# UVM stimulus – driving the DUT from within!

## (a.k.a Faster-Better-Cheaper Verification)

Elihai Maicas

Intel Corp.

**Technical committee special award**

June 6, 2015

SNUG Israel

# Agenda

Background

Motivation

The problems

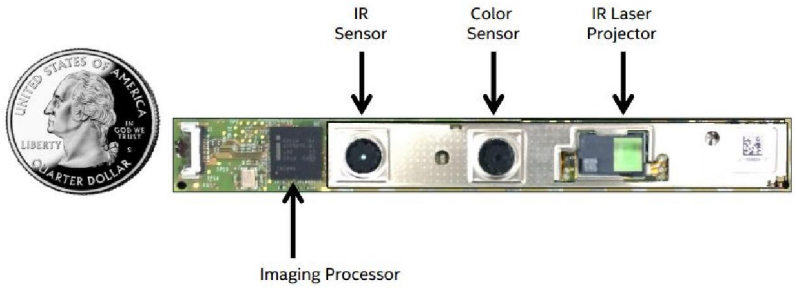
Brute-force solutions

Our solution - Internal DUT stimulus

Summary

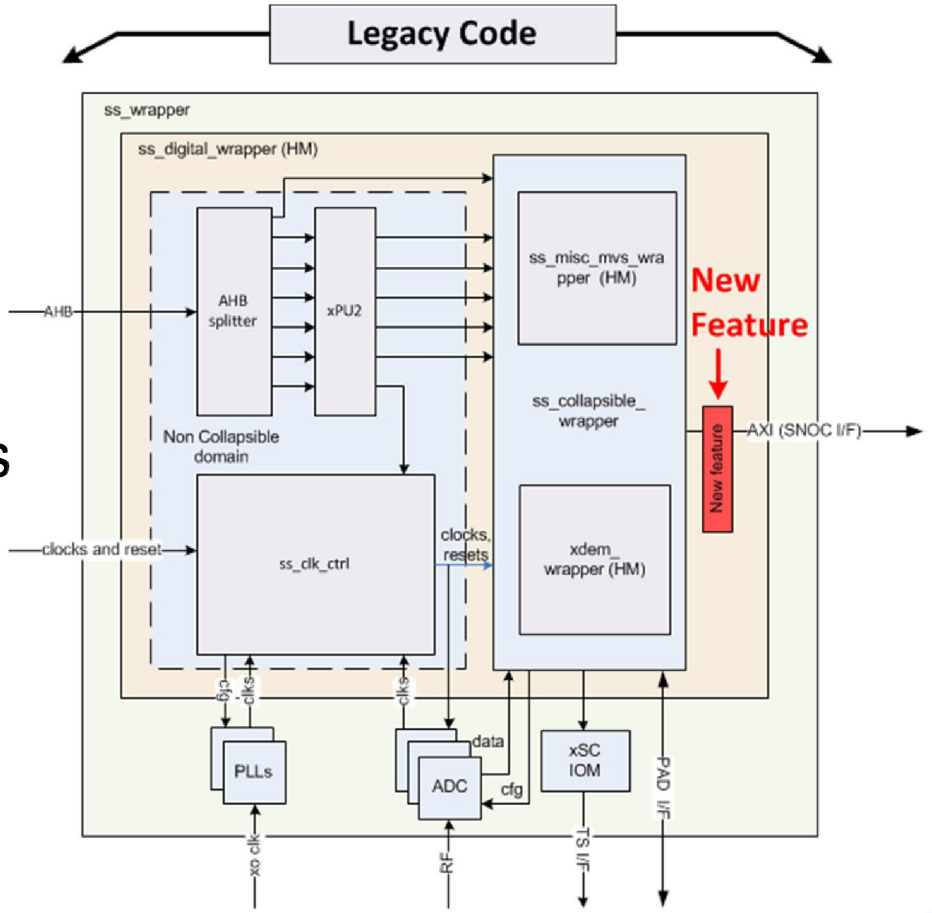
# Background

- Intel® RealSense™ camera fits remarkable technology into a small package. There are three cameras that act like one - a 1080p HD camera, an infrared camera, and an infrared laser projector - they “see” like the human eye to sense depth and track human motion



