Faster-Better-Cheaper Verification

Using internal DUT stimulus

Elihai Maicas

Qualcomm

June 6, 2015

SNUG Israel

Agenda

Motivation

The problems

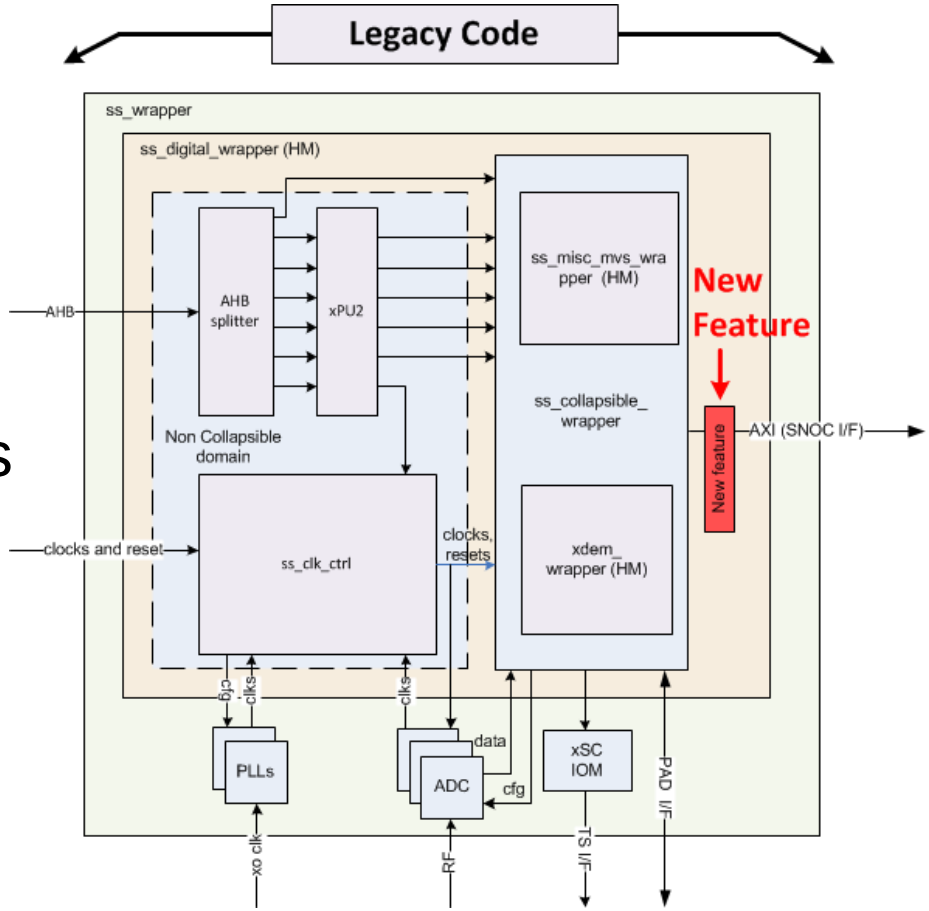
Brute-force solutions

Our solution - Internal DUT stimulus

Summary

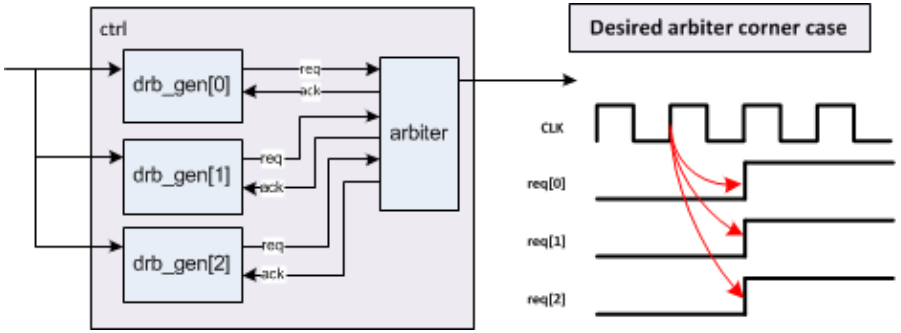
Motivation

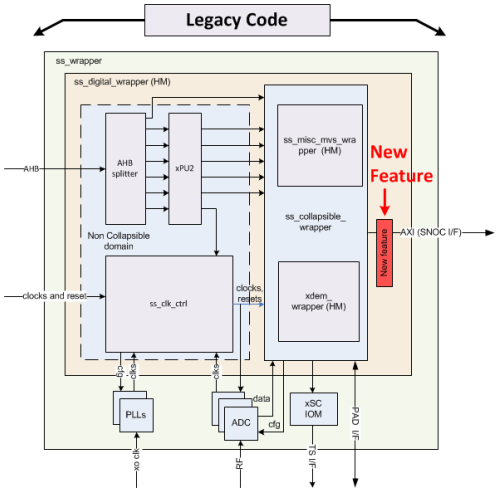
