

Methodology focused

# Intro

- Who am I?
  - Ben Delsol - DV engineer formerly at Intel, Qualcomm, Samsung and Microsoft.
  - Founder of uvmgen.com.
- What I care about?
  - Clean code.
  - Methodology best practices.
  - Not wasting brain energy.
    - Divide, reuse and conquer.
    - Automating redundant problems.



# The idea of UVM is spot on

- Common procedures and methodologies across the industry.
- Clear coding and separation of testbench concerns.
- Reusable protocol agents.
- Reusable block level environments.
- Decades of verification best practices rolled into one methodology.



# Some best practices from the last 15 years...

- Interface harnesses
- Abstract/concrete classes
- DUT parameter passing to VIP
- Scale UVC, sub-env, config and TLM instances at runtime
- Conditional instantiation of static verification elements at compile time
- Pass down config object over config db
- Use sequence, BFM and config factory overrides
- Use slave sequences with late response randomization
- Reset methodology: don't kill sequences with the sequencer
- Use virtual sequences over phase jumping
- No virtual sequencers
- Use objections wisely
- Use constraint policies over inheritance
- Use standalone testbench for UVC development
- And many more...



# Interface harness

- Problem:
  - Code which handles connectivity of the design to interfaces, access to BFM's and DUT parameters is not reusable from the block level to upper levels of integration.
- Solution:
  - An interface harness defines the connectivity of all interface signals to a design and can be reused/bound into the DUT at block level as well as upper levels of integration.
  - It encapsulates VIP to DUT connectivity.
  - It encapsulates access to BFM creation classes.
  - It encapsulates collection of DUT parameters for the testbench.



## ✓ UVCs

SELECT UVC INSTANCE ▾



CONTINUE

3 Sub-environments  
Optional

✓ Predictor

✓ Coverage

6 Configuration and policies  
Optional

7 Signal checker

```
class async_fifo_env_pa
    logic [31:0] dsize;
    logic [31:0] asize;
    logic [31:0] fallt;

    `uvm_object_utils_begin(async_fifo_env_pa)
    `uvm_field_int(dsize, HIDDEN)
    `uvm_field_int(asize, HIDDEN)
    `uvm_field_int(fallt, HIDDEN)
    `uvm_field_int(fallt, HIDDEN)
    `uvm_field_int(fallt, HIDDEN)
endclass

class async_fifo_scoreb
    `uvm_analysis_imp_decl(dsize)
    `uvm_analysis_imp_decl(asize)
    `uvm_analysis_imp_decl(fallt)

    async_fifo_env_config
    async_fifo_predictor

    uvm_analysis_imp_actu
    uvm_analysis_imp_actu

    async_fifo_scoreb
endclass

`ifndef __ASYNC_FIFO_ENV__
`define __ASYNC_FIFO_ENV__

package async_fifo_env
    timeunit 1ns / 1ns;

    import uvm_pkg::*;
    import policy_pkg::*;
    import comparator_pkg::*;

    `ifndef __ASYNC_FIFO_INTERFACE_HARNESS_SV__
    `define __ASYNC_FIFO_INTERFACE_HARNESS_SV__

    module async_fifo_interface_harness
        #(
            parameter DSIZE=0,
            parameter ASIZE=0,
            parameter FALLTHROUGH=0
        )(
            inout wire wclk,
            inout wire wrst_n,
            inout wire winc,
            inout wire [DSIZE-1:0] wdata,
            inout wire wfull,
            inout wire awfull,
            inout wire rclk,
            inout wire rrst_n,
            inout wire rinc,
            inout wire [DSIZE-1:0] rdata,
            inout wire empty,
            inout wire arempty
        );

        timeunit 1ns / 1ns;

        import uvm_pkg::*;
        import async_fifo_env_pkg::*;

        string env_full_name;

        clk_if
        clk_r_if (
            .clk (rclk)
        );
    endmodule
`endif
endpackage
endclass
```

async\_fifo\_interface\_harness.sv

DONE

```
predictor_c extends
    imp_decl(actual_req)
    imp_decl(actual_req)
    imp_decl(actual_req)
    imp_decl(actual_req)

    config_c

    no actual request from
    predictor_c.svh

    env_c extends uvm_en
    config_c env_cfg;

    clk_r_ag
    clk_w_ag
    rst_mast
    rst_mast
    fifor_ma
    fifow_ma

    env_c.svh
```





# UVCs

SELECT UVC INSTANCE ▾



## fifow\_master\_if

Instance of fifow\_if

### Parameters

Name	Value
DATA_MSB	Not set (default: 31)
ON_POSEDGE	Not set (default: 1)

### Ports

Name	Connection
wclk	↔ wclk
wrst_n	↔ wrst_n
winc	⇒ winc
wdata	⇒ wdata
wfull	↔ wfull
wafull	↔ awfull

CONTINUE

class async\_fifo\_env\_pa

```
logic [31:0] dsize
logic [31:0] asize
logic [31:0] fallt
```

```
`uvm_object_utils_beg
`uvm_field_int(dsize
`uvm_field_int(asiz
`uvm_field_int(fall
```

async\_fifo\_env\_pa

class async\_fifo\_scoreb

```
`uvm_analysis_imp_dec
`uvm_analysis_imp_dec
```

```
async_fifo_env_config
async_fifo_predictor
```

```
uvm_analysis_imp_actu
uvm analysis imp actu
```

async\_fifo\_scoreb

```
`ifndef _ASYNC_FIFO_EN
`define _ASYNC_FIFO_EN
```

package async\_fifo\_env

timeunit 1ns / 1ns;

```
import uvm_pkg::*;
import policy_pkg::*;
import comparator_nk
```

async\_fifo\_env\_pkg.sv

```
.clk (wclk),
.rst (wrst_n)
);
```

fifor\_if

```
fifor_master_if (
.rclk (rclk),
.wrst_n (rrst_n),
.rinc (rinc),
.rdata (rdata),
.reempty (reempty),
.raempty (areempty)
);
```

fifow\_if

```
fifow_master_if (
.wclk (wclk),
.wrst_n (wrst_n),
.winc (winc),
.wdata (wdata),
.wfull (wfull),
.wafull (awfull)
);
```

initial begin

wait (env\_full\_name != "");

```
clk_r_if.add_resources_to_config_db({ env_full_name, ".clk_r_agent" })
clk_w_if.add_resources_to_config_db({ env_full_name, ".clk_w_agent" })
rst_master_r_if.add_resources_to_config_db({ env_full_name, ".rst_mast
rst_master_w_if.add_resources_to_config_db({ env_full_name, ".rst_mast
fifor_master_if.add_resources_to_config_db({ env_full_name, ".fifor_ma
fifow_master_if.add_resources_to_config_db({ env_full_name, ".fifow_m
end
```

async\_fifo\_interface\_harness.sv

DONE

predictor\_c extends

```
imp_decl(actual_requ
imp_decl(actual_requ
imp_decl(actual_requ
imp_decl(actual_requ
```

config\_c

no actual request fro

predictor\_c.svh

env\_c extends uvm\_en

config\_c env\_cfg;

clk\_r\_ag

clk\_w\_ag

nt\_c rst\_mast

nt\_c rst\_mast

gent\_c fifor\_ma

gent\_c fifow ma

nv\_c.svh



## ✓ UVCs

SELECT UVC INSTANCE ▾



## fifow\_master\_if

Instance of fifow\_if

## Parameters

Name	Value
DATA_MSB	Not set (default: 31)
ON_POSEDGE	Not set (default: 1)

## Ports

Name	Connection
wclk	↔ wclk
wrst_n	↔ wrst_n
winc	⇒ winc
wdata	⇒ wdata
wfull	↔ wfull
wawfull	↔ awfull

