2018 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

UVM for HLS: An Expedient Approach to the Functional Verification of HLS Designs

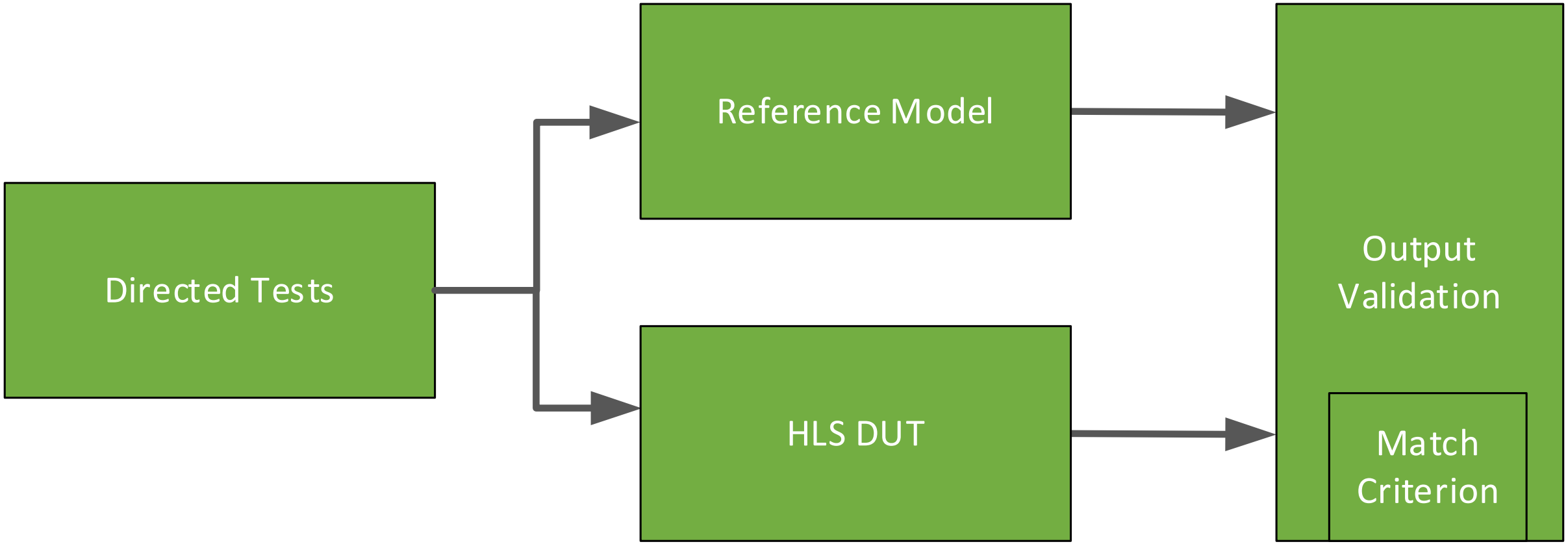
Dave Burgoon, Robert Havlik
Microsoft Corporation

