HeLLO CTF ‘22
Tutorial on Hardware (Sequential) Obfuscation

By accessing the package, you agree not to redistribute any of the components of this package.

Welcome to the **HeLLO CTF ‘22** competition. This document provides a tutorial on how you can use the resources we have provided to see the transformation of a design once it gets obfuscated through sequential obfuscation scheme.

The **warm-up package** contains the followings:

- example_obf.v
  - Netlist of the obfuscated design
- example_org.v
  - Netlist of the original design
- example_obf_unlock_tb.v
  - Test bench for loading the obfuscated design, reading inputs from "example_unlock_inputs.txt", inputting these to the design, and producing/recording the outputs in "example_obfuscated_outputs.txt"
- example_org_tb.v
  - Test bench for loading the original design, reading inputs from "example_original_inputs.txt", inputting these to the design, and producing/recording the outputs in "example_orignal_outputs.txt"
- example_original_inputs.txt
  - A file containing a sequence of 10000 inputs that are loaded by the test bench
- example_unlock_inputs.txt
  - A file containing 10016 inputs, where the first 16 inputs are used to unlock the obfuscated design, and the remaining 10000 inputs are the functional inputs
- internal_lib.v
  - This is a verilog description of the standard cells used in the netlist. This is a proprietary, copyrighted document. **Please do not redistribute** this file as it is made available to you only for this competition.
- example_original_outputs.txt
  - Simulation-generated outputs of the original design (note, running a new simulation will replace this file)
- example_obfuscated_outputs.txt
  - Simulation-generated outputs of the obfuscated design (note, running a new simulation will replace this file)
- keyVectors.txt
  - A file containing the sequence of inputs used to unlock the obfuscated design
- keyPortMapping.txt
  - A file that represents how the bits in a line from keyVectors.txt correspond to the input ports of the design
- obf_summary.txt
  - Limited details/statistics for the obfuscation process

Details:

- “example_org.v” is the oracle netlist. Its testbench, “example_org_tb.v” can be found that instantiates the module (toy) from the “example_org.v” and reads in the 10000 input patterns listed in “example_original_inputs.txt” file. The input patterns are mapped to primary inputs following the mapping order found in the “keyPortMapping.txt” file. A file “example_original_outputs.txt” file is being generated by simulating the oracle using the testbench which has 10000 responses being recorded against 10000 input patterns.

- “example_unlock_inputs.txt” file has (16+10000) patterns where the first 16 patterns are the key patterns which matches with the patterns from “keyVectors.txt” file (except the ‘x’s have been replaced by constants). Rest of the 10000 patterns are the same as in the “example_original_inputs.txt” file.

- The locked circuit “example_obf.v” is simulated using the “example_obf_unlock_tb.v” testbench. This testbench reads in the key patterns followed by the 10000 random patterns - both listed in “example_unlock_inputs.txt” file and generates “example_obfuscated_outputs.txt” file which consists (16+10000) responses from the (un)locked circuit.

- By escaping the first 16 responses from “example_obfuscated_outputs.txt” file, the number of responses can be reduced to 10000 and both “example_original_outputs.txt” and “example_obfuscated_outputs.txt” contain the exact same patterns meaning the correct key has been applied. A simple script (not included) can give you the matching and/or non-matching patterns. You can use the command line program "diff" for this.

- If incorrect key patterns are inserted or no key patterns have been inserted at all, the simulation will result in mismatching outputs between the oracle and locked circuits.

Example simulation script:
Synopsys VCS can be used to simulate designs by invoking the following UNIX commands in terminal (ensuring all the files are in current working directory):

```
$ vcs -full64 example_org.v example_org_tb.v -v internal_lib.v -R
```

One can also use Icarus Verilog:
```
$ iverilog example_org_tb.v example_org.v internal_lib.v -o runme
$ ./runme
```

Tasks
This tutorial is designed to help you become more familiar with the netlists you will receive in the challenge round. NO POINTS WILL BE AWADED FOR EVALUATION OF THE WARM-UP DESIGN. However, you are more than welcome to report the insight on your analysis.

Good luck!