CONTINUE

class async\_fifo\_env\_package; module async\_fifo\_protocol\_checker\_harness

```
logic [31:0] dsize;
logic [31:0] asize;
logic [31:0] fallt;
```

```
parameter DSIZE=0,
parameter ASIZE=0,
parameter FALLTHROUGH=0
```

```
input wire wclk,
input wire wrst_n,
input wire winc,
input wire [DSIZE-1:0] wdata,
input wire wfull,
input wire awfull,
input wire rclk,
input wire rrst_n,
input wire rinc,
input wire [DSIZE-1:0] rdata,
input wire rempty,
input wire arempty;
```

async\_fifo\_env\_package;

class async\_fifo\_scoreboard;

```
`uvm_analysis_imp_decl(ASYNC_FIFO_EN);
`uvm_analysis_imp_decl(ASYNC_FIFO_OUT);
```

```
async_fifo_env_config;
async_fifo_predictor;
```

```
uvm_analysis_imp_async_fifo_en;
uvm_analysis_imp_async_fifo_out;
```

async\_fifo\_scoreboard;

```
`ifndef __ASYNC_FIFO_ENV__
`define __ASYNC_FIFO_ENV__
```

package async\_fifo\_env;

timeunit 1ns / 1ns;

```
import uvm_pkg::*;
import policy_pkg::*;
import comparator_pkg::*;
```

async\_fifo\_env\_pkg.sv

module async\_fifo\_protocol\_checker\_harness

```
#(
parameter DSIZE=0,
parameter ASIZE=0,
parameter FALLTHROUGH=0
)
```

```
input wire wclk,
input wire wrst_n,
input wire winc,
input wire [DSIZE-1:0] wdata,
input wire wfull,
input wire awfull,
input wire rclk,
input wire rrst_n,
input wire rinc,
input wire [DSIZE-1:0] rdata,
input wire rempty,
input wire arempty;
```

);

timeunit 1ns / 1ns;

```
string env_full_name;

fifor_protocol_checker
fifor_master_protocol_checker (
```

```
.rclk (rclk),
.rrst_n (rrst_n),
.rinc (rinc),
.rdata (rdata),
.rempty (rempty),
.raempty (arempty)
);
```

string env\_full\_name;

fifor\_protocol\_checker

```
fifor_master_protocol_checker (
.rclk (rclk),
.rrst_n (rrst_n),
.rinc (rinc),
.rdata (rdata),
.rempty (rempty),
.raempty (arempty)
);
```

fifor\_protocol\_checker

async\_fifo\_protocol\_checker\_harness.sv

DONE

async\_fifo\_interface\_harne...

predictor\_c extends

```
imp_decl(actual_req)
imp_decl(actual_req)
imp_decl(actual_req)
imp_decl(actual_req)
```

config\_c

no actual request fro

redictor\_c.svh

env\_c extends uvm\_en

config\_c env\_cfg;

clk\_r\_ag

clk\_w\_ag

rst\_mast

rst\_mast

fifor\_ma

fifow ma

nv\_c.svh





## ✓ TB basics



async\_fifo

Selected environment

CONTINUE

2

Default settings

Optional

✓

Virtual sequences and policies

Optional

✓

Tests

Optional

5

DUT parameter packages

Optional

6

Generate

./

```
class async_fifo_test_ba
    async_fifo_env_c
    async_fifo_env_config
    async_fifo_top_virtual
    int
    time
    time
```

```
`uvm component utils/
```

```
async_fifo_test_ba
```

```
`ifndef __ASYNC_FIFO_TE
`define __ASYNC_FIFO_TE
```

```
package async_fifo_test
```

```
timeunit 1ns / 1ns;
```

```
import uvm_pkg::*;
import policy_pkg::*;
```

```
async_fifo_test_pk
```

policies/

```
class async_fifo_clk_r
```

```
constraint freq_crv {
    object.freq == 100%
```

```
constraint unit_crv {
    object.unit == CLOCK_FREQUENCY_UN
}
```

```
// ASYNC FIFO Bind statements
```

```
bind async_fifo async_fifo_interface_harness #(
    .DSIZE (DSIZE),
    .ASIZE (ASIZE),
    .FALLTHROUGH (FALLTHROUGH)
) async_fifo_interface_harness (.*);
```

```
`ifdef ENABLE_ASYNC_FIFO_PROTOCOL_CHECKERS
```

```
bind async_fifo async_fifo_protocol_checker_harness #(
    .DSIZE (DSIZE),
    .ASIZE (ASIZE),
    .FALLTHROUGH (FALLTHROUGH)
) async_fifo_protocol_checker_harness (.*);
```

```
`endif
```

async\_fifo\_bind.svh

DONE

```
reset_test_c extends
    utils(async_fifo_res
    string name="async_fif
    e, parent);
```

```
on void build_phase(u
    phase(phase):
```

```
set_test_c.svh
```

```
SV__
SV__
```

```
DISABLE MACROS
-----
```

```
on capabilities are
to disable them.
```

```
nc_fifo_main_virtual
```

```
main_num_fifor_trans
```

```
constraint num_fifor_transactions_c
    object.num_fifor_transactions ==
}
```

```
function new(string name="async_fif
```



# Abstract/concrete classes

- Problem:
  - Any component which uses a virtual interface handle to a parameterized interface must be parameterized, and so too must all its component ancestors (ie test, env, agents, drivers, monitors all must be parameterized!).
- Solution:
  - Define a BFM class in the parameterized interface.
  - Drivers and monitors initiate the BFMs construction with the abstract/concrete design pattern.
  - Ensure the BFM has access to the config object, has context aware UVM printing and can be overridden by the factory.
- Can be used for access of protocol checkers and signal checkers too.



## ✓ Bus protocol

BFMS

PROTOCOL CHECKER

- ✓ Has a master driver
- ✓ Has a slave responder
- ✓ Has a monitor

SELECT BFM ▾



CONTINUE

5

Sequences and policies  
Optional

./

```
typedef class apb_config_c extends uvm_config_c;  
typedef class apb_sequencer_c extends uvm_sequencer_c;  
typedef class apb_monitor_c extends uvm_monitor_c;
```

```
virtual class abstract_apb_bfm_creator_c;
```

```
    apb_config_c    cfg;  
    apb_monitor_c   mon;
```

```
    function new(string name, uvm_analysis_port#(apb_master_driver_c) apb_master_driver_port, uvm_analysis_port#(apb_monitor_c) apb_monitor_port);
```

```
abstract_apb_bfm_creator_c
```

```
class apb_config_c extends uvm_config_c
```

```
    apb_parameters_c
```

```
    protected string  
    protected uvm_active_status_t  
    protected bit
```

```
    `uvm_object_utils_begin  
    `uvm_field_object(n
```

```
apb_config_c.svh
```

```
class apb_monitor_c extends uvm_monitor
```

```
    apb_config_c  
    abstract_apb_bfm_creator_c  
    abstract_apb_bfm_c
```

```
    uvm_analysis_port#(apb_master_driver_c) apb_master_driver_port;  
    uvm_analysis_port#(apb_monitor_c) apb_monitor_port;
```

```
    `uvm_component_utils(apb_monitor_c)
```

```
apb_monitor_c.svh
```

```
virtual class abstract_apb_bfm_creator_c;
```

```
    pure virtual function abstract_apb_bfm_c create_master_bfm(string name, uvm_analysis_port#(apb_master_driver_c) apb_master_driver_port, uvm_analysis_port#(apb_monitor_c) apb_monitor_port);
```

```
    pure virtual function abstract_apb_bfm_c create_slave_bfm(string name, uvm_analysis_port#(apb_master_driver_c) apb_master_driver_port, uvm_analysis_port#(apb_monitor_c) apb_monitor_port);
```

```
    pure virtual function abstract_apb_bfm_c create_monitor_bfm(string name, uvm_analysis_port#(apb_master_driver_c) apb_master_driver_port, uvm_analysis_port#(apb_monitor_c) apb_monitor_port);
```

```
endclass
```

```
    addr_ms
```

```
    addr_ls
```

```
    data_ms
```

```
    addr_ma
```

```
    [MSB_MAX:0]
```

```
    data_ma
```

```
    [MSB_MAX:0]
```

```
    [MSB_MAX:0]
```

```
    [MSB_MAX:0]
```

```
    [MSB_MAX:0]
```

```
    [MSB_MAX:0]
```

```
    [MSB_MAX:0]
```

```
    [MSB_MAX:0]
```

```
    [MSB_MAX:0]
```

```
    [MSB_MAX:0]
```

```
    [MSB_MAX:0]
```

```
    [MSB_MAX:0]
```

```
    [MSB_MAX:0]
```

```
    [MSB_MAX:0]
```

```
    [MSB_MAX:0]
```

```
    [MSB_MAX:0]
```

```
    [MSB_MAX:0]
```

```
    [MSB_MAX:0]
```

```
    [MSB_MAX:0]
```

```
    [MSB_MAX:0]
```

```
    [MSB_MAX:0]
```

```
    [MSB_MAX:0]
```

```
    [MSB_MAX:0]
```

```
    [MSB_MAX:0]
```

```
    [MSB_MAX:0]
```

```
    [MSB_MAX:0]
```

```
    [MSB_MAX:0]
```

```
    [MSB_MAX:0]
```

```
    [MSB_MAX:0]
```

```
    [MSB_MAX:0]
```

```
    [MSB_MAX:0]
```

```
    [MSB_MAX:0]
```

```
    [MSB_MAX:0]
```

```
    [MSB_MAX:0]
```

