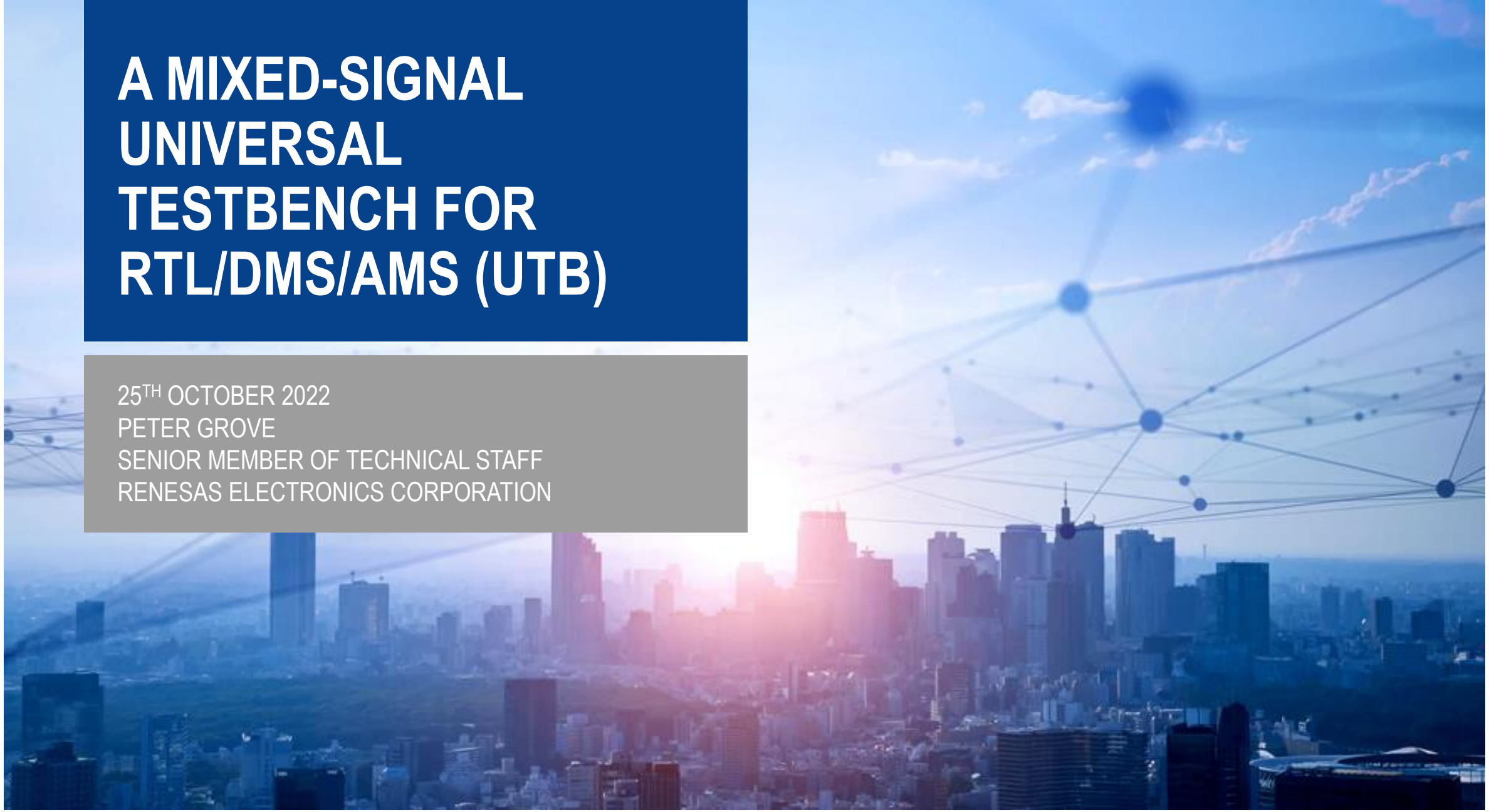
A MIXED-SIGNAL UNIVERSAL TESTBENCH FOR RTL/DMS/AMS (UTB)



25TH OCTOBER 2022
PETER GROVE
SENIOR MEMBER OF TECHNICAL STAFF
RENESAS ELECTRONICS CORPORATION

INTRODUCTION

Acronyms
- Digital Verification (DV)
- Digital-Mixed-Signal (DMS)
- Analog-Mixed-Signal (AMS)

Often what works in DV/DMS often causes issues in AMS

- Verification techniques need to be adjusted to align with multi-discipline usage.
  - Understanding where to create the stimulus and monitor values.
  - Understanding the implications of a verification procedure has in AMS.
  - How out-of-module references effect Connect Modules.
- Testbench structures need to adjustment to enable smooth dual discipline usage
- Todays focus is on a DMS/AMS testbench and automation, but the same setup works for RTL blocks.

