How to Create a Complex Testbench in a Couple of Hours

Tom Fitzpatrick

Graeme Jessiman

Mentor®
A Siemens Business

accellera
SYSTEMS INITIATIVE

2017 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

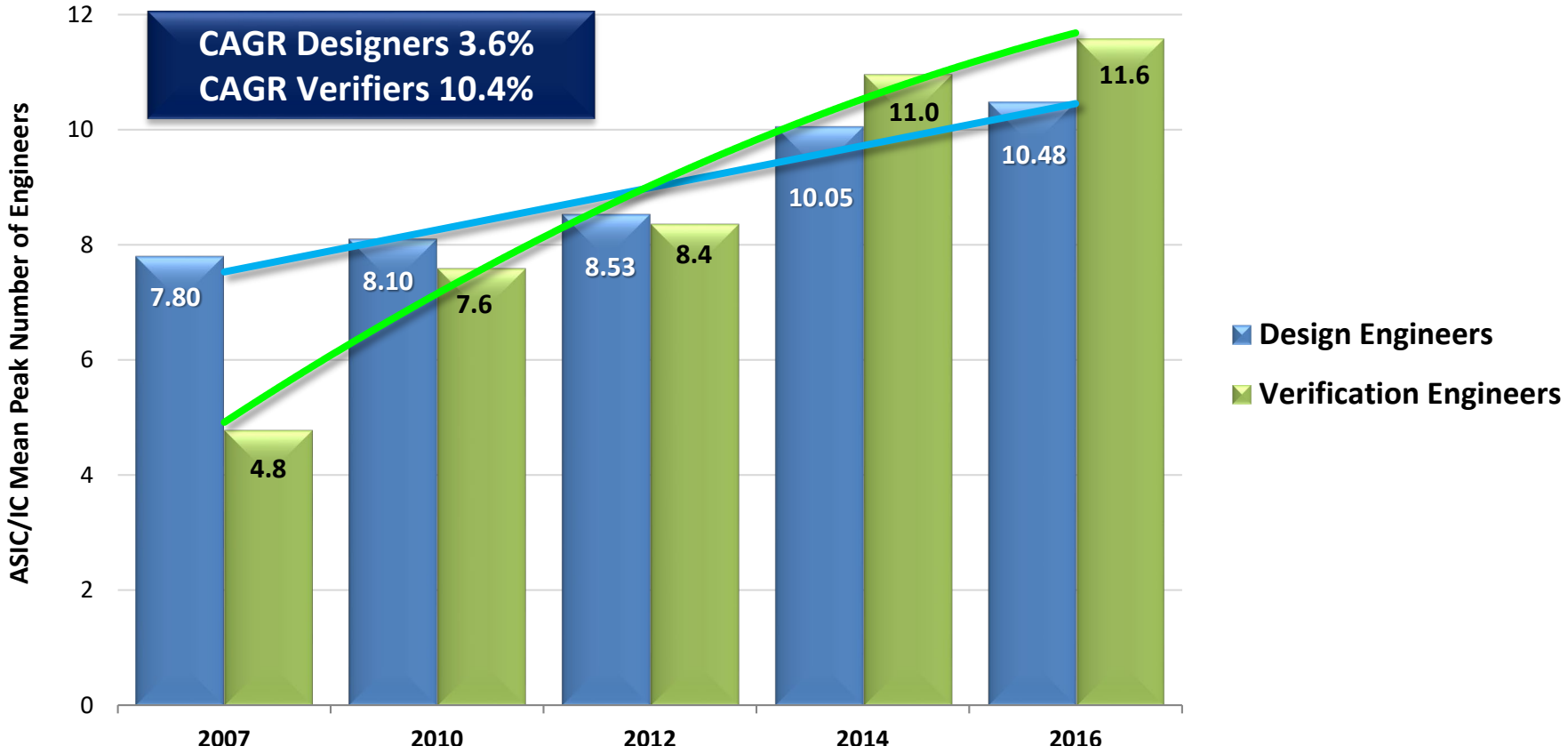
INTRODUCTION

Advanced Verification

The promise 15 years ago

- Reuse
  - From Project to Project
  - From block level to system level
- Coverage and Verification Management
  - How do we know we're done ?
  - Project Management, Metrics
- Constrained Random
  - Directed testing finds the known unknowns
  - Constrained Random finds the unknown unknowns

More Verification Engineers vs Design Engineers



Source: Wilson Research Group and Mentor Graphics, 2016 Functional Verification Study

Why so much Time and Money on Verification ?

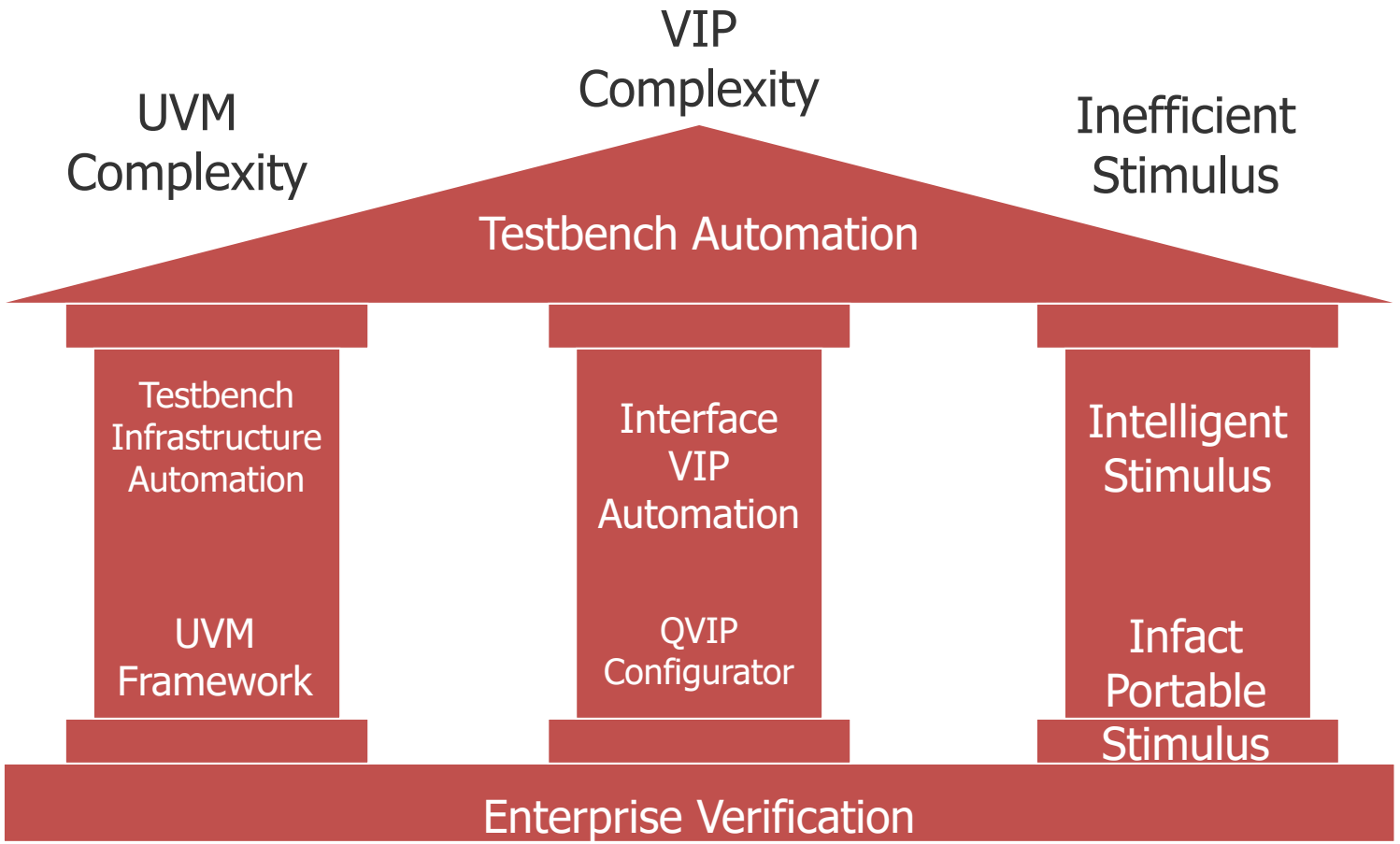
- Complexity of Methodology
  - AVM in 2006 : 6,000 lines of code
  - UVM in 2016 : 75,000 lines of code
  - Delivers re-use but hard to get up and running
- Complexity of VIP
  - Protocols themselves are complex and therefore highly configurable
  - Can be hard to instantiate, configure and bring up
- Constrained Random is dumb
  - Many “useless” repeated tests, consuming compute resource but not improving coverage
  - Directed vs Constrained Random trade-off

accellera SYSTEMS INITIATIVE

2017 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

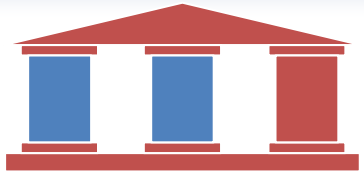
Testbench Automation

Making Verification Easier



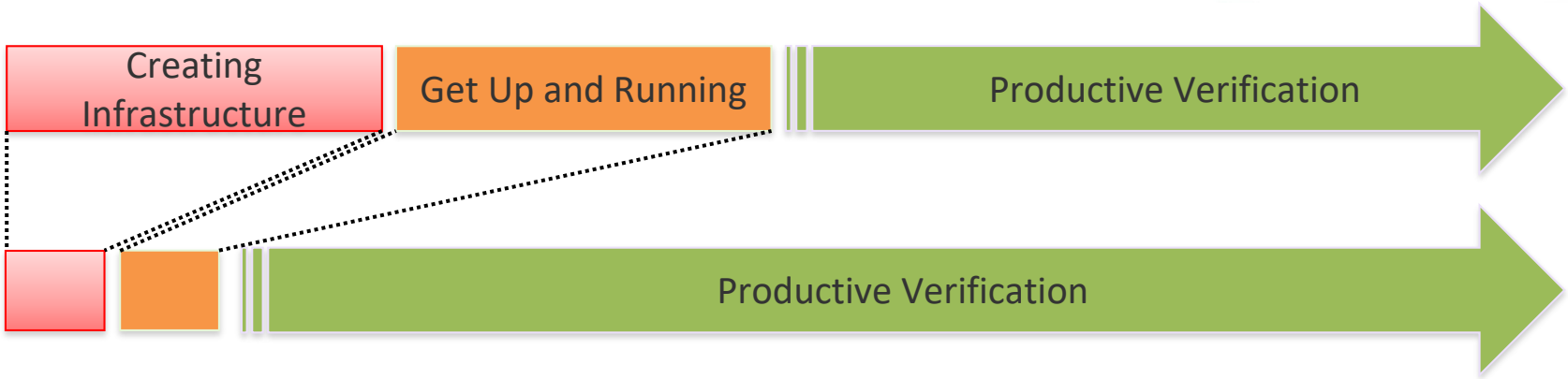
Testbench Generation : UVM-Framework and Configurator

Become Productive in Hours Rather than Weeks



- UVM Framework
  - Generates Testbench for SoC and Proprietary Interfaces

- QVIP Configurator
  - Generates Testbench for Standard Interfaces



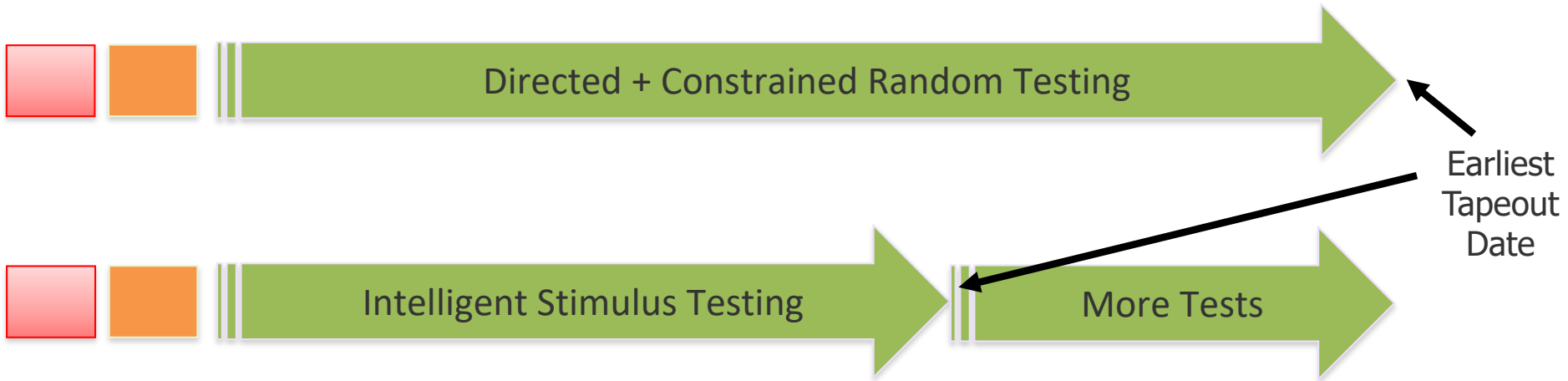
accellera SYSTEMS INITIATIVE

2017 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

Intelligent Stimulus with InFact Portable Stimulus

Earlier Tapeout and/or Higher Quality

- Graph based stimulus combines strengths of directed and constrained random
- Supports Portable Stimulus Standard
- Graph Kits available for QVIPs