- Lower the verification overhead for IP reuse
- Minor changes between flavors should no longer be major verification headaches



The problems

- Reach the corner cases of the new feature

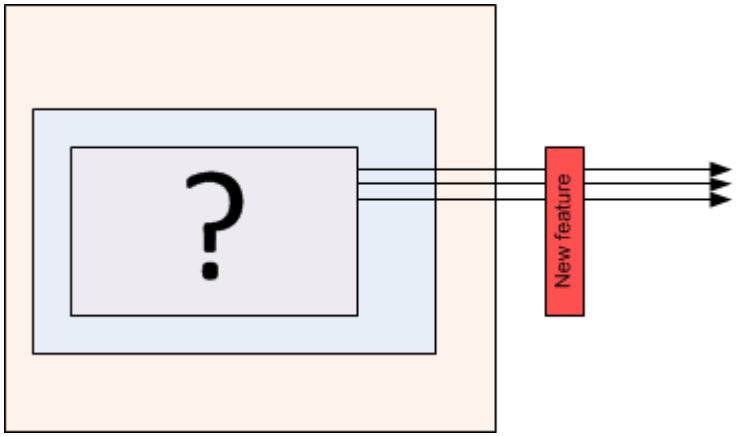


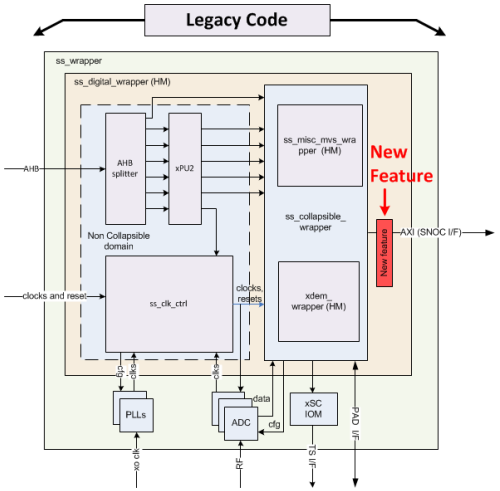


- Error injection – “Who will watch the watchmen?”