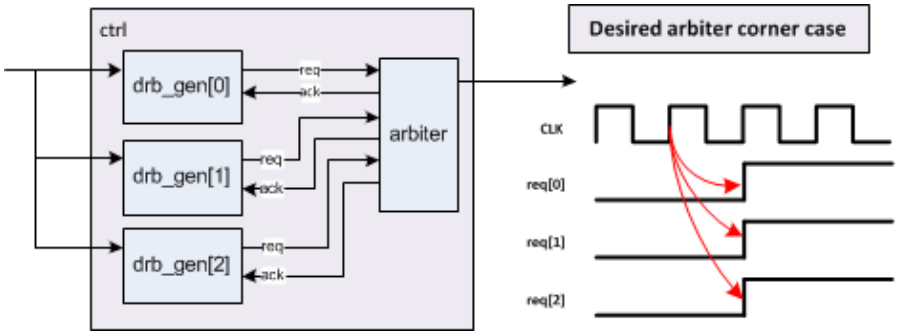
# Motivation

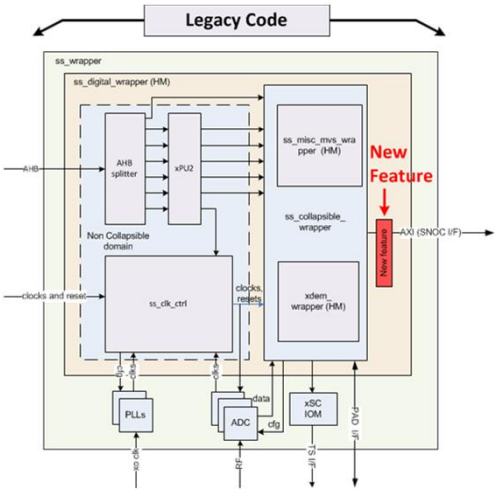
- Lower the verification overhead for IP reuse
- Minor changes between flavors should no longer be major verification headaches



# The problems

- Reach the corner cases of the new feature

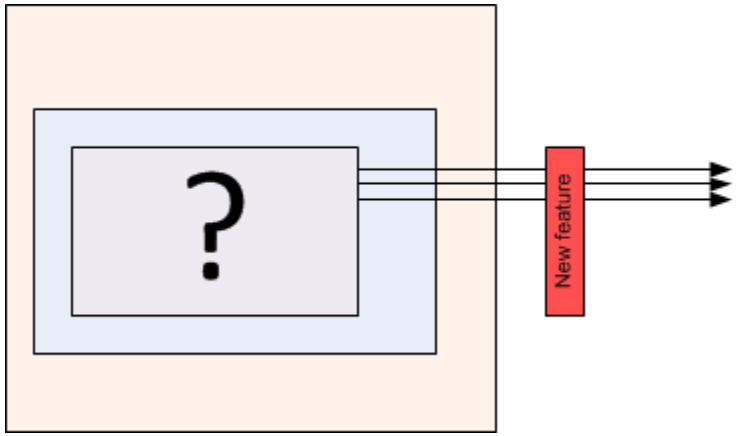


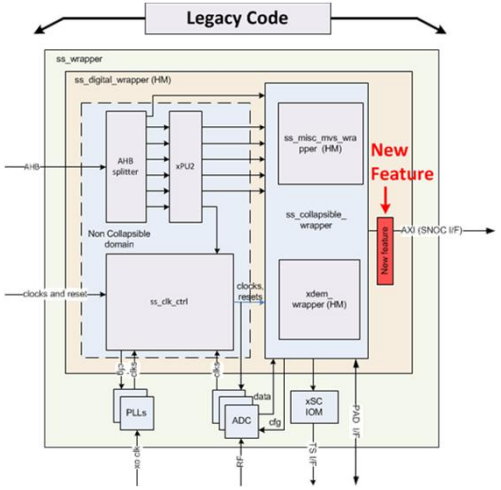




- Error injection – “Who will watch the watchmen?”