abstract\_apb\_bfm\_creator\_c.svh

DONE

```
function new(string name="apb_master_driver", uvm_analysis_port#(apb_master_driver_c) apb_master_driver_port, uvm_analysis_port#(apb_monitor_c) apb_monitor_port):  
    super.new(name, parent);  
    apb_master_driver_port = apb_master_driver_port;  
    apb_monitor_port = apb_monitor_port;
```

```
apb_master_driver_c.svh
```

```
apb_master_agent_c.svh
```



≡ apb



## ✓ Bus protocol

BFMS

PROTOCOL CHECKER

- ✓ Has a master driver
- ✓ Has a slave responder
- ✓ Has a monitor

SELECT BFM ▾



CONTINUE

5

Sequences and policies  
Optional

```
import uvm_pkg::*;
import policy_pkg::*;
```

apb\_pkg.sv

```
class apb_slave_bfm_c extends
```

```
`uvm_component_param_

function new(string name,
    super.new(name, parent);
endfunction
```

```
virtual function void
    super.build_phase(n
```

apb\_slave\_bfm\_c.sv

```
`ifndef __APB_IF_SV__
`define __APB_IF_SV__
```

```
interface apb_if
#(
    parameter ADDR_MSB=
    parameter ADDR_LSB=
    parameter DATA_MSB=
    parameter bit ON_PO
) {
```

apb\_if.sv

policies/

```
class apb_master_policy
```

```
constraint phase_crv {
    object.phase == APB_PHASE_REQUEST
}
```

```
constraint data_crv {
```

```
super.build_phase(phase);
```

```
class apb_bfm_creator_c extends abstract_apb_bfm_creator_c;
```

```
function abstract_apb_bfm_c create_master_bfm(string name, uvm_component
    return apb_master_bfm_c::type_id::create(name, parent);
endfunction
```

```
function abstract_apb_bfm_c create_slave_bfm(string name, uvm_component p
    return apb_slave_bfm_c::type_id::create(name, parent);
endfunction
```

```
function abstract_apb_bfm_c create_monitor_bfm(string name, uvm_component
    return apb_monitor_bfm_c::type_id::create(name, parent);
endfunction
```

```
endclass
```

apb\_bfm\_creator\_c.svh

DONE

```
request_policy_c exte
```

```
constraint pre_transfer_delay_crv {
    object.pre_transfer_delay == 0;
}
```

```
function new(string name="apb_maste
```

```
constraint phase_crv {
    object.phase == APB_PHASE_REQUEST
}
```

```
constraint pre_transfer_delay_crv {
```



## ✓ Bus protocol

BFMS

PROTOCOL CHECKER

- ✓ Has a master driver
- ✓ Has a slave responder
- ✓ Has a monitor

SELECT BFM ▾



CONTINUE

5

Sequences and policies  
Optional

```
import uvm_pkg::*;
import policy_pkg::*;

super.build_phase(phase);
```

apb\_pkg.sv

class apb\_slave\_bfm\_c e

`uvm\_component\_param

```
function new(string n
super.new(name, par
endfunction
```

```
virtual function void
super.build_phase(n
```

apb\_slave\_bfm.c

```
input penable;
input pwrite;
input pwrite;
input pstrb;
input pready;
input prdata;
input pslverr;
endclocking
```

```
`include "apb_bfm_base_c.svh"
`include "apb_master_bfm_c.svh"
`include "apb_slave_bfm_c.svh"
`include "apb_monitor_bfm_c.svh"
```

```
`include "apb_bfm_creator_c.svh"
```

```
function void add_resources_to_config_db(string inst_name);
    add_parameters_to_config_db(inst_name, ".cfg");
    add_bfm_creator_to_config_db(inst_name);
endfunction
```

```
function automatic void add_parameters_to_config_db(string inst_name);
    apb_parameters_c params = apb_parameters_c::type_id::create("params");
    params.set_addr_msb(ADDR_MSB);
    params.set_addr_lsb(ADDR_LSB);
    params.set_data_msb(DATA_MSB);
    uvm_config_db#(apb_parameters_c)::set(null, inst_name, "params", param
endfunction
```

```
function automatic void add_bfm_creator_to_config_db(string inst_name);
    apb_bfm_creator_c bfm_creator = new();
    uvm_config_db#(abstract_apb_bfm_creator_c)::set(null, inst_name, "bfm
endfunction
```

endinterface

policies/

class apb\_master\_policy

```
constraint phase_crv {
    object.phase == APB_PHASE_REQUEST
}
```

```
constraint data_crv {
```

apb\_if.sv

DONE

```
constraint pre_transfer_delay_crv {
    object.pre_transfer_delay == 0;
}
```

```
function new(string name="apb master
```

```
constraint phase_crv {
    object.phase == APB_PHASE_REQUEST
}
```

```
constraint pre_transfer_delay_crv {
```



