Reset Testing Made Simple with UVM Phases

Brian Hunter

Cavium

Synopsys Users Group
SILICON VALLEY 2013

CAVIUM

Agenda

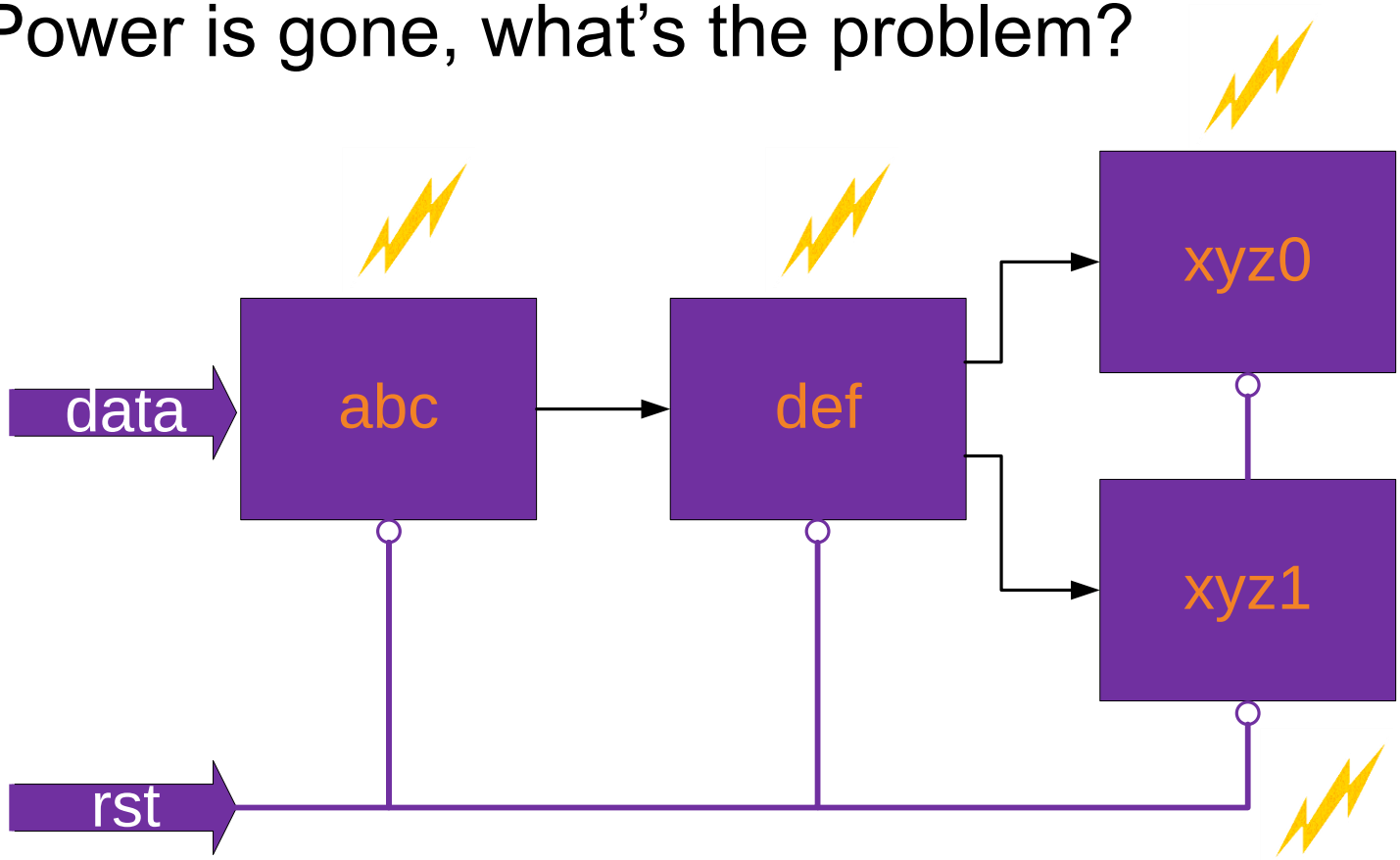
- **Why Test Resets?**
- Typical Challenges
- Solution
- Types of Reset Testing
- Resetting Components
- Multi-Domain Resets
- Re-Randomizing on Reset
- Conclusions
- Q&A

Brian Hunter

CAVIUM

Why Test Resets?

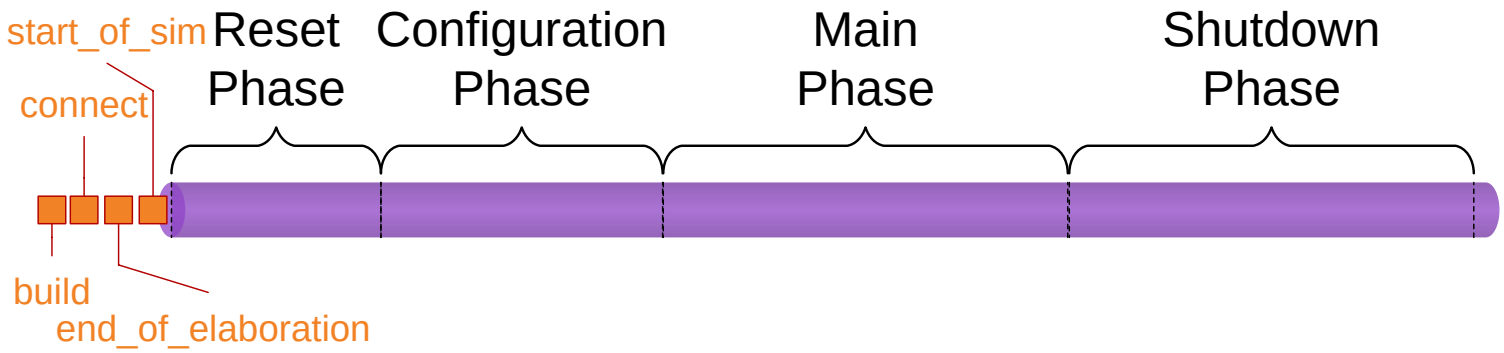
- Power is gone, what's the problem?



Brian Hunter
CAVIUM

Why Test Resets?

- We already do this in every simulation



Brian Hunter

Why Test Resets?

- We always reset our flops anyway

```
always @(posedge clk or negedge rst_n) begin
  if(~rst_n) begin
     data_a0 <= 'b0;
     data_a1 <= 'b0;
     data_a2 <= 'b0;
  end else begin
     data_a0 <= data;
     data_a1 <= data_a0;
     data_a2 <= data_a1;
  end
end
```

Why Test Resets?

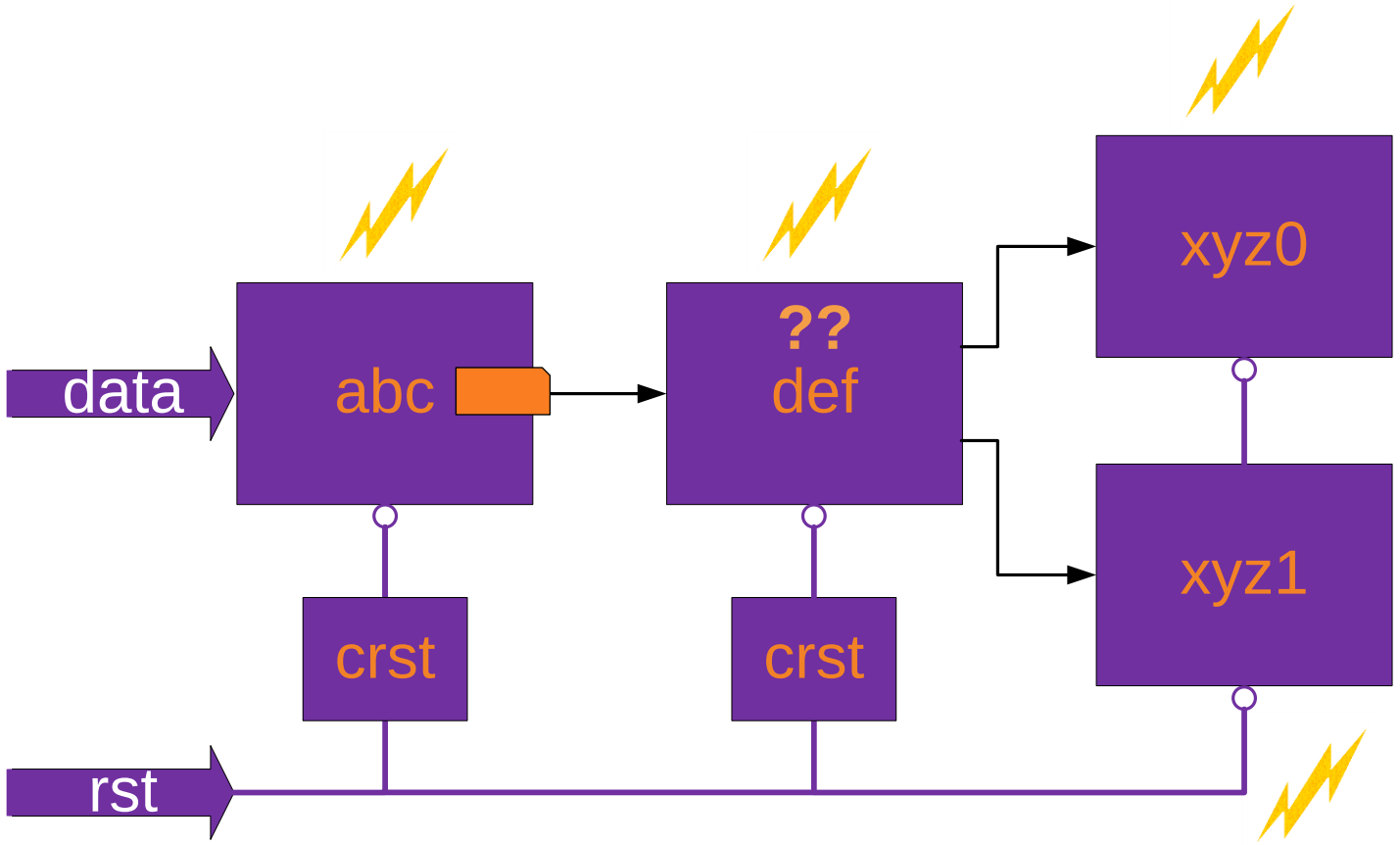
Why am I here listening to this guy?