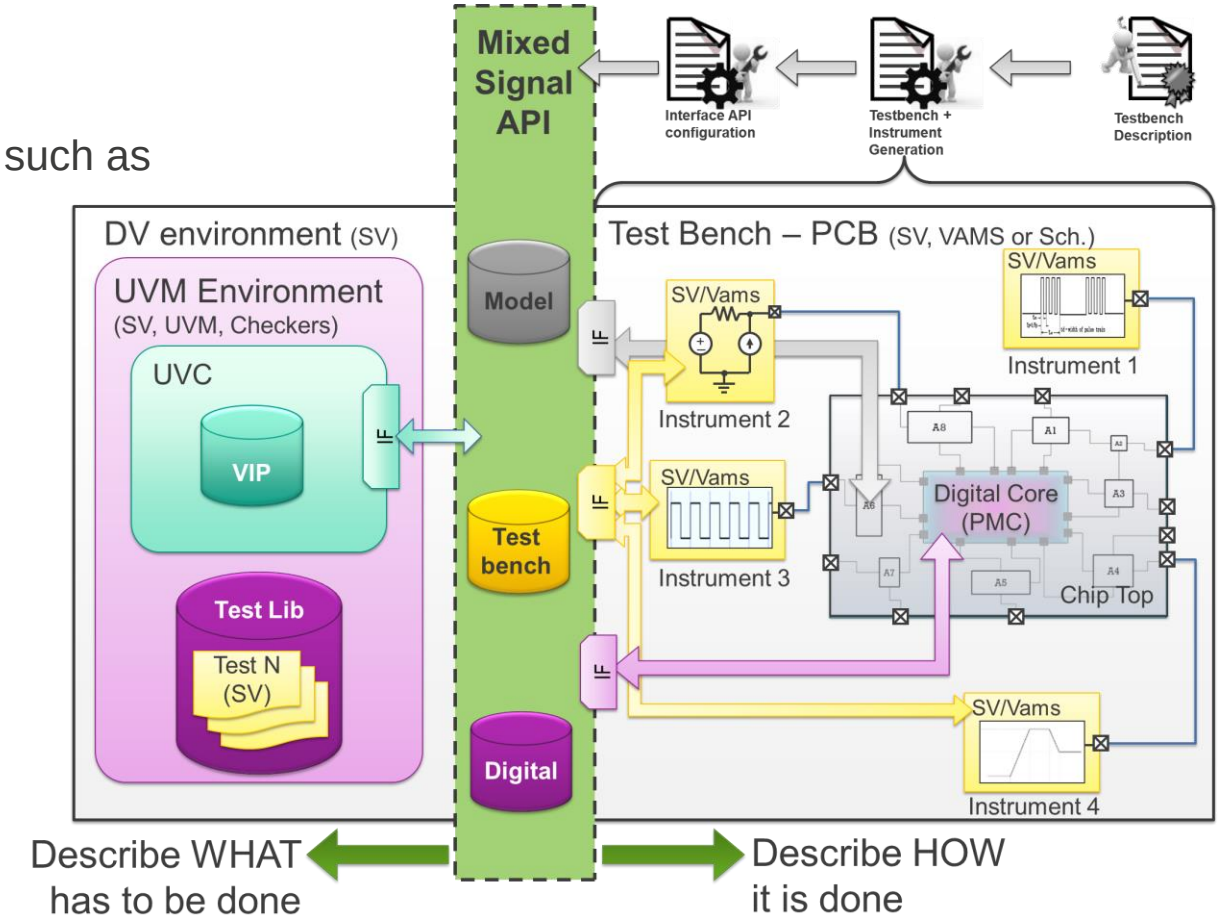
UNIFIED TESTBENCH (UTB)

- Why bother with a Unified TestBench
  - Allows shared resources both in terms of engineers and verification IP between DV, DMS and AMS.
  - Enables fast test case development with DMS modelling before running in AMS at schematic level.
    - Often DMS verification engineers can run the test in AMS without any assistance.
  - Avoids duplication of effort in maintaining numerous testbenches and styles for different disciplines.
  - Creates Verification engineers who are Mixed-Signal skilled. (They can then earn more!)