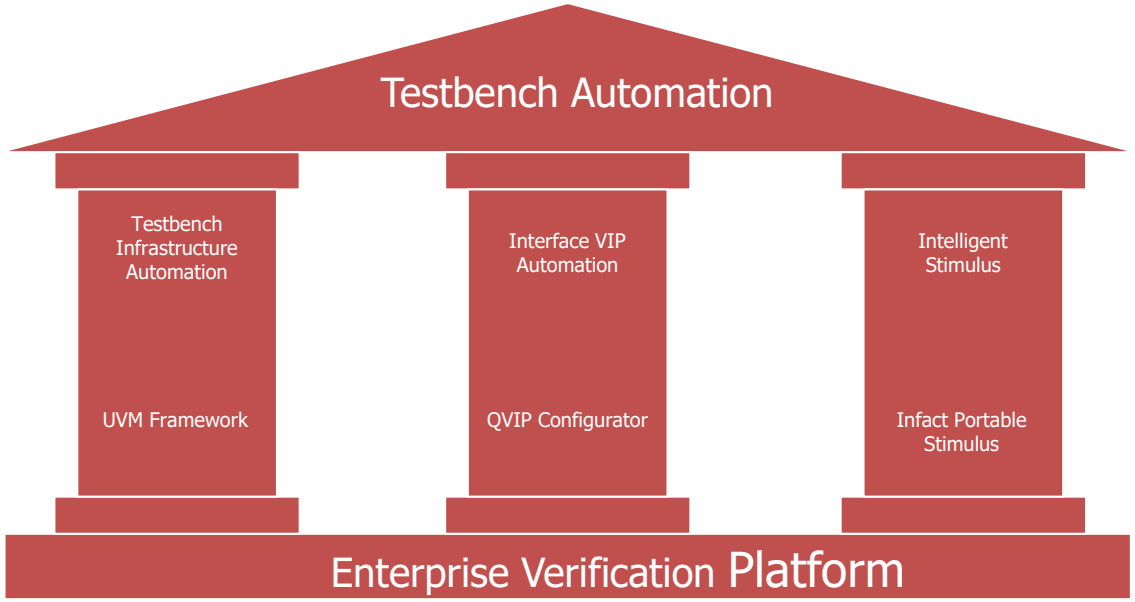


accellera SYSTEMS INITIATIVE

2017 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

Testbench Automation Benefits

For New UVM Users



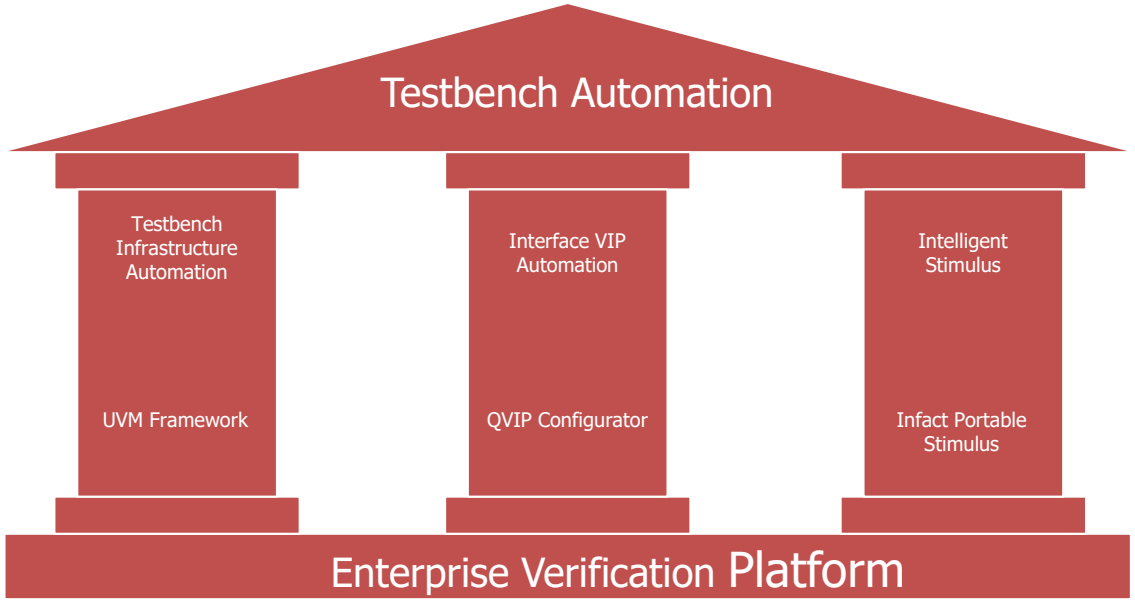
- Reduces Ramp-Up Time
- Reduces Project Risk
- Faster Payback on Investment

accellera SYSTEMS INITIATIVE

2017 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

Testbench Automation Benefits

For Experienced UVM Users



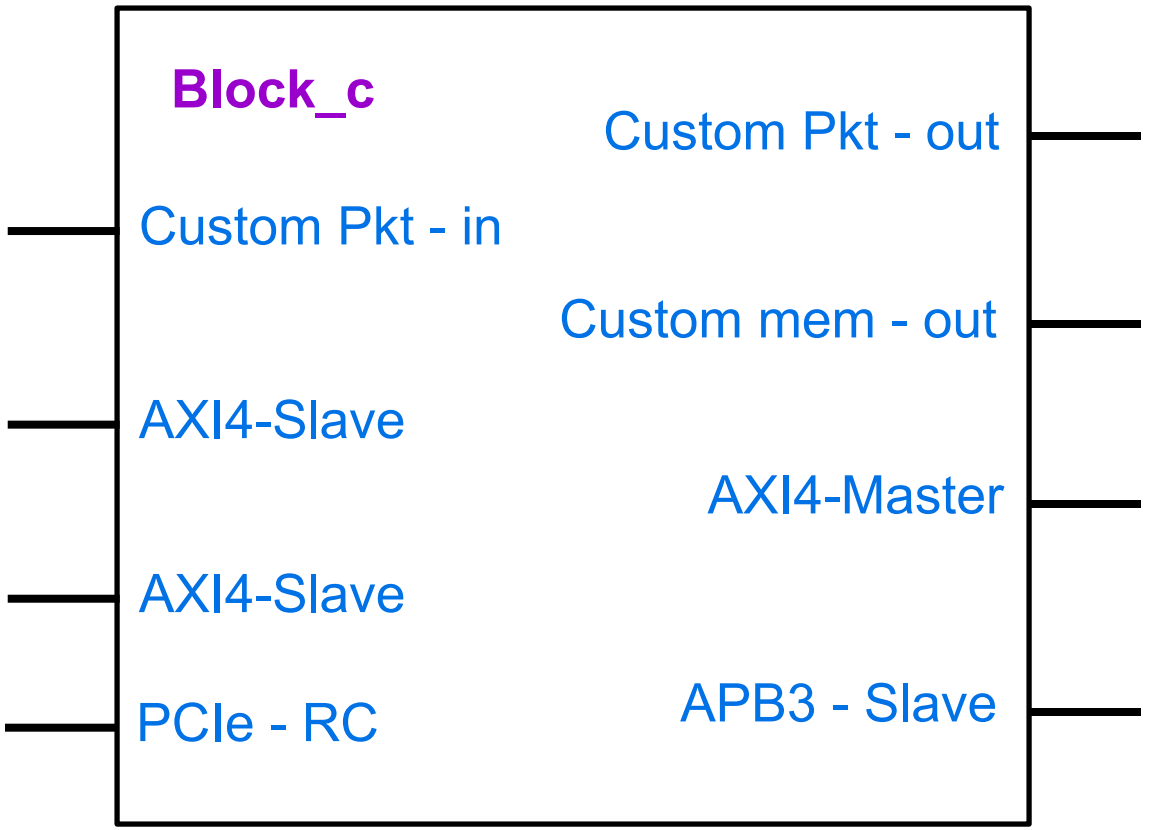
- Reduces Ramp-Up Time
- Maximizes Productivity
- Best Use of Compute Resource

Agenda - Bench Automation

- Describe DUT – Block_c
- Describe Block_c Simulation Environment
- Build Block_c Simulation Environment
  - Interfaces, environments, test bench

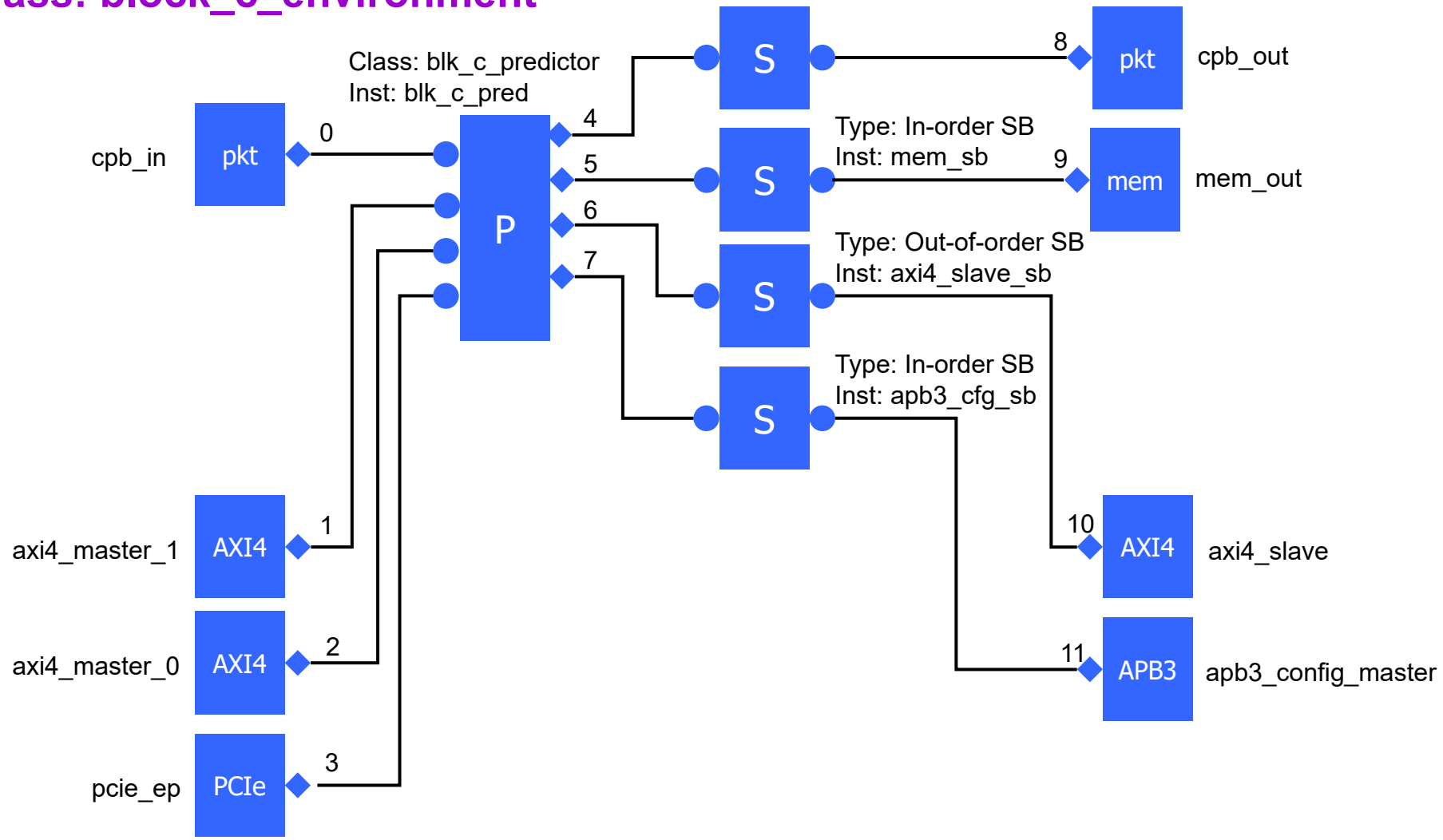
Start Creating Tests ASAP!