Why Test Resets?

- But the times, they are a-changin’

an Hunter

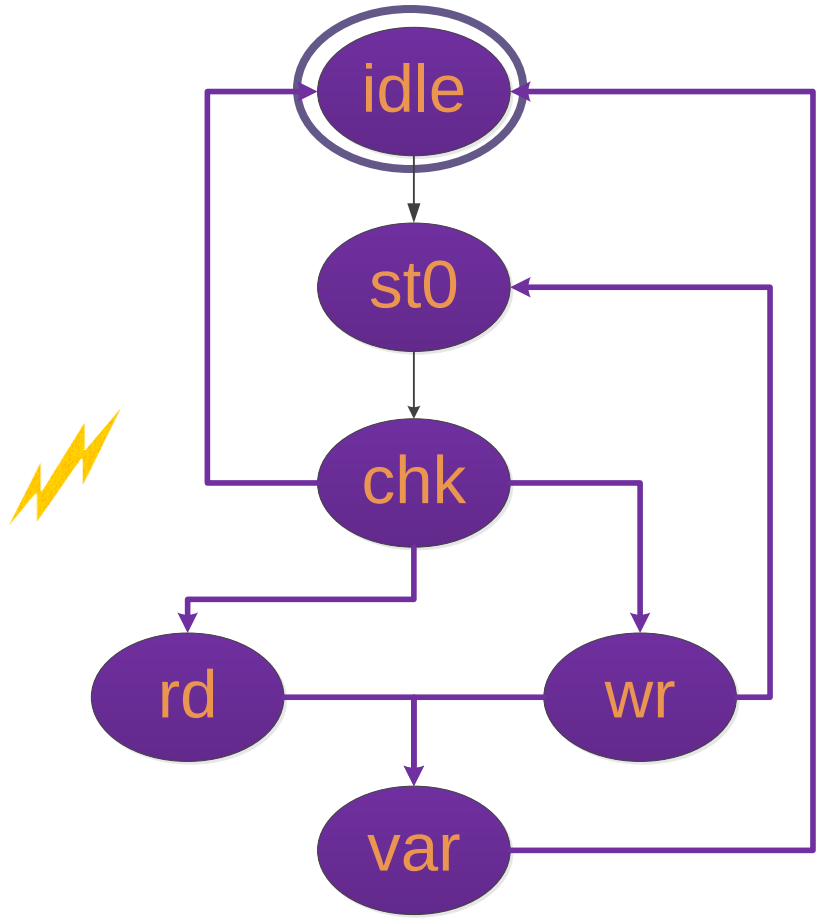
Why Test Resets?



Brian Hunter
CAVIUM

Why Test Resets?

- Are you idle?



Brian Hunter

CAVIUM

Why Test Resets?

- How could that happen?

```
always @(posedge clk or negedge rst_n) begin
  if(~rst_n) begin
     data_a0 <= 'b0;
     data_a1 <= 'b0;
     data_a2 <= 'b0;
  end else begin
     data_a0 <= data;
     data_a1 <= data_a0;
     data_a2 <= data_a1;
     data_a3 <= data_a2;
  end
end
```

unter

CAVIUM

Why Test Resets?

- Memories don’t clear on reset
- FIFOs are not flushed
- Previous data leaks out
- Statistics are not zeroed
- Soft Resets
- Networking link down/up scenarios
- Multi-Domain Resets
- Your Scenario Goes Here

