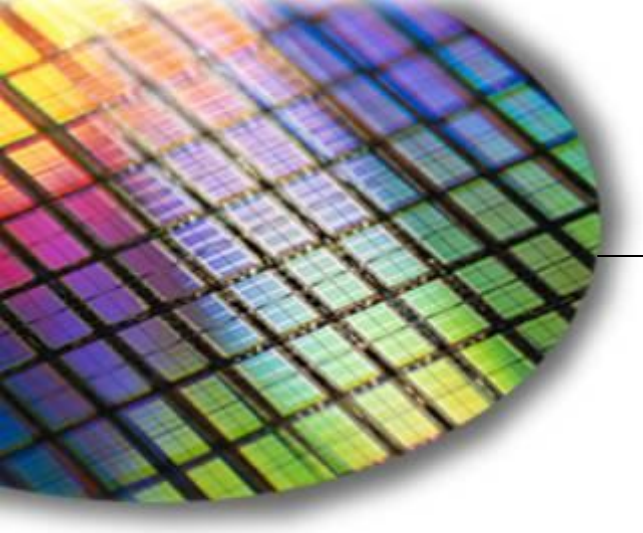


The World Leader in High Performance Signal Processing Solutions

Resetting Anytime with the Cadence UVM Reset Package

Courtney Schmitt, ADI
Stephanie McInnis, Cadence
Uwe Simm, Cadence
Phu Huynh, Cadence

