2022
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
EUROPE
MUNICH, GERMANY
DECEMBER 6 - 7, 2022

The Cost Of Standard Verification Methodology Implementations

Adam Hizzey, Abigail Williams, Svetlomir Hristozkov

Graphcore Ltd, Bristol, UK

GRAPHCORE

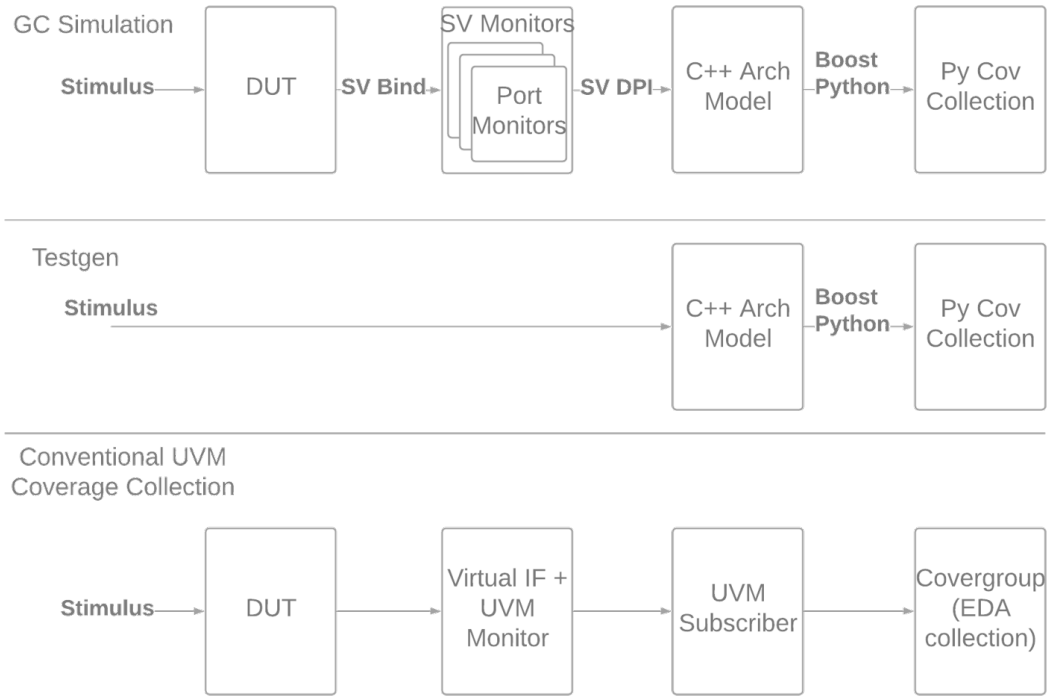
accellera
SYSTEMS INITIATIVE

Acknowledgement

Thanks to Abigail Williams for generating and analysing the data presented here

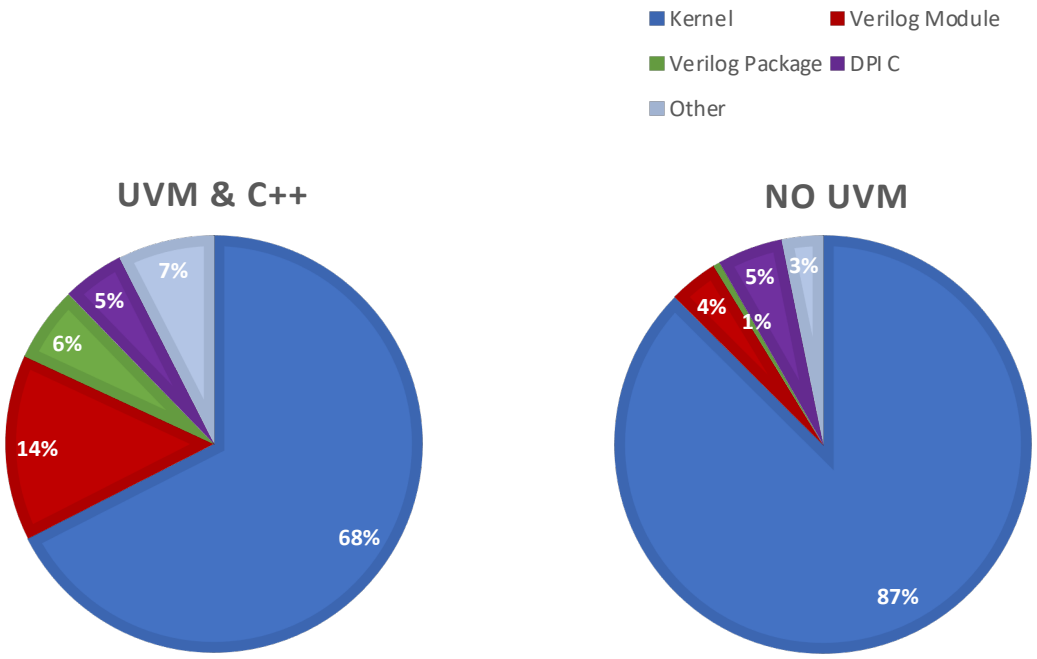
History

- SystemVerilog RTL with custom C++/Python verification methodology
  - Constrained random test generation
  - Metrics driven with coverage
  - CI/CD with >100,000 compute hours daily
- Some effort to optimise performance

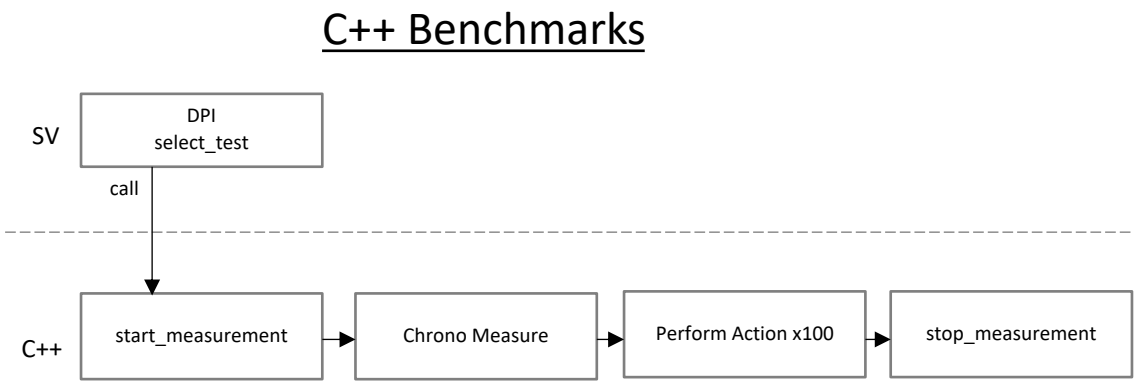


History

- SystemVerilog RTL with custom C++/Python verification methodology
  - Constrained random test generation
  - Metrics driven with coverage
  - CI/CD with >100,000 compute hours daily
- Some effort to optimise performance