2018 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

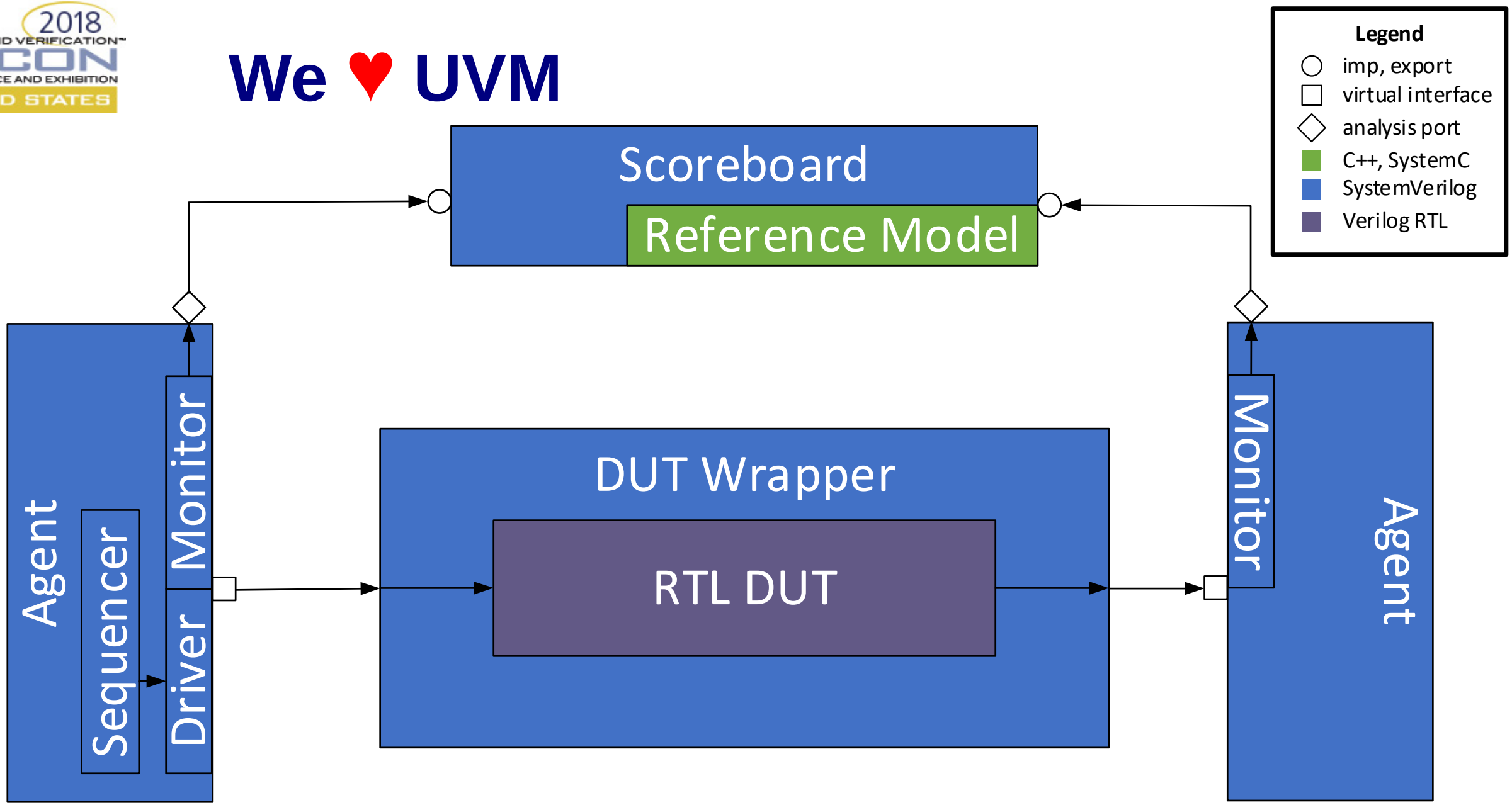
Agenda

- Motivations
- Implementation Overview
- Future Work
- Summary

We ♥ HLS

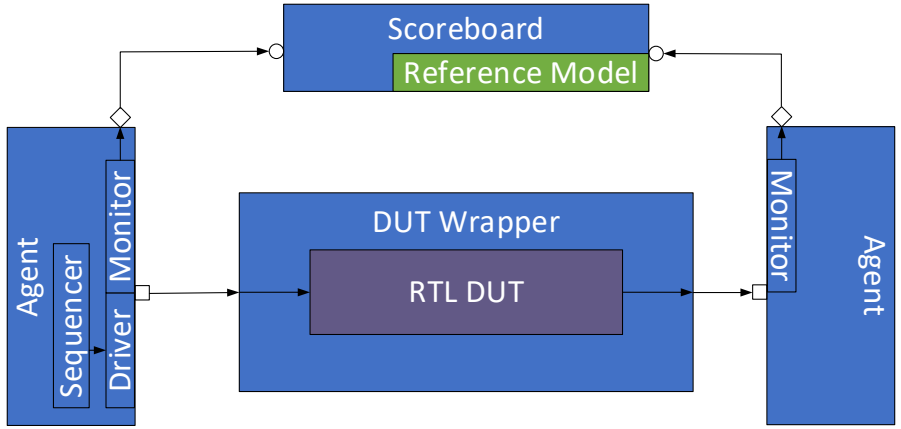

Directed Tests
Reference Model
HLS DUT
Output Validation
Match Criterion

We ♥ UVM

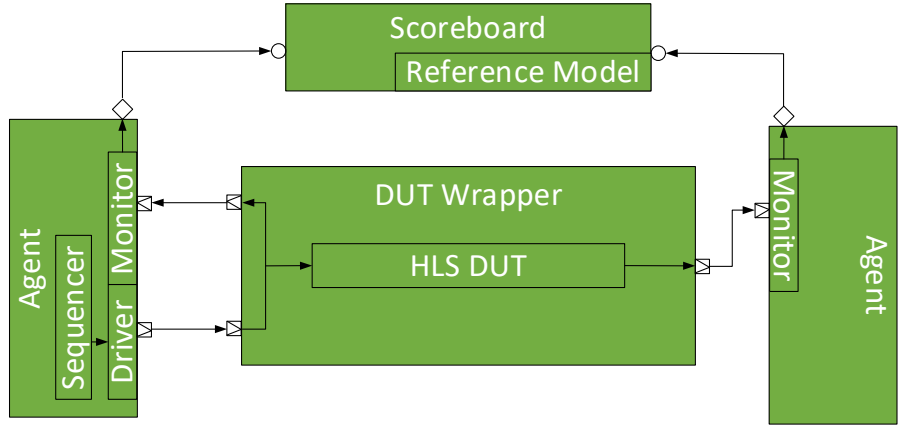

Legend
imp, export
virtual interface
analysis port
C++, SystemC
SystemVerilog
Verilog RTL
