The EPFL Logic Synthesis Libraries

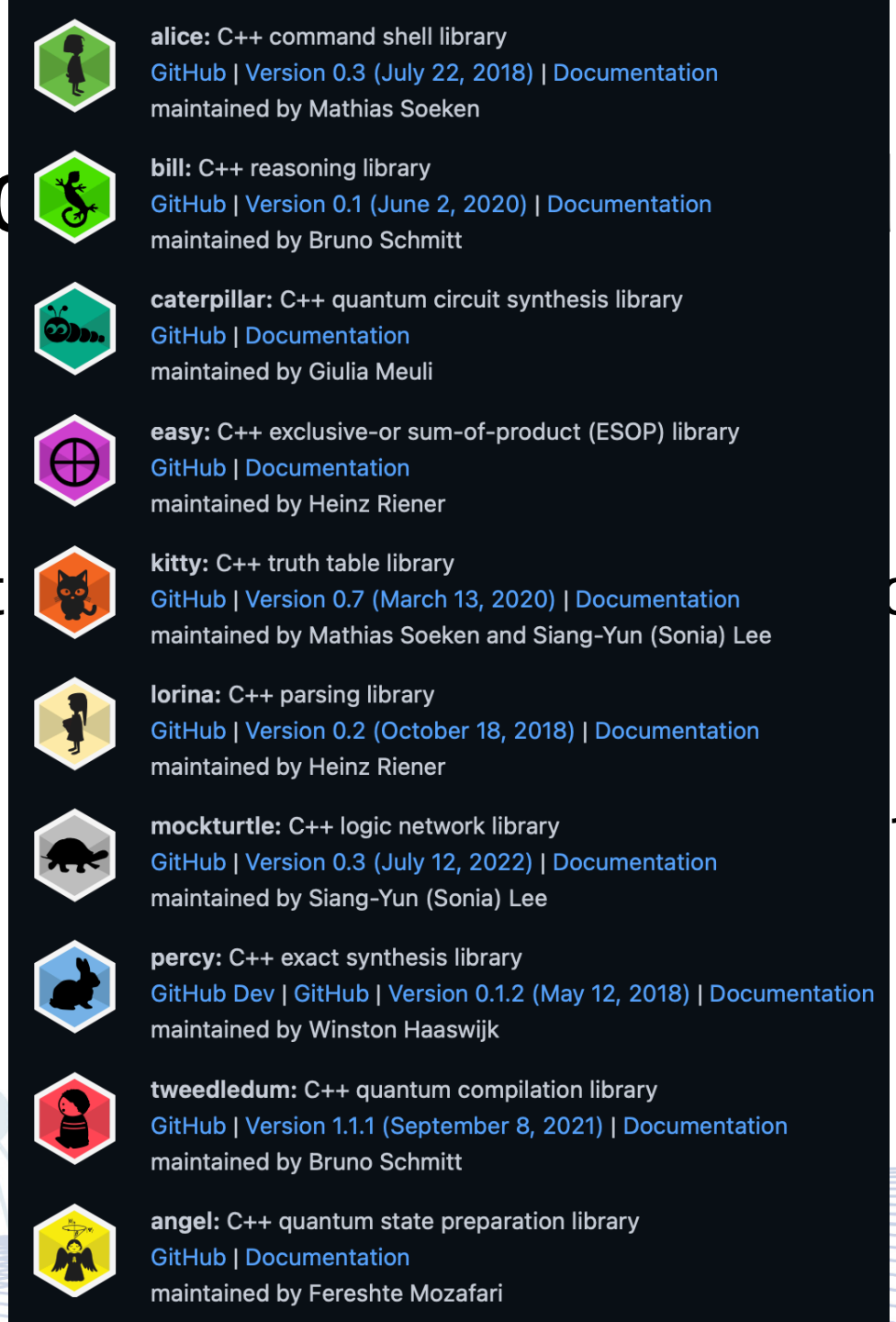
Siang-Yun (Sonia) Lee, EPFL, Switzerland

Birds-of-a-Feather Meeting @ DAC'22

https://github.com/lsils/lstools-showcase

Focus: tec ent logic synthesis

- AIGs, MIGs,
- Truth tables
- Basic infrast ommand shell (`alice`)
- Quantum → , `angel`



alice: C++ command shell library
GitHub | Version 0.3 (July 22, 2018) | Documentation
maintained by Mathias Soeken

bill: C++ reasoning library
GitHub | Version 0.1 (June 2, 2020) | Documentation
maintained by Bruno Schmitt

caterpillar: C++ quantum circuit synthesis library
GitHub | Documentation
maintained by Giulia Meuli

easy: C++ exclusive-or sum-of-product (ESOP) library
GitHub | Documentation
maintained by Heinz Riener

kitty: C++ truth table library
GitHub | Version 0.7 (March 13, 2020) | Documentation
maintained by Mathias Soeken and Siang-Yun (Sonia) Lee

lorina: C++ parsing library
GitHub | Version 0.2 (October 18, 2018) | Documentation
maintained by Heinz Riener

mockturtle: C++ logic network library
GitHub | Version 0.3 (July 12, 2022) | Documentation
maintained by Siang-Yun (Sonia) Lee

percy: C++ exact synthesis library
GitHub Dev | GitHub | Version 0.1.2 (May 12, 2018) | Documentation
maintained by Winston Haaswijk

tweedledum: C++ quantum compilation library
GitHub | Version 1.1.1 (September 8, 2021) | Documentation
maintained by Bruno Schmitt