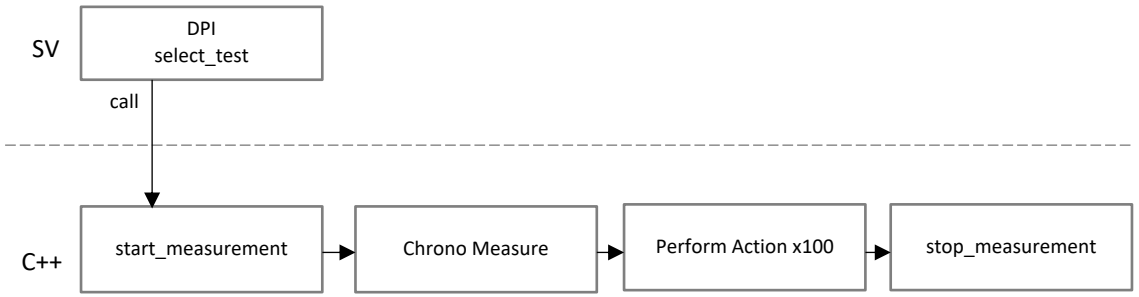
Benchmarks Introduction



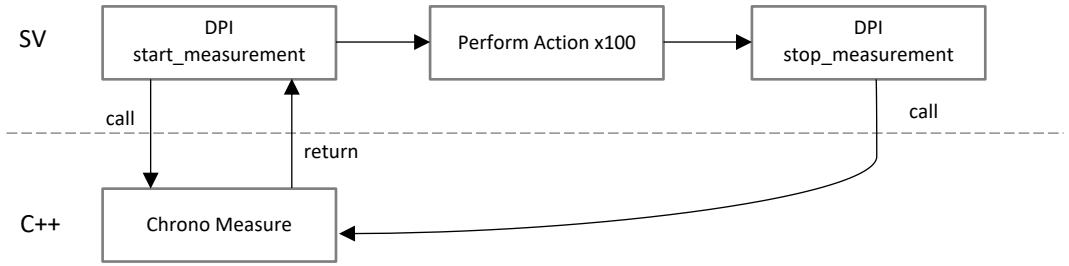
- Timing in C++ with <chrono>

Benchmarks Introduction

C++ Benchmarks



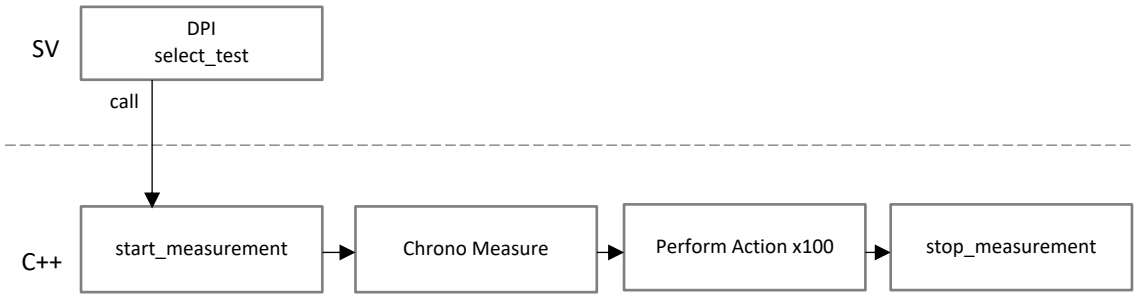
SystemVerilog Benchmarks



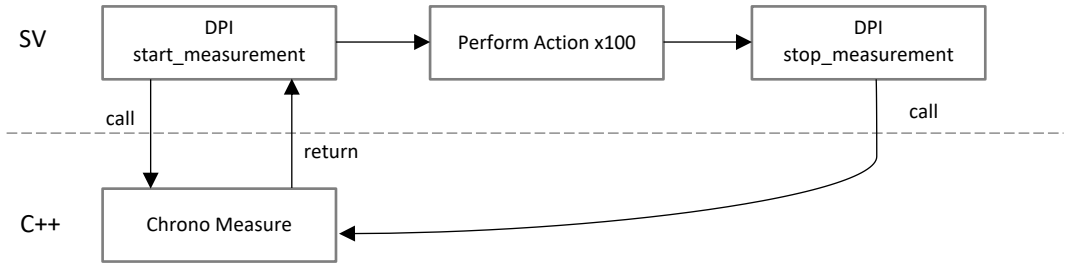
- Timing in C++ with <chrono>
- SystemVerilog tests have extra DPI overhead
  - Minimised by iterating 100 times per timer call