Design Under Test



Block_c Environment

Class: block_c_environment



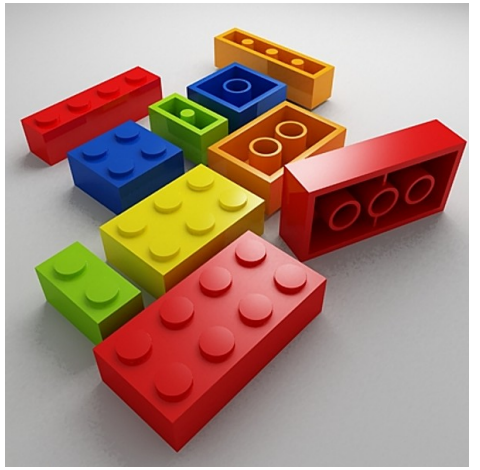
INTERFACE

Custom Interfaces

Create Custom Interface packages
Using UVMF Interface Code Generator

What is the UVM Framework

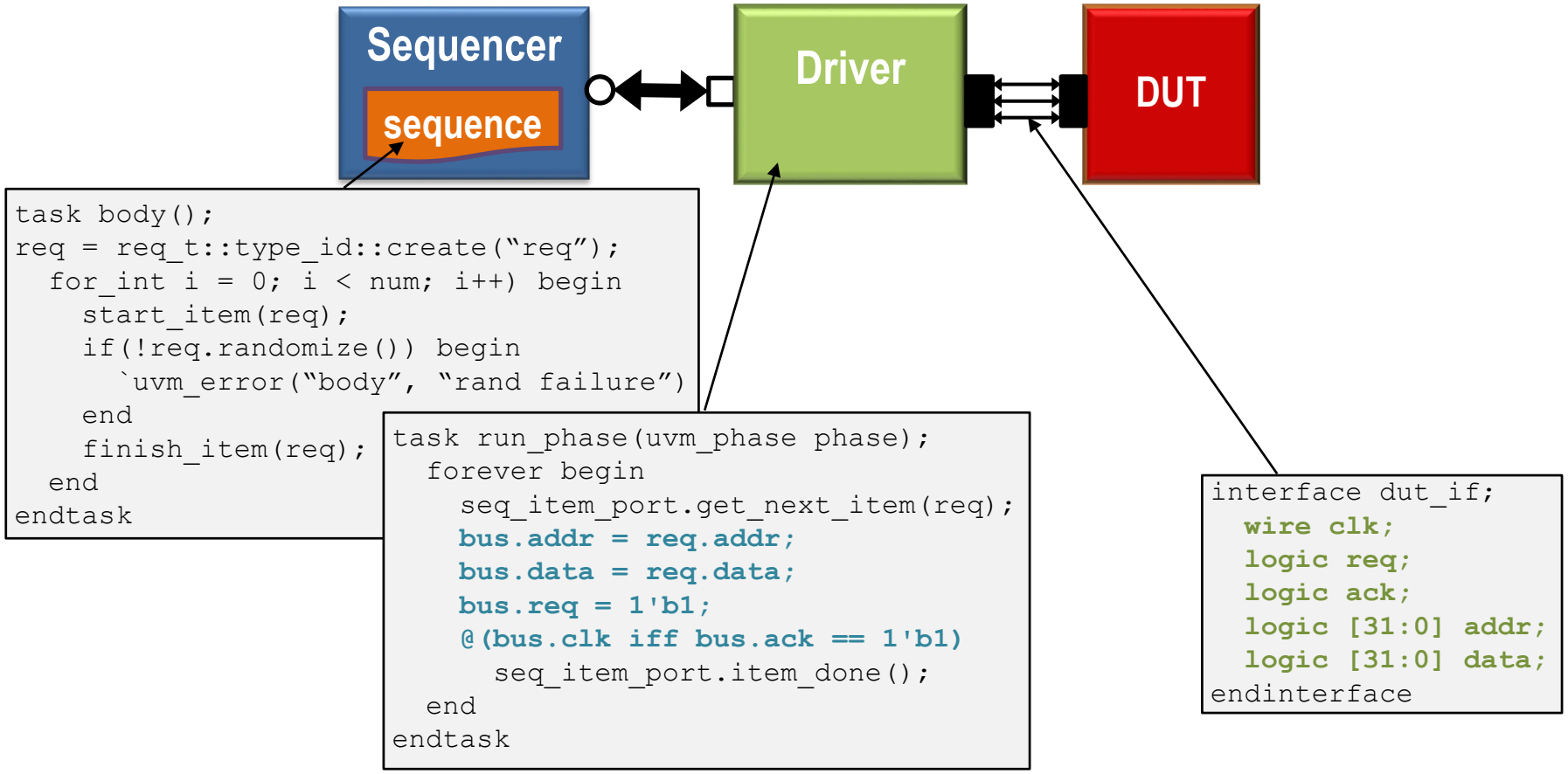
- UVM Code Generator
  - Interfaces, environments, test benches
- UVM Reuse methodology
  - Horizontal and vertical reuse
- UVM Jumpstart
  - Hide UVM details
  - Avoid reuse mistakes
- Free – Open Source



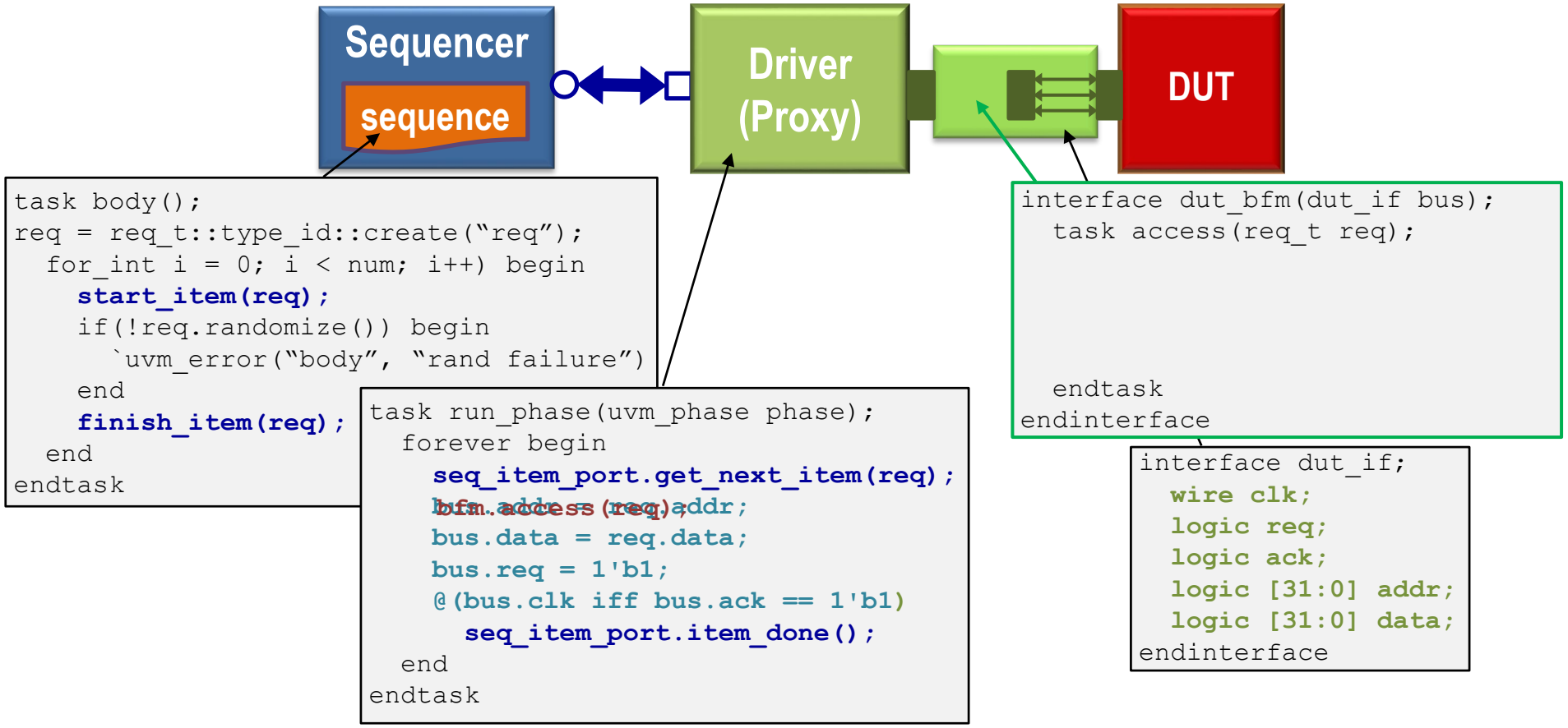
What Are We Creating?

- UVM based interface package
  - Classes: Transaction base, sequence base, random sequence, agent configuration, agent, driver, monitor, coverage, UVM reg adapter, UVM reg predictor
  - Package declaration
- Interface BFM's
  - Interfaces: driver, monitor, signal bundle
- Compile files
  - Compilation file list, makefile
- Operational VIP
  - Transactions are flowing but pins are not wiggling

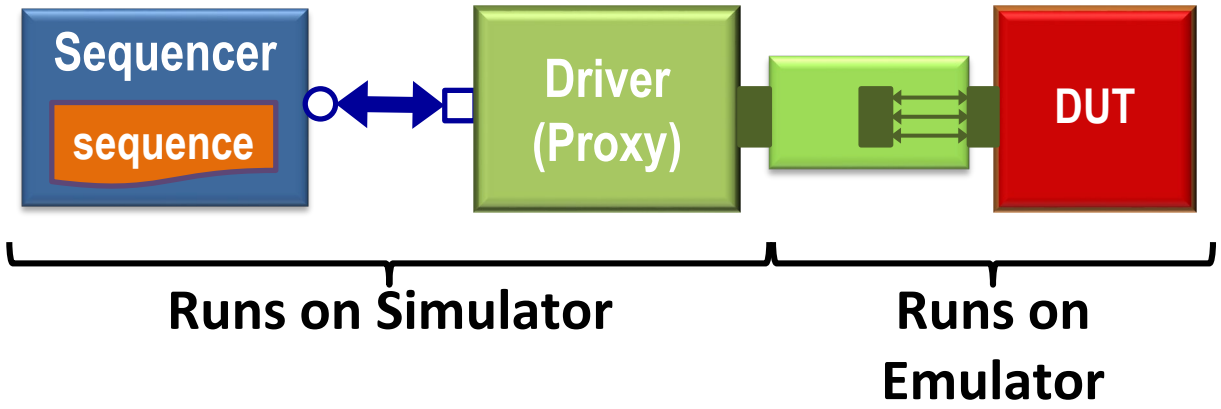
"Traditional UVM"



UVM Framework Arranged for Efficiency

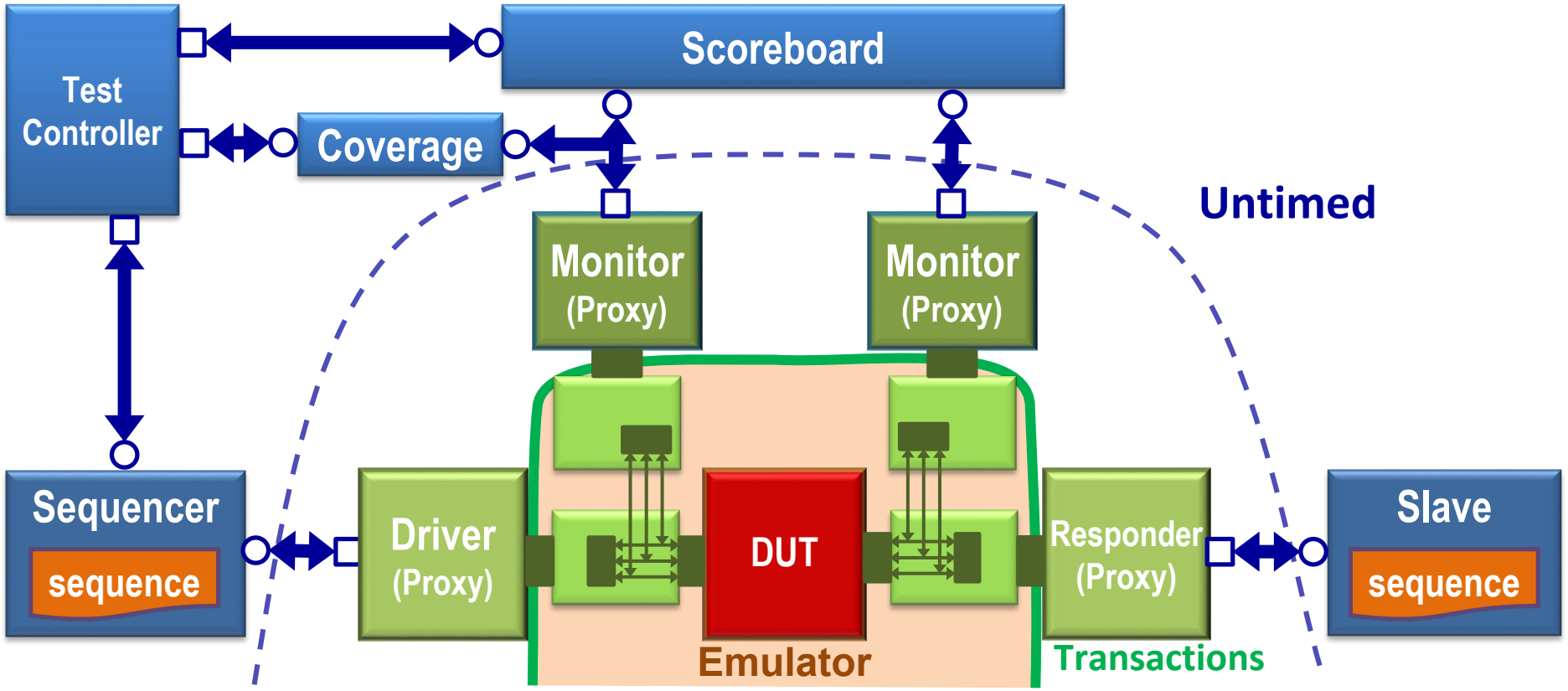


UVM Framework is Emulation-Ready!



- Automatically partitions design and testbench
- Supports both simulation and emulation

UVM Framework is Emulation-Ready!



Creating Custom VIP

- Library of Python API's and variables provided by UVMF
  - Used to characterize VIP
    - Name, clock, reset, signals, parameters, typedefs, variables, constraints

- What's left to complete?
  - Pin level signaling
  - Pin level monitoring
  - Add sequences unique to protocol

Start Creating Tests ASAP!

Interface Name, clk, rst

- Define the protocol name

```
intf = uvmf_gen.InterfaceClass('mem')
```

```
class mem_transaction       #(
        int DATA_WIDTH = 220,
        int ADDR_WIDTH = 210
        ) extends uvmf_transaction_base;
```

- Define the protocol clock and reset characteristics

Name	Value(s)	Description
clock	'signalName'	Name of primary clock. Additional clocks must be added manually.
reset	'resetName'	Name of primary reset. Additional resets must be added manually. If interface has no reset use 'dummy' and remove associated code from interface.
resetAssertionLevel	True|False	Assertion level for this protocol.