## ✓ Bus protocol

BFMS

PROTOCOL CHECKER

- ✓ Has a master driver
- ✓ Has a slave responder
- ✓ Has a monitor

SELECT BFM ▾



CONTINUE

```
class apb_monitor_c extends uvm_monitor
    apb_config_c
    abstract_apb_bfm_creator_c
    abstract_apb_bfm_c

    uvm_analysis_port#(apb_monitor_c)
    `uvm_component_utils

apb_monitor_c.svh

class apb_slave_driver_c extends uvm_driver#(apb_slave_bfm_c)
    apb_config_c
    abstract_apb_bfm_creator_c
    abstract_apb_bfm_c

    `uvm_component_utils

function new(string name, uvm_component parent)
    super.new(name, parent)
endfunction

virtual function void set_cfg(apb_config_c cfg);
    this.cfg = cfg;
endfunction

virtual function void set_bfm_creator(apb_bfm_creator_c bfm_creator);
    this.bfm_creator = bfm_creator;
endfunction

virtual function void build_phase(uvm_phase phase);
    if (cfg == null) begin
        `uvm_fatal(get_type_name(), "Must call set_cfg(apb_config_c cfg).")
    end

    if (bfm_creator == null) begin
        `uvm_fatal(get_type_name(), "Must call set_bfm_creator(apb_bfm_creator_c bfm_creator)")
    end

    bfm = bfm_creator.create_master_bfm("bfm", this);
    bfm.set_cfg(cfg);
endfunction

virtual task run_phase(uvm_phase phase);
    super.run_phase(phase)
endtask

apb_slave_driver_c.svh

package apb_pkg;

timeunit 1ns / 1ns;

import uvm_pkg::*;
import policy_pkg::*;

class apb_slave_bfm_c extends apb_bfm
    apb_config_c
    abstract_apb_bfm_creator_c
    abstract_apb_bfm_c

    `uvm_component_utils

function new(string name, uvm_component parent)
    super.new(name, parent)
endfunction

virtual function void set_cfg(apb_config_c cfg);
    this.cfg = cfg;
endfunction

virtual function void set_bfm_creator(apb_bfm_creator_c bfm_creator);
    this.bfm_creator = bfm_creator;
endfunction

virtual function void build_phase(uvm_phase phase);
    super.build_phase(phase)
endfunction

virtual task run_phase(uvm_phase phase);
    super.run_phase(phase)
endtask

apb_slave_bfm_c.svh

class apb_monitor_bfm_c extends apb_bfm
    apb_config_c
    abstract_apb_bfm_creator_c
    abstract_apb_bfm_c

    `uvm_component_utils

function new(string name, uvm_component parent)
    super.new(name, parent)
endfunction

virtual function void set_cfg(apb_config_c cfg);
    this.cfg = cfg;
endfunction

virtual function void set_bfm_creator(apb_bfm_creator_c bfm_creator);
    this.bfm_creator = bfm_creator;
endfunction

virtual function void build_phase(uvm_phase phase);
    super.build_phase(phase)
endfunction

virtual task run_phase(uvm_phase phase);
    super.run_phase(phase)
endtask

apb_monitor_bfm_c.svh

class apb_master_driver_c extends uvm_driver#(apb_master_bfm_c);
    apb_config_c
    abstract_apb_bfm_creator_c
    abstract_apb_bfm_c

    `uvm_component_utils

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

virtual function void set_cfg(apb_config_c cfg);
    this.cfg = cfg;
endfunction

virtual function void set_bfm_creator(apb_bfm_creator_c bfm_creator);
    this.bfm_creator = bfm_creator;
endfunction

virtual function void build_phase(uvm_phase phase);
    if (cfg == null) begin
        `uvm_fatal(get_type_name(), "Must call set_cfg(apb_config_c cfg).")
    end

    if (bfm_creator == null) begin
        `uvm_fatal(get_type_name(), "Must call set_bfm_creator(apb_bfm_creator_c bfm_creator)")
    end

    bfm = bfm_creator.create_master_bfm("bfm", this);
    bfm.set_cfg(cfg);
endfunction

virtual task run_phase(uvm_phase phase);
    super.run_phase(phase)
endtask

apb_master_driver_c.svh
```

DONE



# DUT parameter passing to VIP

- Problem:
  - The verification environment of a parameterized design must also have access to those parameters. Again, type specialization of many classes becomes very cumbersome to manage.
- Solution:
  - Use your interfaces and interface harness' to collect interface and DUT parameters, respectively.
  - Pass these parameter objects to your UVC and environment config objects via the config db.





## Interface signals

## Interface parameters

Type	Name	Default	
parameter	ADDR_MSB	31	▼
parameter	ADDR_LSB	0	▼
parameter	DATA_MSB	31	▼
parameter bit	ON_POSEDGE	1	▼
ADD			

## Master and slave nets

Source	Type	Name	Reset value	
C	wire	pcik	'x	▼
R	wire	preset_n	'x	▼
M ▼	wire	psel	'x	▼
M ▼	wire [ADDR_MS...	paddr	'x	▼
M ▼	wire	penable	'x	▼
M ▼	wire	pwrite	'x	▼
M ▼	wire [DATA_MSB...	pwdata	'x	▼
M ▼	wire [((DATA_MS...	pstrb	'x	▼
S ▼	wire	pready	'x	▼
S ▼	wire [DATA_MSB...	prdata	'x	▼
S ▼	wire	pslverr	'x	▼

./

```
typedef class apb_config_c;
typedef class apb_sequencer_c;
typedef class apb_monitor_c;
```

```
virtual class abstract_apb_bfm;
```

```
apb_config_c cfg;
apb_monitor_c mon;
```

```
function new(string name);
```

```
abstract_apb_bfm;
```

```
class apb_config_c extends uvm_object;
```

```
apb_parameters_c
```

```
protected string
protected uvm_active
protected bit
```

```
`uvm_object_utils_begin
`uvm_field_object(n
```

```
apb_config_c.svh
```

```
class apb_monitor_c extends uvm_monitor;
```

```
apb_config_c
abstract_apb_bfm_create
abstract_apb_bfm_c
```

```
uvm_analysis_port#(apb_monitor_c)
```

```
`uvm_component_utils(apb_monitor_c)
```

```
apb_monitor_c.svh
```

```
class apb_parameters_c extends uvm_object;
```

```
logic [31:0] addr_msb;
logic [31:0] addr_lsb;
logic [31:0] data_msb;
bit [APB_ADDR_MSB_MAX:0] addr_mask;
bit [APB_DATA_MSB_MAX:0] data_mask;
```

```
`uvm_object_utils_begin(apb_parameters_c)
`uvm_field_int(addr_msb, UVM_DEFAULT)
`uvm_field_int(addr_lsb, UVM_DEFAULT)
`uvm_field_int(data_msb, UVM_DEFAULT)
`uvm_field_int(addr_mask, UVM_DEFAULT | UVM_NOPRINT)
`uvm_field_int(data_mask, UVM_DEFAULT | UVM_NOPRINT)
`uvm_object_utils_end
```

```
function new(string name="apb_parameters_c");
super.new(name);
endfunction
```

```
virtual function void set_addr_msb(logic [31:0] addr_msb);
this.addr_msb = addr_msb;
update_addr_mask();
endfunction
```

```
virtual function void set_addr_lsb(logic [31:0] addr_lsb);
this.addr_lsb = addr_lsb;
update_addr_mask();
endfunction
```

```
virtual function void set_data_msb(logic [31:0] data_msb);
this.data_msb = data_msb;
update_data_mask();
endfunction
```

apb\_parameters\_c.svh

DONE

```
er_c extends uvm_sequencer_c;
```

```
cfg;
```

```
utils(apb_sequencer_
```

```
string name="apb_seque
```

```
ne, parent);
```

```
er_c.svh
```

```
agent_c extends uvm_
```

```
er_c
```

```
uvm_analysis_port#(apb_sequence_ite
```

```
`uvm_component_utils(apb master age
```

```
apb_master_agent_c.svh
```







## Interface signals

## Interface parameters

Type	Name	Default	
parameter	ADDR_MSB	31	▼
parameter	ADDR_LSB	0	▼
parameter	DATA_MSB	31	▼
parameter bit	ON_POSEDGE	1	▼
ADD			

## Master and slave nets

Source	Type	Name	Reset value	
C	wire	clk	'x	▼
R	wire	preset_n	'x	▼
M ▼	wire	psel	'x	▼
M ▼	wire [ADDR_MS...	paddr	'x	▼
M ▼	wire	penable	'x	▼
M ▼	wire	pwrite	'x	▼
M ▼	wire [DATA_MSB...	pdata	'x	▼
M ▼	wire [([DATA_MS...	pstrb	'x	▼
S ▼	wire	pready	'x	▼
S ▼	wire [DATA_MSB...	prdata	'x	▼
S ▼	wire	pslverr	'x	▼

```
virtual function void build_phase(uvm_phase_t phase) {
    super.build_phase(phase);
    input penable;
    input pwrite;
    input pdata;
    input pstrb;
    input pready;
    input prdata;
    input pslverr;
    endlocking
}

`include "apb_bfm_base_c.svh"
`include "apb_master_bfm_c.svh"
`include "apb_slave_bfm_c.svh"
`include "apb_monitor_bfm_c.svh"

`include "apb_bfm_creator_c.svh"

function void add_resources_to_config_db(string inst_name);
    add_parameters_to_config_db({inst_name, ".cfg"});
    add_bfm_creator_to_config_db(inst_name);
endfunction
```

apb\_slave\_bfm\_c.svh

apb\_bfm\_creator\_c.svh

policies/

class apb\_master\_policy

```
constraint phase_crv {
    object.phase == APB_PHASE_REQUEST;
}
```

```
constraint data_crv {
    if (object.write == 1)
        object.data == '0;
}
```