Scoreboard
Reference Model
Agent
Sequencer
Driver
Monitor
DUT Wrapper
RTL DUT
Monitor
Agent

How do we get there from here?

- Debuggers, waveform viewers ✓
- Coverage management ✓
- Formal checkers, Linters ✓
- RTL-to-gates LEC ✓
- UVM-SystemVerilog ✓



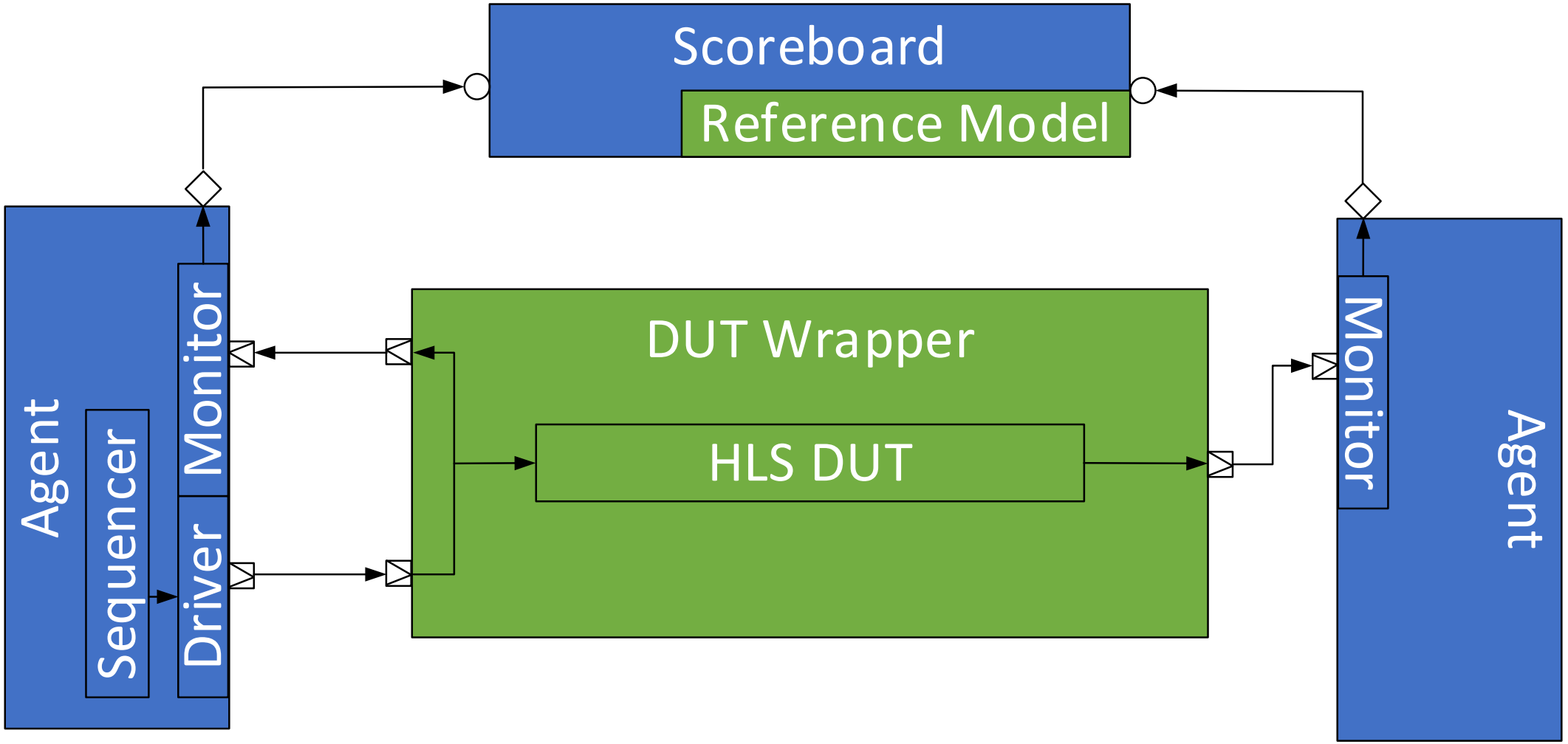
- Portable stimulus standard?
- Debuggers, waveform viewers?
- Coverage management?
- Formal checkers, Linters?
- HLS-to-RTL LEC?
- UVM-SystemC?