- Some key requirements to enable a Unified TestBench
  - Ensure the driver connection to any pin can change based on the abstraction of said pin, and easily.
    - E.g. Swap between Logic, real, UDN, electrical with minimal effort, in the physical module layer.
  - Block OOMR's to nodes where the coerced nettype can be different due to model abstraction.
    - Set nets to used 'interconnect' as OOMR's are not allowed, use the VPI Instead for this, SNUG Paper 2022
    - In AMS avoids 'T' style connect module insertion (merged CM), often a cause of issues.
  - Ensure debug of DC-Op is easy for AMS.
  - Limit the number of Connect Modules inserted in the testbench.

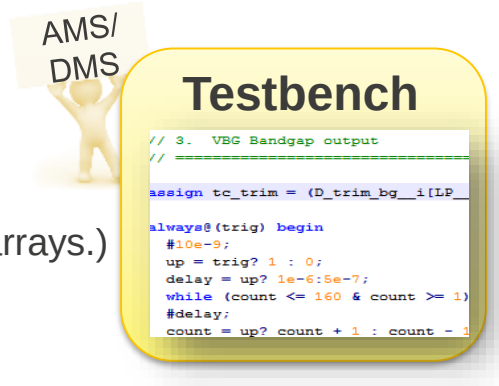
HOLISTIC CONCEPT

- 2 $roots modules (*UVM Dual Top Methodology*)
  - **test_case (TC)** to encapsulate the stimulus requests.
  - **test_env (TB)** to encapsulate the physical environment such as
    - Pin Drivers, monitor, external components.
    - Could reside in Schematic capture tool
- Scalable approach
  - test_case can utilize more of a SW methodology.
  - test_env can use hardware accelerators/emulators.
- Reduces cost by
  - Maximizing reuse between all DUT abstractions
  - Automating a common verification task of creating a TB and UVM environment.



AUTOMATED TESTBENCH REQUIREMENTS

- Able to generate a TB either from an RTL definition, netlist, or schematic symbol view.



- Support complex port structures for pure RTL DUT's. (OpenAccess limits support on schematics to 1-dimensional arrays.)
  - Multi-dimensional arrays mixing packed and unpacked sizes.
  - System Verilog Interfaces
  - Parameterised port sizes whether in a SV IF or not for a parameterised TB.

- Use only valid constructs in 1800-2012 LRM in the generated code. No vendor extensions!
  - Allows other parsers/tools to read the generated code. Vendor extensions are often limited to just one tool not all.

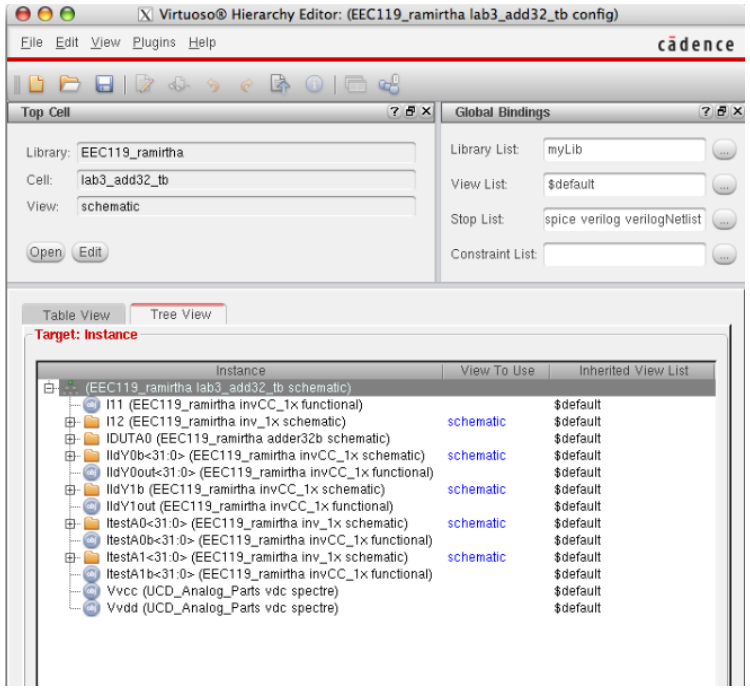
- Easy of use, regeneration and configuration. (Don't be a burden to the users!)
  - Support user code within generated files and structure it so template code is never needed to be hacked.

- Partition filelists/command line arguments so it easy to add/remove some option. E.g. UPF, Code coverage

- Work in harmony with Virtuoso (or your schematic framework tool) for DMS/AMS netlisting and integration.

