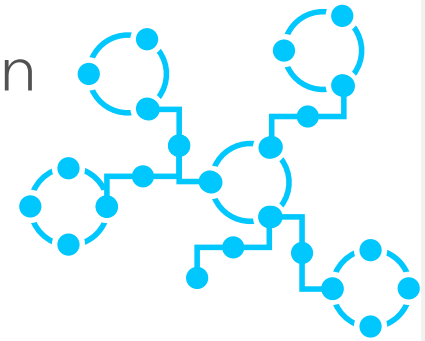
Workshop on Open-Source EDA Technology (WOSET) 2022

Rapid Open Hardware Development Framework

Max Korbel

The Rapid Open Hardware Development (ROHD) Framework

- An ambitious project aiming to replace SystemVerilog as the industry-standard choice for front-end hardware development
- An open-source generator framework for design and verification
- Powerful features include:
  - Logically equivalent, human-readable SystemVerilog generation
  - Fast event-based 4-value simulator with cosimulation and waveform dumping in a modern development environment
- Early data shows big productivity gains
  - Develop hardware up to 50x faster from the same specification
  - Write 50-90% fewer lines of code for the same functionality
  - Build your model over 150x faster
  - Run your simulations over 8x faster

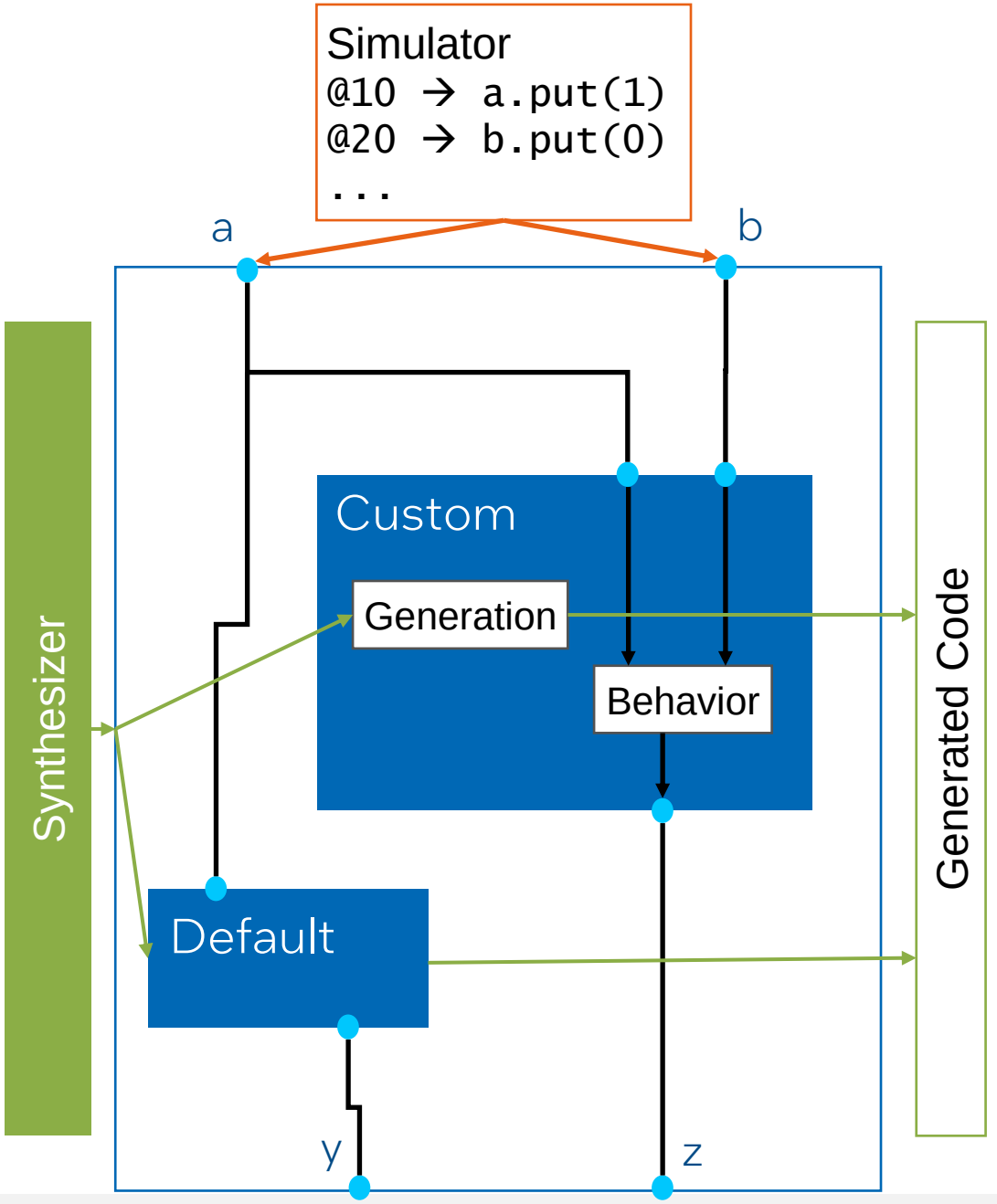


Motivation

- SystemVerilog is not good enough
  - Too limited for hardware description
  - Inadequate debug tooling