Brute-force solutions

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

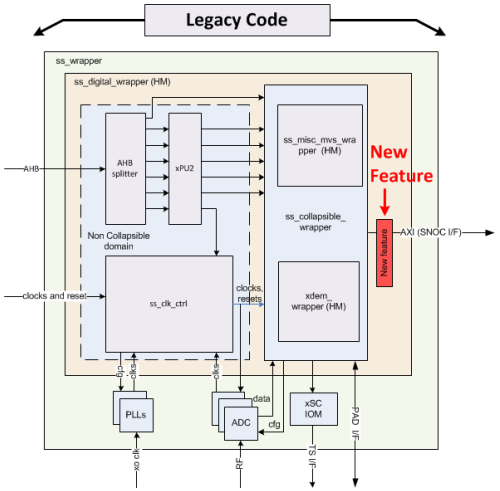




Brute-force solutions

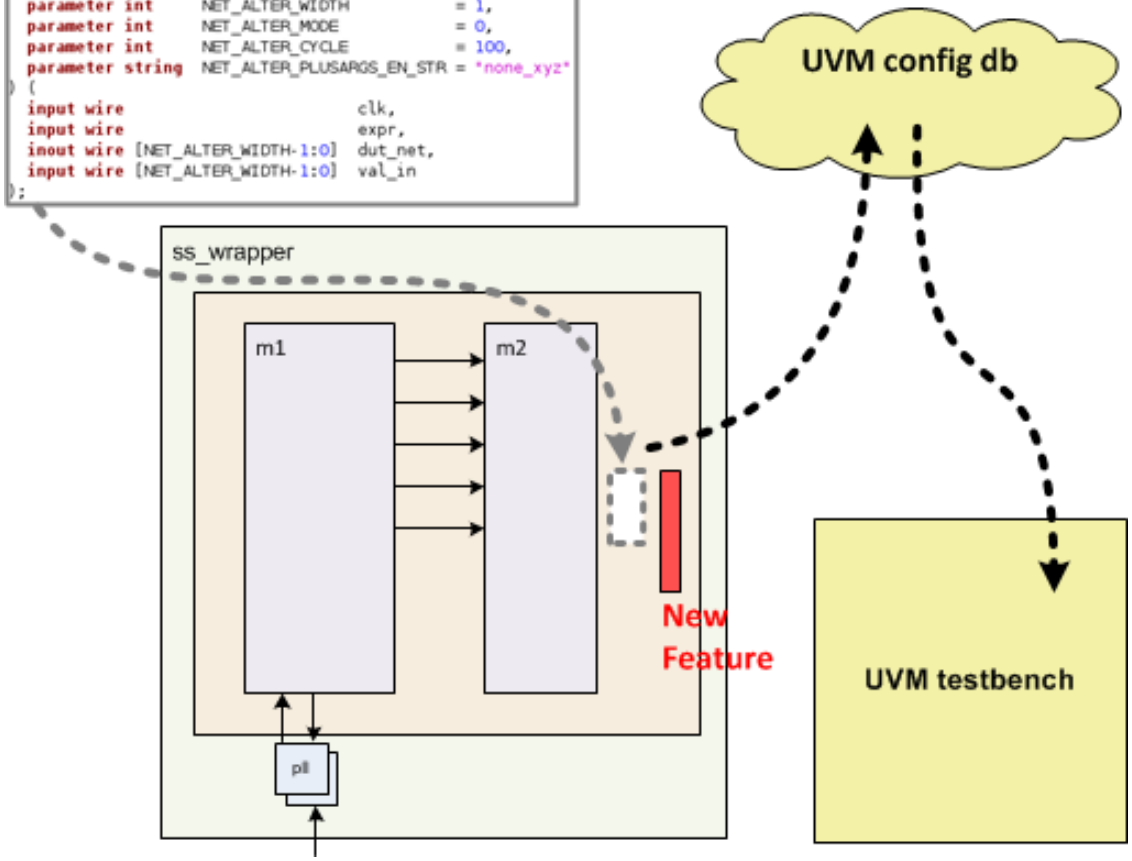
- Regressions -
  not best for all corner cases
  no full controllability on IP behavior

- 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
);
```



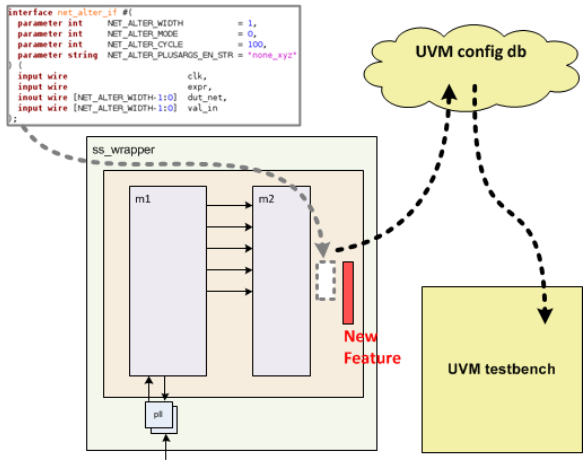
Internal DUT stimulus

How is it done

Internal DUT stimulus

Step I

- Bind a parameterized SV interface



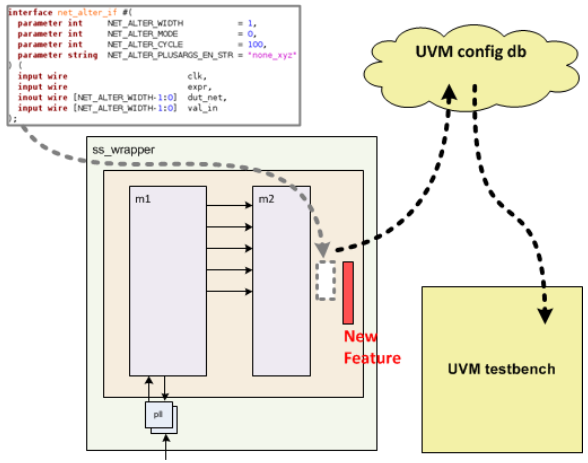