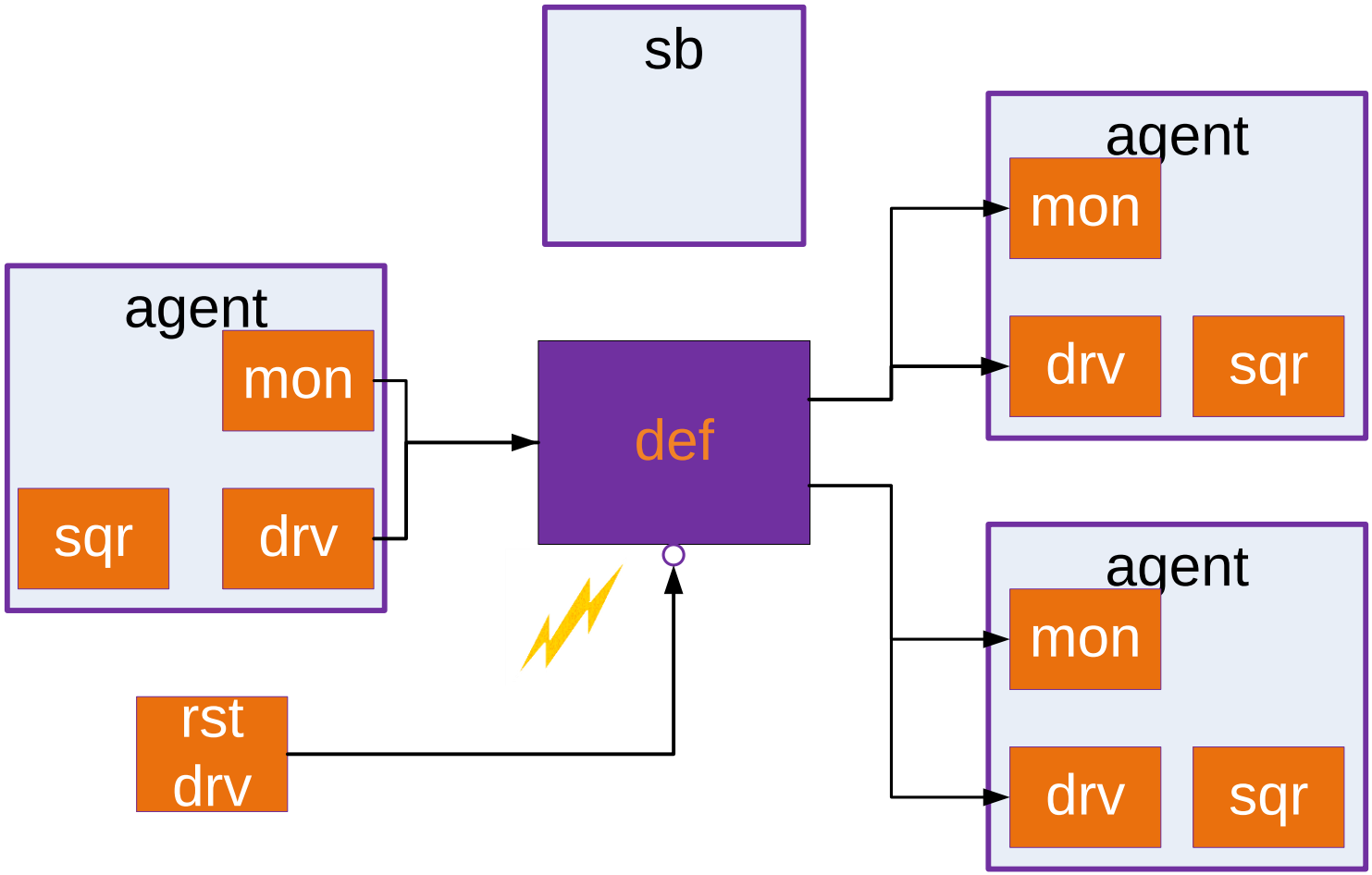
Brian Hunter

CAVIUM

Agenda

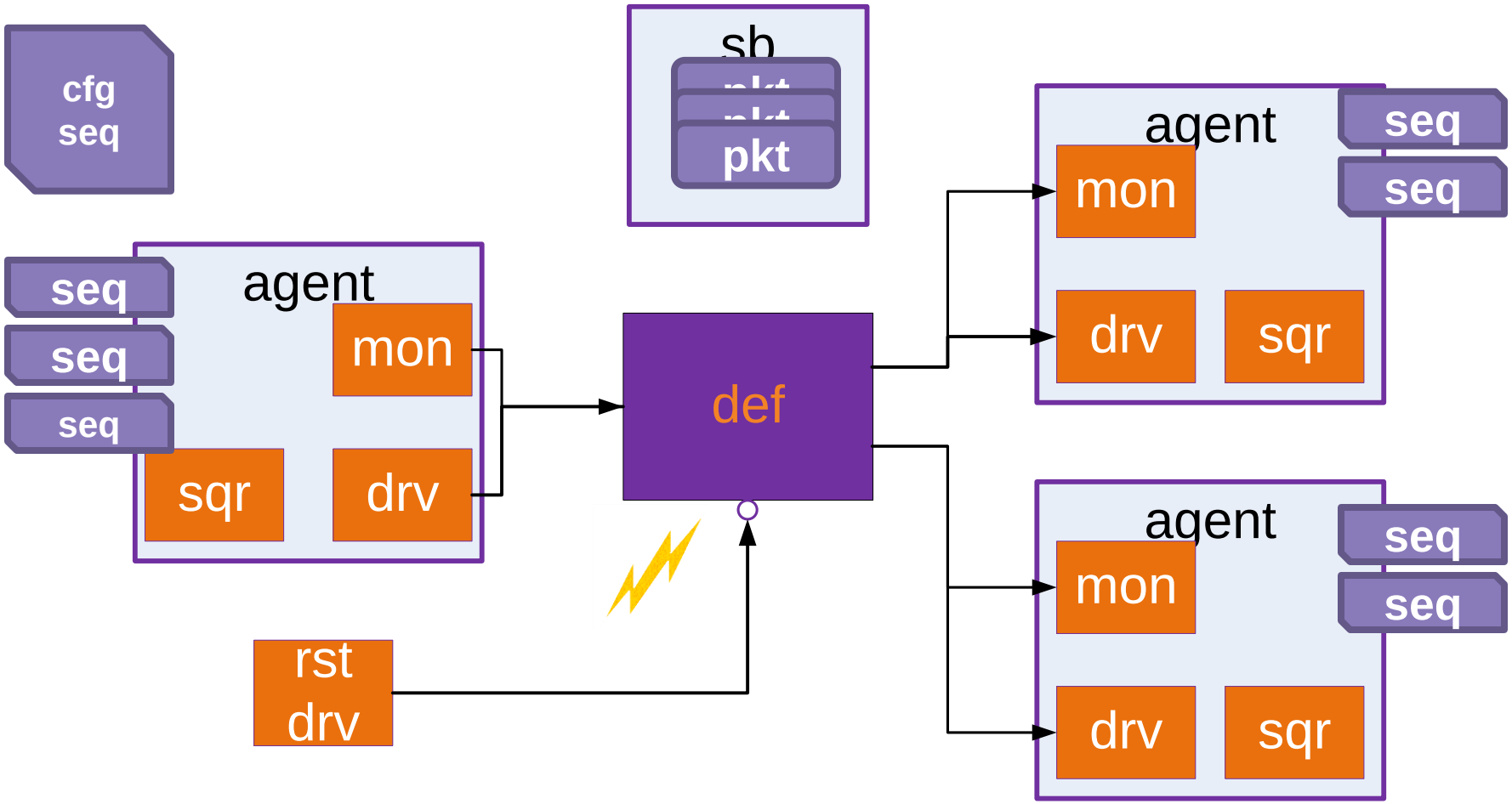
- Why Test Resets?
- **Typical Challenges**
- Solution
- Types of Reset Testing
- Resetting Components
- Multi-Domain Resets
- Re-Randomizing on Reset
- Conclusions
- Q&A

Typical Challenges



Brian Hunter

Typical Challenges



Brian Hunter

Typical Challenges

- Global notification of reset
  - global variables are generally bad practice
  - tlm analysis ports would go *all over the place*
- Kill sequences
- Kill threads
- Work for VIP, too?

Brian Hunter

CAVIUM

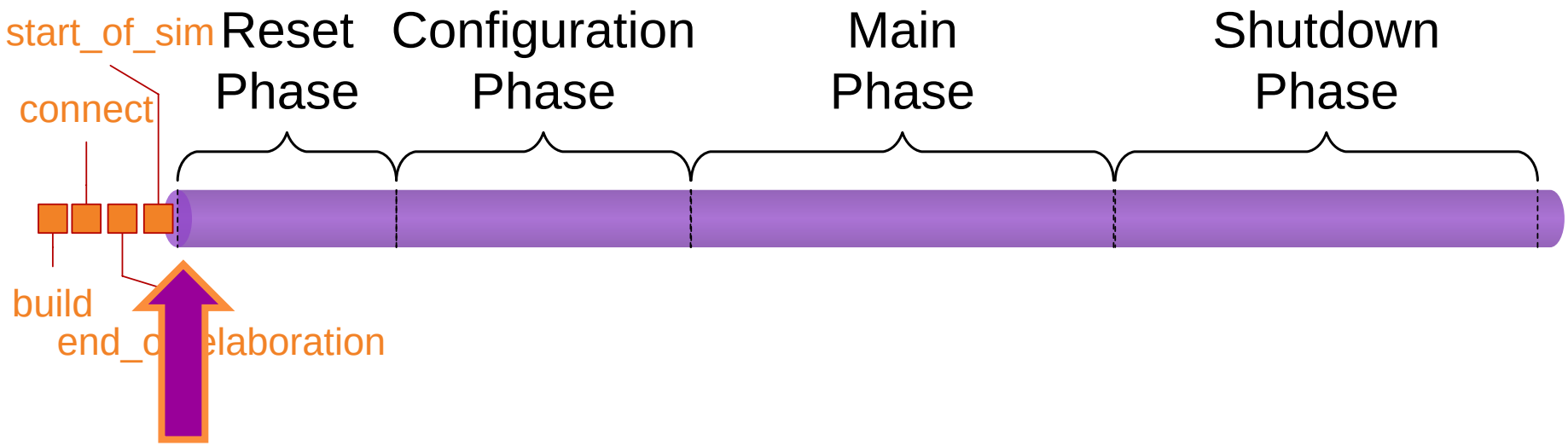
Phase Jumps to the Rescue!

Brian Hunter

CAVIUM

Phase Jumps

- Phases are a major piece of UVM

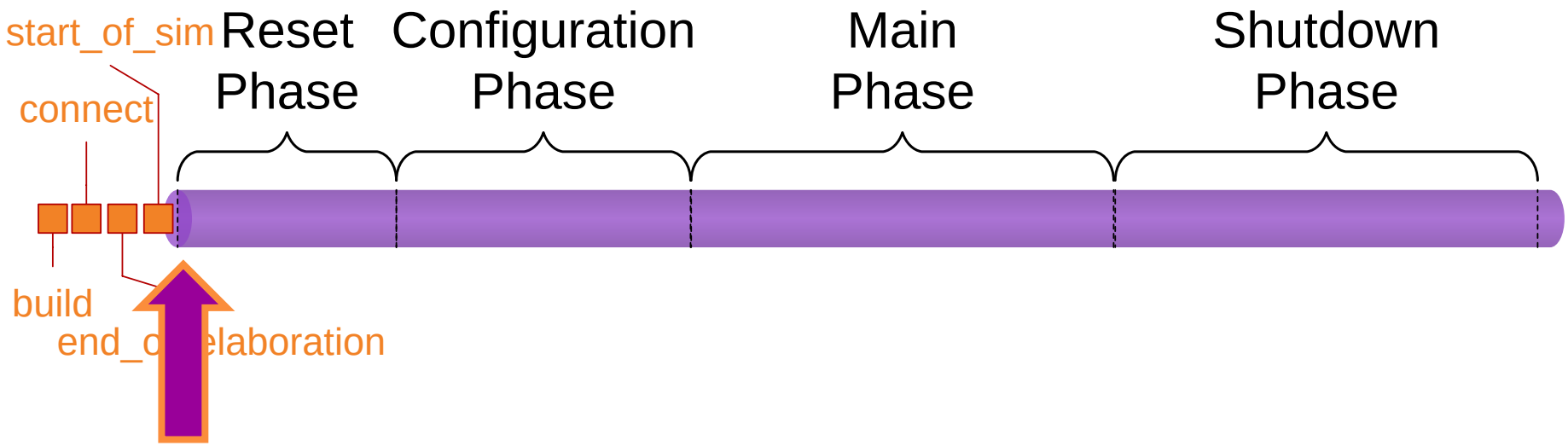


Brian Hunter

CAVIUM

Phase Jumps

- UVM also has a lesser-known feature called phase jumping



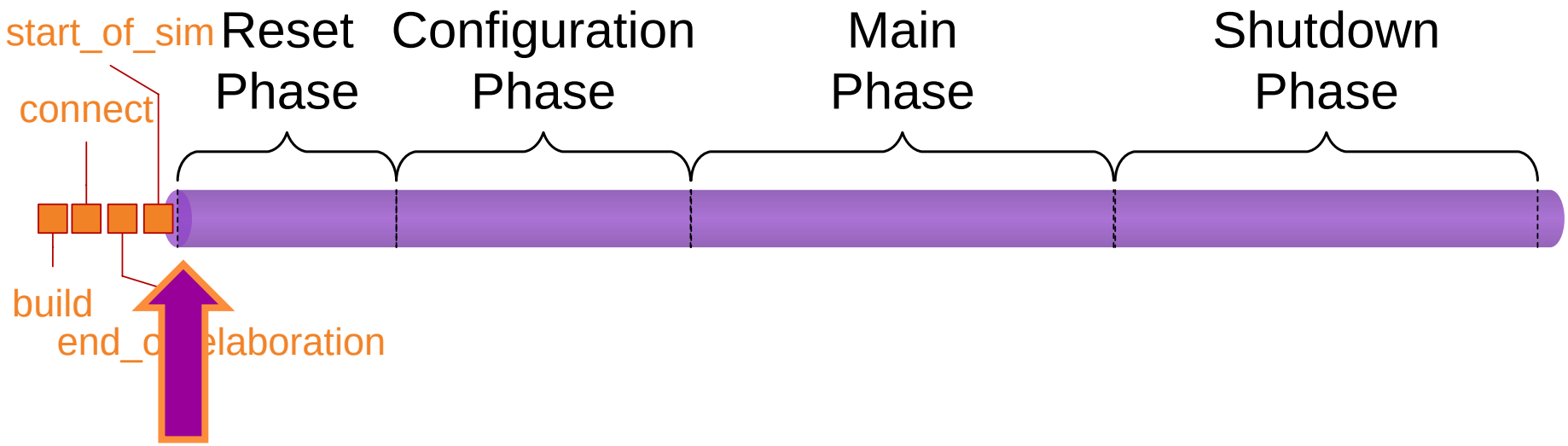
Brian Hunter

Agenda

- Why Test Resets?
- Typical Challenges
- Solution
- **Types of Reset Testing**
- Resetting Components
- Multi-Domain Resets
- Re-Randomizing on Reset
- Conclusions
- Q&A