angel: C++ quantum state preparation library
GitHub | Documentation
maintained by Fereshte Mozafari

https://github.com/lsils/lstools-showcase

Updates

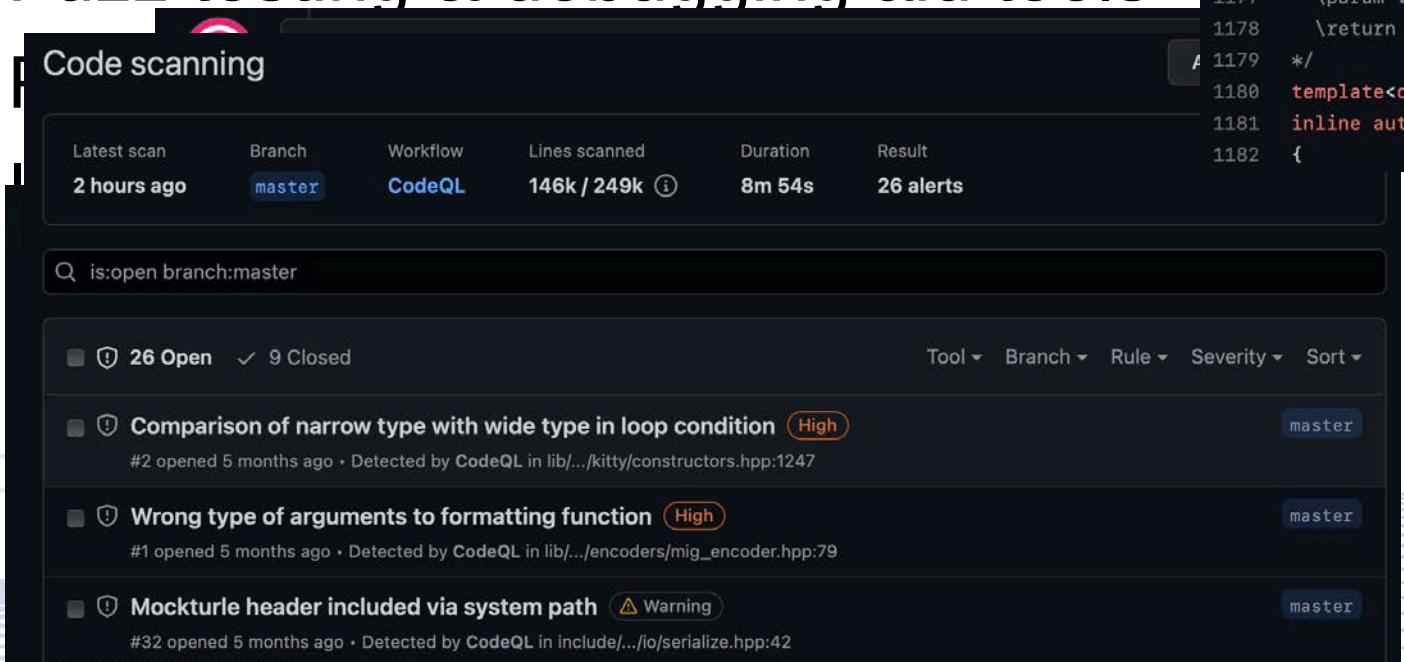
- Updated version of the arXiv paper
- New in `mockturtle` v0.3
  - Partial simulation
  - Versatile technology mapping
  - Superconducting electronic (SCE)-specific algorithms
- New library `angel` for quantum state preparation

arXiv:1805.05121v3

https://github.com/lsils/lstools-showcase

Development & maintainance aids

- Continuous integration (cross-platform build check)
- Code test coverage report
- Code quality analysis
- Fuzz testing & debugging aid tools
-
-




```
1168  /*! \brief Composes a truth table.
1169
1170    Given a function `f`, and a set of truth tables as arguments, computes the
1171    composed truth table.  For example, if `f(x1, x2) = 1001` and
1172    `vars = {x1 = 1001, x2= 1010}`, the function returns 1000. This function can
1173    be regarded as a general operator with arity `vars.size()` where the behavior
1174    of the operator is given by f.
1175
1176    \param f The outer function
1177    \param vars The ordered set of input variables
1178    \return The composed truth table with vars.size() variables
1179  */
1180  template<class TTf, class TTv, typename = std::enable_if_t<is_complete_truth_table<TTf>::value>>
1181  inline auto compose_truth_table( const TTf& f, const std::vector<TTv>& vars )
1182  {
```

```
19    fuzzer(gen, fuzz_ps);
20  bool has_bug = fuzzer.run(opt);
21
22  if (!has_bug) return 0;
23
24  testcase_minimizer_params min_ps;
25  min_ps.file_format = testcase_minimizer_params::aiger;
26  min_ps.init_case = "fuzz";
27  min_ps.minimized_case = "fuzz_min";
28  testcase_minimizer<aig_network> minimizer(min_ps);
29  minimizer.run(opt);
30
31  aig_network aig;
32  lorina::read_aiger("fuzz_min.aig", aiger_reader(aig));
33  write_dot(aig, "fuzz_min.dot");
34  std::system("dot -Tpng -O fuzz_min.dot");
35
36  return 0;
37  }
```

Minimizer

wcase

Contributions are welcomed!

- Fixing existing issues
- Pull request of new features
- Bug reports
- Discussions/suggestions/collaborations …
- Wish list
  - Best MIG optimization flow
  - Parallelize existing algorithms
  - Dynamic visualization of networks
  - Interactive tool (`mockturtle` + `alice`)