```
// a fifo_if interface is instanced in every instance of
fifo module
bind fifo fifo_if fifo_if__bind(.*);

// a fifo_if interface is instanced in the specific
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

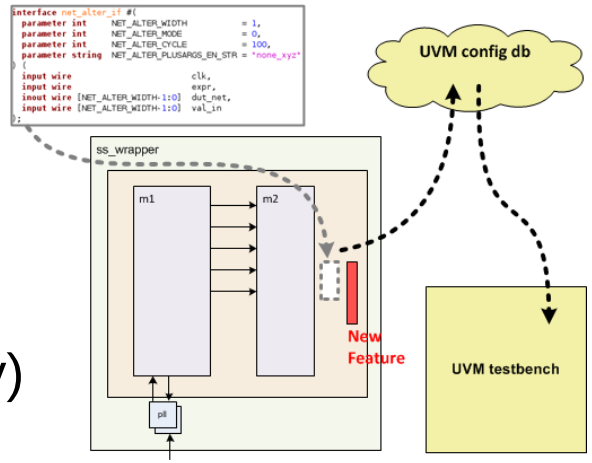



```
string loc_hier = $psprintf("%m");
.
.
uvm_config_db#(virtual
net_alter_if#(...))::set(uvm_root::get(),"*", loc_hier,
net_alter_if);
```

Internal DUT stimulus

Step II – option 2

- Register the interface to the uvm_config_db



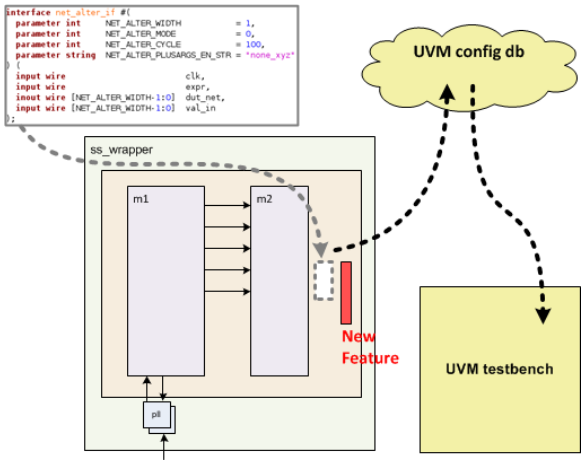
- Using the `::self` syntax
  (similar to `this` of OOP; Synopsys only)

```
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

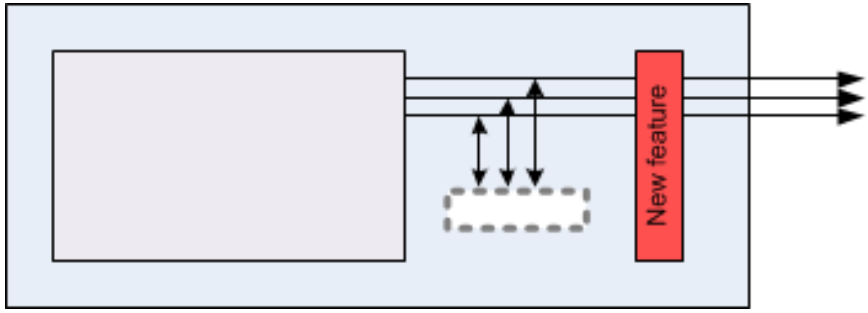


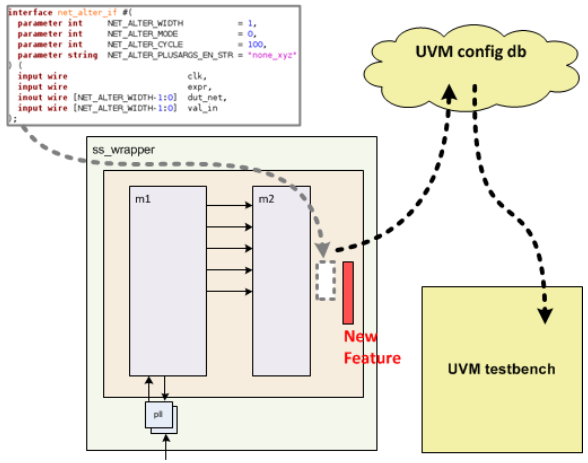