Benchmarks Introduction

C++ Benchmarks



SystemVerilog Benchmarks

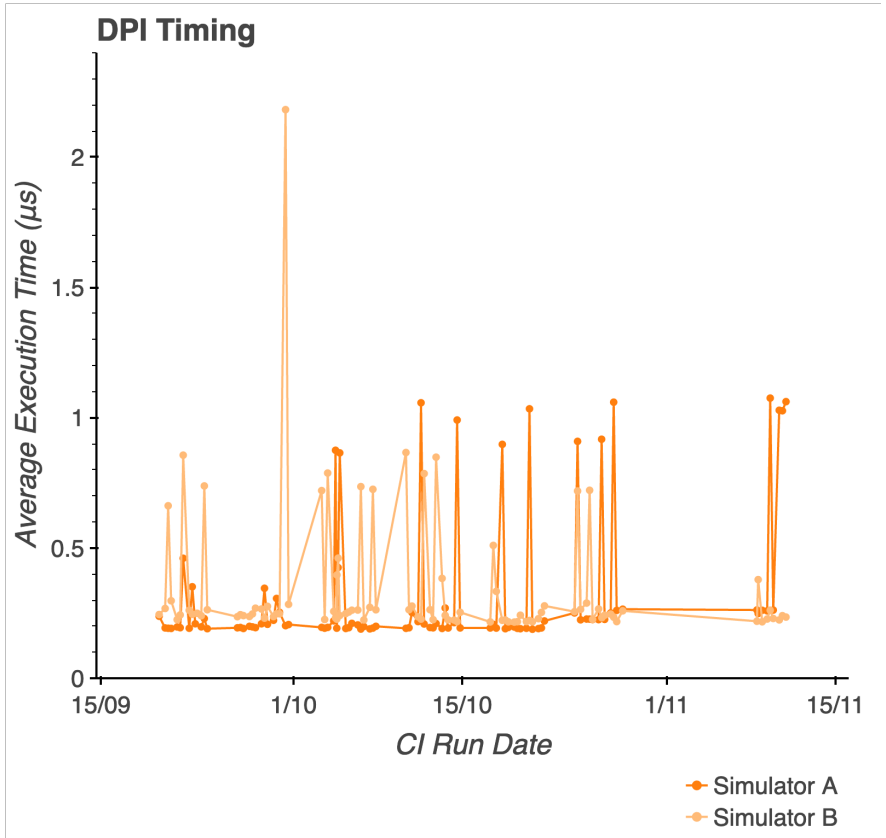


- Timing in C++ with <chrono>
- SystemVerilog tests have extra DPI overhead
  - Minimised by iterating 100 times per timer call
- Performance tests run in CI
  - Dedicated performance machines
  - Results stored in SQL database

Benchmarks Introduction

- “Microbenchmarks”
  - DPI calls of C++ functions from SystemVerilog
  - TLM modelling
    - Passing transactions between components
  - Functional coverage collection
  - String formatting & display
  - Constraint solvers & test generation [future work]
- Identify performance regressions for common tasks
- Not a substitute for profiling real testbenches