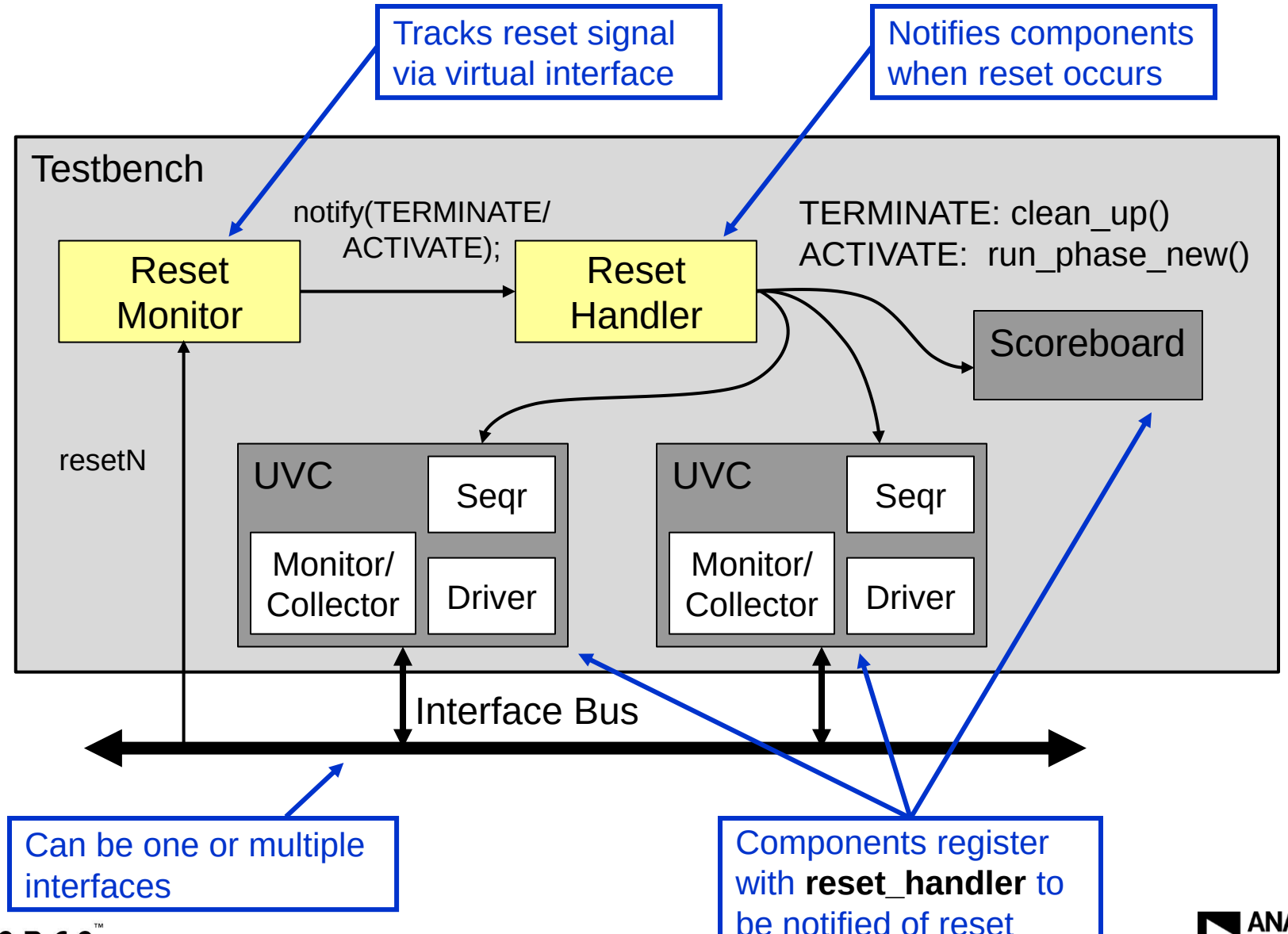
cadence™

ANALOG DEVICES

Reset Requirements

- A common verification requirement is to perform reset
  - At the start of the simulation
  - In the middle of the DUT normal operation
- Special care is required for verification environment during reset:
  - Activities and stimulus needs to stop and possibly restart once the reset is de-asserted
  - Assertions and checkers need to shut down gracefully
  - Scoreboards and data structures need to reset to proper initial values

UVM Thread Reset Methodology

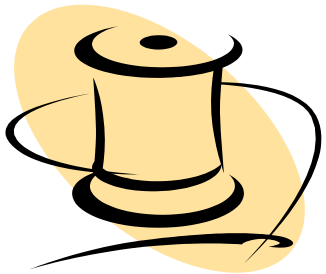


cadence

ANALOG DEVICES

Reset Methodology Components

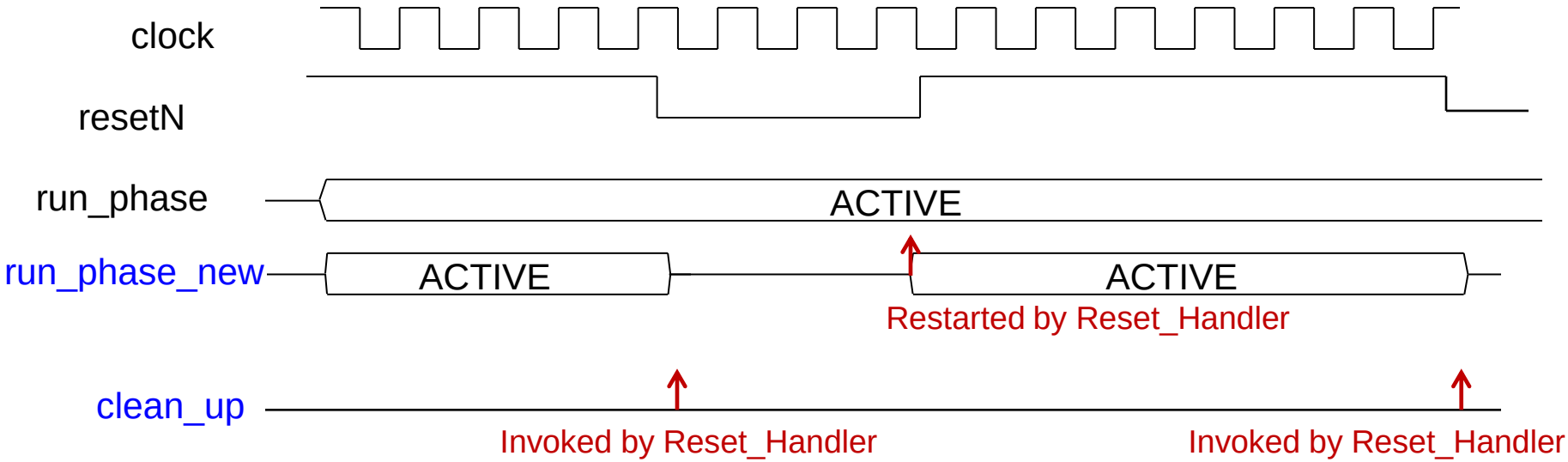
- **Reset Monitor**
  - Monitor reset signal(s)
  - Notify **reset_handler** of reset status



- **Reset Handler**
  - An instance of **uvm_thread**, provided by the Cadence package
  - Manages“reset-aware” components using their reset API:
    - Reset goes away → Invokes run_phase_new()
    - Reset active → Terminates threads, Invokes clean_up()

- **Reset-aware components:**
  - Register themselves with the reset_handler
  - Implement new reset APIs: **clean_up()** and **run_phase_new()**