Types of Reset Testing

- Idle Reset Testing



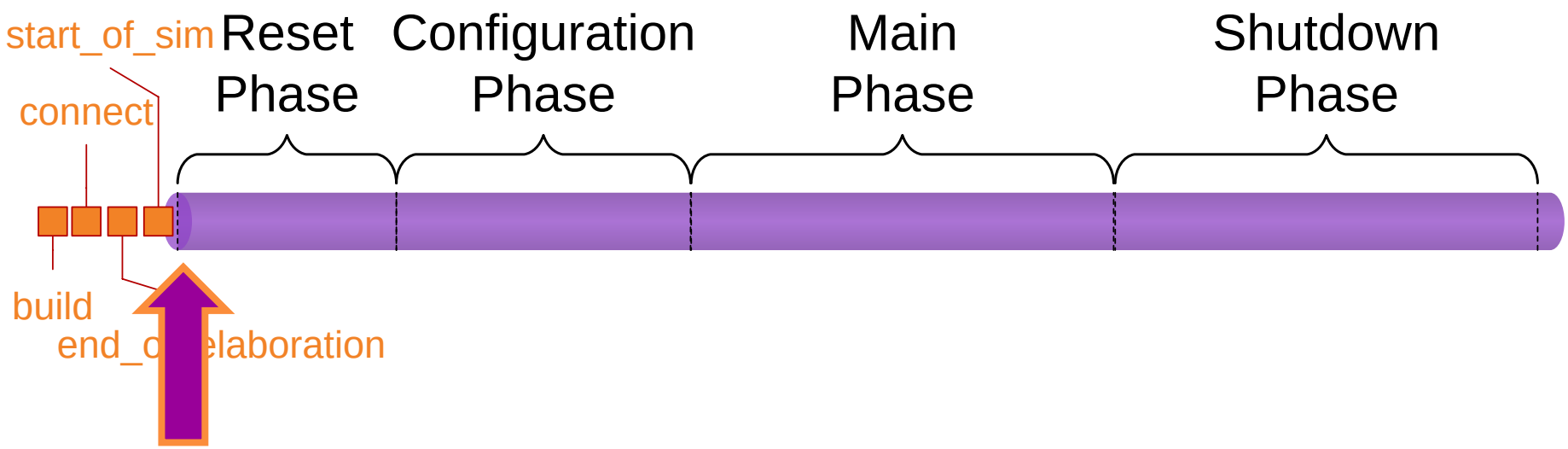
Brian Hunter

Types of Reset Testing


```
class idle_reset_test_c extends basic_test_c;
   `uvm_component_utils(idle_reset_test_c)

   int run_count; // The number of times the test should run

   function new(string name="idle_reset",
                uvm_component parent=null);
      super.new(name, parent);
   endfunction : new

   virtual function void phase_ready_to_end(uvm_phase phase);
      super.phase_ready_to_end(phase);
      if(phase.get_imp() == uvm_shutdown_phase::get()) begin
         if(run_count == 0) begin
            phase.jump(uvm_pre_reset_phase::get());
            run_count++;
         end
      end
   endfunction : phase_ready_to_end
endclass : idle_reset_test_c
```

Types of Reset Testing

- Active Reset Testing



Brian Hunter

CAVIUM

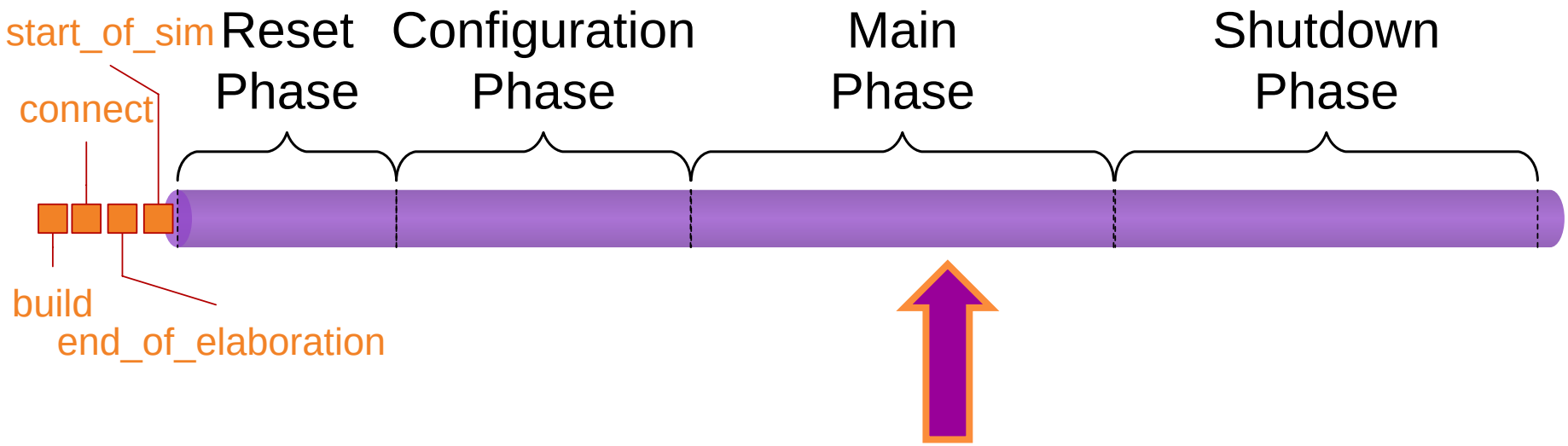
Types of Reset Testing


```
virtual task main_phase(uvm_phase phase);
   fork
      super.main_phase(phase);
   join_none

   if(hit_reset) begin
      phase.raise_objection(this);
      std::randomize(reset_delay_ns) with {
         reset_delay_ns inside {[1000:4000]};
      };
      #(reset_delay_ns * 1ns);
      phase.drop_objection(this);
      phase.get_objection().set_report_severity_id_override(
         UVM_WARNING, "OBJTN_CLEAR", UVM_INFO);
      phase.jump(uvm_pre_reset_phase::get());
      hit_reset = 0;
   end
endtask : main_phase
```