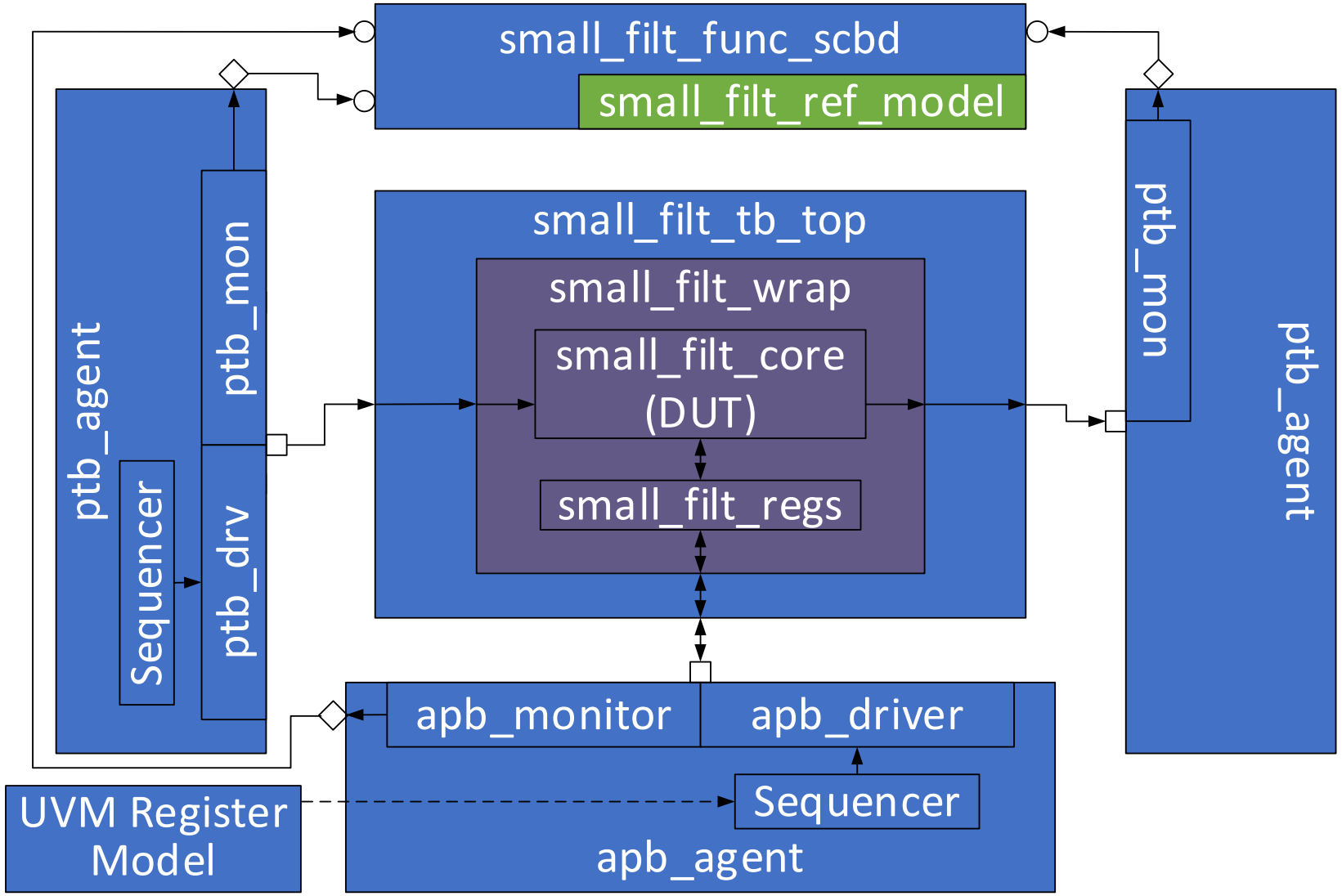
2017

20??