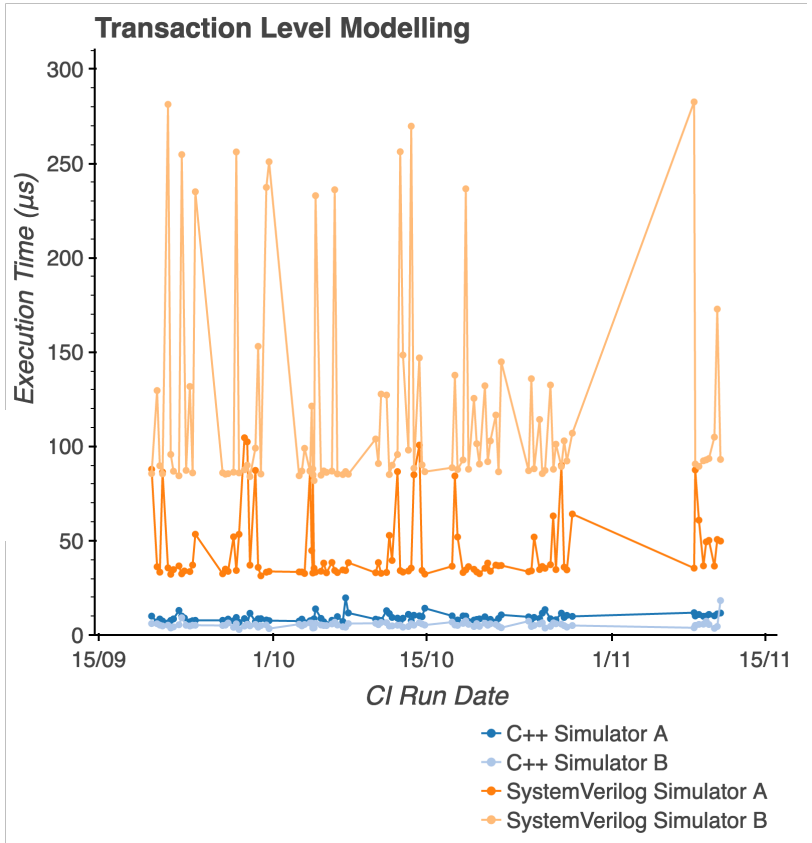
Data – DPI Overhead



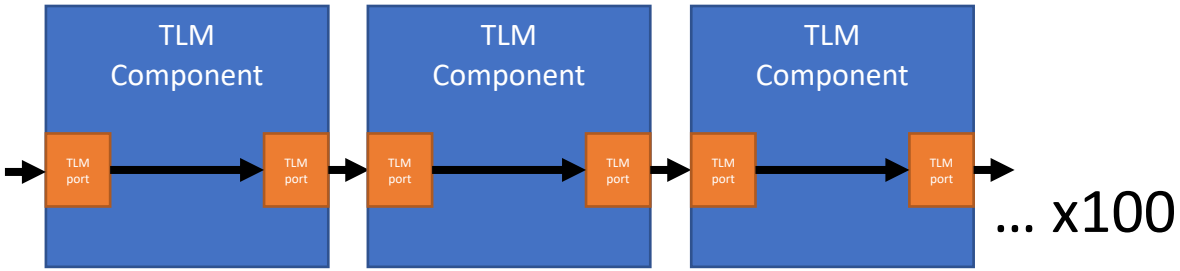
- DPI calls to start the timer then immediately stop it
- Calls from SystemVerilog to C++ will have this ~ 0.2µs overhead

```
// SystemVerilog
for (int j = 0; j < 100; j++) begin
  cpp_timing_lib_start_measurement();
  cpp_timing_lib_stop_measurement();
end
```