apb\_master\_policy

class apb\_slave\_respons

protected apb\_sequence\_item\_c re

```
constraint phase_crv {
    object.phase == APB_PHASE_RESPONS;
}
```

```
function automatic void add_parameters_to_config_db(string inst_name);
    apb_parameters_c params = apb_parameters_c::type_id::create("params");
    params.set_addr_msb(ADDR_MSB);
    params.set_addr_lsb(ADDR_LSB);
    params.set_data_msb(DATA_MSB);
    uvm_config_db#(apb_parameters_c)::set(null, inst_name, "params", params);
endfunction
```

```
function automatic void add_bfm_creator_to_config_db(string inst_name);
    apb_bfm_creator_c bfm_creator = new();
    uvm_config_db#(abstract_apb_bfm_creator_c)::set(null, inst_name, "bfm_creator", bfm_creator);
endfunction
```

endinterface

apb\_if.sv

DONE



6

Configuration  
Optional

## apb\_config\_c

Extends uvm\_object

## Instance variables

Type	Name
No instance variables	

ADD

## Subroutines

ADD

CONTINUE

✓

Test  
Optional

8

## Generate

function new(string name, uvm\_object parent):

abstract\_apb\_bfm\_c

```
virtual function void set_interface_bound(bit interface_bound);  
`ifndef DISABLE_INTERFACE_HARNESS_NOT_BOUND_WARNINGS  
    if (interface_bound == 0) begin  
        `uvm_warning(get_type_name(), "Without an interface the following  
    end  
`endif  
    this.interface_bound = interface_bound;  
endfunction
```

```
virtual function bit get_interface_bound();  
    return interface_bound;  
endfunction
```

```
function void pre_randomize();  
    super.pre_randomize();
```

```
if (inst_name == "") begin  
    `uvm_fatal(get_type_name(), "Must call set_inst_name(string inst_name)  
end  
if (!uvm_config_db#(apb_parameters_c)::get(null, inst_name, "params",  
    `uvm_fatal(get_type_name(), {"apb_parameters_c must be set for ", in  
end  
endfunction
```

```
virtual function bit has_driver();  
    return (interface_bound && (active_state == UVM_ACTIVE));  
endfunction
```

```
virtual function bit has_monitor();  
    return interface_bound;  
endfunction
```

endclass

apb\_config\_c.svh

DONE

class apb\_monitor\_c extends uvm\_monitor\_c

```
    apb_config_c  
    abstract_apb_bfm_c  
    abstract_apb_bfm_c
```

```
    uvm_analysis_port#(apb_sequence_item_c)  
    `uvm_component_utils(apb_monitor_c)
```

apb\_monitor\_c.svh

class apb\_slave\_driver extends uvm\_slave\_driver

```
    apb_config_c  
    abstract_apb_bfm_c  
    abstract_apb_bfm_c
```

```
    `uvm_component_utils(apb_slave_driver)  
  
    function new(string name="apb_slave_driver", uvm_object parent):  
        super.new(name, parent);
```

apb\_monitor\_c

uvm\_analysis\_port#(apb\_sequence\_item\_c)

`uvm\_component\_utils(apb\_slave\_driver)

```
static function string type_name();  
    return "apb_phase_filter_c"(PHASE_FILTER_C)  
endfunction
```



# Use constraint policies over hard coded constraints

- Problem:
  - Hard coding constraints directly into subclasses runs into scenarios where the constraint code must be duplicated. How to reuse constraints?
- Solution:
  - Write your constraint(s) once in a policy object.
  - Apply policy objects on sequence items, sequences or config objects as needed.
  - Use factory overrides to instantiate the objects which apply these policies.



## 5 Sequences and policies

Optional

SEQUENCES

POLICIES

SELECT POLICY ▾



## apb\_slave\_response\_policy\_c

Extends policy\_base\_c#(apb\_sequence\_item\_c)

Constraints

Add constraint...

apb\_sequence\_item\_c rand instance variables

CONTINUE

## 6 Configuration

Optional

## 7 Test

Optional

## 8 Generate

apb\_if.sv

policies/

class apb\_master\_policy\_c

constraint phase\_crv {

object.phase == APB\_PHASE\_RESPONSE;

}

}

constraint data\_crv {

if (object.write ==

object.data == '0

}

apb\_master\_policy\_c

seqs/

function void pre\_rand

super.pre\_randomize

if (cfg == null) begin

`uvm\_fatal(get\_type\_name(), "Mu

end

endfunction

class apb\_slave\_response\_policy\_c extends policy\_base\_c#(apb\_sequence\_item\_c

protected apb\_sequence\_item\_c request\_item;

constraint phase\_crv {

object.phase == APB\_PHASE\_RESPONSE;

}

constraint data\_crv {

if (object.write) {

object.data == request\_item.data;

}

}

constraint sel\_crv {

object.sel == request\_item.sel;

}

constraint addr\_crv {

object.addr == request\_item.addr;

}

constraint write\_crv {

object.write == request\_item.write;

}

constraint strb\_crv {

object.strb == request\_item.strb;

}

function new(string name="apb\_slave\_response\_policy\_c");

super.new(name);

endfunction

apb\_slave\_response\_policy\_c.svh

DONE

function void pre\_rand

super.pre\_randomize

if (cfg == null) begin

`uvm\_fatal(get\_type\_name(), "Mu

end

endfunction

`uvm\_object\_utils/apb\_master\_sequen

function new(string name="apb\_maste

super.new(name);

endfunction

apb\_config\_c cfg;

`uvm\_object\_utils\_begin(apb\_slave\_r

`uvm\_field\_object(cfg, UVM\_DEFAULT

`uvm\_object\_utils\_end



## 5 Sequences and policies

Optional

SEQUENCES

POLICIES

SELECT SEQUENCE ▾



## apb\_slave\_response\_sequence\_base\_c

Extends uvm\_sequence#(apb\_sequence\_item\_c)

Apply policies

Constrains apb\_sequence\_item\_c req (apb\_slave\_resp... ▾

Select or create a new policy

CONTINUE

## 6 Configuration

Optional

## 7 Test

Optional

## 8 Generate

```
protected apb_sequence_item_c req;
constraint pre_transfer_delay_crv {
    // ...
}

constraint phase_crv {
    object.phase == APB_SLAVE_RESPONSE_SEQUENCE;
}

constraint data_crv {
    if (object.write) {
        // ...
    }
}

apb_slave_response_sequence_base_c

seqs/

function void pre_randomize()
    super.pre_randomize();
endfunction

if (cfg == null) begin
    `uvm_fatal(get_type_name(), "Must call set_cfg(apb_config_c cfg) before randomization")
end

if (request_item == null) begin
    `uvm_fatal(get_type_name(), "Must call set_request_item(apb_sequence_item_c request_item) before randomization")
end

virtual task body();
    req = apb_sequence_item_c::type_id::create("apb_sequence_item");
    req.set_cfg(cfg);

    start_item(req);
    do_req(req);
    if (!req.randomize()) begin
        `uvm_error(get_type_name(), { req.get_type_name(), " randomization failed" })
    end

    finish_item(req);

    get_response(rsp);
endtask

virtual function void do_req(apb_sequence_item_c req);
    apb_slave_response_policy_c apb_slave_response_policy = new("apb_slave_response_policy");

    apb_slave_response_policy.set_request_item(request_item);
    req.add_policy(apb_slave_response_policy);
endfunction

endclass
```

apb\_slave\_response\_sequence\_base\_c.svh

DONE

apb\_slave\_response\_sequence\_base\_c.svh

apb\_slave\_response\_sequence\_base\_c.svh



apb\_interface\_harness.sv

apb.sv

tb/

class apb\_test\_base\_c extends

```
apb_env_c
apb_env_config_c
apb_top_virtual_sequence
int
time
time
```

```
`uvm_component_utils
```

apb\_test\_base\_c.sv

```
`ifndef _APB_TEST_PKG
`define _APB_TEST_PKG
```

package apb\_test\_pkg;

```
timeunit 1ns / 1ns;

import uvm_pkg::*;
import policy_pkg::*;
```

apb\_test\_pkg.sv

tb/policies/

class apb\_clk\_frequency

```
constraint freq_crv {
    object.freq == 100;
}

constraint unit_crv {
```

class apb\_test\_c extends apb\_test\_base\_c;

```
`uvm_component_utils(apb_test_c)
```