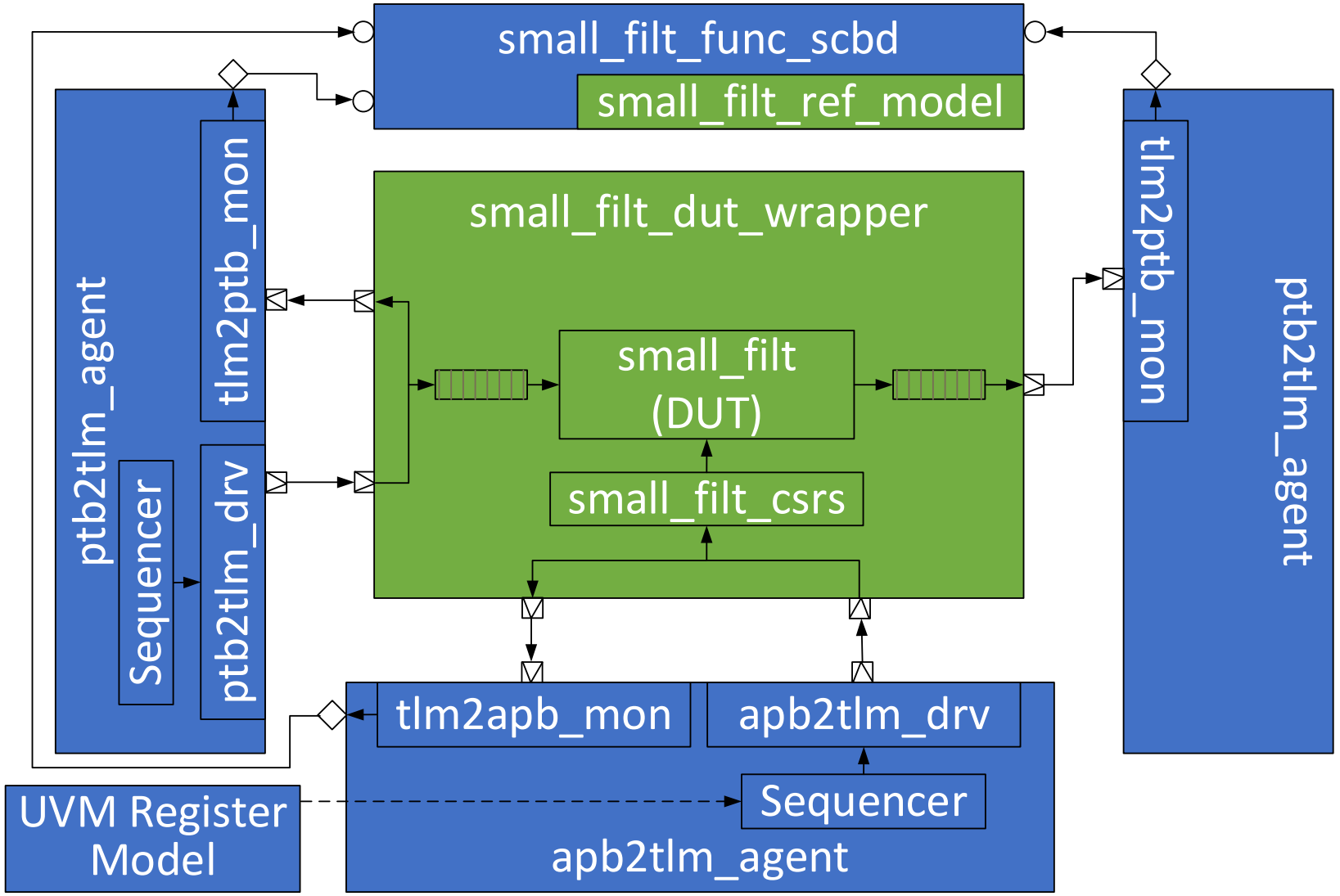
Answer: “UVM for HLS” via UVM Connect



"small_filt" Guinea Pig—RTL DUT



"small_filt" Guinea Pig—HLS DUT



How We Did It

- TLM Conversion
- Subclasses and Type Overrides
- DUT Wrapper
- UVM Connect “Wiring”

How We Did It—TLM Conversion (Driver)

```
virtual function void to_tlm_gp(ref uvm_tlm_gp txn);
```

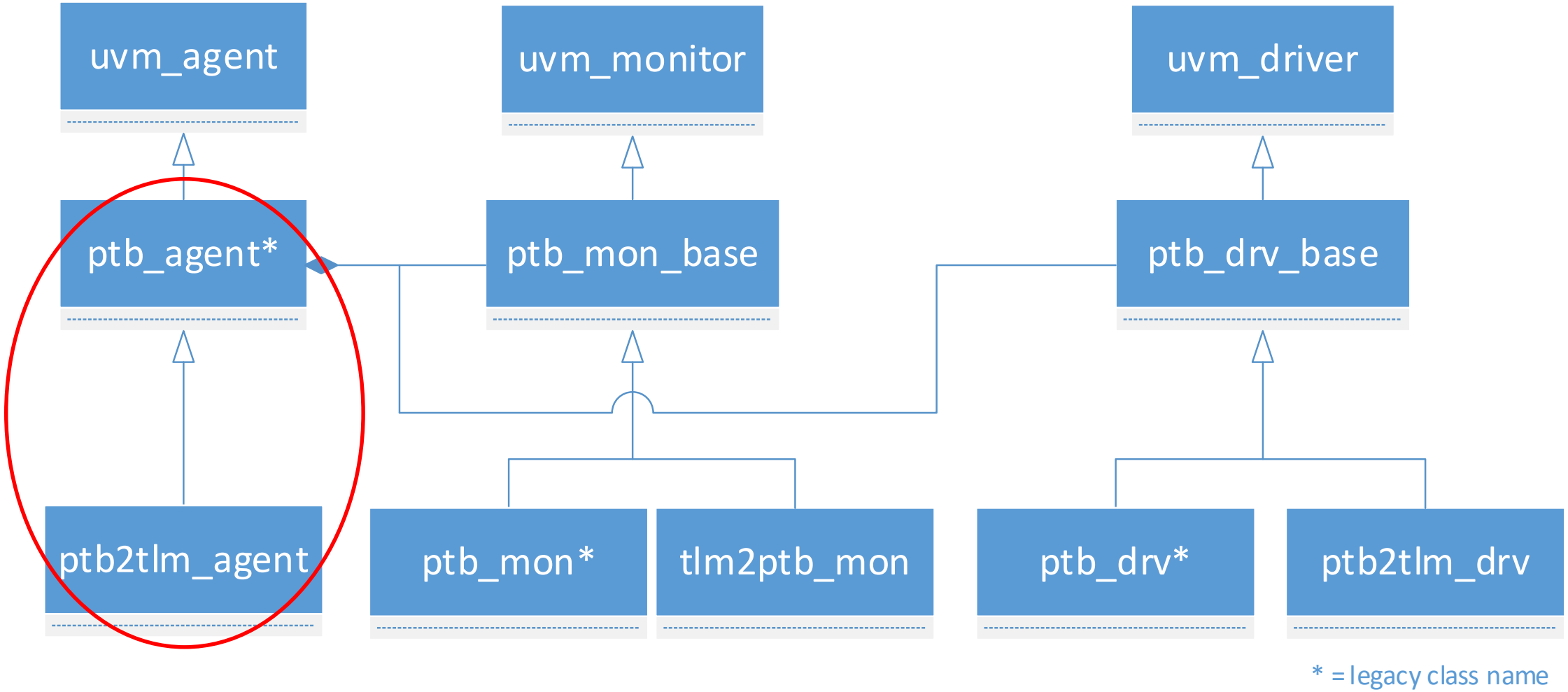
```
task ptb2tlm_drv::run_phase(uvm_phase phase);
  // TLM2 Generic Payload Transaction
  uvm_tlm_gp tlm_gp_req_item = new("tlm_gp_req_item");

  uvm_tlm_time delay = new("delay", 1e-12);

  forever begin
    seq_item_port.get_next_item(req_item);
    req_item.to_tlm_gp(tlm_gp_req_item);
    tlm_out.b_transport(tlm_gp_req_item, delay);

    rsp_item.copy(req_item);
    rsp_item.set_id_info(req_item);
    seq_item_port.item_done(rsp_item);
  end //end forever

endtask: run_phase
```