Brian Hunter

Types of Reset Testing

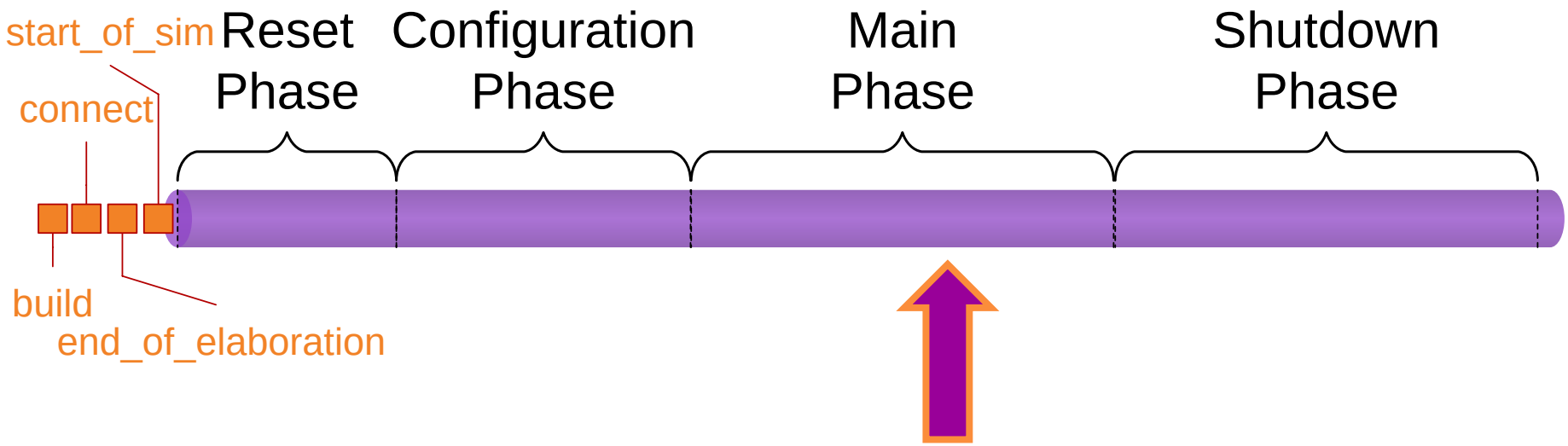
- Soft Reset Testing



Brian Hunter

Types of Reset Testing

- Soft Reset Testing



Brian Hunter

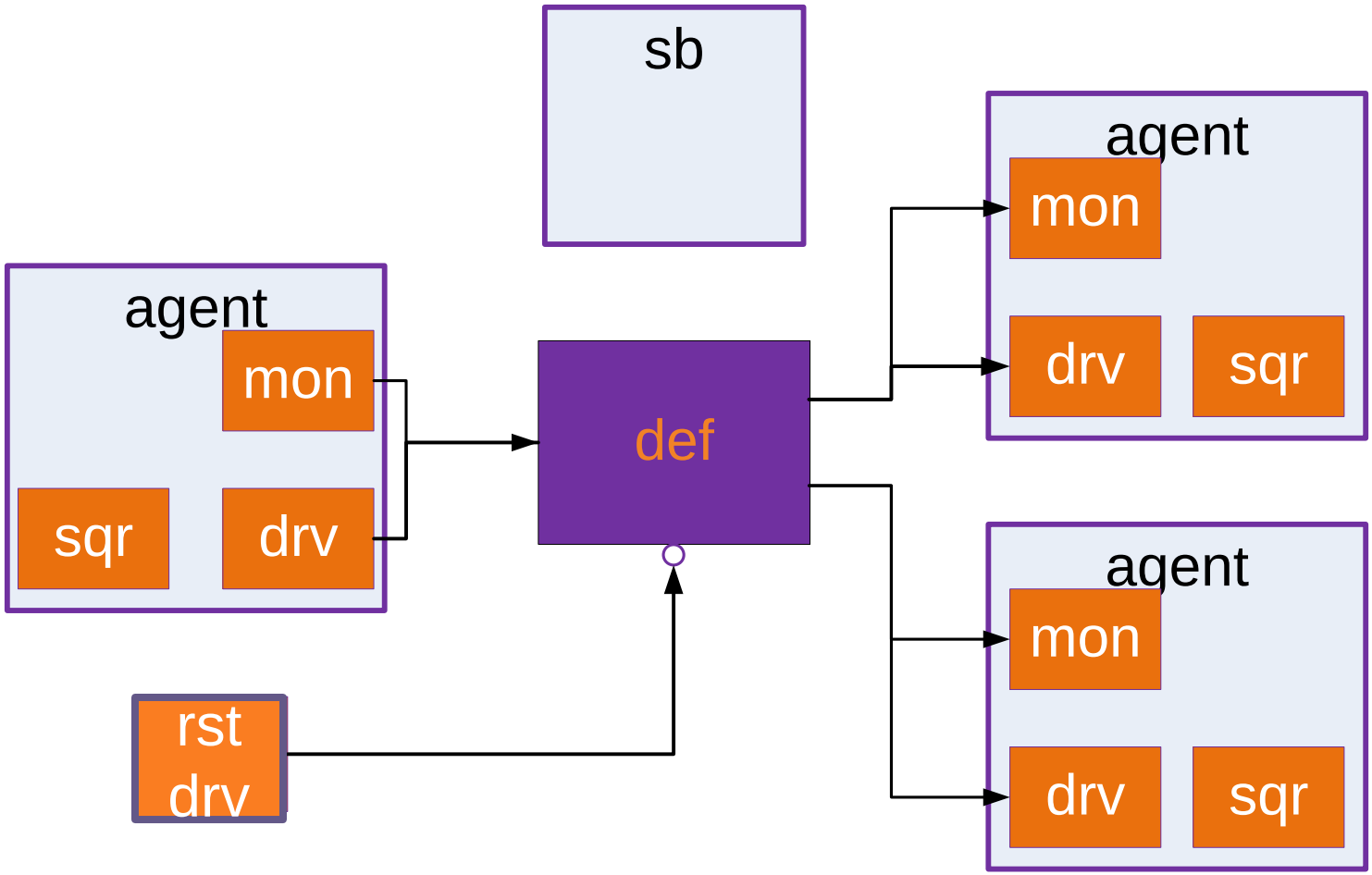
Types of Reset Testing

- Link-Down, Link-Up

Agenda

- Why Test Resets?
- Typical Challenges
- Solution
- Types of Reset Testing
- **Resetting Components**
- Multi-Domain Resets
- Re-Randomizing on Reset
- Conclusions
- Q&A

Resetting Components



Brian Hunter

Resetting Components

Synopsys Users Group
SILICON VALLEY 2013

```
class rst_drv_c extends uvm_driver;
   `uvm_component_utils_begin(rst_drv_c)
      `uvm_field_string(intf_name,   UVM_DEFAULT)
      `uvm_field_int(reset_time_ps,  UVM_DEFAULT)
   `uvm_component_utils_end

   // var: rst_vi
   // Reset virtual interface
   virtual rst_intf rst_vi;

   // var: intf_name
   string intf_name = "rst_i";

   virtual function void build_phase(uvm_phase phase);
      super.build_phase(phase);
      // get the interface
      uvm_resource_db#(virtual rst_intf)::read_by_name("rst_intf",
                                          intf_name, rst_vi)
   endfunction : build_phase
```

CAVIUM

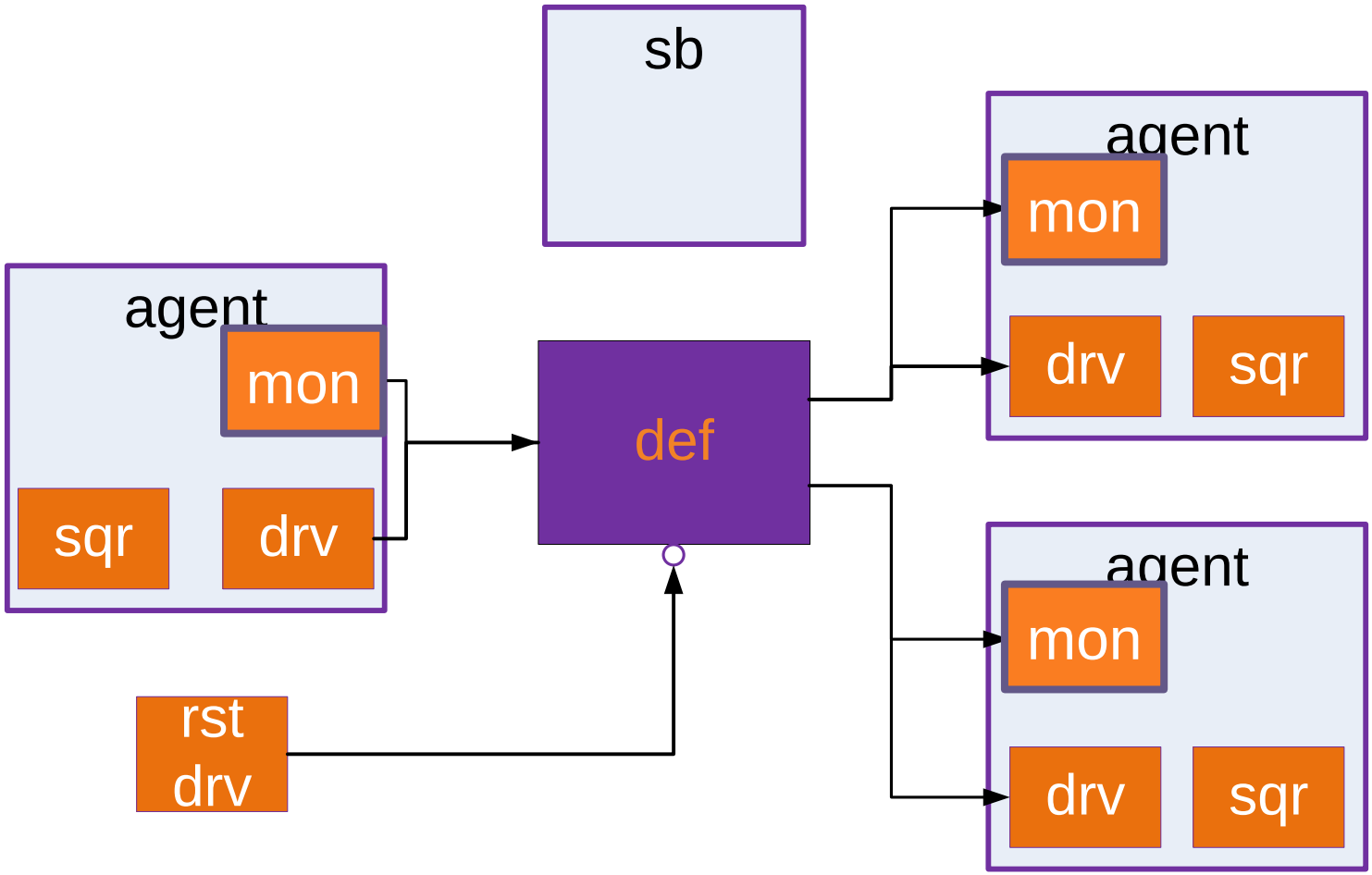
Resetting Components


```
   // var: reset_time_ps
   // The length of time, in ps, that reset will stay active
   rand int reset_time_ps;

   // Base constraints
   constraint rst_cnstr { reset_time_ps inside {[1:1000000]}; }

   virtual task reset_phase(uvm_phase phase);
      phase.raise_objection(this);
      rst_vi.rst_n <= 0;
      #(reset_time_ps * 1ps);
      rst_vi.rst_n <= 1;
      phase.drop_objection(this);
   endtask : reset_phase
endclass : rst_drv_c
```