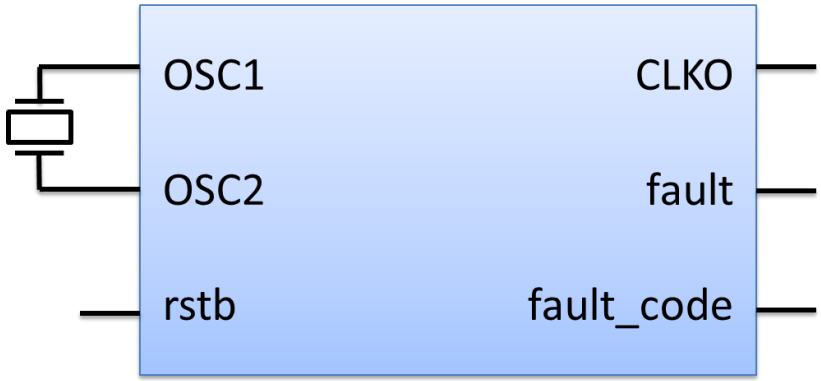
run_phase_new & clean_up



- **Thread management instead of phase management**
  - **Works with existing UVM components to handle reset within the run_phase**
  - **Can be extended to multiple reset domains**
  - **No changes to the UVM library are required**

ADI Evaluation Block

- **System Oscillator with Watchdog**
  - Provides the main system clock from a crystal oscillator input
  - Oscillator watchdog detects faults on the input clock
  - Active low rstb disables the block



- **Verification Requirements**
  - Check CLKO frequency based on crystal frequency
  - Activate and check the detection of various fault conditions
  - Check that the block shuts down gracefully when reset is asserted
  - Check that the block restarts correctly when reset is de-asserted

wd_osc_env

```
class wd_osc_env extends uvm_env;

  wd_osc_agent agent_wd_osc;
  wd_osc_scoreboard wd_osc_sb;
  virtual wd_osc_interface vif;

  reset_monitor reset_mon;
  uvm_thread reset_handler;

 function void build_phase(uvm_phase phase);
   …
   reset_mon = reset_monitor::type_id::create("reset_mon", this);
   reset_handler = uvm_thread::type_id::create("reset_handler", this);
   uvm_config_db#(uvm_thread)::set(this, "*", "reset_handler", reset_handler);

 endfunction : build_phase
 …
endclass
```

Pointer to reset_handler and reset_monitor.

Registers reset_mon and reset_handler with the factory

Store the handle to reset_handler in the config_db

reset_monitor

```
class reset_monitor extends uvm_monitor;
  virtual reset_if vif;
  uvm_thread reset_handler;

 function void connect_phase(uvm_phase phase);
    uvm_config_db#(virtual reset_if)::get(this, "", "vif", vif);
    uvm_config_db#(uvm_thread)::get(this, "", "reset_handler", reset_handler);
 endfunction

virtual task run_phase(uvm_phase phase);
    forever begin
        @(vif.resetN);
        if (vif.resetN)      reset_handler.notify(ACTIVATE);
        else                 reset_handler.notify(TERMINATE);
    end
 endtask
```

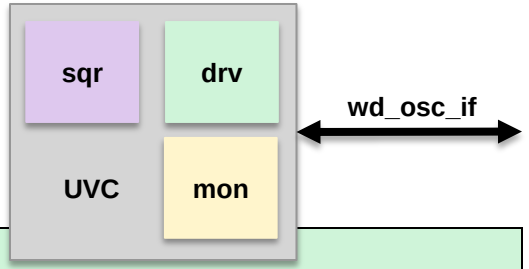
Virtual Interface contains reset signal

Pointer to reset_handler

Get vif and reset_handler from config_db

activate UVCs when reset goes away
terminate UVC processes when reset starts

wd_osc_drv (1 of 2)



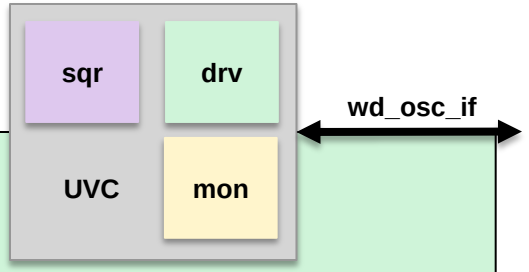

```
class wd_osc_drv extends uvm_driver #(wd_osc_data);
  uvm_thread reset_handler;
  local uvm_thread_imp#(wd_osc_drv) reset_export;

 function new(string name, uvm_component parent);
    reset_export = new("reset_export", this);
 endfunction : new virtual

 function void connect_phase(uvm_phase phase);
     uvm_config_db#(uvm_thread)::get(this, "", "reset_handler", reset_handler)
     reset_handler.register(reset_export, uvm_thread_pkg::default_map);
 endfunction
```

Pointer to reset_handler

TLM-like port reset_export for notification

Get reset_handler & register this component

wd_osc_drv (2 of 2)