# Brute-force solutions

- Unit level testbench -
  optimal, but sometimes cannot be justified

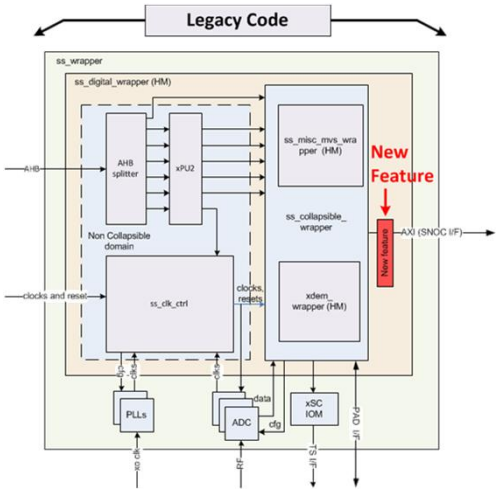


# Brute-force solutions

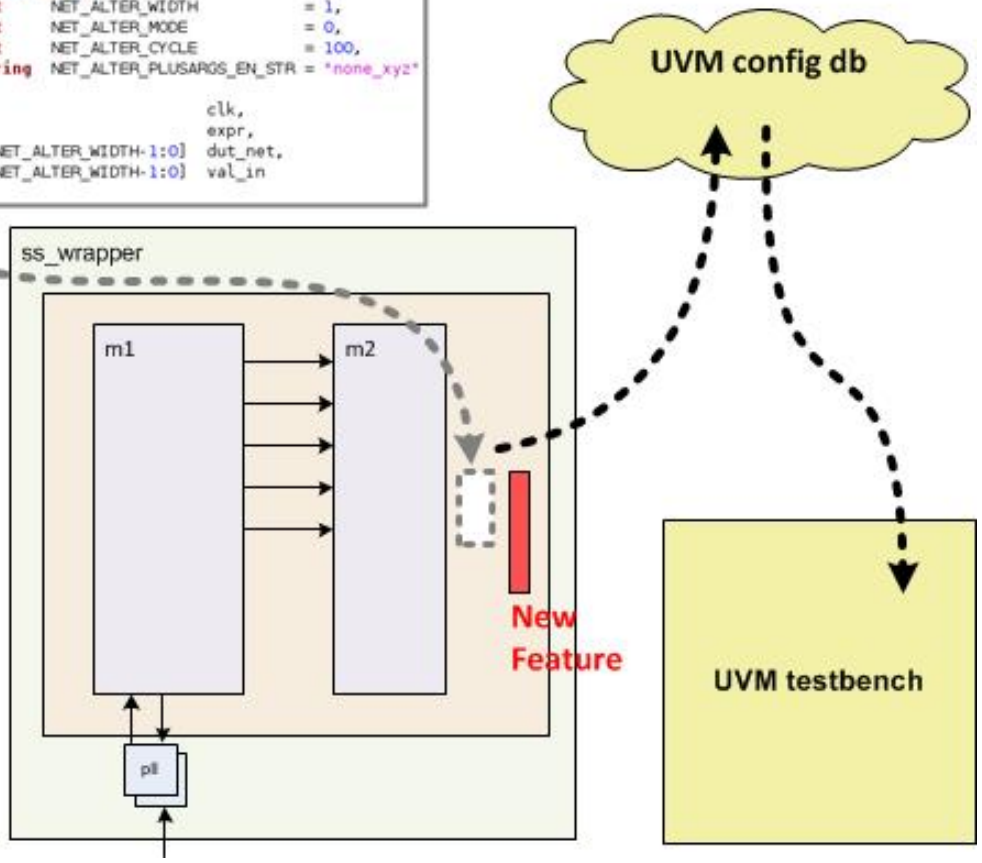
- Regressions -
  not best for all corner cases
  no full controllability on IP behavior

- Force-release statements -
  cumbersome
  hard to reuse
  bad performance



# Internal DUT stimulus

```
interface net_alter_if #(
  parameter int      NET_ALTER_WIDTH          = 1,
  parameter int      NET_ALTER_MODE           = 0,
  parameter int      NET_ALTER_CYCLE          = 100,
  parameter string   NET_ALTER_PLUSARGS_EN_STR = "none_xyz"
) (
  input wire                          clk,
  input wire                          expr,
  inout wire [NET_ALTER_WIDTH-1:0]    dut_net,
  input wire [NET_ALTER_WIDTH-1:0]    val_in
);
```



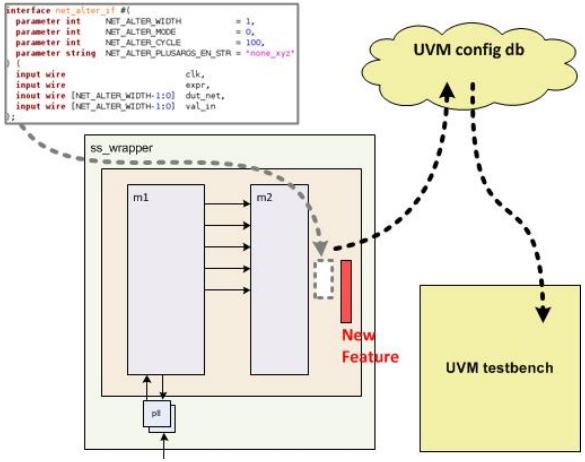
# Internal DUT stimulus

## How is it done

# Internal DUT stimulus

## Step I

- Bind a parameterized SV interface



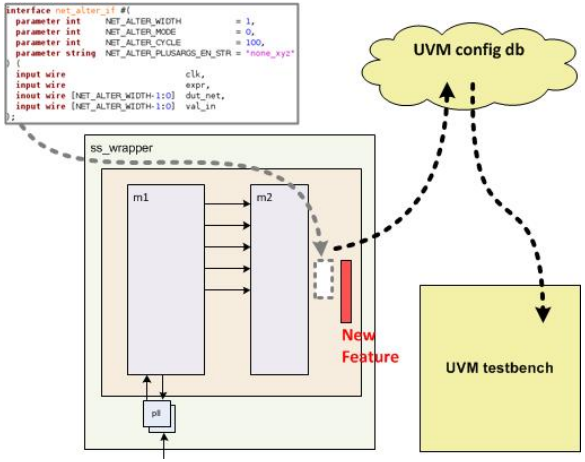

```
// a fifo_if interface is instanced in every instance of fifo module
bind fifo fifo_if fifo_if__bind(.*);

// a fifo_if interface is instanced in the listed instances of fifo
bind fifo: fifo1, fifo2  fifo_if fifo_if__bind(.*);
```