Brian Hunter

CAVIUM

Resetting Components



Brian Hunter

CAVIUM

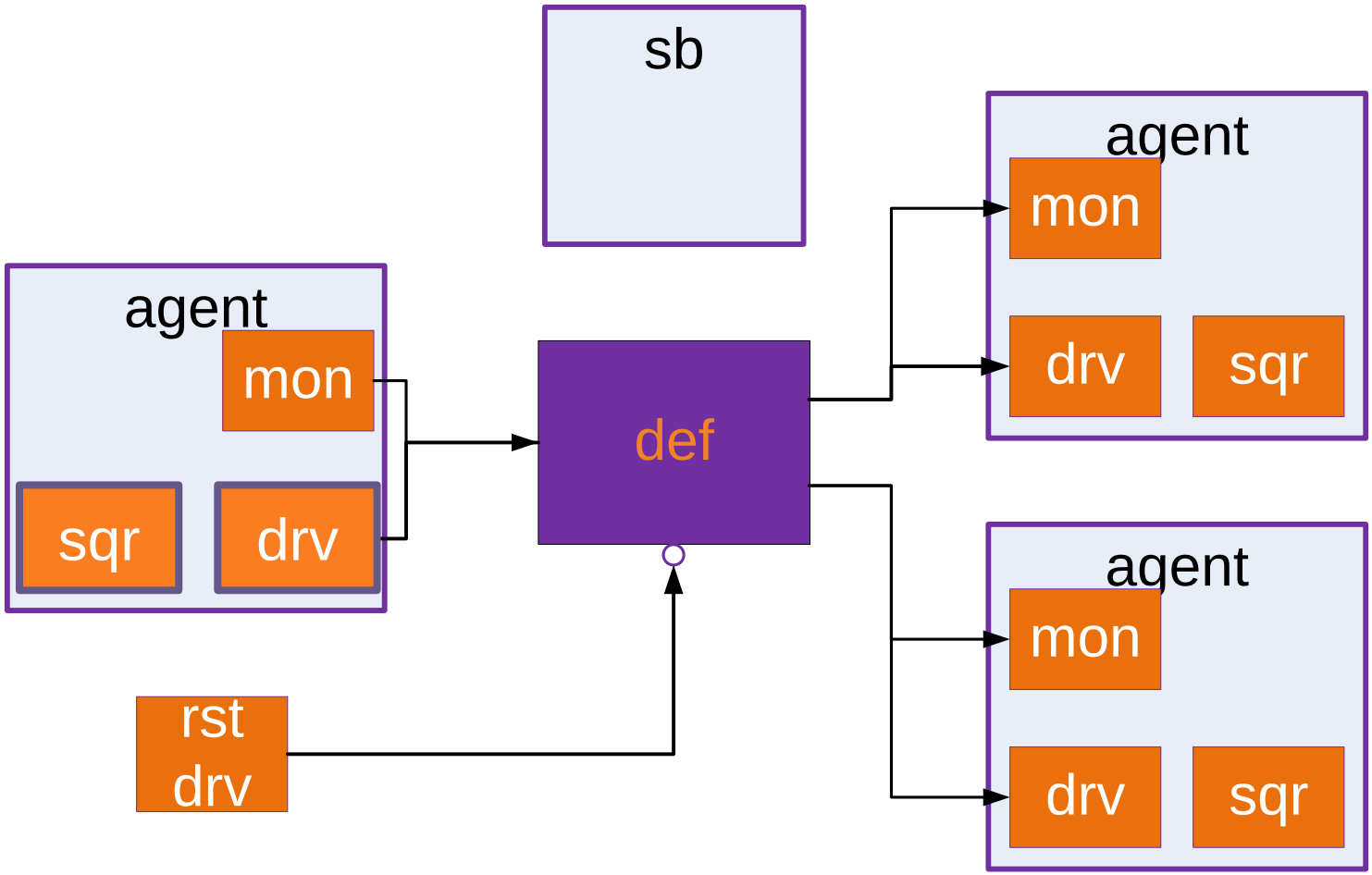
Resetting Components


```
class mon_c extends uvm_monitor;
   `uvm_component_utils(mon_c)

   ...

   virtual task run_phase(uvm_phase phase);
      forever begin
         @(posedge my_vi.rst_n);

         fork
            monitor_items();
         join_none

         @(negedge my_vi.rst_n);
         disable fork;
         cleanup();
      end
   endtask : run_phase
endclass : mon_c
```

Resetting Components


```
class drv_c extends uvm_driver;
   `uvm_component_utils(drv_c)

   event reset_driver;
   ...

   virtual task run_phase(uvm_phase phase);
      forever begin
         @(posedge my_vi.rst_n);

         fork
            drive_items();
         join_none

         @(reset_driver);
         disable fork;
         cleanup();
      end
   endtask : run_phase
endclass : mon_c
```

Resetting Components



Brian Hunter

Resetting Components


```
class agent_c extends uvm_agent;
   `uvm_component_utils(agent_c)
   sqr_c sqr;
   drvc drv;
   ...
   virtual task pre_reset_phase(uvm_phase phase);
      if(sqr && drv) begin
         sqr.stop_sequences();
         ->drv.reset_driver;
      end
   endtask : pre_reset_phase
endclass : agent_c
```

Brian Hunter

Resetting Components

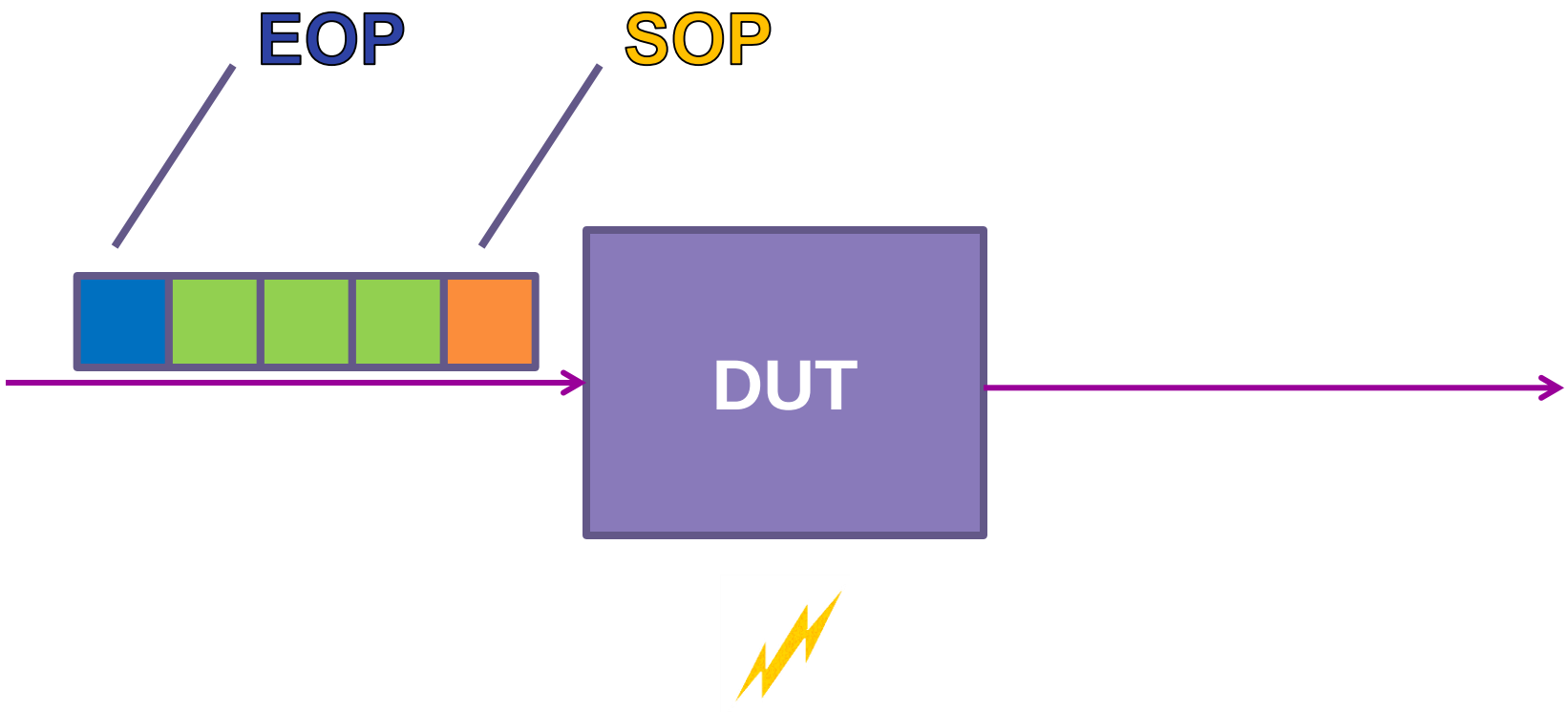
- Scoreboards and Predictors are also components
- Simply erase the contents of expected queues upon detecting resets

```
class sb_c extends uvm_scoreboard;
   `uvm_component_utils(sb_c)

   // expected packets
   pkt_c exp_pkts[$];

   // clear expected upon reset
   virtual task pre_reset_phase(uvm_phase phase);
      exp_pkts.delete();
   endtask : pre_reset_phase
endclass : sb_c
```

Resetting Components

- Some predictions can be very difficult
  - In one simulation, no packets come out after resets
  - In another simulation, the first expected packet happens to come out on the same cycle as the pre_reset phase
- All tests must be self-checking. All tests must pass.
- Try a *ternary* scoreboard
- Mark traffic as *unpredictable*