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

```
virtual function void build_phase(uvm_phase phase);
    super.build_phase(phase);
endfunction
```

```
virtual function void end_of_elaboration_phase(uvm_phase phase);
    super.end_of_elaboration_phase(phase);
```

```
apb_top_virtual_sequence_c::type_id::set_type_override(apb_top_no_reset
apb_run_virtual_sequence_base_c::type_id::set_type_override(apb_run_vir
apb_clk_sequence_base_c::type_id::set_type_override(apb_clk_sequence_c
apb_rst_initial_sequence_base_c::type_id::set_type_override(apb_rst_ini
rst_master_sequence_base_c::type_id::set_type_override(rst_master_seque
apb_master_sequence_base_c::type_id::set_type_override(apb_master_sequ
apb_slave_response_sequence_base_c::type_id::set_type_override(apb_slave
```

endclass

apb\_test\_c.svh

DONE

```
class apb_main_num_apb_transactions_p
    constraint num_apb_transactions_crv
        object.num_apb_transactions == 10
}

constraint unit_crv {
```

```
class apb_run_no_resets_policy_c exte
    constraint num_mid_sim_resets_crv {
        object.num_mid_sim_resets == 0;
    }
}
```



≡ apb

✓ Bus protocol

5 Sequences and policies  
Optional6 Configuration  
Optional✓ Test  
Optional☒ Generate a development testbench

DEFAULTS

TESTS

PARAMS

SELECT TEST ▾



apb\_test\_c

Extends apb\_test\_base\_c

Factory overrides



# Scale testbench components/objects at runtime

- Problem:
  - Compile-time instance scaling requires class type specializations. And as we know, that's no fun.
- Solution:
  - Collect DUT parameters at runtime and make them available to env and agent configs.
  - Using these parameter variables and dynamic arrays to:
    - Construct agents and sub-env instances.
    - Construct env-configs and agent configs.
    - Construct/connect scoreboard, predictor and coverage TLM.
    - Construct sub-env predictors.





## axir\_master\_if

Instance of axir\_if

## Parameters

Name	Value
ARID_MSB	Not set (default: 3)
ARADDR_MSB	ARADDR_MSB
ARADDR_LSB	ARADDR_LSB
ARLEN_MSB	ARLEN_MSB
RID_MSB	RID_MSB
RDATA_MSB	RDATA_MSB
RRESP_MSB	RRESP_MSB
ON_POSEDGE	Not set (default: 1)

## Ports

Name	Connection
clk	clk
reset_n	reset_n
arid	arid_slvx[d]
araddr	araddr_slvx[d]
arlen	arlen_slvx[d]

```
virtual function void do_axir_maste
axir master max num
virtual function void do_axir_maste
axir master max num
virtual function void do_axir_maste
axir master max num
```

hub\_default\_env.c

class hub\_predictor\_c

```
`uvm_analysis_imp_dec
`uvm_analysis_imp_dec
`uvm_analysis_imp_dec
`uvm_analysis_imp_dec
`uvm_analysis_imp_dec
`uvm_analysis_imp_dec
`uvm_analysis_imp_dec
`uvm_analysis_imp_dec
```

hub\_predictor\_c.s

```
`ifndef __HUB_ENV_PKG_S
`define __HUB_ENV_PKG_S
```

package hub\_env\_pkg;

timeunit 1ns / 1ns;

```
import uvm_pkg::*;
import policy_pkg::*;
import comparator_pkg::*;
```

hub\_env\_pkg.sv

```
`ifndef __HUB_SIGNAL_CH
`define __HUB_SIGNAL_CH
```

module hub\_signal\_check

```
#(
parameter ADDRESS_WIDTH=0,
parameter DATA_WIDTH=0,
parameter ID_MSB=0,
parameter NUM_X=0,
parameter NUM_Y1=0,
```

```
for (genvar d1 = 4; d1 < (NUM_Y2 - 2); d1++) begin : GEN_IF_RESOURCES1
initial begin
wait (env_full_name != "");
```

```
GEN_IF1_D0[D0].GEN_IF1_D1[D1].sideband_master_y_if.add_resources_t
GEN_IF1_D0[D0].GEN_IF1_D1[D1].sideband_slave_y_if.add_resources_to
GEN_IF1_D0[D0].GEN_IF1_D1[D1].apb_master_y_if.add_resources_to_con
GEN_IF1_D0[D0].GEN_IF1_D1[D1].apb_slave_y_if.add_resources_to_conf
GEN_IF1_D0[D0].GEN_IF1_D1[D1].axir_master_y_if.add_resources_to_co
GEN_IF1_D0[D0].GEN_IF1_D1[D1].axir_slave_y_if.add_resources_to_con
GEN_IF1_D0[D0].GEN_IF1_D1[D1].axiw_master_y_if.add_resources_to_co
GEN_IF1_D0[D0].GEN_IF1_D1[D1].axiw_slave_y_if.add_resources_to_con
```

```
end
end
end
```

```
function void add_resources_to_config_db(string env_full_name);
hub_interface_harness.env_full_name = env_full_name;
add_parameters_to_config_db(env_full_name);
endfunction
```

```
function automatic void add_parameters_to_config_db(string inst_name);
hub_env_parameters_c params = hub_env_parameters_c::type_id::create("p
params.set_address_width(ADDRESS_WIDTH);
params.set_data_width(DATA_WIDTH);
params.set_id_msb(ID_MSB);
```

```
params.set_num_x(NUM_X);
params.set_num_y1(NUM_Y1);
params.set_num_y2(NUM_Y2);
params.set_insert_bug(INSERT_BUG);
uvm_config_db#(hub_env_parameters_c)::set(null, inst_name, "env_params
endfunction
```

endmodule

hub\_interface\_harness.sv

DONE



## axir\_master\_if

Instance of axir\_if

## Parameters

Name	Value
ARID_MSB	Not set (default: 3)
ARADDR_MSB	ARADDR_MSB
ARADDR_LSB	ARADDR_LSB
ARLEN_MSB	ARLEN_MSB
RID_MSB	RID_MSB
RDATA_MSB	RDATA_MSB
RRESP_MSB	RRESP_MSB
ON_POSEDGE	Not set (default: 1)

## Ports

Name	Connection
clk	clk
reset_n	reset_n
arid	arid_slvx[d]
araddr	araddr_slvx[d]
arlen	arlen_slvx[d]

class hub\_predictor\_c

```

`uvm_analysis_imp_dec
`uvm_analysis_imp_dec
`uvm_analysis_imp_dec
`uvm_analysis_imp_dec
`uvm_analysis_imp_dec
`uvm_analysis_imp_dec
`uvm_analysis_imp_dec
`uvm_analysis_imp_dec
`uvm_analysis_imp_dec
`uvm_analysis_imp_dec

```

hub\_predictor\_c.sv

```

`ifndef _HUB_ENV_PKG_S
`define _HUB_ENV_PKG_S

```

package hub\_env\_pkg;

timeunit 1ns / 1ns;

```

import uvm_pkg::*;
import policy_pkg::*;
import comparator_pkg::*;

```

hub\_env\_pkg.sv

```

`ifndef _HUB_SIGNAL_CHECKER_S
`define _HUB_SIGNAL_CHECKER_S

```

module hub\_signal\_checker

```

#(
    parameter ADDRESS_WIDTH=0,
    parameter DATA_WIDTH=0,
    parameter ID_MSB=0,
    parameter NUM_X=0,
    parameter NUM_Y1=0,

```

hub\_signal\_checker.sv

```

if (env_cfg.get_signal_checker_bound(bound))
    env_cfg.set_signal_checker_bound(bound);
end

```

```

if (env_cfg.has_agents()) begin
    clk_agent = clk_agent_c::type_id::create("clk_agent", this);
    clk_agent.set_cfg(env_cfg.clk_cfg);

```

```

    rst_master_agent = rst_master_agent_c::type_id::create("rst_master_agent", this);
    rst_master_agent.set_cfg(env_cfg.rst_master_cfg);