# Internal DUT stimulus

## Step II – option 1

- Register the interface to the uvm_config_db

  - Using the wrapper module

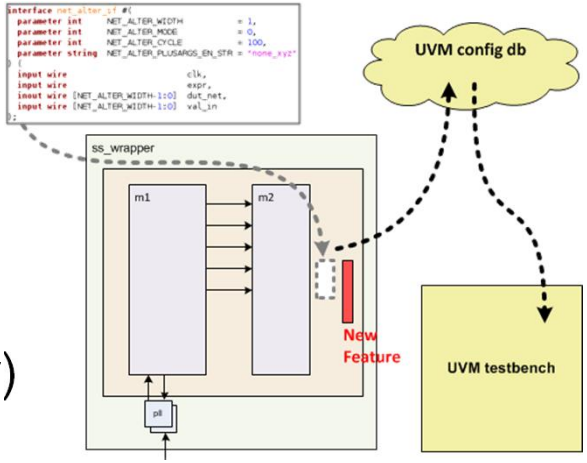



```
module if_wrap(/*  */);
  string loc_hier = $psprintf("%m");
  net_alter_if net_alter_if(/*  */);
  .
  uvm_config_db#(virtual net_alter_if#(...))::set
  (uvm_root::get(),"*", loc_hier, net_alter_if);
```

# Internal DUT stimulus

## Step II – option 2

- Register the interface to the uvm_config_db



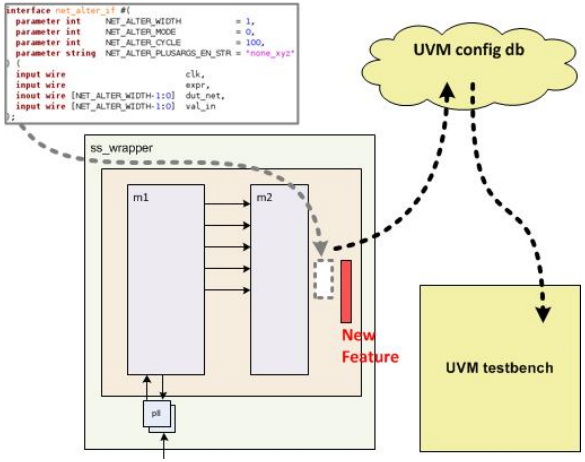

- Using the `::self` syntax
  (similar to `this` of OOP; Synopsys only)

```
net_alter_if (/*  */);
  string loc_hier = $psprintf("%m");
  .
  .
  uvm_config_db#(virtual net_alter_if#(...))::set
  (uvm_root::get(),"*", loc_hier, interface::self());
```

# Internal DUT stimulus

## Step III

- Get the interface in a designated agent

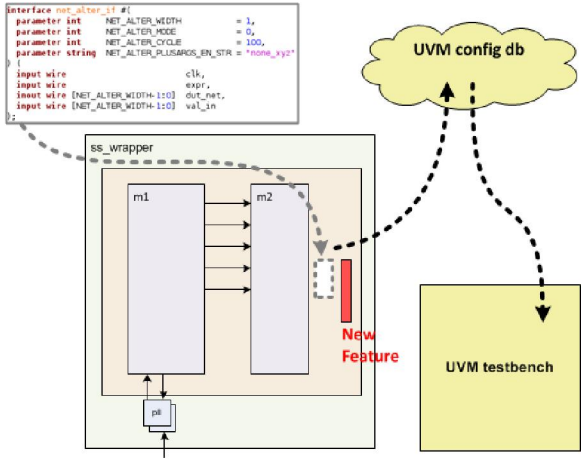


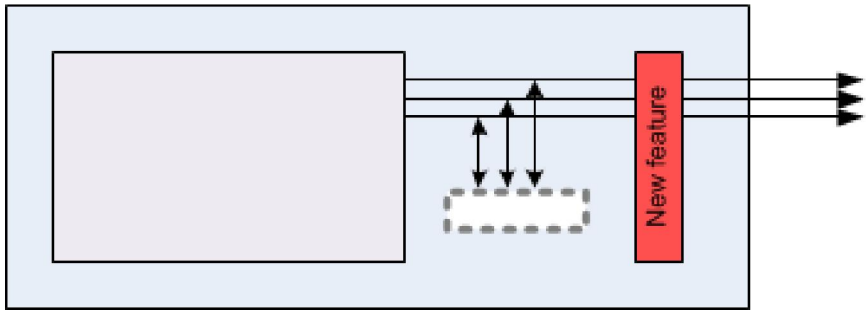

```
uvm_config_db#(virtual net_alter_if#(...))::get(this, "",
m_net_alter_if_bind_name, m_net_alter_vif);
```