```
uvm_config_db#(virtual net_alter_if#(...))::get(this, "",
m_net_alter_if_bind_name, m_net_alter_vif);
```

Internal DUT stimulus

Final result

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





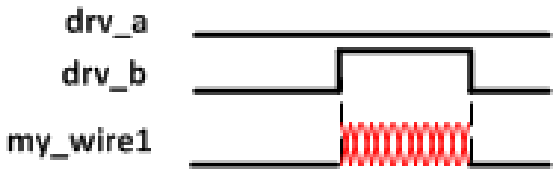
- Multiple drivers to the same net?

Internal DUT stimulus

Multiple drivers to the same net …

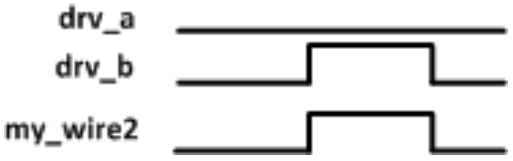
- Standard assignment

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

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

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



- Strength aware assignment

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

initial begin
  drv_b = 'bz; #100;
  drv_b = 'b1; #100;
  drv_b = '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;
```

15th Anniversary
snug
Synopsys Users Group
Israel

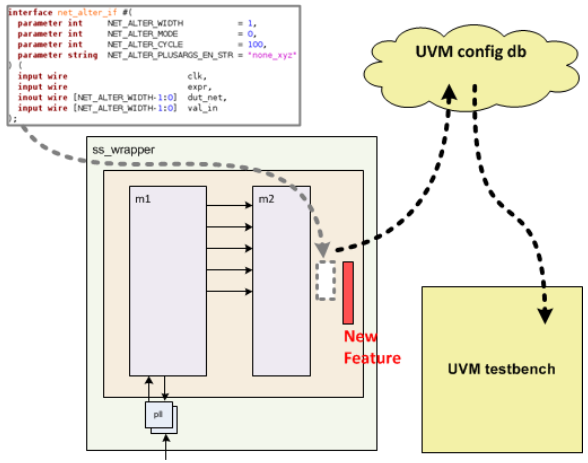
Internal DUT stimulus

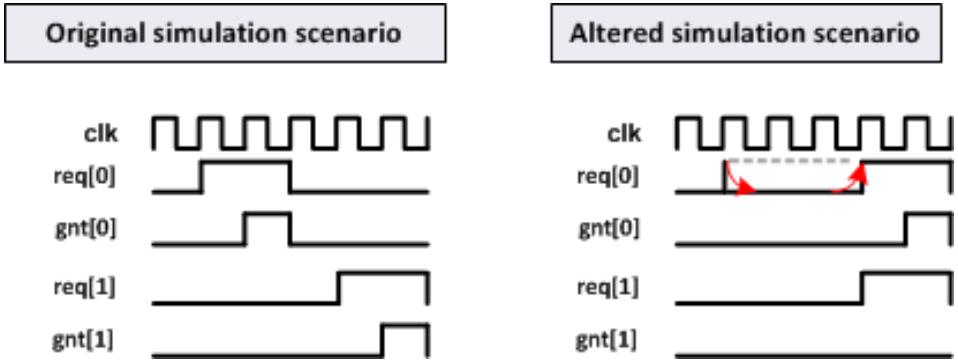
What can be achieved

Internal DUT stimulus

Manipulate the arbiter…

- Reach the desired rare corner case in the arbiter

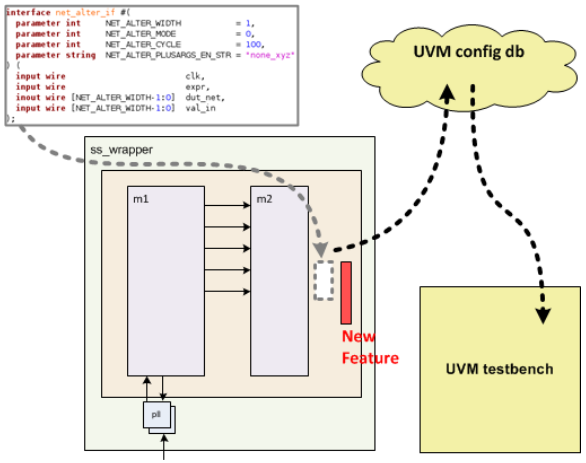




Internal DUT stimulus

Manipulate an AXI burst

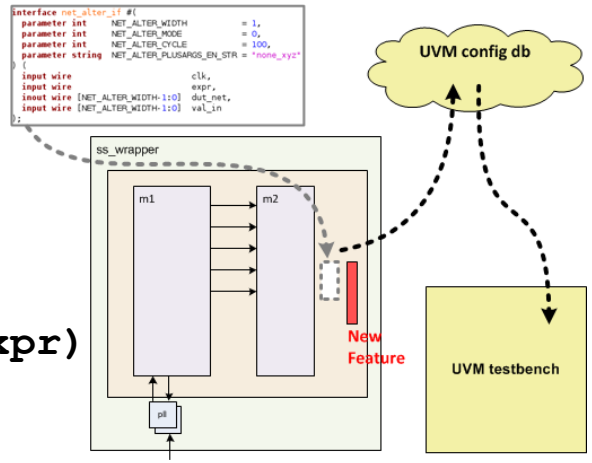
- Reach desired rare type of AXI burst



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

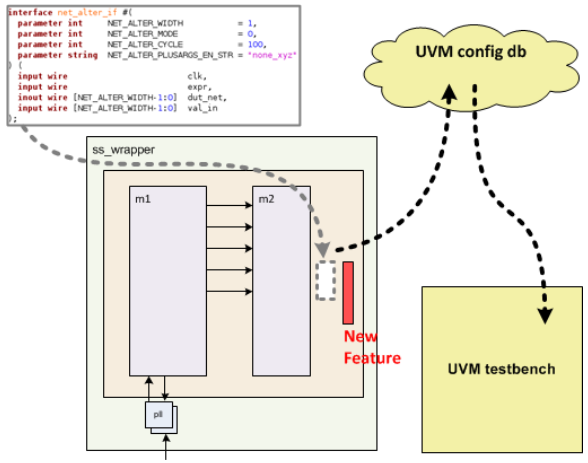



```
loc_net = {NET_WIDTH{1'bz}};
forever begin
  repeat (ALTER_FREQ-1) @(posedge expr)

  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

COMPANY LOGO

15th Anniversary
snug
Synopsys Users Group
Israel

Thank You

15th Anniversary
snug