```

```

    sideband_master_agent = sideband_master_agent_c::type_id::create("sideband_master_agent", this);
    sideband_master_agent.set_cfg(env_cfg.sideband_master_cfg);

```

```

    apb_slave_agent = apb_slave_agent_c::type_id::create("apb_slave_agent", this);
    apb_slave_agent.set_cfg(env_cfg.apb_slave_cfg);

```

```

    axir_master_agents = new[env_cfg.params.num_x];
    foreach (axir_master_agents[a]) begin
        axir_master_agents[a] = axir_master_agent_c::type_id::create($sprintf("axir_master_agent_%d", a), this);
        axir_master_agents[a].set_cfg(env_cfg.axir_master_cfgs[a]);
    end

```

```

    axir_slave_agents = new[env_cfg.params.num_x];
    foreach (axir_slave_agents[a]) begin
        axir_slave_agents[a] = axir_slave_agent_c::type_id::create($sprintf("axir_slave_agent_%d", a), this);
        axir_slave_agents[a].set_cfg(env_cfg.axir_slave_cfgs[a]);
    end

```

```

    axiw_master_agents = new[env_cfg.params.num_x];
    foreach (axiw_master_agents[a]) begin
        axiw_master_agents[a] = axiw_master_agent_c::type_id::create($sprintf("axiw_master_agent_%d", a), this);
        axiw_master_agents[a].set_cfg(env_cfg.axiw_master_cfgs[a]);
    end

```

```

    axiw_slave_agents = new[env_cfg.params.num_x];

```

hub\_env\_c.svh

DONE



## Sub-environments

Optional

SELECT SUB-ENV ▾



## relay\_x\_envs[]

Instance of relay\_env\_c

HDL path

GEN\_X[NUM\_X].relay\_x

CONTINUE

## Predictor

## 5 Coverage

## 6 Configuration and policies

Optional

## Signal checker

Optional

class hub\_predictor\_c

```

`uvm_analysis_imp_dec
`uvm_analysis_imp_dec
`uvm_analysis_imp_dec
`uvm_analysis_imp_dec
`uvm_analysis_imp_dec
`uvm_analysis_imp_dec
`uvm_analysis_imp_dec
`uvm_analysis_imp_dec
`uvm_analysis_imp_dec
`uvm_analysis_imp_dec

```

hub\_predictor\_c.sv

```

`ifndef __HUB_ENV_PKG_S
`define __HUB_ENV_PKG_S

```

package hub\_env\_pkg;

timeunit 1ns / 1ns;

```

import uvm_pkg::*;
import policy_pkg::*;
import comparator_pkg::*;

```

hub\_env\_pkg.sv

```

`ifndef __HUB_SIGNAL_CH
`define __HUB_SIGNAL_CH

```

module hub\_signal\_checker

```

#(
    parameter ADDRESS_WIDTH=0,
    parameter DATA_WIDTH=0,
    parameter ID_MSB=0,
    parameter NUM_X=0,
    parameter NUM_Y1=0,

```

hub\_signal\_checker.sv

end

```

axi_slave_y_agents = new[(env_cfg.params.num_y1 - 1) - 3];
foreach (axi_slave_y_agents[a0]) begin
    axi_slave_y_agents[a0] = new[(env_cfg.params.num_y2 - 2) - 4];
    for (int a1 = 0; a1 < axi_slave_y_agents[a0].size(); a1++) begin
        axi_slave_y_agents[a0][a1] = axi_slave_agent_c::type_id::create(
            axi_slave_y_agents[a0][a1].set_cfg(env_cfg.axi_slave_y_cfgs[a0]
        end
    end
end
end

```

```

if (env_cfg.relay_env_cfg.has_env()) begin
    relay_env = relay_env_c::type_id::create("relay_env", this);
    relay_env.set_env_cfg(env_cfg.relay_env_cfg);
end

```

```

relay_x_envs = new[env_cfg.params.num_x];
foreach (relay_x_envs[e]) begin
    if (env_cfg.relay_x_env_cfgs[e].has_env()) begin
        relay_x_envs[e] = relay_env_c::type_id::create($sformatf("relay_x_
        relay_x_envs[e].set_env_cfg(env_cfg.relay_x_env_cfgs[e]);
    end
end

```

```

relay_y_envs = new[(env_cfg.params.num_y1 - 1) - 3];
foreach (relay_y_envs[e0]) begin
    relay_y_envs[e0] = new[(env_cfg.params.num_y2 - 2) - 4];
    for (int e1 = 0; e1 < relay_y_envs[e0].size(); e1++) begin
        if (env_cfg.relay_y_env_cfgs[e0][e1].has_env()) begin
            relay_y_envs[e0][e1] = relay_env_c::type_id::create($sformatf("r
            relay_y_envs[e0][e1].set_env_cfg(env_cfg.relay_y_env_cfgs[e0][e1]
        end
    end
end

```

hub\_env\_c.svh

DONE



## ✓ Predictor

## hub\_predictor\_c

Extends uvm\_component

## Subroutines

- function void write\_actual\_request\_from\_rst\_master(rst\_sequen... ▾
- function void write\_actual\_request\_from\_sideband\_master(side... ▾
- function void write\_actual\_response\_from\_apb\_slave(apb\_sequ... ▾
- function void write\_actual\_request\_from\_axir\_master\_x(axir\_seq... ▾
- function void write\_actual\_response\_from\_axir\_slave\_x(axir\_seq... ▾
- function void write\_actual\_request\_from\_axiw\_master\_x(axiw\_s... ▾
- function void write\_actual\_data\_from\_axiw\_master\_x(axiw\_sequ... ▾
- function void write\_actual\_response\_from\_axiw\_slave\_x(axiw\_s... ▾
- function void write\_actual\_request\_from\_sideband\_master\_y(si... ▾
- function void write\_actual\_response\_from\_sideband\_slave\_y(sid... ▾
- function void write\_actual\_request\_from\_apb\_master\_y(apb\_se... ▾
- function void write\_actual\_response\_from\_apb\_slave\_y(apb\_sequ... ▾
- function void write\_actual\_request\_from\_axir\_master\_y(axir\_seq... ▾
- function void write\_actual\_response\_from\_axir\_slave\_y(axir\_seq... ▾
- function void write\_actual\_request\_from\_axiw\_master\_y(axiw\_s... ▾

endclass

logic [31:0] num\_y2;

if (!clk\_cfg.randomize()) begin

get\_type\_name(), { c

abstract\_hub\_sign

fig\_c.svh

class hub\_default\_env\_c

'uvm\_object\_utils(hub

```
function new(string n
    super.new(name);
endfunction
```

```
virtual function void
    axir_master_max_num
```

hub\_default\_env\_c

class hub\_predictor\_c e

```
'uvm_analysis_imp_dec
'uvm_analysis_imp_dec
'uvm_analysis_imp_dec
'uvm_analysis_imp_dec
'uvm_analysis_imp_dec
'uvm_analysis_imp_dec
'uvm_analysis_imp_dec
'uvm_analysis_imp_dec
```

hub\_predictor\_c.sv

'ifndef \_HUB\_ENV\_PKG\_S

'define \_HUB\_ENV\_PKG\_S

package hub\_env\_pkg;

timeunit 1ns / 1ns;

```
import uvm_pkg::*;
import policy_pkg::*;
import comparator_pkg::*;
```

```
virtual function void set_env_cfg(hub_env_config_c env_cfg);
    this.env_cfg = env_cfg;
endfunction
```

```
virtual function void build_phase(uvm_phase phase);
    if (env_cfg == null) begin
        `uvm_fatal(get_type_name(), "Must call set_env_cfg(hub_env_config_c
    end
```