  - Poor language for testbenches
- Testbenches are software
  - Should be written in a modern, fully-featured programming language
  - Should run as software, not in a simulator
- Integration and reuse are too difficult
- Slow development iteration time
- Existing alternatives are insufficient
  - Verification often treated as second-class
  - Need an execution-focused solution, not just academic
  - Other HDLs and generators don’t solve the problems entirely
- Open-source hardware community is lacking

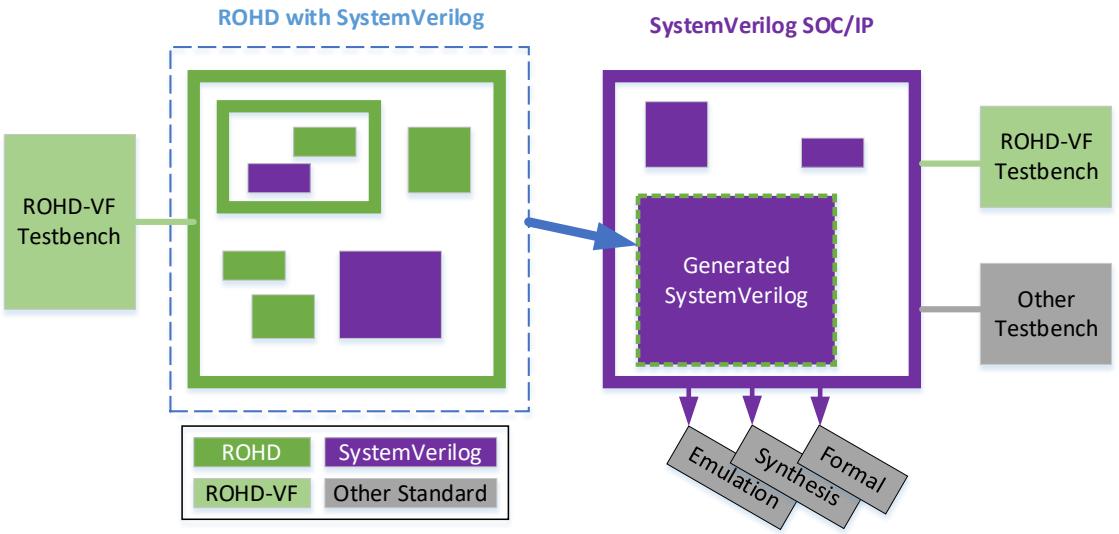
Features

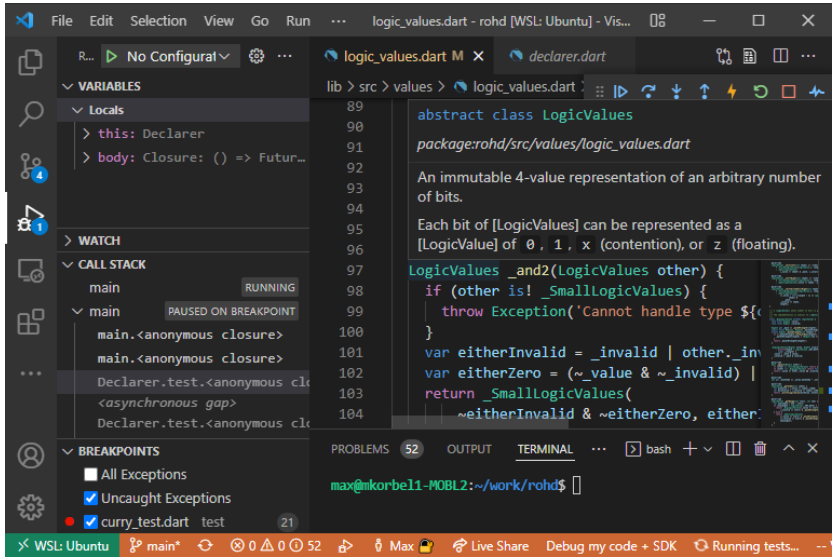
- A framework in Dart language for describing and verifying hardware
  - Dart is a modern, fast language developed by Google
- Build connectivity between module objects
  - Generate with unrestricted software
- ROHD modules map one-to-one
  - Structurally similar, human readable SV
  - Extensible generation (CIRCT, UPF, etc.)
- Built-in fast simulator
  - Includes waveform dumper
  - Cosimulation with SV modules via VPI