Synopsys Users Group
Israel

Backup

Internal DUT stimulus

Some code…

```
interface net_alter_if #(
  parameter int NET_WIDTH           = 1,          // width of net to alter
  parameter int ALTER_MODE          = 0,          // 0: drive bitwise not;  1: drive x;   2: drive value
  parameter int ALTER_FREQ          = 100,        // count "expr==1" freq times, and alter net for expr width
  parameter string PLUSARGS_EN_STR  = "none_xyz"  // net will modify upon this plusargs
) (
  input wire                  clk,
  input wire                  expr,
  inout wire [NET_WIDTH-1:0]  dut_net,
  input wire [NET_WIDTH-1:0]  val_in
);

// importing the uvm_pkg, so the interface could register itself in the db
import uvm_pkg::*;

// local attributes
logic [NET_WIDTH-1:0] loc_net;
string                loc_hier;

// drive dut signal with internal net
assign (supply1,supply0) dut_net = loc_net;
```

Internal DUT stimulus

Some code…

```
// set the interface in the uvm db
// this interface can be explicitly get by any component in the env
initial begin
  loc_hier = $psprintf("%m");
  uvm_config_db#(virtual
net_alter_if#(NET_WIDTH,ALTER_MODE,ALTER_FREQ,PLUSARGS_EN_STR))::set(uvm_root::get(),"*",loc_hier, interface::self());
end
// frequently alter net
initial begin  // initially, do not drive
  loc_net = {NET_WIDTH{1'bz}};
  if($test$plusargs(PLUSARGS_EN_STR)) begin
    forever begin
      repeat (ALTER_FREQ-1) @(posedge expr);
      $display("[%0t] altering dut net at %m", $time);
      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
  end
end
endinterface
```