actual\_request\_from\_rst\_master\_imp = new("actual\_request\_from\_rst\_maste

```
rst_master_export = new("rst_master_export", this);
sideband_master_export = new("sideband_master_export", this);
apb_slave_export = new("apb_slave_export", this);
```

```
axir_master_exports = new[env_cfg.params.num_x];
foreach (axir_master_exports[e]) begin
    axir_master_exports[e] = new($sformatf("axir_master_exports[%0d]", e
end
```

```
axir_slave_exports = new[env_cfg.params.num_x];
foreach (axir_slave_exports[e]) begin
    axir_slave_exports[e] = new($sformatf("axir_slave_exports[%0d]", e),
end
```

```
axiw_master_exports = new[env_cfg.params.num_x];
foreach (axiw_master_exports[e]) begin
    axiw_master_exports[e] = new($sformatf("axiw_master_exports[%0d]", e
end
```

```
axiw_slave_exports = new[env_cfg.params.num_x];
foreach (axiw_slave_exports[e]) begin
    axiw_slave_exports[e] = new($sformatf("axiw_slave_exports[%0d]", e),
end
```

hub\_scoreboard\_c.svh

DONE

```
parameter ADDRESS_WIDTH=0,
parameter DATA_WIDTH=0,
parameter ID_MSB=0,
parameter NUM_X=0,
parameter NUM_Y1=0,
```

```
parameter ADDRESS_WIDTH=0,
parameter DATA_WIDTH=0,
parameter ID_MSB=0,
parameter NUM_X=0,
parameter NUM_Y1=0,
```



# Conditional instantiation of static verification elements at compile time

- Problem:
  - Sub-environments and SVA may not be needed in every test regression and can bog down full-chip simulation performance.
- Solution:
  - Make instantiation of static verification sub-elements, such as interfaces, protocol checkers and signal checkers, conditional at compile time.
  - Create clear, easy to use macro definitions to disable binding of verification elements individually or all at once.
  - Enable verification sub-environments and/or SVA as needed for debug.



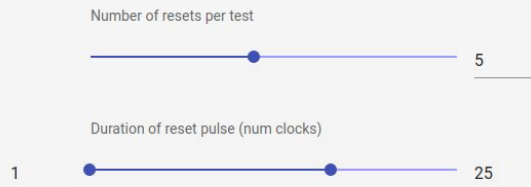
## 2 Default settings

Optional

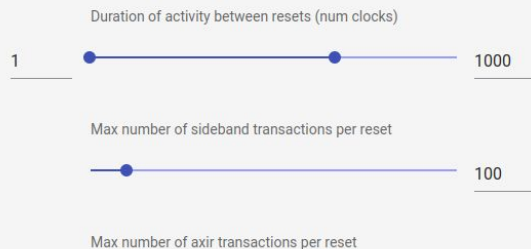
## Clock



## Reset



## Transactions



```
virtual function void build_phase(uvm_component* c) {  
    hub_env_config c;   
    hub_in_order_test  
    build_phase(uvm_component* c) {  
        .BRESP_MSB (BRESP_MSB)  
        ) hub_signal_checker (.*);  
    }  
`endif
```

// RELAY Bind statements

```
`ifdef ENABLE_RELAY_FUNCTIONAL_VERIFICATION  
    bind relay: hub.relay1 relay_interface_harness #(  
        .ADDRESS_WIDTH (ADDRESS_WIDTH),  
        .DATA_WIDTH (DATA_WIDTH),  
        .INSERT_BUG (INSERT_BUG),  
        .ARID_FOO_MSB (ARID_FOO_MSB),  
        .AWID_MSB (AWID_MSB),  
        .SHOULD_NOT_BE_IGNORED (SHOULD_NOT_BE_IGNORED),  
        .ARADDR_MSB (ARADDR_MSB),  
        .ARADDR_LSB (ARADDR_LSB),  
        .ARLEN_MSB (ARLEN_MSB),  
        .RID_MSB (RID_MSB),  
        .RDATA_MSB (RDATA_MSB),  
        .RRESP_MSB (RRESP_MSB),  
        .AWADDR_MSB (AWADDR_MSB),  
        .AWADDR_LSB (AWADDR_LSB),  
        .AWLEN_MSB (AWLEN_MSB),  
        .WID_MSB (WID_MSB),  
        .WDATA_MSB (WDATA_MSB),  
        .BID_MSB (BID_MSB),  
        .BRESP_MSB (BRESP_MSB)  
    ) relay_interface_harness (.*);  
`endif
```

```
`ifdef ENABLE_RELAY_PROTOCOL_CHECKERS  
    bind relay: hub.relay1 relay_protocol_checker_harness #(  
        .ADDRESS_WIDTH (ADDRESS_WIDTH),  
        .DATA_WIDTH (DATA_WIDTH),  
        .INSERT_BUG (INSERT_BUG),  
        .ARID_FOO_MSB (ARID_FOO_MSB),  
        .AWID_MSB (AWID_MSB),  
        .SHOULD_NOT_BE_IGNORED (SHOULD_NOT_BE_IGNORED),  
        .ARADDR_MSB (ARADDR_MSB),  
        .ARADDR_LSB (ARADDR_LSB),  
        .ARLEN_MSB (ARLEN_MSB),  
        .RID_MSB (RID_MSB),  
        .RDATA_MSB (RDATA_MSB),  
        .RRESP_MSB (RRESP_MSB),  
        .AWADDR_MSB (AWADDR_MSB),  
        .AWADDR_LSB (AWADDR_LSB),  
        .AWLEN_MSB (AWLEN_MSB),  
        .WID_MSB (WID_MSB),  
        .WDATA_MSB (WDATA_MSB),  
        .BID_MSB (BID_MSB),  
        .BRESP_MSB (BRESP_MSB)  
    ) relay_protocol_checker_harness (.*);  
`endif
```

hub\_bind.svh

DONE

policies/

class hub\_clk\_frequency

```
constraint freq_crv {  
    object.freq == 100;  
}
```

```
constraint unit_crv {  
    object.unit == CLOG  
}
```

hub\_clk\_frequency

typedef class hub\_main

class hub\_main\_num\_axi\_transactions

```
constraint num_axi_transactions_cr  
foreach (object.num_axi_transact  
    object.num_axi_transactions[n]
```

class hub\_main\_num\_sideband\_y\_transac

```
constraint num_sideband_y_transacti  
foreach (object.num_sideband_y_tr  
    object.num_sideband_y_transacti
```

class hub\_main\_num\_apb\_y\_transactions

```
constraint num_apb_y_transactions_c  
foreach (object.num_apb_y_transac  
    object.num_apb_y_transactions[n]
```



# Does your company use all these best practices?

- Interface harnesses
- Abstract/concrete classes
- DUT parameter passing to VIP
- Scale UVC, sub-env, config and TLM instances at runtime
- Conditional instantiation of static verification elements at compile time
- Pass down config object over config db
- Use sequence, BFM and config factory overrides
- Use slave sequences with late response randomization
- Use test methodology: don't kill sequences with the sequencer
- Use virtual sequences over phase jumping
- No virtual sequencers
- Use objections wisely
- Use constraint policies over inheritance
- Use standalone testbench for UVC development
- And many more...

NOPE!



# The UVM dream is not today's reality

- Tight schedules. Large designs. Lots of new features.
- Some UVM best practices are not known to engineers.
- Some UVM best practices are daunting to implement.
- Hard for a large organization to be in sync on methodology.
- Reuse per the UVM dream is not easy and takes a lot of code!



# Today's reality

- ~~Careful implementation of best practices for reuse and scalability.~~
- Make it work! Get it verified. On time. However possible.
- Monolithic verification decisions made to hit deadlines.
  - It makes sense. Reuse is not quick or easy to implement.
- Hacks to fix hacks.
- Code rot ensues...





# Introducing UVMGen Technology

- With the best DV practices across the industry distilled and encoded into the UVMGen code generator, users can stand on the shoulders of DV experts.
- Generate world-class VIP in an instant, reuse at a click, integrate and scale with ease.
- Now everybody can code like a guru and capitalize on the UVM promise.
- Verify more features.
- In less time.
- With higher confidence.
- And less brain juice :p

**Go to [uvmgen.com](https://uvmgen.com) and  
Generate your UVCs for free!**



[uvmgen.com](http://uvmgen.com)