Data – TLM Ports



- Total time for a transaction to pass through a chain of 100 TLM components

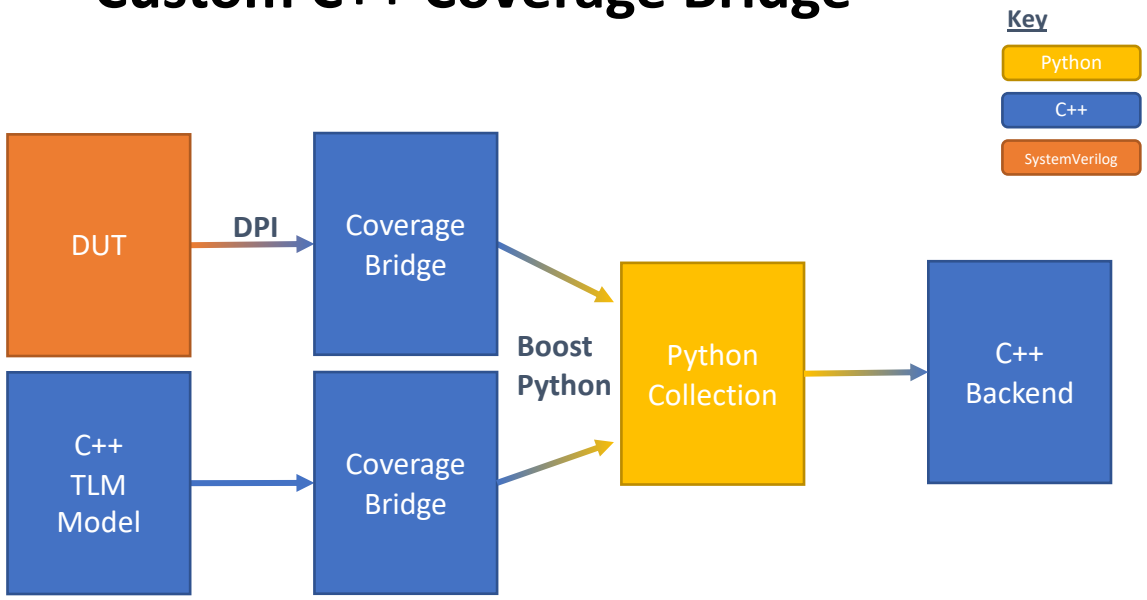


Data – Functional Coverage Collection

Standard SystemVerilog

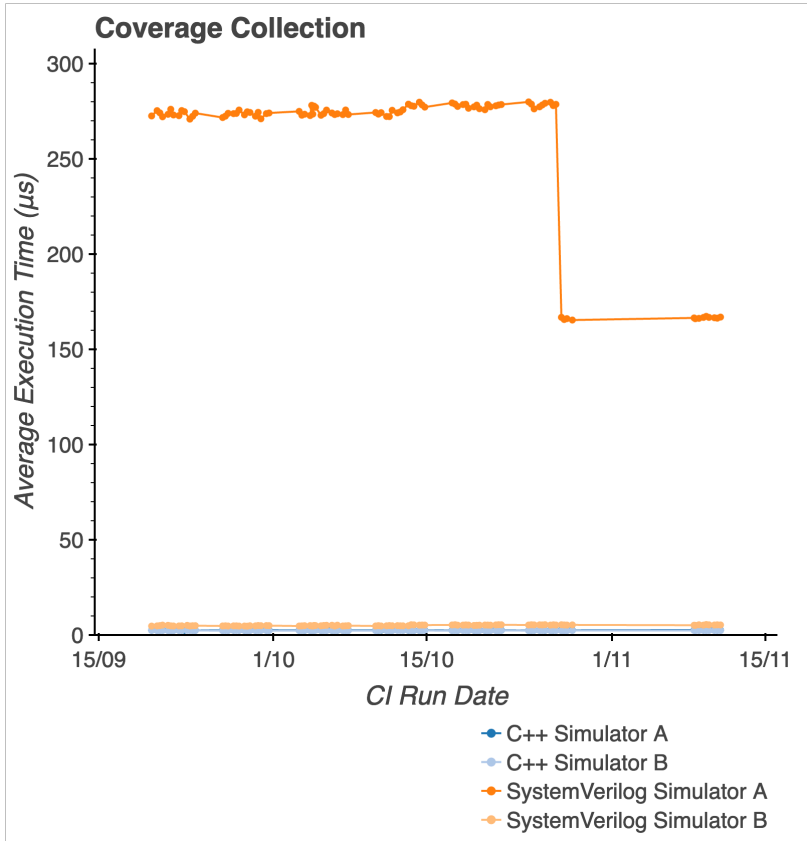
```
// SystemVerilog
// 256x256 bin cross

logic[15:0] my_var;

covergroup cg;
  first_cp : coverpoint my_var[15:8];
  second_cp : coverpoint my_var[7:0];