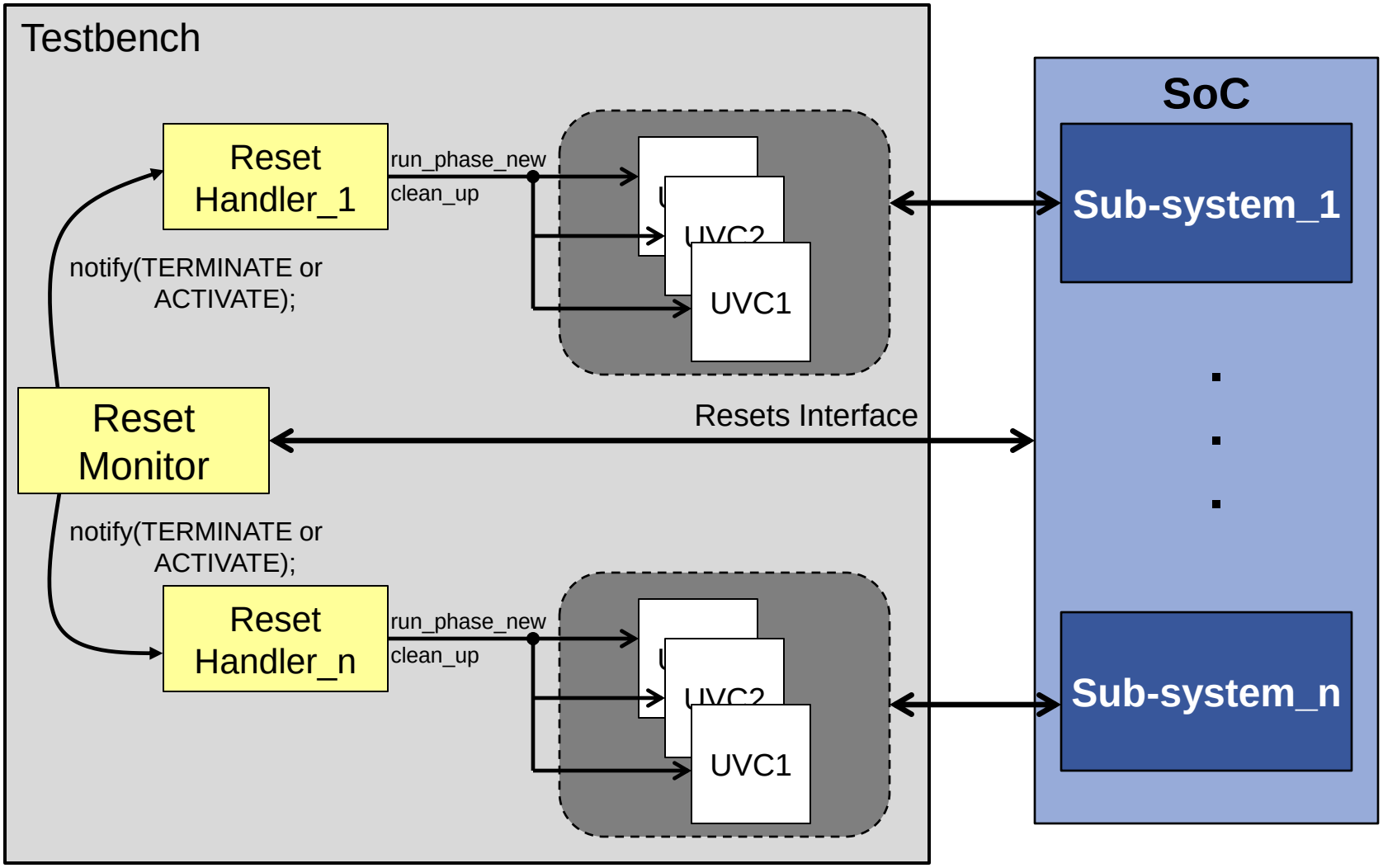



```
virtual function void clean_up();
   vif.OSC1 <= $urandom_range(1,0);
   vif.OSC2 <= $urandom_range(1,0);
endfunction : clean_up

 virtual task run_phase(uvm_phase phase);
 endtask : run_phase

 virtual task run_phase_new(uvm_phase phase);
   super.run_phase(phase);
   get_and_drive();
 endtask : run_phase_new

endclass
```

Invoked by the reset_handler when reset is active

Invoke these activities when reset goes away

Handling Multiple Reset Regions



Summary & Conclusions

Cadence uvm_thread Reset Package provides a good methodology for adding reset verification to UVM testbenches

- **Flexible and extendible**
  - You define reset triggers based on project requirements
  - Can extend to multiple resets; active low or active high reset
  - clean_up and run_phase_new are coded based on specific UVC
- **Works with standard UVM library source code (LRM compliant)**
- **Provides testbench standardization for reset verification**
- **Easy to implement in new or existing testbench environments**
  - Only two additional classes
  - Minimal modifications to existing UVCs

Resources for More Information

- **Cadence uvm_thread package and simple reset examples is available in the "Accellera UVM Uploads (Users contribution area)":**
  - http://forums.accellera.org/files/category/3-uvm/