# Internal DUT stimulus

## Final result

- The ports of the new module are monitored and driven internally by the bound interface



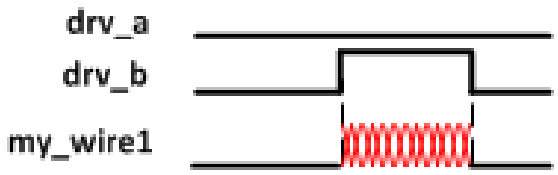

- Multiple drivers to the same net?

# Internal DUT stimulus

## Multiple drivers to the same net …

- Standard assignment

```
wire  my_wire1;
logic drv_a = 0;
logic drv_b;

initial begin
  drv_b = 1'bz; #100;
  drv_b = 1'b1; #100;
  drv_b = 1'bz;
end

assign my_wire1 = drv_a;
assign my_wire1 = drv_b;
```

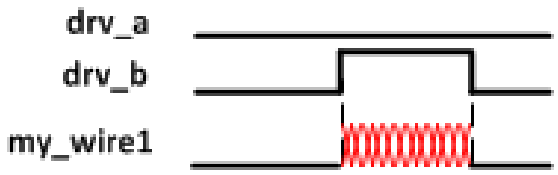


# Internal DUT stimulus

## Multiple drivers to the same net …

- Standard assignment

```
wire  my_wire1;
logic drv_a = 0;
logic drv_b;

initial begin
  drv_b = 1'bz; #100;
  drv_b = 1'b1; #100;
  drv_b = 1'bz;
end

assign my_wire1 = drv_a;
assign my_wire1 = drv_b;
```




- Strength aware assignment

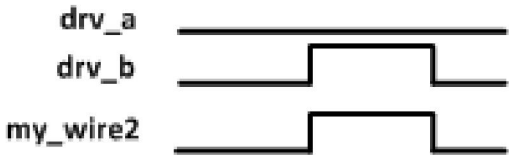
```
wire  my_wire2;
logic drv_a = 0;
logic drv_b;

initial begin
  drv_b = 1'bz; #100;
  drv_b = 1'b1; #100;
  drv_b = 1'bz;
end

assign my_wire2 = drv_a;
assign (supply0,supply1)
my_wire2 = drv_b;
```

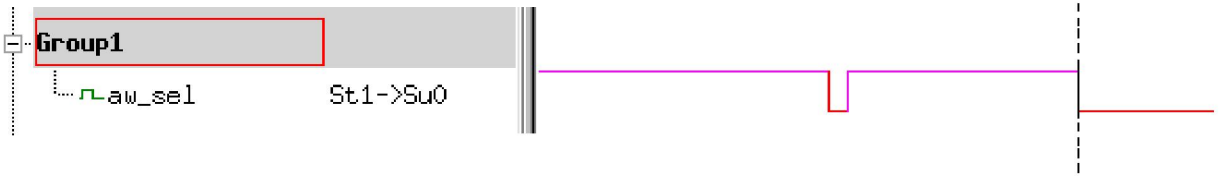
# Internal DUT stimulus

## Multiple drivers to the same net …

- Strength aware assignment

```
wire  my_wire2;
...
assign                        my_wire2 = drv_a;

assign (supply0,supply1) my_wire2 = drv_b;
```