How We Did It—TLM Conversion (Monitor)

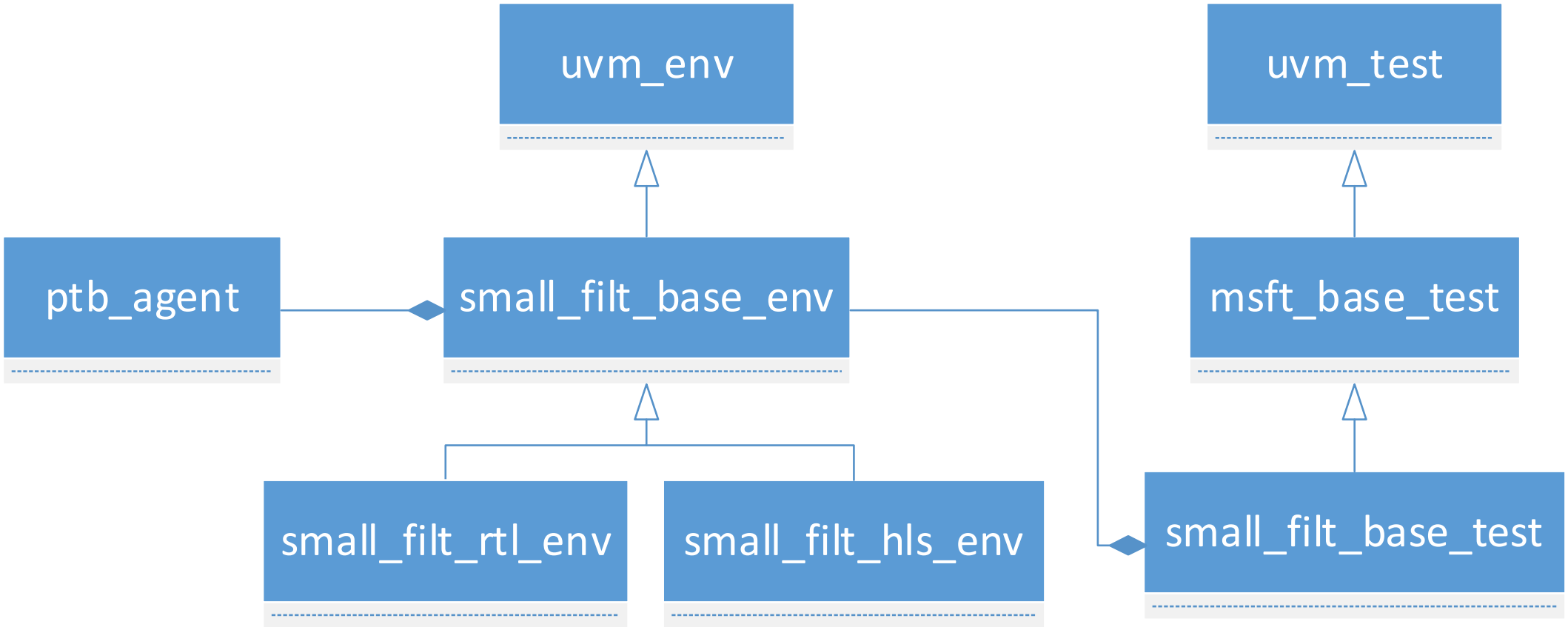
```
virtual function void set_from_tlm_gp(const ref uvm_tlm_gp in);
```

```
// Blocking task called via the "tlm_in" socket
task tlm2ptb_mon::b_transport(uvm_tlm_gp txn_in, uvm_tlm_time delay);
  ptb_txn = ptb_seq_item
`PTB_AGENT_PARAMS_PASS::type_id::create("ptb_txn",this);

  ptb_txn.set_from_tlm_gp(txn_in);
  mon_items_ap.write(ptb_txn);

endtask: b_transport
```