Parameters and Typedefs

Name	addParamDef
Description	This API adds a parameter to the interface classes and BFM's.

Usage	`addParamDef(`
	`'parameterName',`
	`'parameterType',`
	`'parameterDefalutValue'`
	`)`

Arguments	
`'parameterName'`	**Description:** Name of the parameter. **Value:** Any valid SV parameter name.
`'parameterType'`	**Description:** Type of the parameter. **Value:** Any valid SV parameter type.
`'parameterDefalutValue'`	**Description:** The default value for this parameter. **Value:** Any valid SV value for the parameter type.

```
class mem_transaction       #(
      int DATA_WIDTH = 220,
      int ADDR_WIDTH = 210
      ) extends uvmf_transaction_base;
```

Name	addHdlTypedef
Description	This API adds adds a typedef that is visible to UVM content and BFM's.

Usage	`addHdlTypedef(`
	`'typedefName',`
	`'typedefDefinition'`
	`)`

Arguments	
`'typedefName'`	**Description:** The name of the typedef. **Value:** Any valid SV typedef name.
`'typedefDefinition'`	**Description:** Type definition of the typedef. **Value:** Any valid SV typedef definition.

```
typedef byte my_byte_t;
typedef bit [15:0]  my_word_t;
```

Define Interface Signals

Name	addPort
Description	This API adds a signal to the interface package. Use this API for each signal in the protocol with the exception of the clock and reset defined in the interface variables.
Usage	`addPort( 'signalName', 'signalWidth', 'signalDirection' )`
Arguments	
'signalName'	**Description:** Name of the signal. **Value:** Any valid SV signal name.
'signalWidth'	**Description:** Width of the signal. **Value:** Any valid SV signal width value. This value can be one of the parameters defined using addParamDef API.
'signalDirection'	**Description:** The direction of the signal from the perspective of the test bench as an initiator. **Value:** input|output|inout.

```
interface  mem_if       #(
       int DATA_WIDTH = 220,
       int ADDR_WIDTH = 210
       )
 (
  input tri clock,
  input tri reset
  ,inout tri  cs
  ,inout tri  rwn
  ,inout tri  rdy
  ,inout tri [ADDR_WIDTH-1:0] addr
  ,inout tri [DATA_WIDTH-1:0] wdata
  ,inout tri [DATA_WIDTH-1:0] rdata
);

endinterface
```

Transaction Variables

Name	addTransVar
Description	This API adds a variable to the transaction class. This variable is automatically added to the convert2string function. A coverpoint for this variable is added to the generated coverage component. This variable is automatically added to transaction viewing in the waveform viewer.

Usage	`addTransVar(`
	`'variableName',`
	`'variableType',`
	`isrand=True|False,`
	`isCompare=True|False`
	`)`

Arguments	
'variableName'	**Description:** Name of the variable. **Value:** Any valid SV variable name.
'variableType'	**Description:** Type of the variable. **Value:** Any valid SV variable type.
isrand	**Description:** Determines if the variable is randomizable. **Value:** True|False
isCompare	**Description:** Determines if the variable is included in the do_compare function. **Value:** True|False

```
class mem_transaction       #(
      int DATA_WIDTH = 220,
      int ADDR_WIDTH = 210
      ) extends uvmf_transaction_base;

  `uvm_object_param_utils( mem_transact
                           DATA_WIDTH,
                           ADDR_WIDTH
                            ))

  bit [DATA_WIDTH-1:0] read_data;
  bit [DATA_WIDTH-1:0] write_data;
  rand bit [ADDR_WIDTH-1:0] address;
  rand bit [3:0] byte_enable;
  int chksum;
```

Transaction Constraints

Name	addTransVarConstraint
Description	This API adds a constraint to the transaction class.

Usage	`addTransVarConstraint(`
	`'constraintName',`
	`'constraintBody'`
	`)`

Arguments	
`'constraintName'`	**Description:** Name of the constraint. **Value:** Any valid SV constraint name.
`'constraintBody'`	**Description:** Body of the constraint. **Value:** Any valid SV constraint body.

```
//Constraints for the transaction variables:
  constraint address_word_align_c { address[1:0]==0; }
```

Example Interface Generator File

```
1  #! /usr/bin/env python
2  import uvmf_gen
3  ## The input to this call is the name of the desired interface
4  intf = uvmf_gen.InterfaceClass('mem')
5  ## Specify parameters for this interface package.
6  intf.addParamDef('DATA_WIDTH','int','220')
7  ## Specify the clock and reset signal for the interface
8  intf.clock = 'clock'
9  intf.reset = 'reset'
10 intf.resetAssertionLevel = True
11 ## Specify the ports associated with this interface.
12 intf.addPort('cs',1,'output')
13 intf.addPort('rwn',1,'output')
14 ...
15 ## Specify typedef for inclusion in typedefs_hdl file
16 intf.addHdlTypedef('my_byte_t','byte')
17 ## Specify transaction variables for the interface.
18 intf.addTransVar('read_data','bit [DATA_WIDTH-1:0]',israND=False,iscompare=True)
19 ## Specify transaction variable constraint
20 intf.addTransVarConstraint('address_word_align_c','{ address[1:0]==0; }')
21 ...
```

Files Generated

./mem_pkg:

- Makefile
- mem_filelist_hdl.f
- mem_filelist_hvl.f
- mem_pkg.sv
- mem_pkg_hdl.sv
- src

./mem_pkg/src:

- mem2reg_adapter.svh
- mem_agent.svh
- mem_configuration.svh
- mem_driver.svh
- mem_driver_bfm.sv
- mem_if.sv
- mem_monitor.svh
- mem_monitor_bfm.sv
- mem_random_sequence.svh
- mem_responder_sequence.svh
- mem_sequence_base.svh
- mem_transaction.svh
- mem_transaction_coverage.svh
- mem_typedefs.svh
- mem_typedefs_hdl.svh

Implement Driver Signaling

```
// ****************************************************************************
  task do_transfer(                  input bit [DATA_WIDTH-1:0] read_data,
                   input bit [DATA_WIDTH-1:0] write_data,
                   input bit [ADDR_WIDTH-1:0] address,
                   input bit [3:0] byte_enable,
                   input int chksum                 );
  // UVMF_CHANGE_ME : Implement protocol signaling.
  // Reference code;
  //    while (control_signal == 1'b1) @(posedge clock_i);
  //    INITIATOR mode input signals
  //    rdy_i;         //
  //    rdy_o <= xyz; //
  //    INITIATOR mode output signals
  //    addr_i;         //   [ADDR_WIDTH-1:0]
  //    addr_o <= xyz; //   [ADDR_WIDTH-1:0]
  //    Bi-directional signals

  @(posedge clock_i);
  @(posedge clock_i);
  @(posedge clock_i);
  @(posedge clock_i);
  @(posedge clock_i);
endtask
```

Replace with protocol signaling

Implement Signal Monitoring

```
// ****************************************************************************
    task do_monitor(
                    output bit [DATA_WIDTH-1:0] read_data,
                    output bit [DATA_WIDTH-1:0] write_data,
                    output bit [ADDR_WIDTH-1:0] address,
                    output bit [3:0] byte_enable,
                    output int chksum                    );
// UVMF_CHANGE_ME : Implement protocol monitoring.
// Reference code;
//    while (control_signal == 1'b1) @(posedge clock_i);
//    xyz = cs_i;  //
//    xyz = rwn_i;  //
//    xyz = rdy_i;  //
//    xyz = addr_i;  //    [ADDR_WIDTH-1:0]
//    xyz = wdata_i;  //    [DATA_WIDTH-1:0]
//    xyz = rdata_i;  //    [DATA_WIDTH-1:0]

    @(posedge clock_i);
    @(posedge clock_i);
    @(posedge clock_i);
    @(posedge clock_i);

    endtask
```

Replace with protocol monitoring

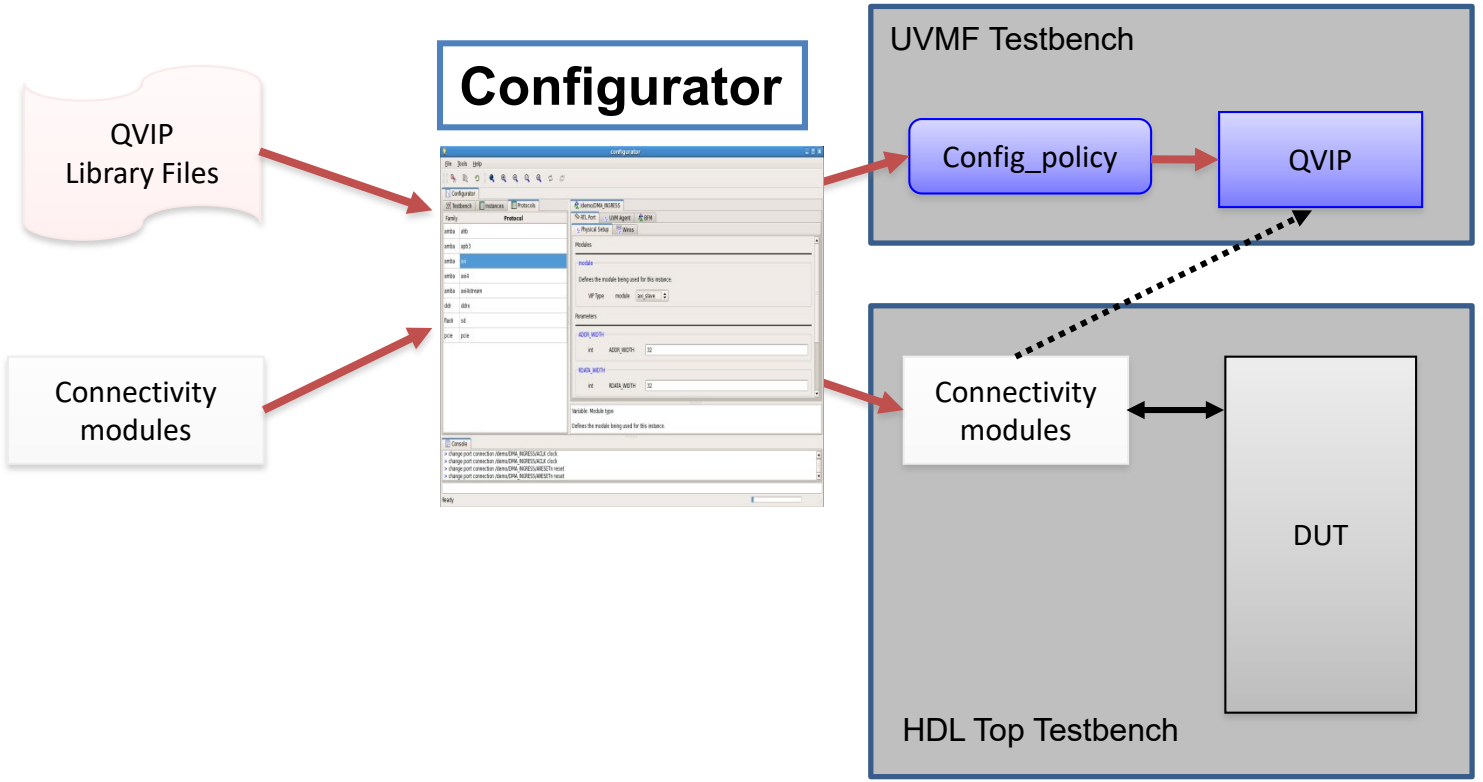
STANDARD INTERFACES

Standard Interfaces

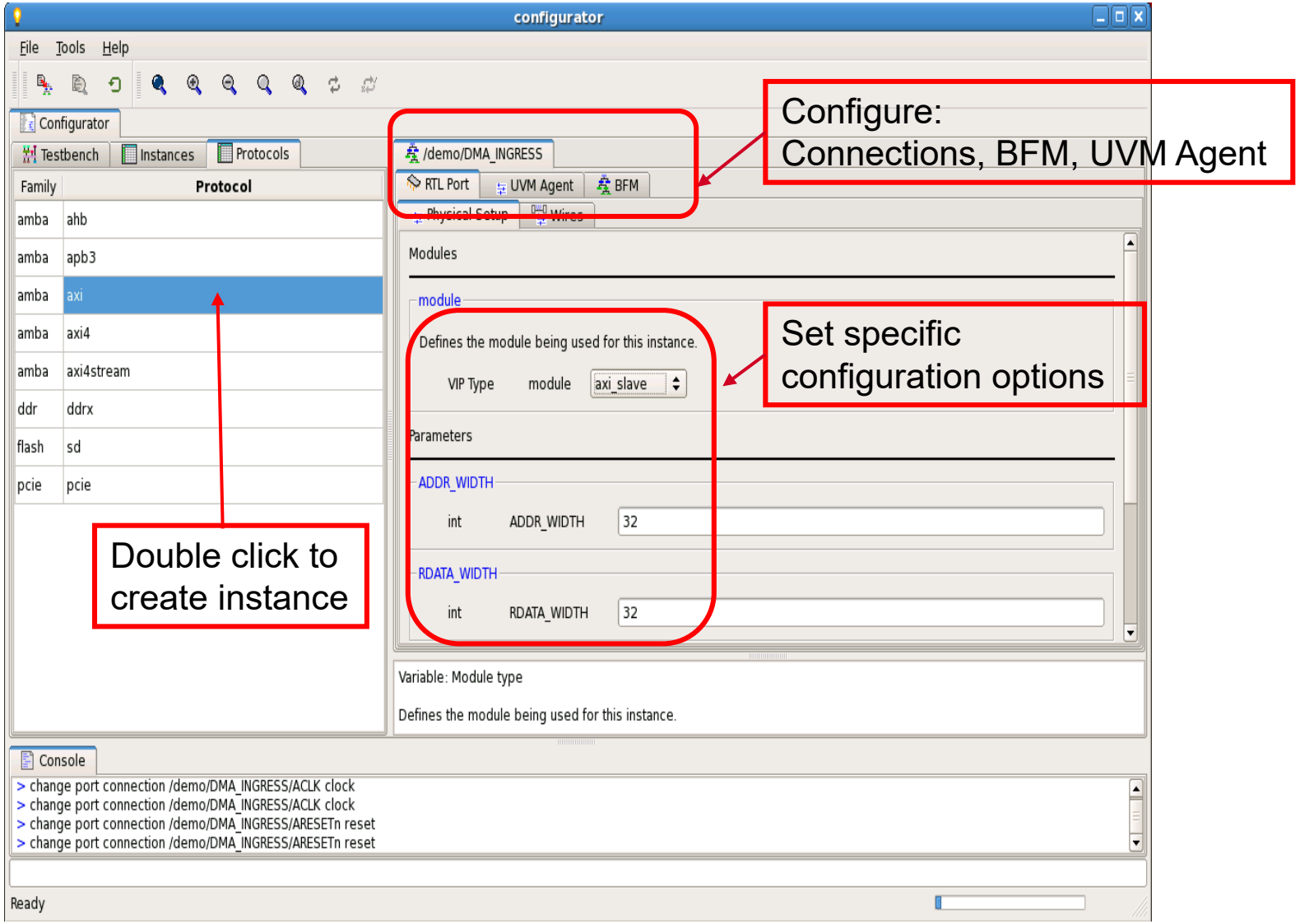
Standard Interface packages provided By
Questa VIP and Instantiated using Configurator