Resetting Components



Brian Hunter
CAVIUM

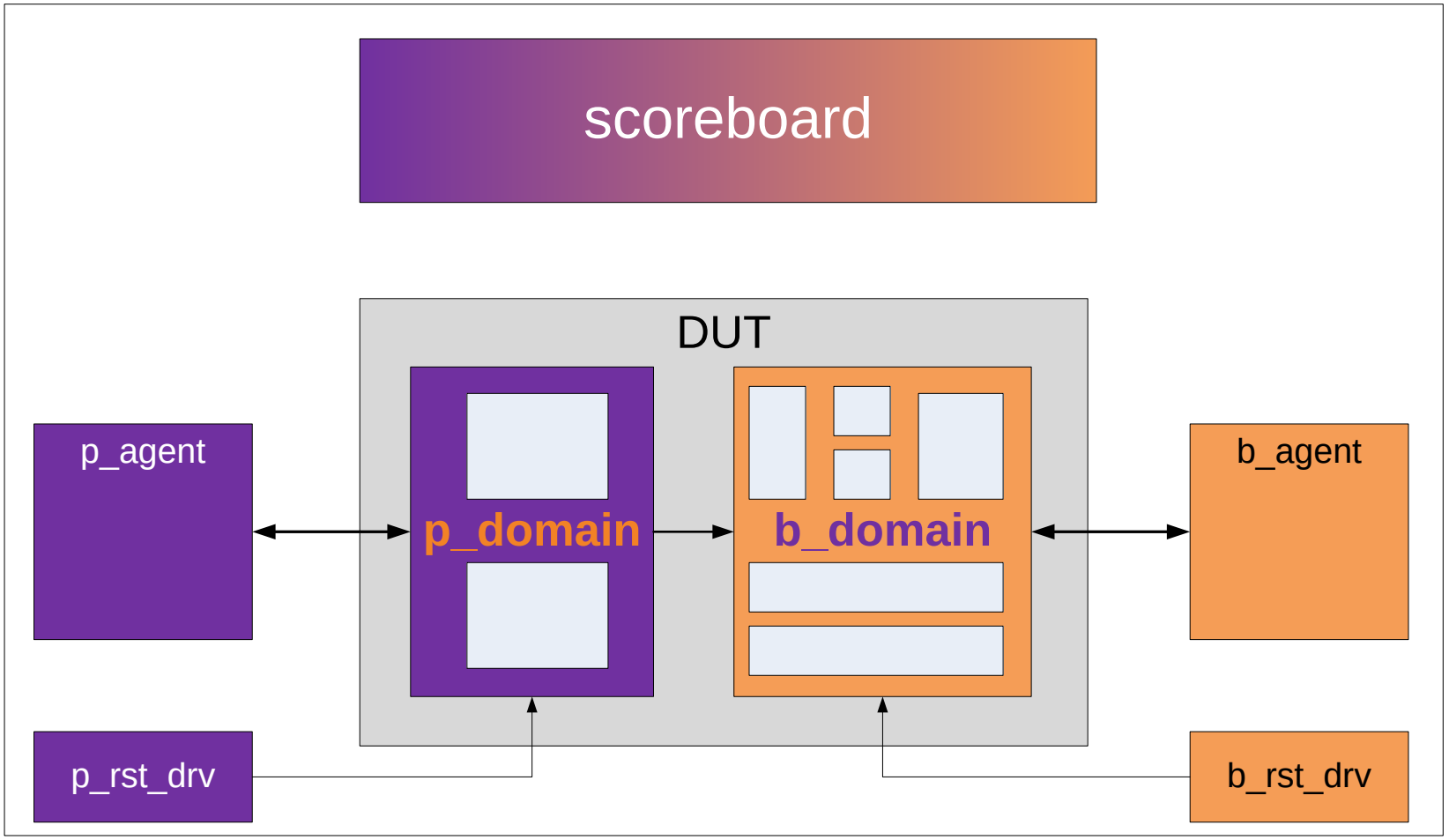
Resetting Components


```
class sb_c extends uvm_scoreboard;
   `uvm_component_utils(sb_c)

   uvm_analysis_imp_rcvd_pkt #(pkt_c, sb_c) rcvd_pkt_imp;
   pkt_c exp_pkts[$];

   // mark all outstanding packets as unpredictable
   virtual task pre_reset_phase(uvm_phase phase);
      foreach(exp_pkts[num])
         exp_pkts[num].unpredictable = 1;
   endfunction : write_soft_reset

   function void write_rcvd_pkt(pkt_c _pkt);
      pkt_c exp_pkt = exp_pkts.pop_front();
      // unpredictable packets are ignored
      if(exp__pkt.unpredictable)
         return;
      else if(exp_pkt.compare(_pkt) == 0)
            `uvm_error(get_full_name(), "Packet Miscompare.")
   endfunction : write_rcvd_pkt
endclass : sb_c
```

Agenda

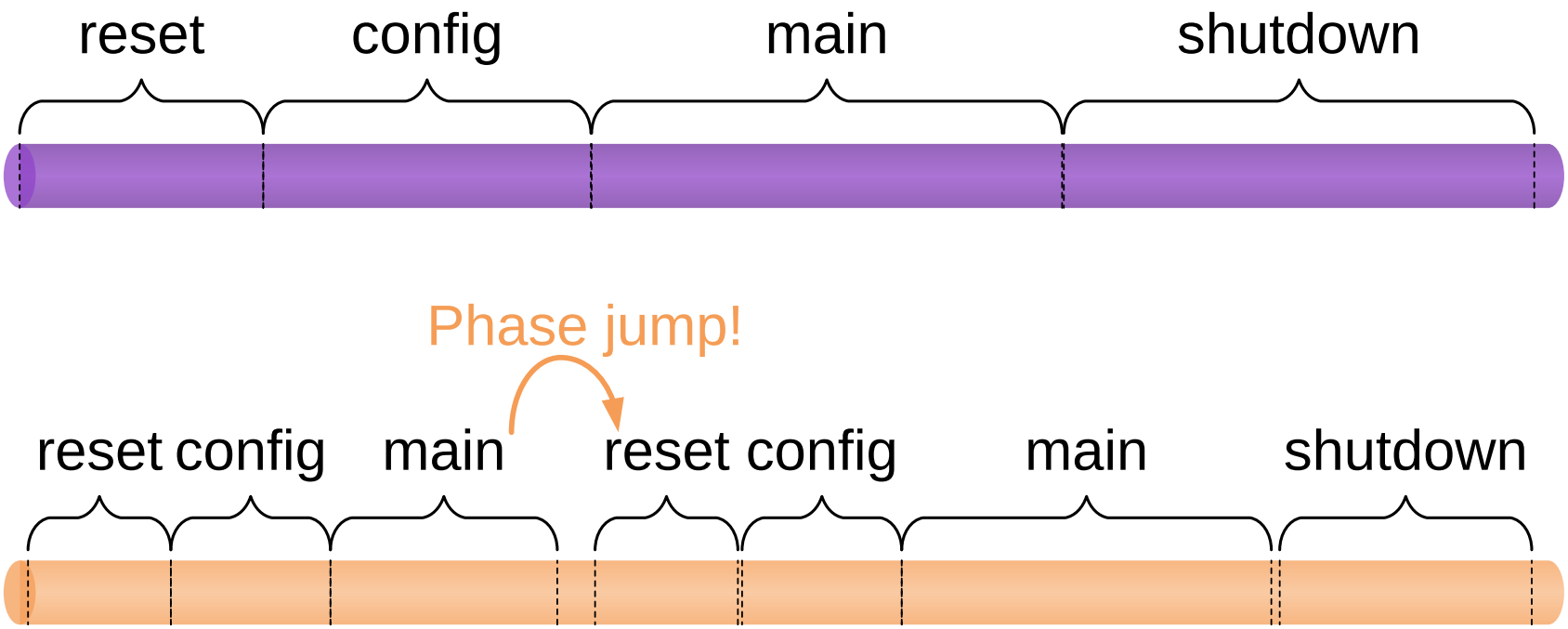
- Why Test Resets?
- Typical Challenges
- Solution
- Types of Reset Testing
- Resetting Components
- **Multi-Domain Resets**
- Re-Randomizing on Reset
- Conclusions
- Q&A