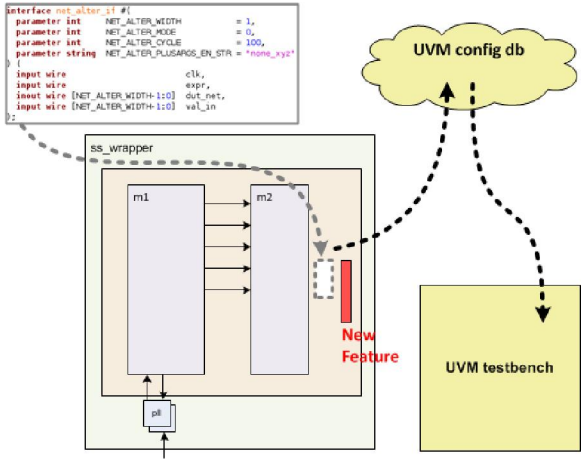
# Internal DUT stimulus

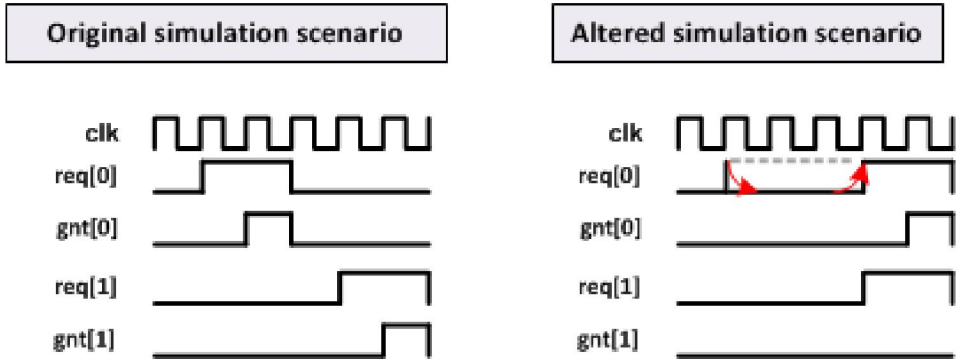
## What can be achieved

# Internal DUT stimulus

## Manipulate the arbiter…

- Reach the desired rare corner case in the arbiter

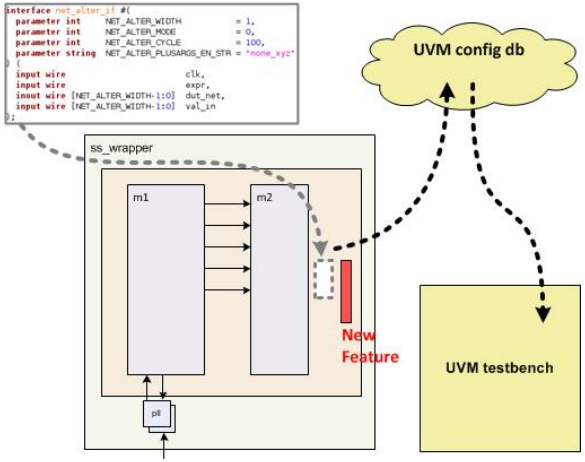


# Internal DUT stimulus

## Manipulate an AXI burst

- Reach desired rare type of AXI burst

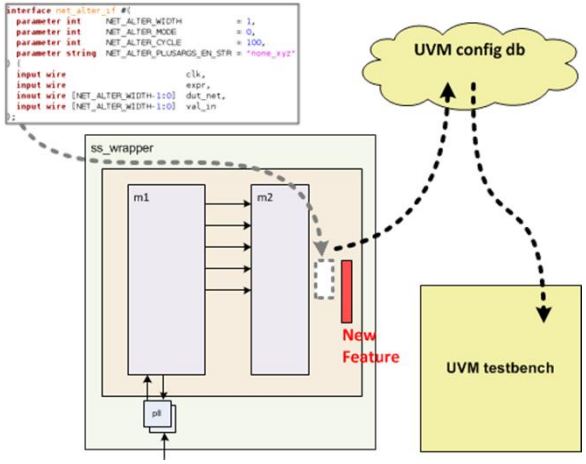



| | | | | | |
|---|---|---|---|---|---|
| **Original Data Stream:** | Data burst | Data burst | Data burst | Data burst | Data burst |
| **Altered Data Stream:** | Data burst | Event Burst | Data burst | Data burst | Data burst |

# Internal DUT stimulus

## Autonomous error injection

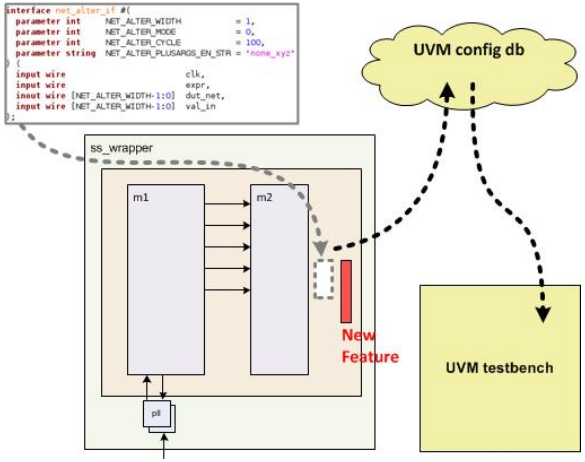
```
loc_net = {NET_WIDTH{1'bz}};
forever begin
  repeat (ALTER_FREQ-1) @(posedge expr);

  case (ALTER_MODE)
    0:  loc_net = ~dut_net;
    1:  loc_net = {NET_WIDTH{1'bx}};
    2:  loc_net = val_in;
  endcase
  @(negedge expr);
  loc_net = {NET_WIDTH{1'bz}};
end
```



# Internal DUT stimulus

## Summary

- Verification at the unit level is optimal but not always possible




- Using internal DUT stimulus we can achieve a unit-level verification while running in a subsystem level environment

- In addition, this method is a good solution for cluster to top verification reuse