  firstXsecond: cross first_cp, second_cp;
endgroup
```

Custom C++ Coverage Bridge



See DVCon paper ***No Country For Old Men*** for more on coverage
https://dvcon-proceedings.org/document/no-country-for-old-men-a-modern-take-on-metrics-driven-verification/

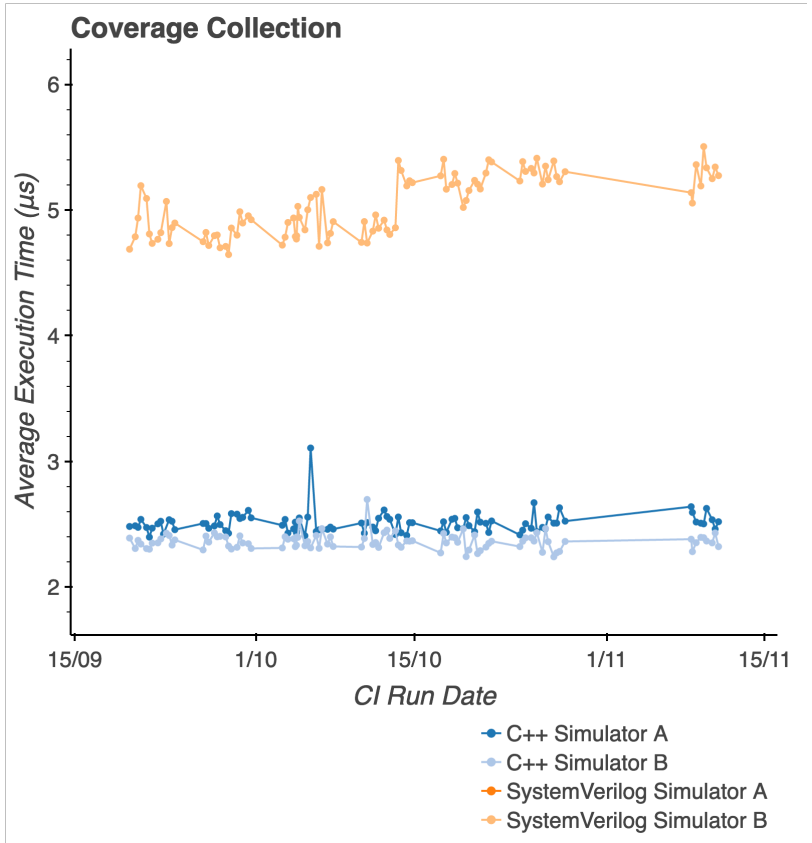
Data – Functional Coverage Collection



- Version upgrade of Simulator A on October 26th
  - Correlates with 40% performance boost 🚀