Questa VIP Configurator

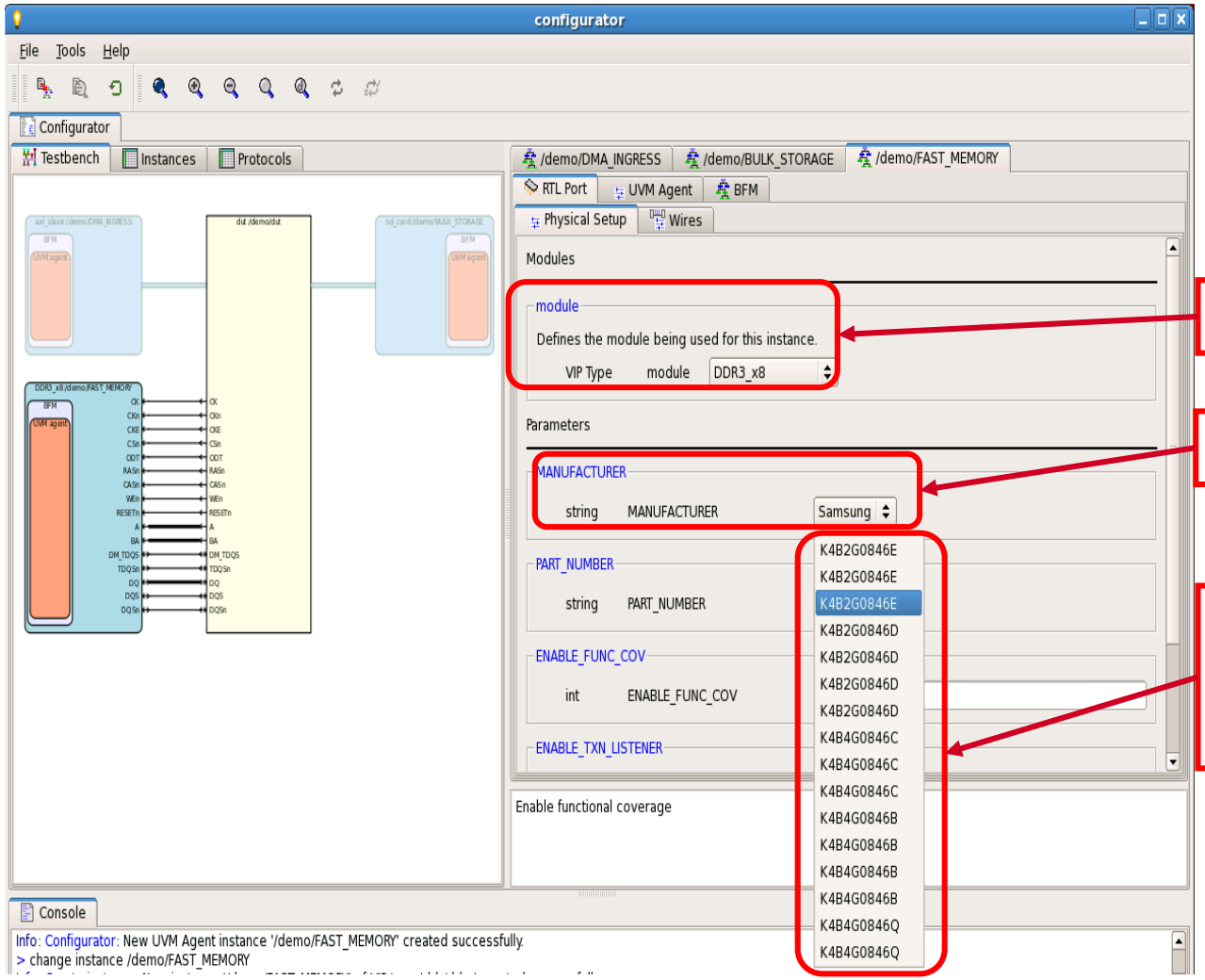
- Configuration GUI for protocols and memory models
- Configure multiple QVIPs in one session
- Generate a complete UVMF testbench



Select and Configure Protocols



Select Memory Models



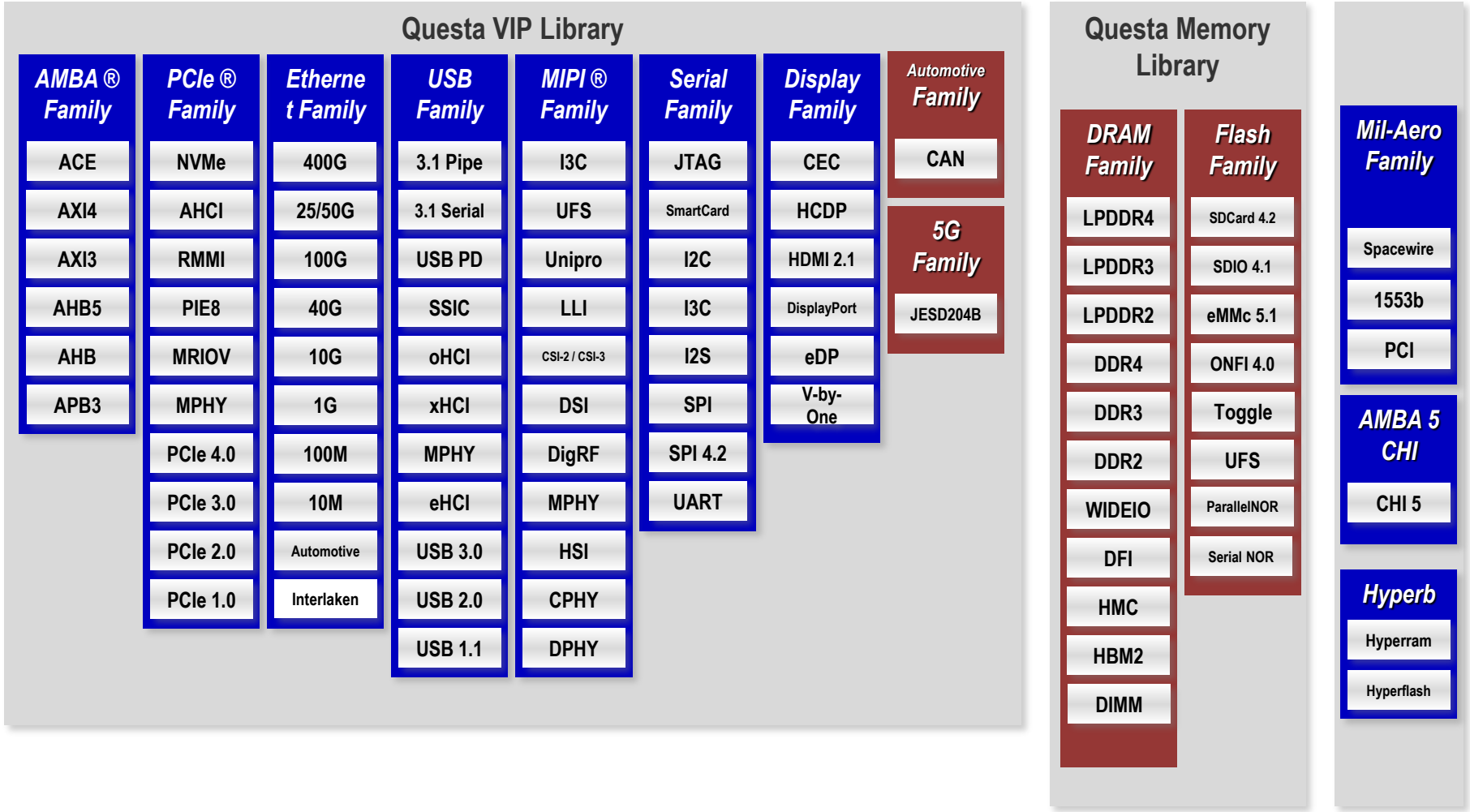
Select memory type

Select Manufacturer

Select part number from list sorted by manufacturer

Questa Verification IP

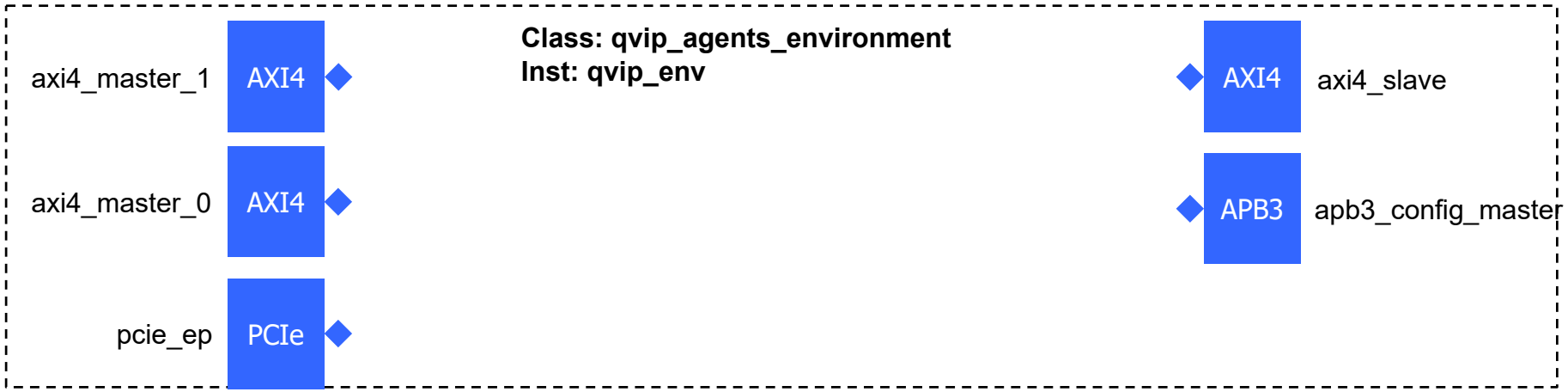
A Complete Library for ASIC and FPGA Designs



accellera SYSTEMS INITIATIVE

2017 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

Create Via QVIP Configurator



Files Generated

qvip_agents_dir/
- config_policies
- uvm_tb
- uvmf

qvip_agents_dir/config_policies:
- apb3_config_master_config_policy.svh
- axi4_master_0_config_policy.svh
- axi4_master_1_config_policy.svh
- axi4_slave_config_policy.svh
- pcie_ep_config_policy.svh
- qvip_agents_params_pkg.sv

qvip_agents_dir/uvmf:
- Makefile
- default_clk_gen.sv
- default_reset_gen.sv
- hdl_qvip_agents.sv
- hvl_qvip_agents.sv
- questa_run
- qvip_agents_env_configuration.svh
- qvip_agents_environment.svh
- qvip_agents_filelist.f
- qvip_agents_pkg.sv
- qvip_agents_test_base.svh
- qvip_agents_vseq_base.svh
- test_packages.svh

API's Generated For UVMF Use

```
1  //
2  // File: qvip_agents_pkg.sv
3  //
4  // Generated from Mentor VIP Configurator (20161207)
5  // Generated using Mentor VIP Library ( 10_5c : 12/12/2016:13:30 )
6  //
7  // ## The following code is used to add this qvip_configurator generated output into an
8  // ## encapsulating UVMF Generated environment.  The addQvipSubEnv function is added to
9  // ## the python configuration file used by the UVMF environment generator.
10 // env.addQvipSubEnv('sub_env_instance_name', 'qvip_agents', ['pcie_ep', 'axi4_master_0',
11 //
12 // ## The following code is used to add this qvip_configurator generated output into an
13 // ## encapsulating UVMF Generated test bench.  The addQvipBfm function is added to
14 // ## the python configuration file used by the UVMF bench generator.
15 // ben.addQvipBfm('pcie_ep', 'qvip_agents', 'ACTIVE')
16 // ben.addQvipBfm('axi4_master_0', 'qvip_agents', 'ACTIVE')
17 // ben.addQvipBfm('axi4_master_1', 'qvip_agents', 'ACTIVE')
18 // ben.addQvipBfm('axi4_slave', 'qvip_agents', 'ACTIVE')
19 // ben.addQvipBfm('apb3_config_master', 'qvip_agents', 'ACTIVE')
20 package qvip_agents_pkg;
21     import uvm_pkg::*;
22
23     `include "uvm_macros.svh"
24
25     import addr_map_pkg::*;
```

ENVIRONMENT

What Are We Creating?