Features

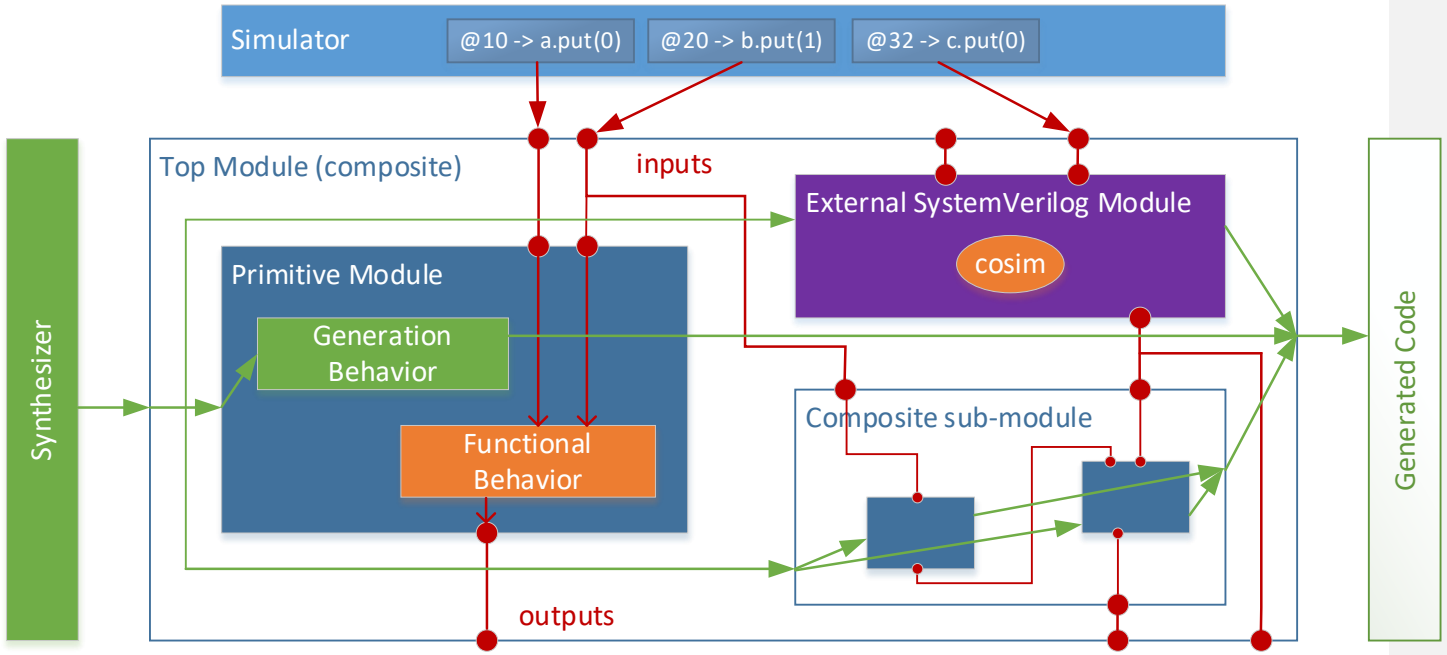
- Easy IP integration and interfaces
  - Just depend on and then import an IP to integrate it
- Verification collateral simpler to develop and debug
  - Testbenches in Dart with a UVM-like framework
  - Interact with signals and events intuitively
- Includes convenient abstractions
  - Pipelining, FSMs, and more
- Use pub.dev package manager
  - Share code without friction
  - Automatically manage dependencies
- Fast and simple build
  - No complex build systems or EDA vendor tools
- Develop code in a modern IDE
  - Powerful static analysis and debug features





Implementation

- All modules extend “Module”
- Hierarchy determined by connectivity
- No software reflection
- Modules compose by default
  - Can define custom behavior
- Simulation is chronological execution of events
- Signal changes propagate to downstream signals