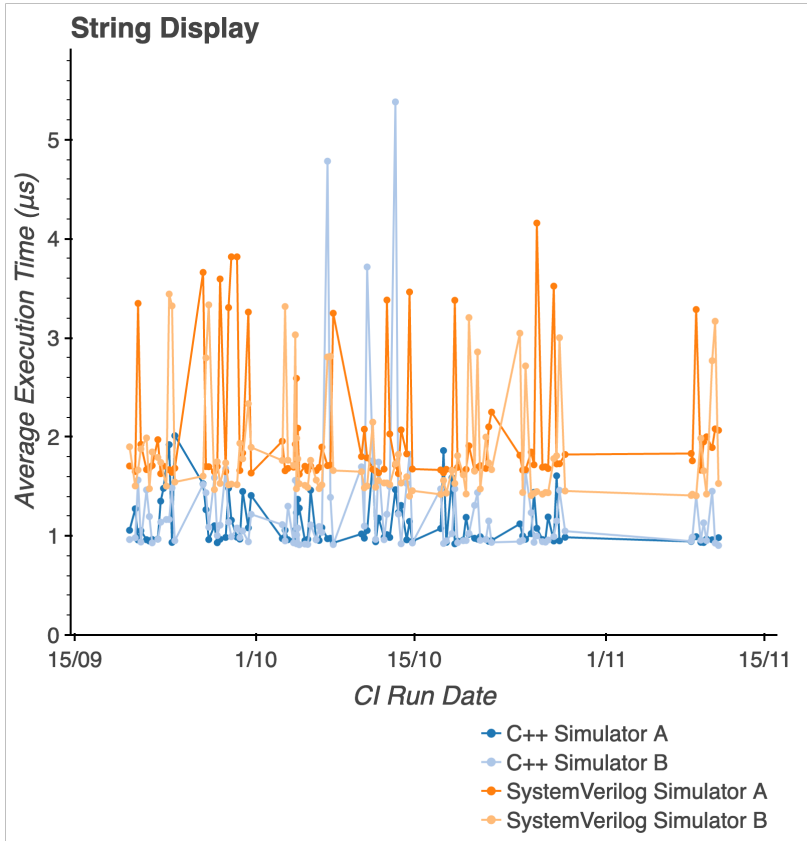
```
// SystemVerilog
// 256x256 bin cross

logic[15:0] my_var;

covergroup cg;
  first_cp : coverpoint my_var[15:8];
  second_cp : coverpoint my_var[7:0];
  firstXsecond: cross first_cp, second_cp;
endgroup
```

Data – Functional Coverage Collection



- Same plot as previous slide but with Simulator A removed
- Version upgrade of Simulator B on October 12th
  - Possibly correlates with slight regression in performance
  - Tools don't always get faster

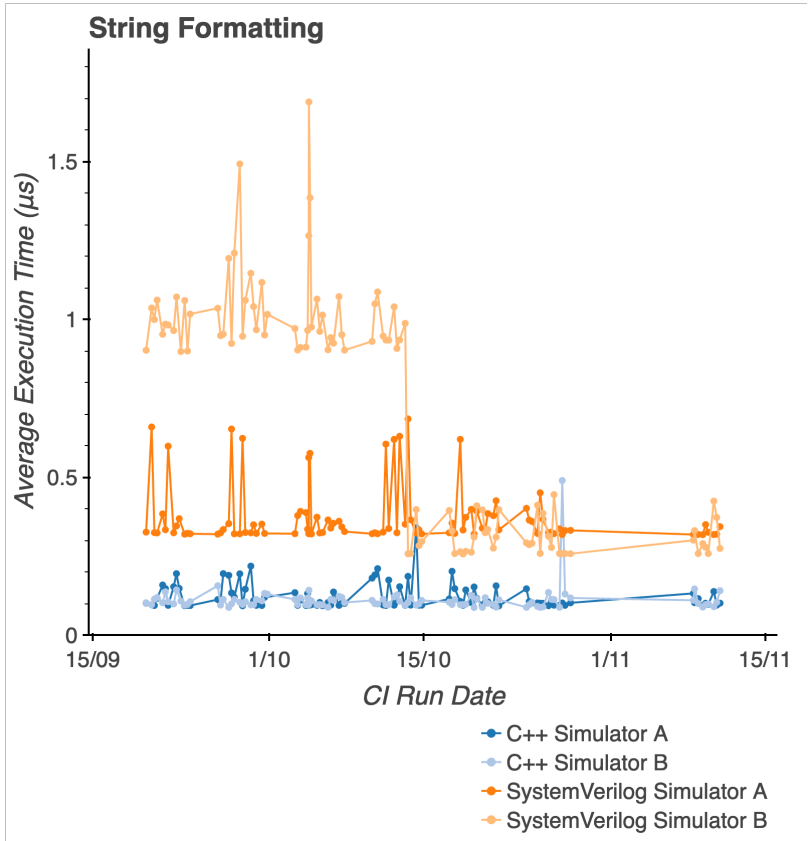
Data – String Display




```systemverilog
// SystemVerilog
for (int j = 0; j < 1000; j++) begin
  $display(test_string);
end
```

```cpp
// C++
for (int j = 0; j < 1000; j++) {
  printf(test_string);
}
```