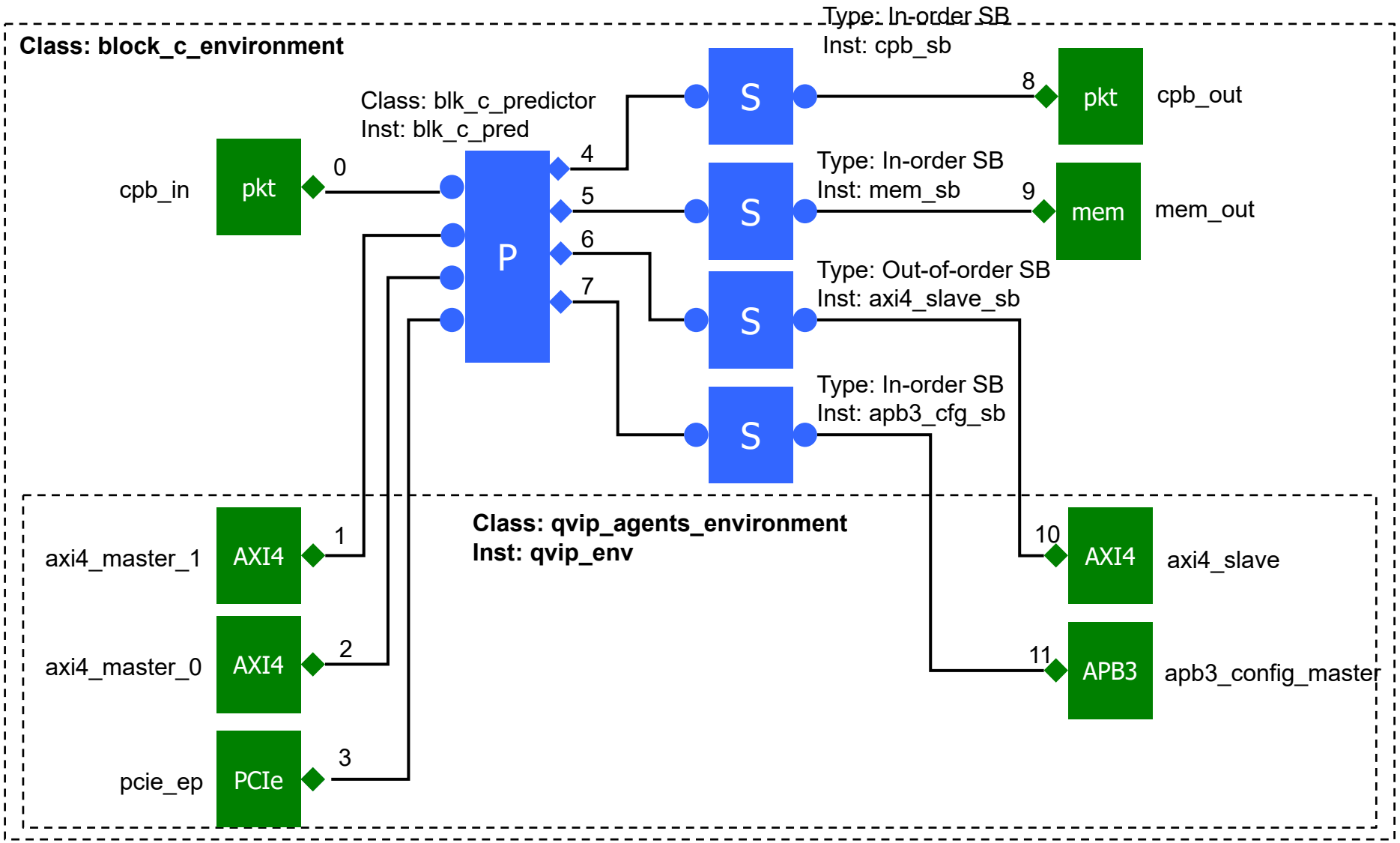
- UVM based environment package
  - Classes: Environment, environment configuration, environment sequence base, predictors, coverage
  - Package declaration
- Compile files
  - Compilation file list, makefile
- Fully constructed and connected environment

Creating The Environment

- Library of Python API's and variables provided by UVMF
  - Used to characterize environment
    - Name, parameters, agents, predictors, coverage components, scoreboards, sub-environments
- What's left to complete?
  - Write prediction model
    - SystemVerilog, C, SystemC, …
  - Add reusable sequences unique to DUT

Start Creating Tests ASAP!

Block_c Environment



Environment Name/Params

- Define the environment name

```
env = uvmf_gen.EnvironmentClass('block_c')
```

```
class block_c_environment
extends uvmf_environment_base #(.CONFIG_T( block_c_env_configuration
                                ));

  `uvm_component_utils( block_c_environment );
```

- Define environment parameters

Name	addParamDef
Description	This API adds a parameter to the environment classes.

Usage	`addParamDef(`
	`'parameterName',`
	`'parameterType',`
	`'parameterDefalutValue'`
	`)`

Arguments	
`'parameterName'`	**Description:** Name of the parameter. **Value:** Any valid SV parameter name.
`'parameterType'`	**Description:** Type of the parameter. **Value:** Any valid SV parameter type.
`'parameterDefalutValue'`	**Description:** The default value for this parameter. **Value:** Any valid SV value for the parameter type.

Instantiate Custom Agents

Name	addAgent
Description	This API adds an agent to the environment.
Usage	addAgent('*instanceName*', '*protocol*', '*clock*', '*reset*', {'*protocolParam1*':'*protocolParam1Override*', '*protocolParam2*':'*protocolParam2Override*'})
Arguments	
'*instanceName*'	**Description:** Instance name of this agent. **Value:** Any valid SV instance name.
'*protocol*'	**Description:** Name of the protocol package. Example: mem_pkg would use the value mem **Value:** Any valid SV package name.
'*clock*'	**Description:** The name of the primary clock for this protocol. **Value:** Any valid SV signal name.
'*reset*'	**Description:** The name of the primary reset for this protocol. **Value:** Any valid SV signal name.
'*protocolParamN*'	**Description:** Name of the interface package parameter to be overridden. **Value:** Must match the interface parameter name to be overridden.
protocolParamNOverride'	**Description:** The value used to override the parameters default value. **Value:** Any valid SV value for overriding the identified parameter. Example: '5' or '*testLevelParameterName*'

```
class block_c_environment
extends uvmf_environment_base #(.CONF
                                ));

  `uvm_component_utils( block_c_envir

  typedef mem_agent mem_in_agent_t;
  mem_in_agent_t mem_in;

  typedef mem_agent mem_out_agent_t;
  mem_out_agent_t mem_out;

  typedef pkt_agent pkt_out_agent_t;
  pkt_out_agent_t pkt_out;
```

accellera SYSTEMS INITIATIVE

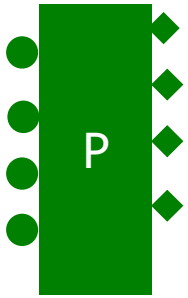
2017 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

Define Analysis Component

Name	defineAnalysisComponent
Description	This API is used to define an analysis component class. This can be used to create predictors, coverage components, etc. It creates a component that is an extension of uvm_component. Any number of analysis_exports and analysis_ports can be added to this component. This API creates the class definition and adds the class to the environment package.

Usage	defineAnalysisComponent(
	keyword,
	'*className*',
	{
	'*analysisExport1Name*':'*transactionType1*',
	'*analysisExport2Name*':'*transactionType2*'
	},
	{
	'*analysisPort1Name*':'*transactionType3*',
	'*analysisPort2Name*':'*transactionType4*'
	}
	)

Class: blk_c_predictor




```
class block_c_predictor
extends uvm_component;

  // Factory registration of this cla
  `uvm_component_utils( block_c_predi

  // Instantiate a handle to the conf
  block_c_env_configuration  configur

  // Instantiate the analysis exports
  uvm_analysis_imp_mem_in_ae #(mem_tr
  uvm_analysis_imp_axi4_master_1_ae #
  uvm_analysis_imp_axi4_master_0_ae #

  // Instantiate the analysis ports
  uvm_analysis_port #(pkt_transaction
  uvm_analysis_port #(mvc_sequence_it
  uvm_analysis_port #(mvc_sequence_it
  uvm_analysis_port #(mem_transaction
```

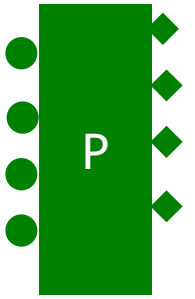
Instantiate Analysis Comps

Name	addAnalysisComponent
Description	This API instantiates an analysis component.

Usage	`addAnalysisComponent(`
	`'instanceName',`
	`'classType'`
	`)`

Arguments	
`'instanceName'`	**Description:** Instance name for the component. **Value:** Any valid SV instance name.
`'classType'`	**Description:** Class name of the component to be instantiated. **Value:** The class name of the component to be instantiated.

Class: blk_c_predictor
Inst: blk_c_pred




```
class block_c_environment
extends uvmf_environment_base #(.CONFIG_T( block_c_env_configuration
                                ));

  `uvm_component_utils( block_c_environment );


typedef block_c_predictor  blk_c_pred_t;
blk_c_pred_t blk_c_pred;
```

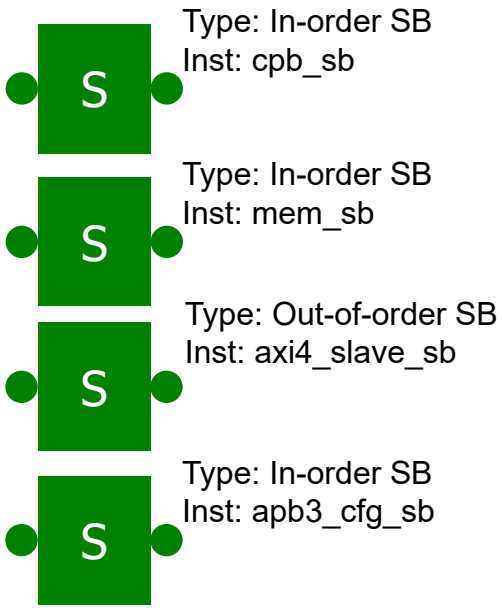
Instantiate Scoreboards

Name	addUvmfScoreboard
Description	This API instantiates a UVMF scoreboard

Usage	`addUvmfScoreboard(`
	`'instanceName',`
	`'scoareboardType',`
	`'transactionType'`
	`)`

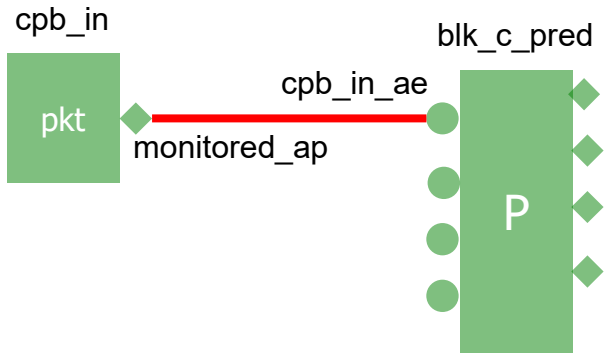
Arguments	
`'instanceName'`	**Description:** Instance name given to the scoreboard. **Value:** Any valid SV instance name.
`'scoareboardType'`	**Description:** Type of scoreboard to be instantiated. **Value:** Any of the UVMF scoreboard types: uvmf_in_order_scoreboard, uvmf_out_of_order_scoreboard, uvmf_in_order_scoreboard_array, uvmf_in_order_race_scoreboard. Custom scoreboards extended from uvmf_scoreboard_base can also be used.
`'transactionType'`	**Description:** Type of the transaction to be received by the scoreboard. The type parameterization must match that of the component connected to this scoreboard. **Value:** Transaction type matching component connecting to this scoreboard.

```
typedef uvmf_in_order_scoreboard #(mem_transaction)  mem_sb_t;
mem_sb_t mem_sb;
```



Connecting Custom Comps

Name	addConnection
Description	This API connects two UVM components
Usage	`addConnection( 'componentAInstanceName', 'componentAConnectionPoint', 'componentBInstanceName', 'componentBConnectionPoint' )`
Arguments	
`'componentAInstanceName'`	**Description:** Instance name of the component to be connected. **Value:** Instance name of the component to be connected.
`'componentAConnectionPoint'`	**Description:** Connection point of component to be connected. **Value:** Any valid UVM connection port or export.
`'componentBInstanceName'`	**Description:** Instance name of the component to be connected. **Value:** Instance name of the component to be connected.
`'componentBConnectionPoint'`	**Description:** Connection point of component to be connected. **Value:** Any valid UVM connection port or export.




