DPI Protected Verilog Instead of Encryption

A non-broken and open source friendly alternative to IEEE 1735

Todd Strader
@ ORConf 2019
IP is everywhere

- Modular hardware components
- Licensed by a third party
- Term clearly coined by lawyers
- IEEE P1735 standardizes RTL encryption
Open source tools can’t play
Possible solutions:

- Use closed source simulator
- $€$
- Emulate functionality
- Gate-level sim
- Negotiate for source
- Give up
What if we could compile the secret Verilog and use it via a standardized interface?
Usual Verilator flow
Building the library

VERILATOR

C++

secret. sv

secret. cpp

GCC

secret. a
Using the library

SystemVerilog

secret.sv

secret.a

VERILATOR

Big EDA

[Simulation waveform]

04100120 049160+ 0580+ 052+
08d040048 012458+ 1160+
Let’s add it all up

- Portable across DPI-capable simulators
- Including open source simulators
- Compiled library is pretty indecipherable
- No possibility of leaked RTL
- Stable API via the DPI
- Could lead to fewer trusted keys
Is everything fine?

- Everything is not fine
Top-level parameters

- Verilator requires fixed top-level parameters
- Possible solutions:
  - Don’t do that
  - Build libraries on-demand
  - Convert parameters to wires?
  - Dynamically construct hierarchy?
Build matrix

- OS
- Machine architecture
- C++ ABI
- Static or shared library
Try it out

$ git clone -b protect-lib https://github.com/toddstrader/verilator-dev.git
$ # build Verilator
$ make -C examples/dpi_protect_lib/
$ test_regress/t/t_prot_lib.pl
$ test_regress/t/t_prot_lib.pl --xsim
Next steps

- Land upstream
- Larger tests
  - verilator_ext_tests
  - Benchmarking
- Test more commercial simulators
- Support x’s and z’s
- Isolate Verilator runtime
- Performance optimizations
- Better obfuscation
- VCD replay
- Support top-level parameters
Unless
Further reading

- [https://acmccs.github.io/papers/p1533-chhotarayA.pdf](https://acmccs.github.io/papers/p1533-chhotarayA.pdf)