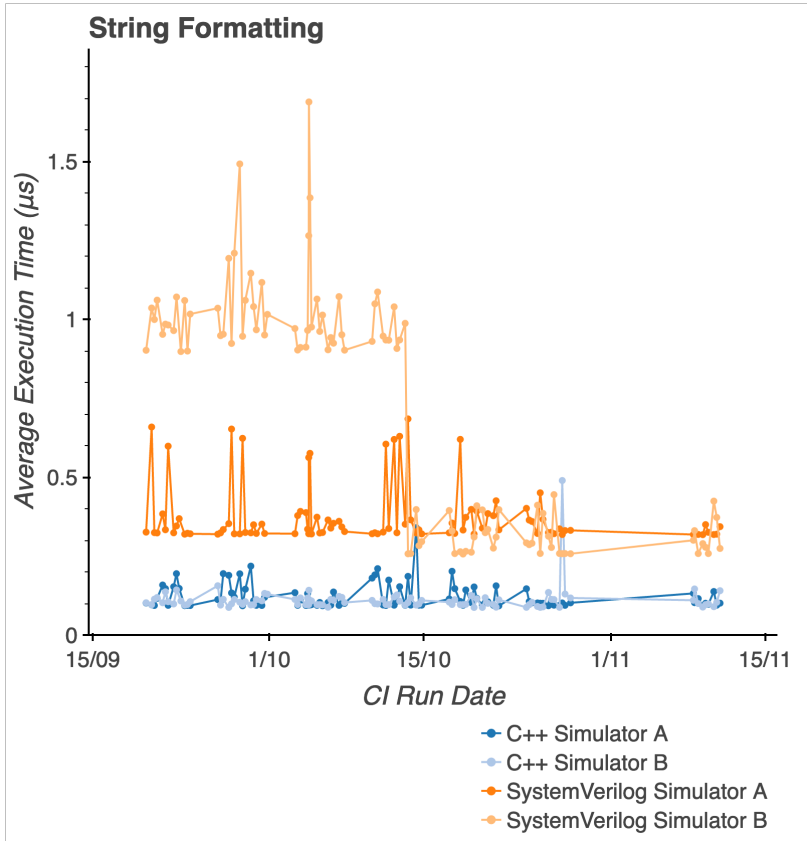
Data – String Formatting




```systemverilog
// SystemVerilog
for (int j = 0; j < 1000; j++) begin
  my_str = $sformatf(
    "Hello world %s %d",
    test_string, j);
end
```

```cpp
// C++
for (int j = 0; j < 1000; j++) {
  my_str = fmt::format(
    "Hello world {} {}",
    test_string, j);
}
```

Data – String Formatting



- Version upgrade of Simulator B on October 12th
  - Correlates with 60% performance boost
- Consider *lazy formatting* in C++

Conclusion

- The DPI is your friend
- Write yourself some benchmarks
  - Evaluate upgrades to newer releases of your tool
  - Quantify gains and losses from custom methodologies
  - Spot long-term trends
- Demand more performance

2022 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

MUNICH, GERMANY
DECEMBER 6 - 7, 2022

Thank You

The Cost Of Standard Verification Methodology Implementations

accellera SYSTEMS INITIATIVE