An Interesting Example

ROHD with Dart

```dart
class TreeOfTwoInputModules extends Module {

  final Logic Function(Logic a, Logic b) _op;
  final List<Logic> _seq = [];
  Logic get out => output('out');

  TreeOfTwoInputModules(List<Logic> seq, this._op) : super(name: 'tree_of_two_input_modules') {
    if(seq.isEmpty) throw Exception("Don't use TreeOfTwoInputModules with an empty sequence");

    for(var i = 0; i < seq.length; i++) {
      _seq.add(addInput('seq$i', seq[i], width: seq[i].width));
    }
    addOutput('out', width: seq[0].width);

    if(_seq.length == 1) {
      out <= _seq[0];
    } else {
      var a = TreeOfTwoInputModules(_seq.getRange(0,           _seq.length~/2).toList(), _op).out;
      var b = TreeOfTwoInputModules(_seq.getRange(_seq.length~/2, _seq.length).toList(), _op).out;
      out <= _op(a, b);
    }
  }
}}

void main() {
  // You could instantiate this module with some code such as:
  var tree = TreeOfTwoInputModules(
    List<Logic>.generate(16, (index) => Logic(width: 8)),
    (Logic a, Logic b) => Mux(a > b, a, b).y
  );
}
```

First-class functions

No parameters required

Dynamic port generation

ROHD enables simplicity and less code

Generated SystemVerilog

```
//////////////////
...
//////////////////
module TreeOfTwoInputModules_2(
input logic [7:0] seq0,
input logic [7:0] seq1,
input logic [7:0] seq2,
input logic [7:0] seq3,
input logic [7:0] seq4,
input logic [7:0] seq5,
input logic [7:0] seq6,
input logic [7:0] seq7,
output logic [7:0] out
);
logic [7:0] out_1;
logic [7:0] out_0;

assign out = (out_1 > out_0) ? out_1 : out_0;  // mux
TreeOfTwoInputModules_1  tree_of_two_input_modules(
.seq0(seq0),.seq1(seq1),.seq2(seq2),.seq3(seq3),
.out(out_1));
TreeOfTwoInputModules_1  tree_of_two_input_modules_0(
.seq0(seq4),.seq1(seq5),.seq2(seq6),.seq3(seq7),
.out(out_0));
endmodule : TreeOfTwoInputModules_2
//////////////////
...
//////////////////

module TreeOfTwoInputModules(
input logic [7:0] seq0,
output logic [7:0] out
);

assign out = seq0;

endmodule : TreeOfTwoInputModules
```

ROHD Verification Framework (ROHD-VF)

- A separate framework in Dart language for building testbenches
  - Runs on top of ROHD with fast ROHD simulator
  - Testbench code is native Dart code: fast and simple build, static analysis, IDE integration, debug, etc.
- Build testbenches that are structurally similar to UVM testbenches
  - Does *not* reimplement full UVM API, less opinionated with focus on value
- Take advantage of native Dart language asynchronous support
  - Futures, async/await, Streams, etc.
- Testbench does *not* convert to SystemVerilog UVM
  - No restriction on what kind of code you can run, it need not be convertible

Results

- Used for multiple purposes
  - Microarchitectural modelling
  - Testbench development
  - Hardware design
- Multiple teams within Intel
- Multiple external users
- Configurable designs
  - Logic generated from machine readable specifications

Dramatic Productivity Improvements

IP "A"	Original	New with ROHD	Improvement
Development Time	~1Q initial SV	< 2 days initial	97.8% reduction
Lines of Code (design)	~1500 lines SV	~450 lines ROHD	70% reduction
Build time (design + TB)	~300 seconds, on server	~1.65 seconds, on laptop	182x speedup
Simulation (>50k cycles random)	~170 CPS vendor simulator on server	~1400 CPS on ROHD, on laptop	8.2x speedup

Hardware	Original Lines of Code	Lines of Code (ROHD)	Improvement
AXI Interface	1161 lines Tcl	229 lines ROHD	80% reduction
Interface "C"	588 lines Tcl	283 lines ROHD	50% reduction
IP "D" HIP instantiation and connectivity	941 lines SV	114 lines ROHD	88% reduction

Ecosystem

- ROHD & ROHD-VF are free and open-source on GitHub and pub.dev
  - ROHD: https://github.com/intel/rohd
  - ROHD-VF: https://github.com/intel/rohd-vf
- Encouraging hardware developers to participate in open-source
  - Collaborating develops better solutions with less effort duplication
  - Learning about open-source helps us understand software and our customers
- Sharing publicly, growing a community around ROHD
  - Shared hardware designs and modular components
  - Reusable verification collateral and tools
- Actively accepting contributions in the form of issues & pull requests
  - Existing packages
  - New packages

Looking Ahead

- It's easy to get started with ROHD
  - Everything is free and open source, no licenses or vendor tools required
  - Join the ROHD Forum and ROHD Discord to discuss
- ROHD & the ecosystem are open for contributions!
  - Great opportunity to help redefine the way things should be done from a clean slate
- More big innovations are in the pipeline
  - CIRCT integration, power-aware simulation, registers & RAL, coverage, randomized constraint solving, HLS integration, component libraries, more abstractions, etc.
- ROHD can be adopted for small pieces of a design or testbench
  - Does not have to be all-or-nothing