# Thank You

# Backup

# Internal DUT stimulus

## Unit to top reuse – conventional way

- tb_top code

```
 6 alg_pipe_common_if fa_out_if();
 7
 8 assign fa_out_if.resetn = factl_rst_al_n;
 9 assign fa_out_if.clk    = alclk_fa;
10
11 `ifdef FC_SIM
12 assign fa_out_if.data    = `PATH_MC_CORE.mc_alg_i.fa_db_fa_amp_vcmd;
13 assign fa_out_if.data_en = `PATH_MC_CORE.mc_alg_i.fa_db_prt3_val;
14
15 `else  // unit level
16 force  `PATH_MC_CORE.mc_alg_i.fa_db_fa_amp_vcmd = fa_out_if.data;
17 force  `PATH_MC_CORE.mc_alg_i.fa_db_prt3_val    = fa_out_if.data_en;
18
19 `endif
```

# Internal DUT stimulus

## Unit to top reuse – bind interface working example

- tb_top code

```
 6 bind `PATH_MC_CORE.mc_alg_i alg_pipe_common_if fa_out_dut_bind_if (
 7   .config_db_id ("fa_out_dut"           ),
 8   .resetn       (factl_rst_al_n         ),
 9   .clk          (alclk_fa               ),
10   .data         (fa_db_fa_amp_vcmd      ),
11   .data_en      (fa_db_prt3_val         )
12 );
13
14 bind memsctl_alg_gm_i alg_pipe_common_if fa_out_gm_bind_if (
15   .config_db_id ("fa_out_gm"            ),
16   .resetn       (~reset_fa              ),
17   .clk          (clk_fa                 ),
18   .data         ( FA_CL_Out             ),
19   .data_en      (clk_enable_fa          )
20 );
21
22 ////////////////////////////////////////////////////////////////////
23 /////////////////        fa filter test out      ///////////////////
24 ////////////////////////////////////////////////////////////////////
25
26 bind `PATH_MC_CORE.mc_alg_i alg_pipe_common_if fa_filter_test_out_dut_bind_if (
27   .config_db_id ("fa_filter_test_out_dut" ),
28   .resetn       (factl_rst_al_n           ),
29   .clk          (alclk_fa                 ),
30   .data         (filter_sa_test_out       ),
31   .data_en      (filter_sa_test_out_val   )
32 );
33
```

# Internal DUT stimulus

## Unit to top reuse – bind interface working example

- Interface code

```
1 //------------------------------------------------------------------------------
2 // interface alg_pipe_common_if
3 //------------------------------------------------------------------------------
4
5 interface alg_pipe_common_if (
6   input [256*8-1:0] config_db_id,                  // Interface ID for uvm_config_db registration
7   input             resetn,                        // Main reset port
8   input             clk,                           // Main clk port
9   inout [31:0]      data,                          // Main data port. Used to monitor and/or drive data to the dut
0   inout             data_en                        // Data enable port
1 );
2
3
4   logic [31:0]  data_drv           = 32'hz;       // Local data variable, to be toggled by the driver
5   logic         data_en_drv        = 1'hz;        // Local data enable variable
6
7   assign (supply1,supply0) data    = data_drv;    // Drive data port by local variable
8   assign (supply1,supply0) data_en = data_en_drv; // Drive data enable port by local variable
9
0
1   //------------------------------------------------------------------------------
2   // Self registration to interface uvm_config_db
3   //------------------------------------------------------------------------------
4   initial begin
5     @(config_db_id);
6     uvm_config_db#(virtual alg_pipe_common_if)::set(uvm_root::get(),"*", string'(config_db_id), interface::self());
7   end
```

# Internal DUT stimulus

## Unit to top reuse – bind interface working example

- Interface code

```
41   //---------------------------------------------------------------------------
42   // Clocking block and modport for the driver
43   //---------------------------------------------------------------------------
44   clocking bfm_cb @ (posedge clk);
45     default input #1ps output #1ps;
46     output  data_drv;
47     output  data_en_drv;
48     input   data;
49   endclocking: bfm_cb
50
51   modport bfm_mp (clocking bfm_cb, input resetn);
52
53
54   //---------------------------------------------------------------------------
55   // Clocking block and modport for the monitor
56   //---------------------------------------------------------------------------
57   clocking mon_cb @ (posedge clk);
58     default input #1ps output #1ps;
59     input  data;
60     input  data_en;
61     input  ts_cntr;
62   endclocking: mon_cb
63
64   modport mon_mp (clocking mon_cb, input resetn);
65
66
67   // sva checker instance
68   alg_pipe_common_sva_checker sva_checker (.*);
69
70 endinterface : alg_pipe_common_if
```