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

 mem_in.monitored_ap.connect(blk_c_pred.mem_in_ae);
 blk_c_pred.mem_sb_ap.connect(mem_sb.expected_analysis_export);
 blk_c_pred.pkt_sb_ap.connect(pkt_sb.expected_analysis_export);
```

Instantiating Sub-Envs

Name	addSubEnv
Description	This API adds a UVMF environment to this environment.

Usage	`addSubEnv(`
	`'subEnvironmentInstanceName',`
	`'subEnvironmentPackageName',`
	`numberOfAgents,`
	`{`
	`'subEnvironmentParameter1':'parameter1Override',`
	`'subEnvironmentParameter2':'parameter2Override'`
	`})`

Arguments	
`'subEnvironmentInstanceName'`	**Description:** Instance name for this environment. **Value:** Any valid SV instance name.
`'subEnvironmentPackageName'`	**Description:** Package name for this environment. **Value:** The name of the environment package that contains this environment. Example: for package named block_b_env_pkg use the value block_b
`numberOfAgents`	**Description:** Number of agents in this environment and any environments encapsulated by this environment. **Value:** Total number of agents.
`'subEnvironmentParameterN'`	**Description:** Parameter of this sub environment. **Value:** The parameter of this environment class.
`'parameterNOverride'`	**Description:** The value to override the specified environment parameter. **Value:** Any valid SV value for overridding the parameter.

```
typedef block_b_environment #(.CP_IN_DATA_WIDTH(CHIP_CP_IN_DATA_WIDTH),.CP_OUT_ADDR_WIDTH(CHIP_
block_b_env_t block_b_env;
```

Example Environment Generator File

```
1  #! /usr/bin/env python
2  import uvmf_gen
3  env = uvmf_gen.EnvironmentClass('block_c')
4  ##  The addQvipSubEnv() line below was copied from the comments in the QVIP c
5  env.addQvipSubEnv('qvip_env', 'qvip_agents', ['pcie_ep', 'axi4_master_0', 'ax
6  ## Specify the agents contained in this environment
7  env.addAgent('mem_in',  'mem', 'clock', 'reset')
8  env.addAgent('mem_out', 'mem', 'clock', 'reset')
9  env.addAgent('pkt_out', 'pkt', 'pclk', 'prst')
10 ## Define the predictors contained in this environment (not instantiate, yet,
11 env.defineAnalysisComponent('predictor','block_c_predictor',
12         {'mem_in_ae':'mem_transaction #()',
13         'axi4_master_0_ae':'mvc_sequence_item_base',
14         'axi4_master_1_ae':'mvc_sequence_item_base'},
15         {'mem_sb_ap':'mem_transaction #()',
16         'pkt_sb_ap':'pkt_transaction #()',
17         'axi4_slave_ap':'mvc_sequence_item_base',
18         'apb3_config_master_ap':'mvc_sequence_item_base'})
19 ## Instantiate the components in this environment
20 ## addAnalysisComponent(<name>,<type>)
21 env.addAnalysisComponent('blk_c_pred','block_c_predictor')
22 ...
```

Files Generated

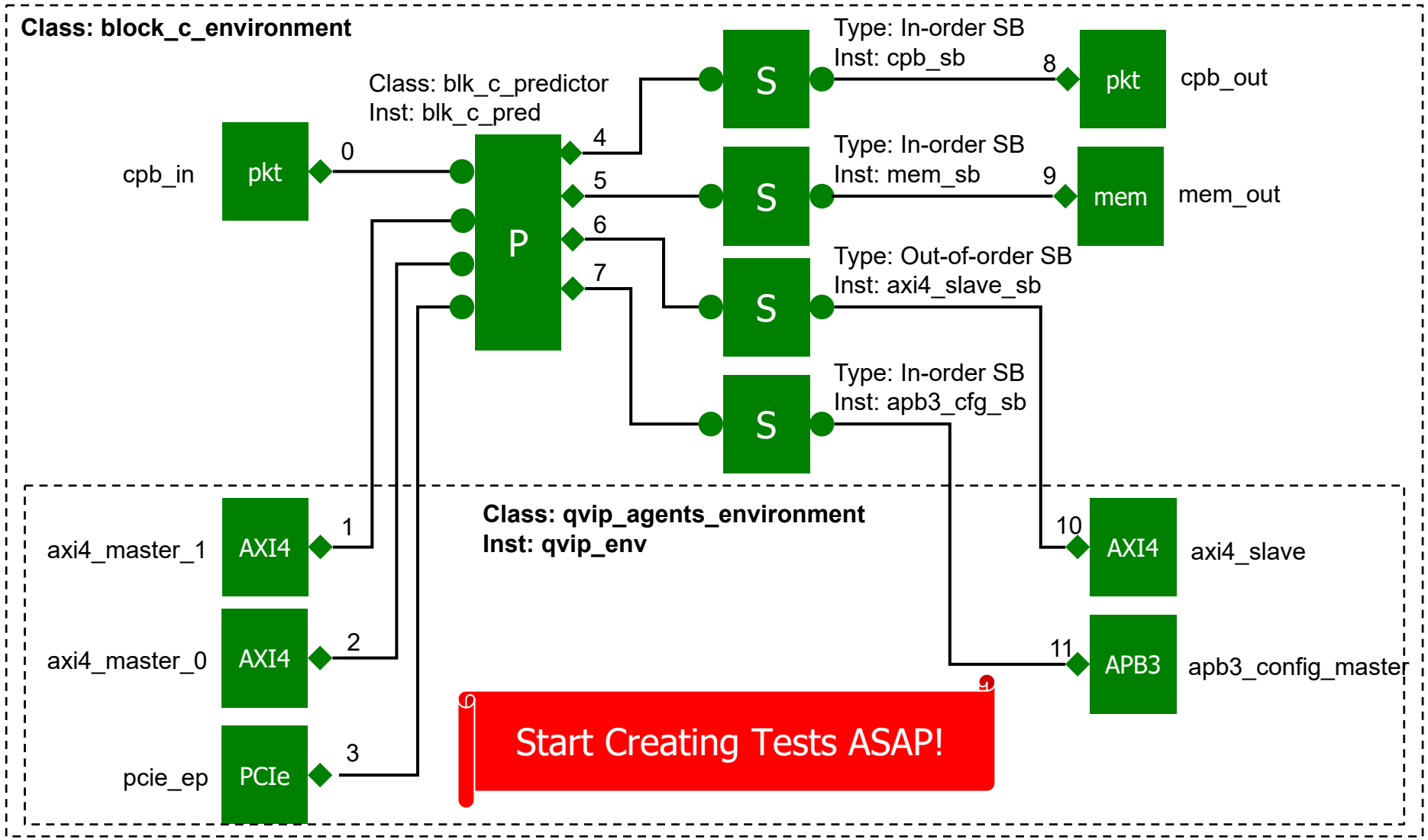
./block_c_env_pkg:

- Makefile
- block_c_env_pkg.sv
- src

./block_c_env_pkg/src:

- block_c_env_configuration.svh
- block_c_env_sequence_base.svh
- block_c_environment.svh
- block_c_infact_env_sequence.svh
- block_c_predictor.svh

Environment Created



TEST BENCH

What Are We Creating?

- UVM based test packages
  - Top level UVM test package
  - Top level UVM sequence package
  - Top level parameters package
- Top level modules
- Compile files
  - Compilation file lists, makefile
- Operational simulation
  - How operational? What does it do

Creating The Bench

- Library of Python API's and variables provided by UVMF
  - Used to characterize test bench
    - Name, clock, reset, parameters, top environment, BFM's
- What's left to complete?
  - Instantiate DUT
  - Add test sequences

Start Creating Tests ASAP!

Test Bench Name/Params

- Define the test bench name and environment used

```
ben=uvmf_gen.BenchClass('block_c','block_c',{})
```

```
class test_top extends uvmf_test_base #(.CONFIG_T(block_c_env_configuration_t),
                                        .ENV_T(block_c_environment_t),
                                        .TOP_LEVEL_SEQ_T(block_c_bench_sequence_base));

  `uvm_component_utils( test_top );
```

- Define bench parameters

Name	addParamDef
Description	This API adds a parameter to the test level parameter package.
Usage	`addParamDef( 'parameterName', 'parameterType', 'parameterDefalutValue' )`
Arguments	
`'parameterName'`	**Description:** Name of the parameter. **Value:** Any valid SV parameter name.
`'parameterType'`	**Description:** Type of the parameter. **Value:** Any valid SV parameter type.
`'parameterDefalutValue'`	**Description:** The default value for this parameter. **Value:** Any valid SV value for the parameter type.

Defining Clock and Reset

Name	Value(s)	Description
clockHalfPeriod	'timeValue'	Time duration of half period. Example: '6ns', or '6'
clockPhaseOffset	'timeValue'	Time duration before first clock edge. Exaple: '25ns' or '25'
resetAssertionLevel	True|False	Assertion level of reset signal driven by test bench.
resetDuration	'timeValue'	Time duration reset is asserted at start of simulation. Example: '100ns', or '100'

```
module hdl_top;
// pragma attribute hdl_top par

bit rst = 0;
bit clk;
    // Instantiate a clk driver
    // tbx clkgen
    initial begin
        #9ns;
        clk = ~clk;
        forever #5ns clk = ~clk;
    end
    // Instantiate a rst driver
    initial begin
        #200ns;
        rst <= ~rst;
    end
```

Instantiating Custom BFM's

Name	addBfm
Description	This API adds a monitor BFM to the top level module, hdl_top. If the *activity* argument is ACTIVE then a driver BFM is also added to hdl_top. The test bench clock and reset are connected to these BFM's. If the *activity* argument is ACTIVE then the top level virtual sequence will automatically start a random sequence on this BFM.

Usage	`addBfm(`
	`'instanceName',`
	`'protocol',`
	`'clock',`
	`'reset',`
	`'activity',`
	`{'protocolParam1':'protocolParam1Override',`
	`'protocolParam2':'protocolParam2Override'}`
	`)`

Arguments	
`'instanceName'`	**Description:** Instance name of this BFM. **Value:** Any valid SV instance name.
`'protocol'`	**Description:** Name of the protocol package. Example: mem_pkg would use the value mem **Value:** Any valid SV package name.
`'clock'`	**Description:** The name of the primary clock given in the user input file for this protocol. **Value:** Must match value given to clock variable in user input file for this protocol.

```
mem_if  mem_in_bus(.clock(clk), .rese
mem_if  mem_out_bus(.clock(clk), .res
pkt_if  pkt_out_bus(.pclk(clk), .prst

mem_monitor_bfm  mem_in_mon_bfm(mem_i
mem_monitor_bfm  mem_out_mon_bfm(mem_
pkt_monitor_bfm  pkt_out_mon_bfm(pkt_

mem_driver_bfm  mem_in_drv_bfm(mem_in
mem_driver_bfm  mem_out_drv_bfm(mem_o
pkt_driver_bfm  pkt_out_drv_bfm(pkt_o
```

Instantiating QVIP BFM's

Name	addQvipBfm
Description	Add a QVIP interface BFM. This API is generated by the QVIP Configurator. It is located in the package as SV comments.

Usage	`addQvipBfm(`
	`'instanceName',`
	`'envPackage',`
	`'activity'`
	`)`

Arguments	
`'instanceName'`	**Description:** Instance name of this BFM. **Value:** Name given to instance in QVIP Configurator.
`'envPackage'`	**Description:** Name of the environment package generated using the QVIP Configurator. **Value:** Value given test bench in QVIP Configurator.
`'activity'`	**Description:** Determines the ACTIVE/PASSIVE state of the BFM. **Value:** ACTIVE|PASSIVE