Multi-Domain Resets



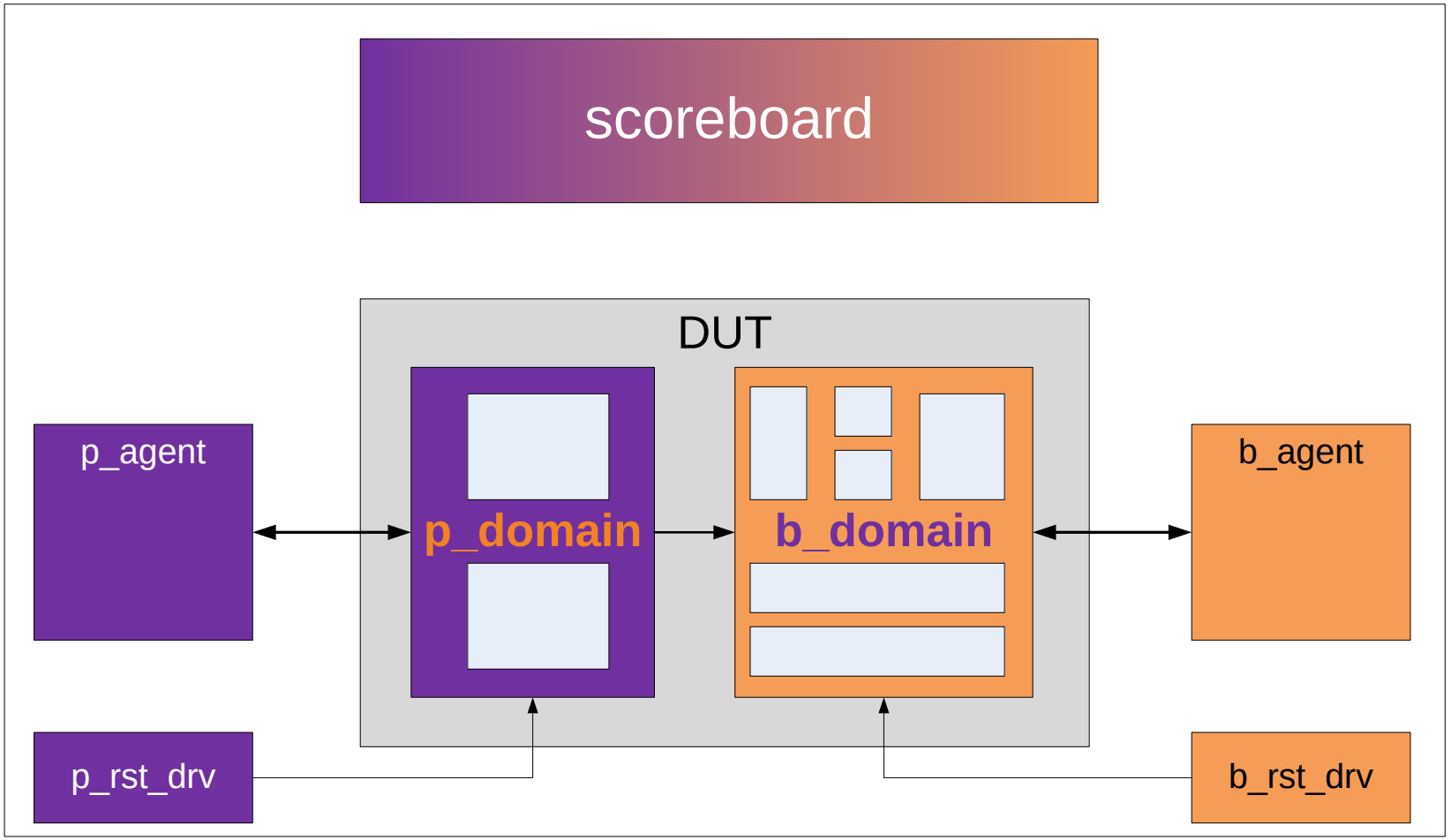
Brian Hunter

Multi-Domain Resets



Brian Hunter

Multi-Domain Resets



Brian Hunter

Multi-Domain Resets


```
virtual task main_phase(uvm_phase phase);
      fork
         super.main_phase(phase);
      join_none

      if(run_count == 0) begin
         phase.raise_objection(this);
         randomize();
         #(reset_delay_ns * 1ns);
         phase.drop_objection(this);
         p_domain.jump(uvm_pre_reset_phase::get());
         run_count++;

         // tell scoreboard that a reset occurred
         -> scoreboard.p_domain_reset;
      end
   endtask : main_phase
```

Agenda

- Why Test Resets?
- Typical Challenges
- Solution
- Types of Reset Testing
- Resetting Components
- Multi-Domain Resets
- **Re-Randomizing on Reset**
- Conclusions
- Q&A

Re-Randomizing on Reset

Synopsys Users Group
SILICON VALLEY 2013

```
class cfg_c extends uvm_object;
   // The number of transactions to send
   rand int num_trans;

   // All of the configuration space registers can be randomized
   rand reg_block_c config_space;

   // how fast is your clock?
   rand int period_ps;

   // Run in PCI, PCI/X, or PCI Express modes
   rand pci_mode_e pci_mode;

   // The number of PCI agents to create
   rand num_pci_agents;

endclass : cfg_c
```

Brian Hunter

CAVIUM

Re-Randomizing on Reset


```
class cfg_c extends uvm_object;
   // The number of transactions to send
   rand int num_trans;

   // All of the configuration space registers can be randomized
   rand reg_block_c config_space;

   // how fast is your clock?
   rand int period_ps;

   // Run in PCI, PCI/X, or PCI Express modes
   rand pci_mode_e pci_mode;

   // The number of PCI agents to create
   rand num_pci_agents;

endclass : cfg_c
```

200

device_id = 0x18770020

640

PCIE

4

Brian Hunter

CAVIUM

Re-Randomizing on Reset

Synopsys Users Group
SILICON VALLEY 2013

```
class cfg_c extends uvm_object;
   // The number of transactions to send
   rand int num_trans;                          200

   // All of the configuration space registers can be randomized
   rand reg_block_c config_space;               device_id = 0x1497202

   // how fast is your clock?
   rand int period_ps;                          400

   // Run in PCI, PCI/X, or PCI Express modes
   rand pci_mode_e pci_mode;                    PCIE

   // The number of PCI agents to create
   rand num_pci_agents;                         4

endclass : cfg_c
```

Brian Hunter

CAVIUM

Re-Randomizing on Reset


```
class cfg_c extends uvm_object;
   // The number of transactions to send
   rand int num_trans;

   // All of the configuration space registers can be randomized
   rand reg_block_c config_space;

   // how fast is your clock?
   rand int period_ps;

   // Run in PCI, PCI/X, or PCI
   rand pci_mode_e pci_mode;

   // The number of PCI agents
   rand num_pci_agents;

endclass : cfg_c
```

200

device_id = 0x14972048

490

Structural Variable

Structural Variable

Brian Hunter

CAVIUM

Re-Randomizing on Reset


```
class cfg_c extends uvm_object;
   // The number of transactions to send
   rand int num_trans;

   // All of the configuration space registers can be randomized
   rand reg_block_c config_space;

   //
   ra

   //
   ra

   // The number of PCI agents to create
   rand num_pci_agents;

endclass : cfg_c
```

```
// Ensure that structural variables are only randomized once
function void post_randomize();
   pci_mode.rand_mode(0);
   num_pci_agents.rand_mode(0);
endfunction : post_randomize
```

Brian Hunter

CAVIUM

Re-Randomizing on Reset


```
class active_reset_test_c extends base_test_c;
   rand cfg_c cfg;

   virtual task pre_reset_phase(uvm_phase phase);
      randomize();
   endtask : pre_reset_phase

   virtual task main_phase(uvm_phase phase);
      fork
         super.main_phase(phase);
      join_none

      if(hit_reset) begin
         phase.raise_objection(this);
         std::randomize(reset_delay_ns) with {
            reset_delay_ns inside {[1000:4000]};
         };
         #(reset_delay_ns * 1ns);
```

Brian Hunter

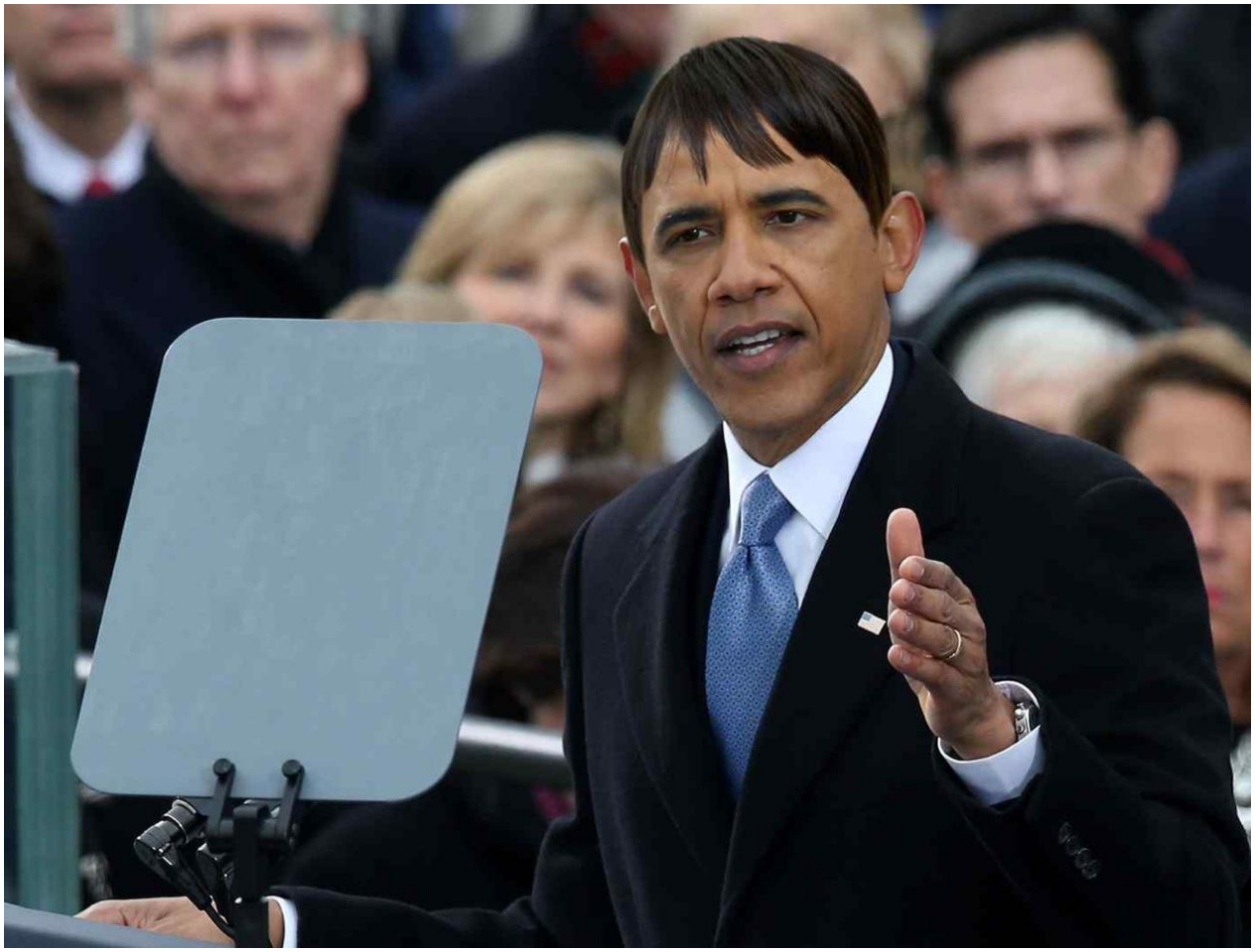
Conclusions

- Functional reset testing is getting more important
- Reset testing used to be very complicated
- UVMs phases and phase jumping make these tests easy-peasy
- Some *minor* component changes need to be made
- If you avoid the use of structural variables, you can re-randomize everything all in one simulation!

Brian Hunter

CAVIUM

Q&A



Brian Hunter