# Internal DUT stimulus

## Unit to top reuse – bind interface working example

- Agent code

```
27 //------------------------------------------------------------------------------
28 // build_phase()
29 //------------------------------------------------------------------------------
30 function void alg_pipe_common_agent::build_phase(uvm_phase phase);
31
32   virtual interface alg_pipe_common_if vif_loc;
33
34   // get cfg object
35   if(!uvm_config_db #(alg_pipe_common_cfg)::get(this, "", "alg_pipe_cfg", m_cfg)) begin
36     `uvm_error(get_type_name(), "alg_pipe_cfg agent config not found")
37   end
38
39   // get vif
40   if(!uvm_config_db #(virtual alg_pipe_common_if)::get(this, "", m_cfg.m_vif_cfg_db_name, vif_loc)) begin
41     `uvm_error(get_type_name(), $psprintf("%0s vif not found", m_cfg.m_vif_cfg_db_name))
42   end
43
44   // create the monitor
45   if (m_cfg.m_has_monitor) begin
46     m_monitor          = alg_pipe_common_monitor::type_id::create("m_monitor", this);
47     m_monitor.m_vif_mp  = vif_loc;
48   end
49
50   // create sequencer/driver
51   if (m_cfg.m_is_active == UVM_ACTIVE) begin
52     m_sequencer       = alg_pipe_common_sequencer::type_id::create("m_sequencer", this);
53     m_driver          = alg_pipe_common_driver::type_id::create("m_driver", this);
54     m_driver.m_vif_mp = vif_loc;
55   end
```

# Internal DUT stimulus

## Unit to top reuse – bind interface working example

- Driver code (relevant for unit level)

```
 1 //------------------------------------------------------------------------------
 2 // class alg_pipe_common_driver
 3 //------------------------------------------------------------------------------
 4
 5 class alg_pipe_common_driver extends uvm_driver #(alg_pipe_common_sequence_item);
 6   `uvm_component_utils(alg_pipe_common_driver)
 7
 8   alg_pipe_common_cfg                          m_cfg;          // Agent's config object
 9   virtual interface alg_pipe_common_if.bfm_mp m_vif_mp;       // Interface modport
10

50 //------------------------------------------------------------------------------
51 // run_phase()
52 //------------------------------------------------------------------------------
53 task alg_pipe_common_driver::run_phase(uvm_phase phase);
54
55   bfm_init();
56
57   forever begin
58     seq_item_port.try_next_item(req);
59
60     if (req == null) begin
61       drive_erratic_data(m_cfg.m_non_valid_data_drv_type);
62       @(m_vif_mp.bfm_cb);
63     end else begin
64       cnt_hold_data = m_cfg.m_num_of_hold_data_after_valid;
65       process_item(req);
66       seq_item_port.item_done();
67     end
68
69   end
70
71 endtask : run_phase
72
```

# Internal DUT stimulus

## Unit to top reuse – bind interface working example

- Driver code (relevant for unit level)

```
 86 //------------------------------------------------------------------------------
 87 // process_item()
 88 //------------------------------------------------------------------------------
 89 task alg_pipe_common_driver::process_item(alg_pipe_common_sequence_item req);
 90
 91   foreach (req.m_data[ii]) begin
 92
 93     //---------------------------------------
 94     // delay before data valid
 95     //---------------------------------------
 96     repeat (req.m_delay_bw_valid[ii]) begin
 97       drive_erratic_data(m_cfg.m_non_valid_data_drv_type);
 98       @(m_vif_mp.bfm_cb);
 99     end
100
101     //---------------------------------------
102     // drive valid data
103     //---------------------------------------
104     m_vif_mp.bfm_cb.data_drv    <= req.m_data[ii];
105     m_vif_mp.bfm_cb.data_en_drv <= 1'b1;
106     @(m_vif_mp.bfm_cb);
107
108
109   end
110
111 endtask : process_item
```

# Internal DUT stimulus

## Unit to top reuse – bind interface working example

- Monitor code (relevant for unit AND top level)

```
51 //------------------------------------------------------------------------------
52 // run_phase()
53 //------------------------------------------------------------------------------
54 task alg_pipe_common_monitor::run_phase(uvm_phase phase);
55
56   forever begin
57     m_item_collected = alg_pipe_common_sequence_item::type_id::create("m_item_collected");
58     m_item_collected.build_item(m_cfg.m_num_of_data_to_collect);
59
60     // collect item
61     foreach (m_item_collected.m_data[ii]) begin
62
63       do begin
64         @(m_vif_mp.mon_cb);
65       end while (m_vif_mp.mon_cb.data_en !== 1'b1);
66
67       m_item_collected.m_data[ii]     = m_vif_mp.mon_cb.data;
68       m_item_collected.m_data_ts[ii]  = m_vif_mp.mon_cb.ts_cntr;
69
70     end
71
72     // report item
73
74     m_item_collected_analysis_port.write(m_item_collected);
75
76   end
77
78 endtask : run_phase
79
```