```
module hdl_top;
// pragma attribute hdl_top partition_module_xrtl

hdl_qvip_agents qvip_qvip_agents();
```

Example Test Bench Generator File

```
1  #! /usr/bin/env python
2  import uvmf_gen
3  ## The input to this call is the name of the desired bench and the name of the to
4  ## environment package
5  ##   BenchClass(<bench_name>,<env_name>)
6  ben = uvmf_gen.BenchClass('block_c','block_c',{})
7  ## Import QVIP protocol packages so that the test bench can use sequence items an
8  ben.addImport('mgc_apb3_v1_0_pkg')
9  ben.addImport('mgc_pcie_v2_0_pkg')
10 ben.addImport('mgc_axi4_v1_0_pkg')
11 ## The addQvipBfm() lines below were copied from comments in the QVIP Configurato
12 ben.addQvipBfm('pcie_ep', 'qvip_agents', 'ACTIVE')
13 ben.addQvipBfm('axi4_master_0', 'qvip_agents', 'ACTIVE')
14 ben.addQvipBfm('axi4_master_1', 'qvip_agents', 'ACTIVE')
15 ben.addQvipBfm('axi4_slave', 'qvip_agents', 'ACTIVE')
16 ben.addQvipBfm('apb3_config_master', 'qvip_agents', 'ACTIVE')
17 ## Specify the agents contained in this bench
18 ##   addBfm(<agent_handle_name>,<agent_type_name>,<clock_name>,<reset_name>,<acti
19 ben.addBfm('mem_in',  'mem', 'clock', 'reset', 'ACTIVE')
20 ben.addBfm('mem_out', 'mem', 'clock', 'reset', 'ACTIVE')
21 ben.addBfm('pkt_out', 'pkt', 'pclk', 'prst', 'ACTIVE')
22 ## This will prompt the creation of all bench files in their specified locations
23 ben.create()
```

accellera SYSTEMS INITIATIVE

2017 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

Files Generated

./block_c:
- docs
- registers
- rtl
- sim
- tb

./block_c/sim:
- Makefile
- default.rmdb
- run.do
- tbx.config
- testlist
- veloce.config
- velrunopts.ini
- wave.do

./block_c/tb:
- parameters
- sequences
- testbench
- tests

./block_c/tb/sequences:
- block_c_sequences_pkg.sv
- src/
  - block_c_bench_sequence_base.svh
  - example_derived_test_sequence.svh
  - infact_bench_sequence.svh

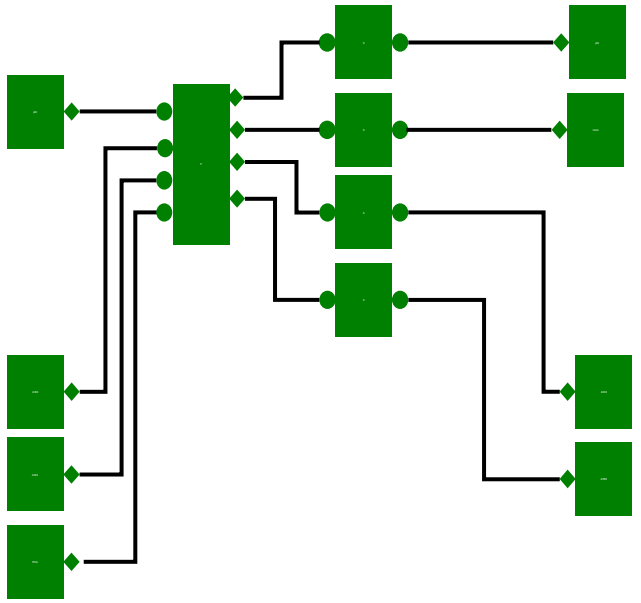
./block_c/tb/testbench:
- hdl_top.sv
- hvl_top.sv
- top_filelist_hdl.f
- top_filelist_hvl.f

./block_c/tb/tests:
- block_c_test_pkg.sv
- src/
  - example_derived_test.svh
  - test_top.svh

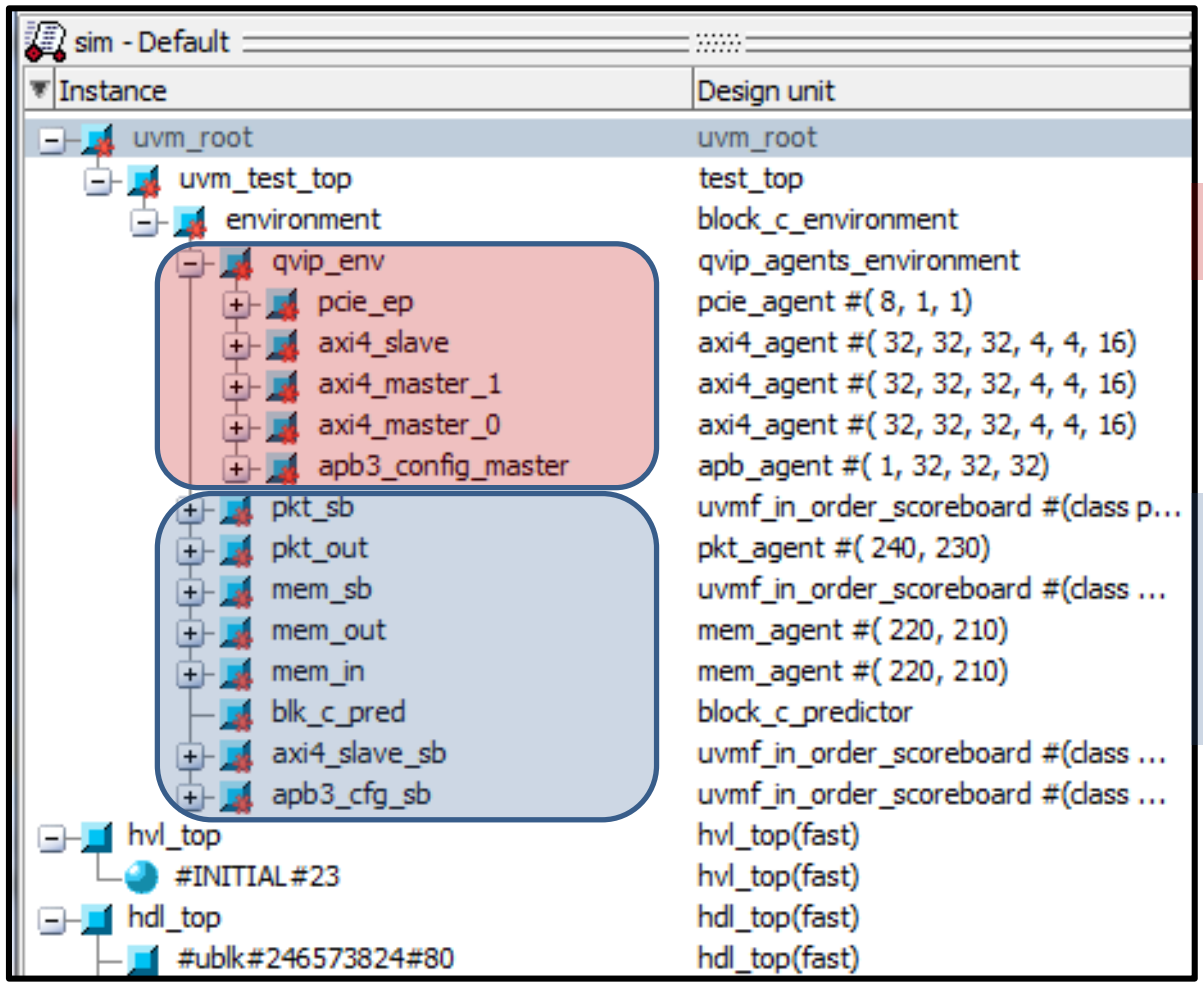
What We Built

- Built Block_c Simulation Environment
  - Interfaces
  - Environment
  - Test Bench

Start Creating Tests ASAP!



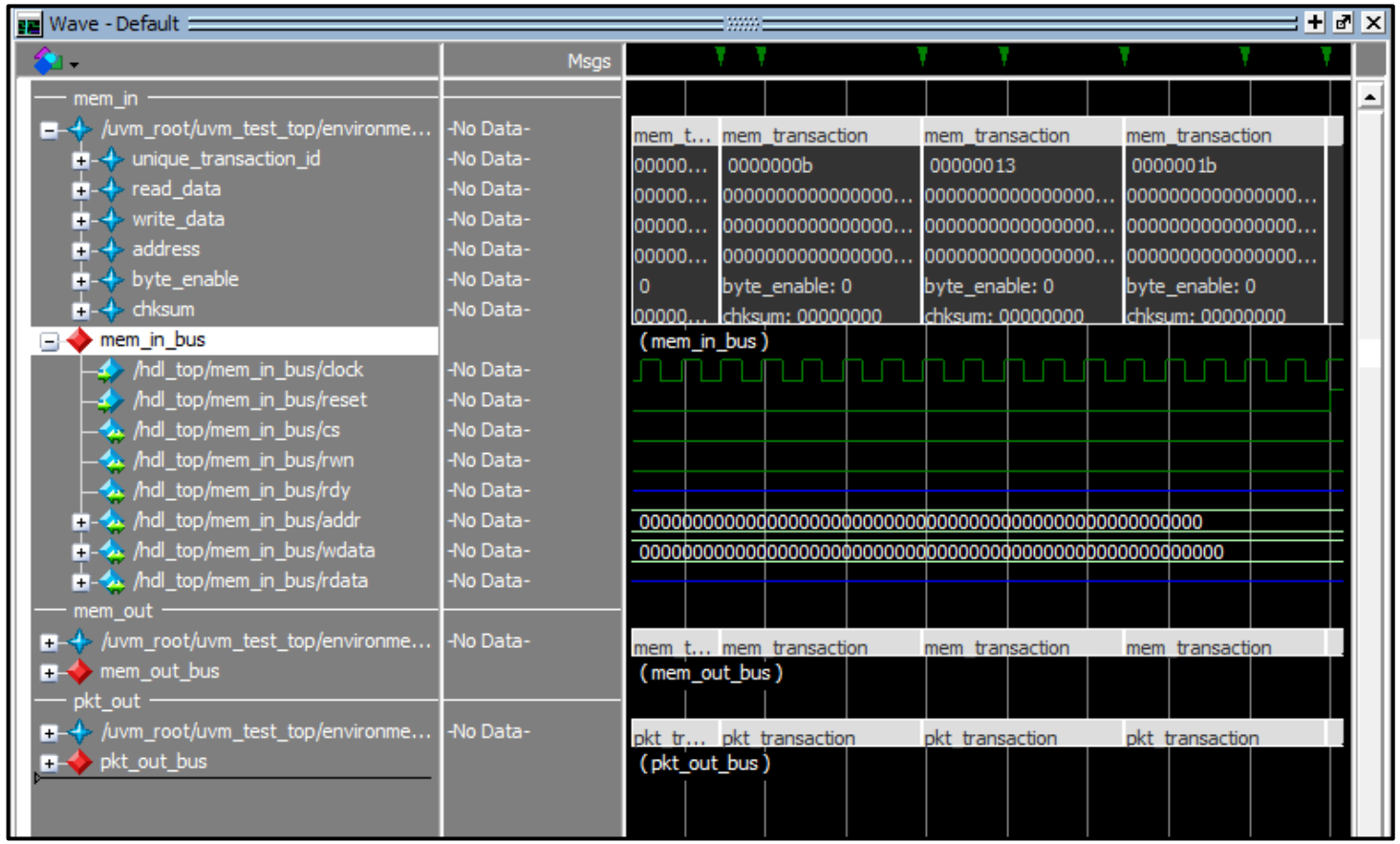
Generated Code In Simulation



QVIP Sub Environment

Custom Agents, scoreboards & predictors

Generated Code In Simulation



Transactions & signals from custom interface agents automatically added to waveforms

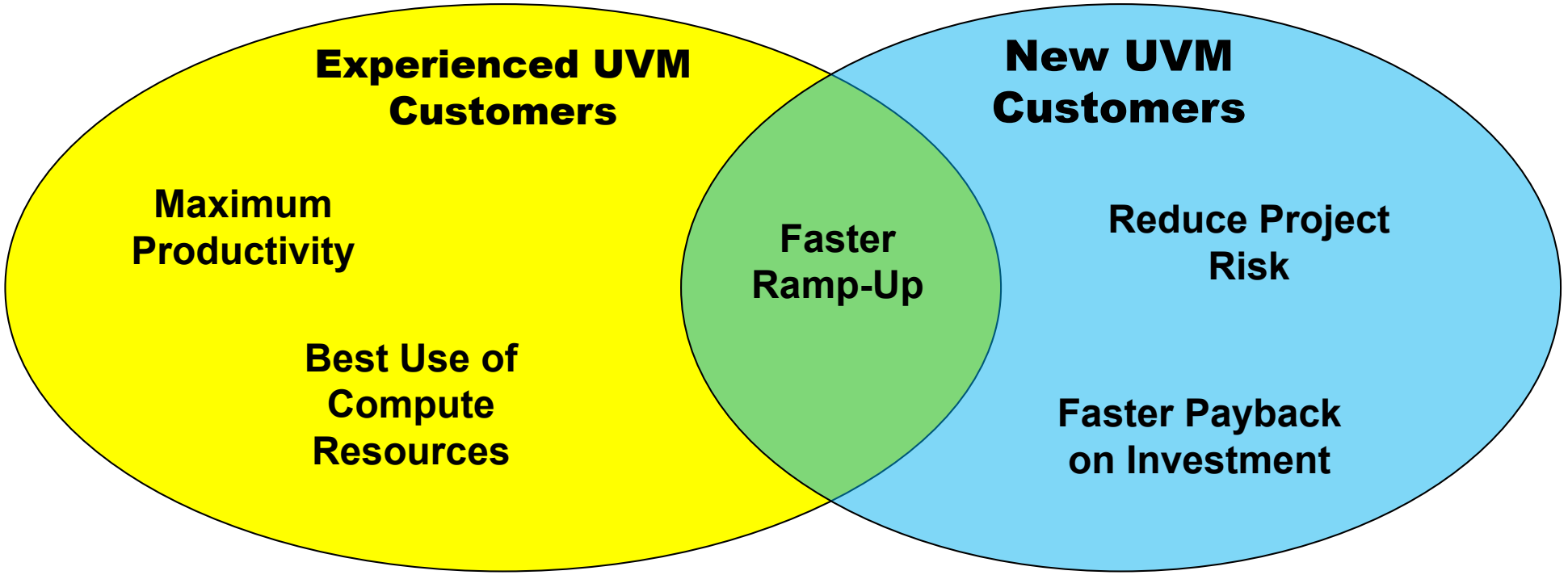
What We Saw

- Saves time
- Guaranteed reuse
- Guaranteed emulation readiness

Start Creating Tests ASAP!

SUMMARY

Testbench Automation
-: Key Benefits



Testbench Automation References & Links

White Paper:
http://go.mentor.com/4h1rr

Video tutorial:
https://www.youtube.com/watch?v=UStXG0qFL5Q
https://verificationacademy.com/seminars/uvm-forum/uvm-framework

UVMF User Guide, UVMF API Reference Manual
$QUESTA_INSTALL_DIR/examples/UVM_Framework/UVM_3.6f/docs

Completed Examples
$QUESTA_INSTALL_DIR/examples/UVM_Framework/UVM_3.6f/base_examples

EMAIL:
tom_fitzpatrick@mentor.com
graeme_jessiman@mentor.com

accellera SYSTEMS INITIATIVE

2017 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE