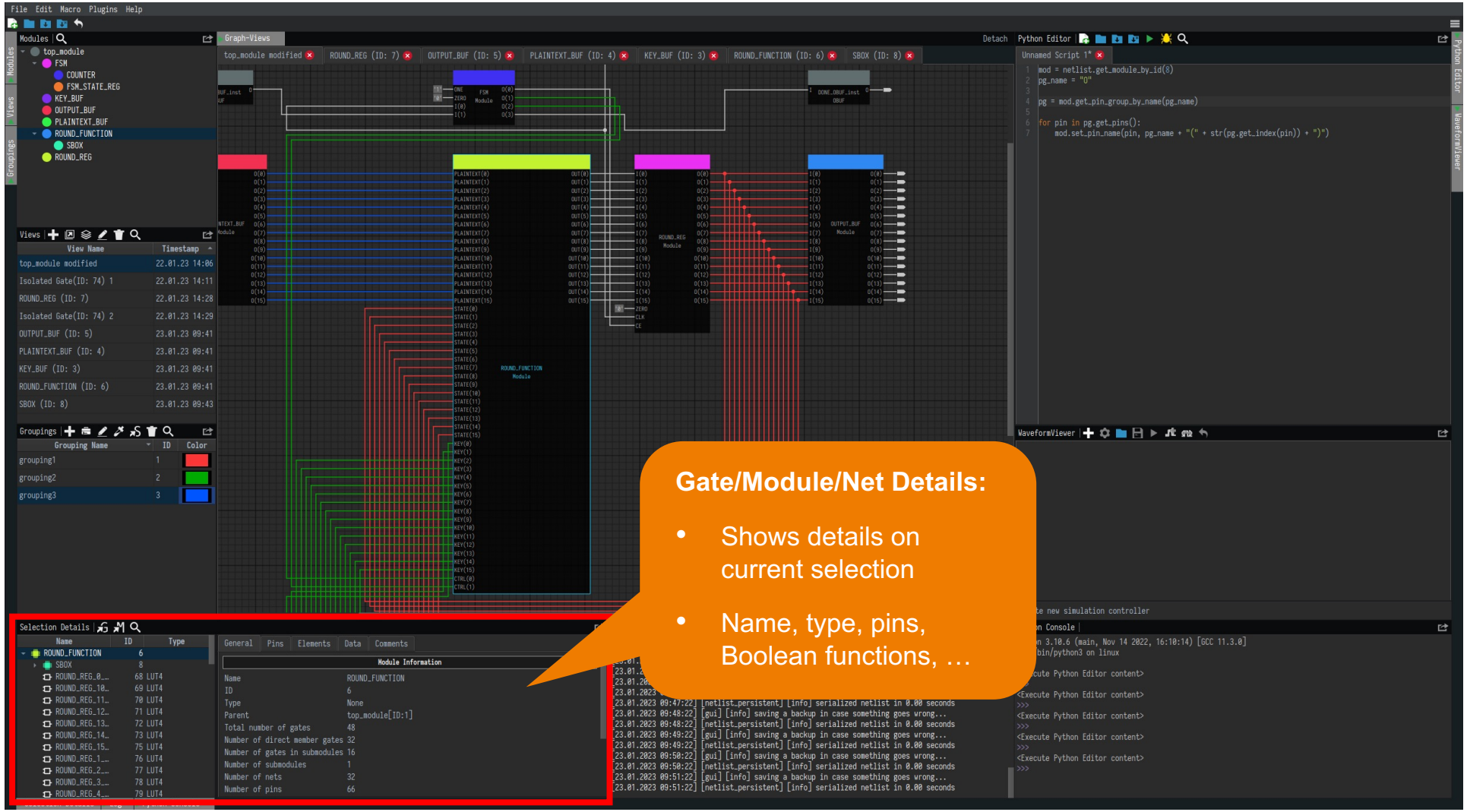
HAL – The Hardware Analyzer

Modules:

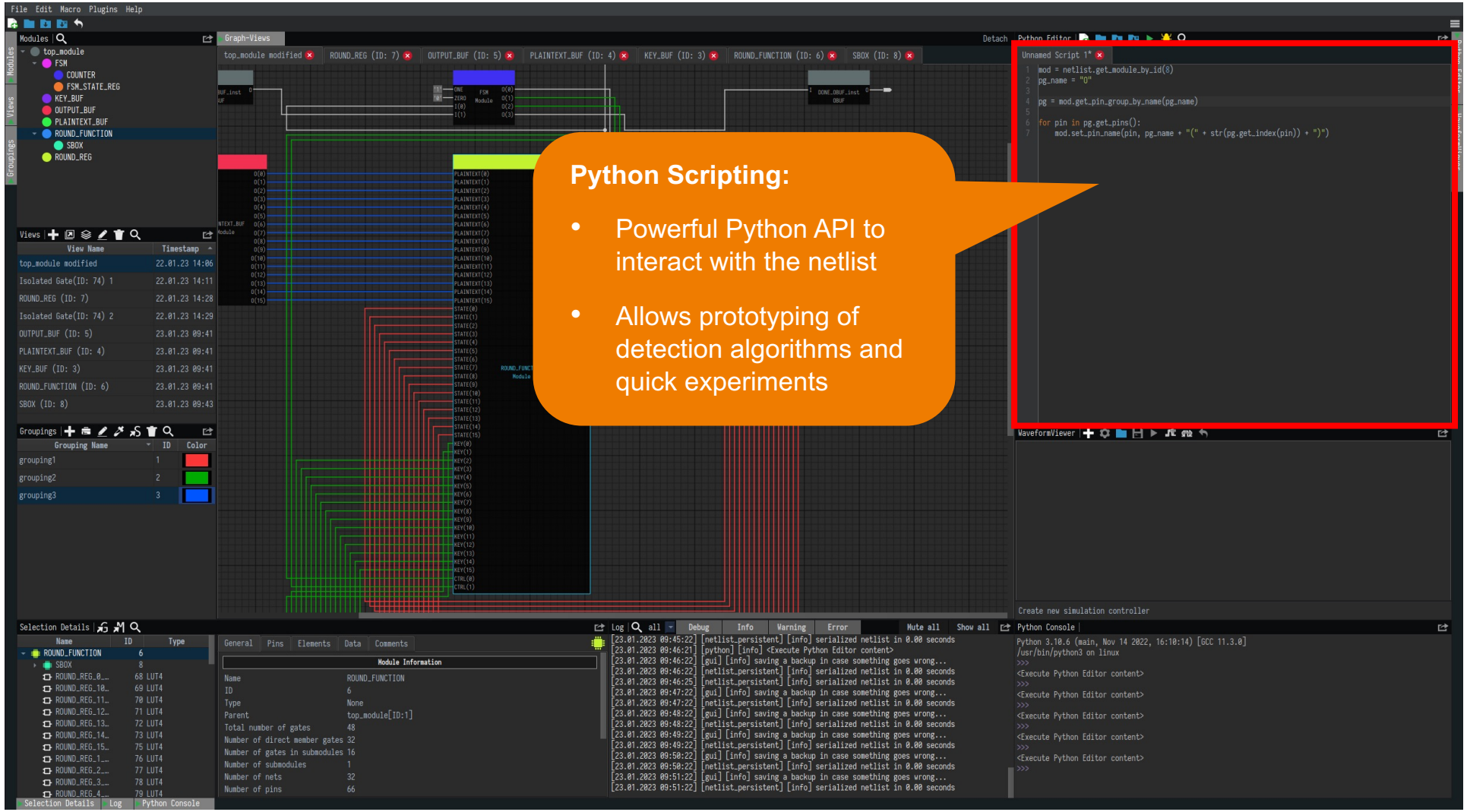
- Collections of functionally related gates
- Re-introduce hierarchy to netlist using manual and automated approaches



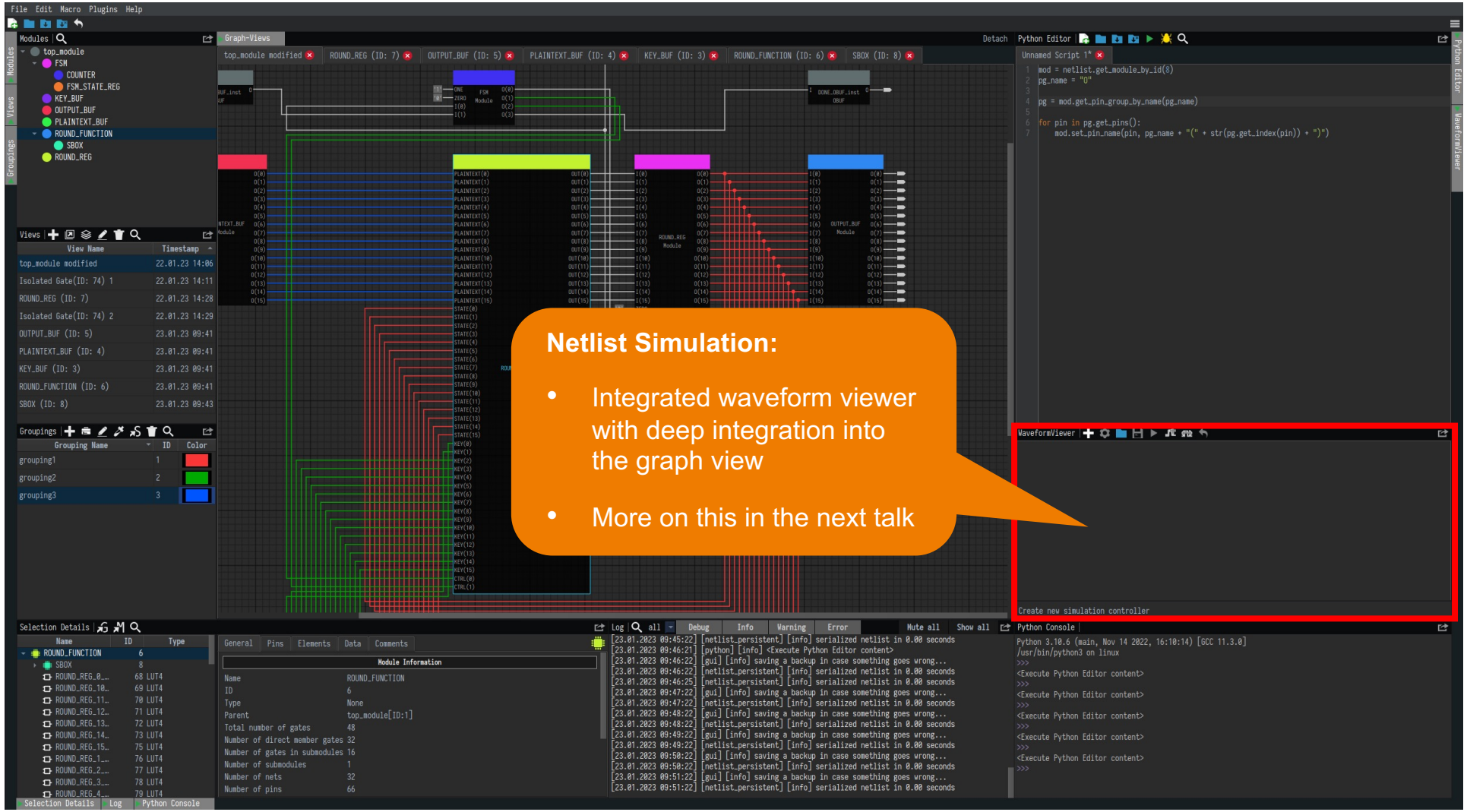
HAL – The Hardware Analyzer



HAL – The Hardware Analyzer



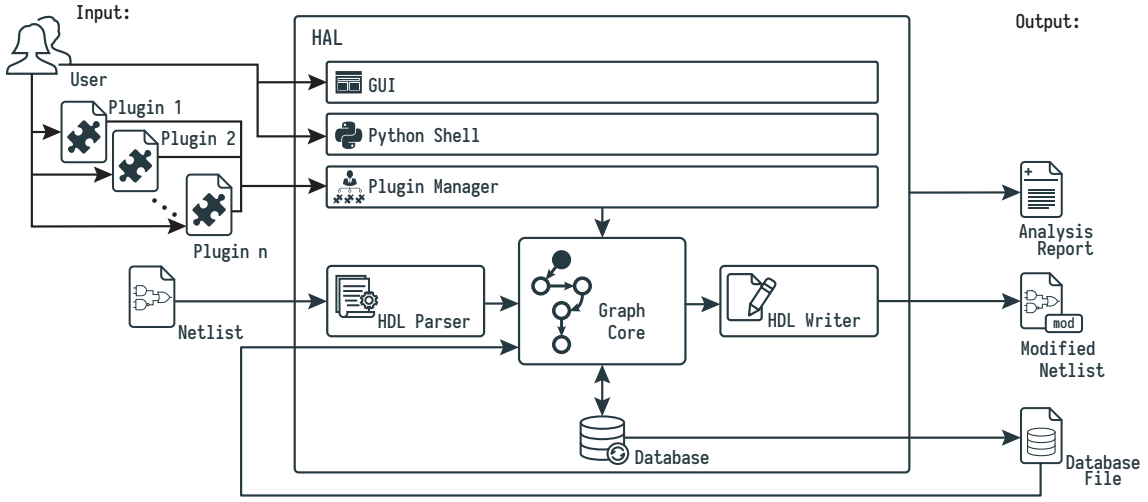
HAL – The Hardware Analyzer



Under the Hood – Core

Provides Functionalities for Recurring Tasks

- Creation and management of HAL projects
- Netlist and gate library parsing and writing
- Netlist interaction and manipulation
  - Interpretation of the netlist as a graph
  - Allows easy traversal of the netlist
  - Enables execution of graph algorithms
- Boolean function handling and evaluation
- Module creation and management



Under the Hood – Plugin System

C++ API allows to customize and extend HAL functionality

- The HAL core can be extended according to your requirements using C++ plugins
- Allows you to keep plugins private while still running an up-to-date HAL
- Enables interacting with the netlist, running automated detection algorithms, …
- (Experimental) GUI support for plugin functionality
  - Add your plugin functions to context menus and the toolbar
- **Already available:** Dataflow analysis (DANA), Simulator, Pre-Processing, Parsers & Writers, …

Under the Hood – Boolean Functions

Out-of-the-Box Support for Multi-Bit Boolean Functions

- Allows for functional analysis of word-level combinational sub-circuits
- Utility functions to get Boolean functions of sub-graphs
- Interpretation of LUT configurations and translation to Boolean functions
- Support for
  - Arithemtic operations: AND, OR, XOR, NOT, ADD, SUB, MUL …
  - Comparison operations: EQ, ULT, SLT, ULE, SLE
  - Slice, Concat, zero extension, sign extension

Under the Hood – SMT Solving

Native Support for SMT Solving

- Allow to check functional equivalence of Boolean functions
  - **Example:** verify whether a multi-bit function implements an addition
- HAL API enables interfacing with arbitrary SMT solvers
  - Supported by default: z3, Boolector, Bitwuzla
- No need for conversion of Boolean functions, **HAL takes care**

What HAL Can and Cannot Do for You

Framework to provide functionality to analyze netlists

- Parsing of netlists and gate libraries
- Core features provides functions to interact with netlist
- Boolean function support (including SMT/SAT integration)
- Simulation support
- Plugin system to automate certain tasks

HAL is not a one-click solution

- HAL provides functions to analyze netlist, but does not do the analysis for you

MAX PLANCK INSTITUTE
FOR SECURITY AND PRIVACY

Any Questions?