How We Did It—Agent Subclasses



How We Did It—Test & Env. Subclasses



How We Did It—Overrides in Base Test

```
void'($value$plusargs("CONF=%s", conf));
if (uvm_is_match("*_rtl_dut", conf)) begin
  small_filt_tb_cfg::is_rtl_dut = 1;
  `uvm_info(REPORT_TAG, "RTL DUT", UVM_NONE)
end
else begin
  small_filt_tb_cfg::is_rtl_dut = 0;
  `uvm_info(REPORT_TAG, "HLS DUT", UVM_NONE)
end

if (small_filt_tb_cfg::is_rtl_dut)
  small_filt_base_env::type_id::set_type_override(small_filt_rtl_env::get_type());
else
  small_filt_base_env::type_id::set_type_override(small_filt_hls_env::get_type());
`uvm_info(REPORT_TAG, "Creating Environment", UVM_MEDIUM)

env = small_filt_base_env::type_id::create("env", this);
```

How We Did It—HLS-DUT Wrapper

```
virtual void b_transport(tlm_generic_payload& gp, sc_time& t) {
  // Tee the incoming transaction into data_in_mon
  data_in_mon->b_transport(gp, t);

  // Send incoming data into the pipeline
  small_filt_ns::ptb_txn data_in(gp);
  ch_ptb_in.write(data_in);

  // Process any data in ch_ptb_in
  dut.StepAC(ch_ptb_in, csrs, ch_ptb_out);

  // Send output to data_out_mon
  while (ch_ptb_out.available(1)) {
    send_to_monitor(ch_ptb_out.read(), data_out_mon, t);
  }

  // Complete the TLM transaction
  gp.set_response_status(TLM_OK_RESPONSE);
}
```

How We Did It—SV UVM Connect'ions

```
function void ptb2tlm_agent::connect_phase(uvm_phase phase);
  super.connect_phase(phase);

  if (is_master == UVM_ACTIVE) begin
    uvmc_tlm #(uvm_tlm_gp,
               uvm_tlm_phase_e,
               uvmc_xl_tlm_gp_converter)::connect(driver.get_tlm_out(),
driver.get_full_name());
  end

  uvmc_tlm #(uvm_tlm_gp,
             uvm_tlm_phase_e,
             uvmc_xl_tlm_gp_converter)::connect(monitor.get_tlm_in(),
monitor.get_full_name());

endfunction: connect_phase
```

How We Did It—SC UVM Connect'ions

```
int sc_main(int argc, char* argv[]) {
  small_filt_dut_wrapper& dut_wrapper(*(new small_filt_dut_wrapper("dut_wrapper")));

  uvmc::uvmc_connect<uvmc_xl_converter<tlm_generic_payload> >(
    dut_wrapper.data_in,
    "uvm_test_top.env.pixin_agent.driver"
  );
  uvmc::uvmc_connect<uvmc_xl_converter<tlm_generic_payload> >(
    dut_wrapper.data_in_mon,
    "uvm_test_top.env.pixin_agent.monitor"
  );
  uvmc::uvmc_connect<uvmc_xl_converter<tlm_generic_payload> >(
    dut_wrapper.data_out_mon,
    "uvm_test_top.env.pixout_agent.monitor"
  );

  ● ● ●

  sc_start();
  return 0;
}
```

2018 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Future Work

- Continue to improve simulation run-time performance.
- Evolve our in-house UVM code generator.
- Reduce cost of writing and maintaining programming-model adapter code.
- Ramp up on portable stimulus standard and commercial offerings.

Summary

- Our UVM-for-HLS approach is an expedient way to *directly* verify HLS DUTs *today*, and avoids the risk and uncertainty of nascent approaches.
- UVM Connect, class inheritance, and type overrides minimize the incremental effort:
  - UVM Connect avoids custom DPI code.
  - Effort confined to development of TLM conversion methods, subclasses, and SystemC wrapper.
- Leveraged base classes, majority of test cases, and entire scoreboard.
- Enabled early development of test environment and test codes.

Thank You!