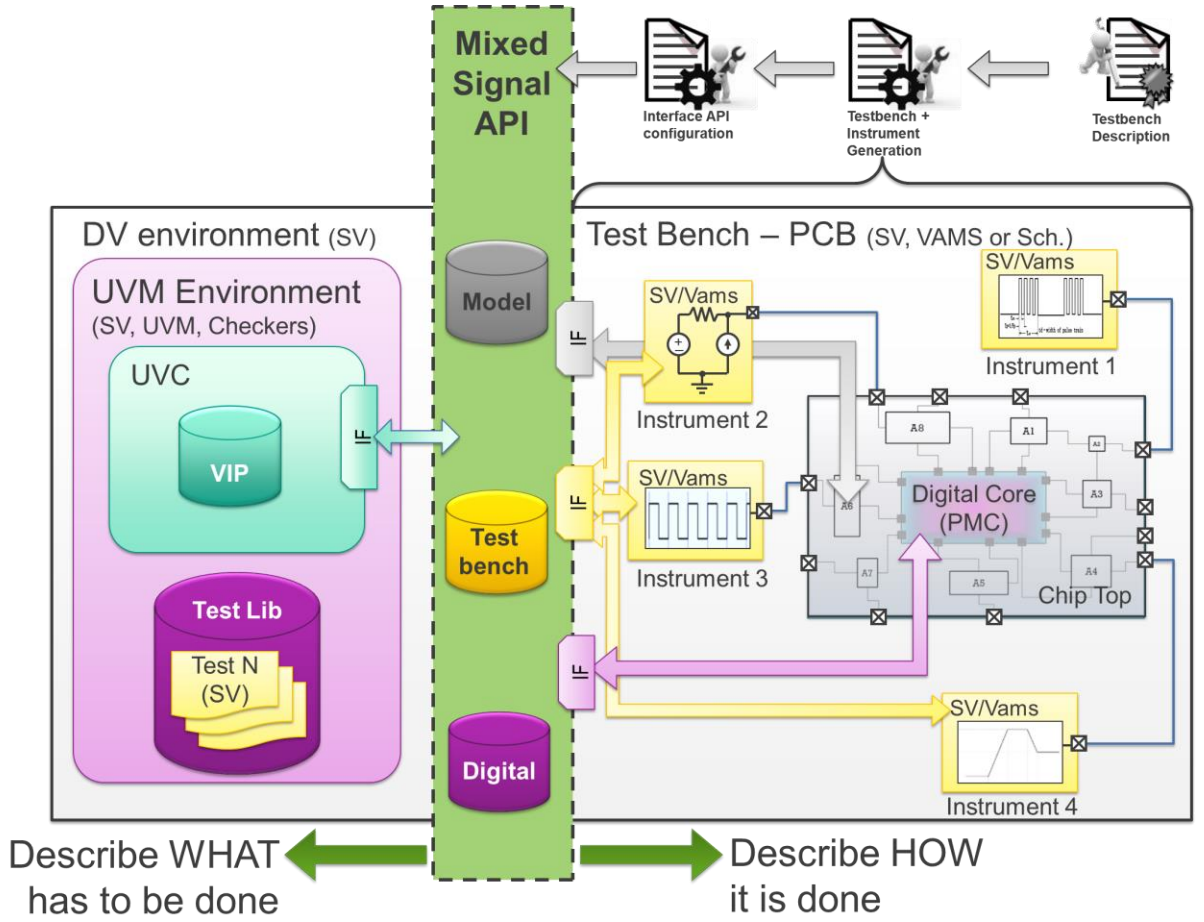
WHY IMPORT DMS/AMS TESTBENCH INTO SCHEMATIC TOOL?

- Schematic Capture tools come with netlisters and hierarchy editors (HED).
  - The hierarchy editor provides a nice and easy way to configure the design via a GUI.
  - Netlisters use the hierarchy editor to extract a netlist with a binding configuration file.
  - Makes sharing the testbench between DMS and AMS a lot easier.
  - Support text views just as well as schematic based views
    - So can work for digital-on-top or analog-on-top
- Pro's
  - Creating a binding file by hand can be error prone so use a tool.
    - E.g. lib.map and Verilog-Configuration files.
  - Far easier to stitch a design together for AMS, where spice uses order connection.
  - Graphical view of the hierarchy to easily root cause binding mistakes.
- Con's
  - Conditional typology within text views will not work, whether with a parameter or generate statement.
  - Generate statements can cause issues as the netlister/HED will not know what path to follow.
  - Not what a 'digital' guy would want to use.



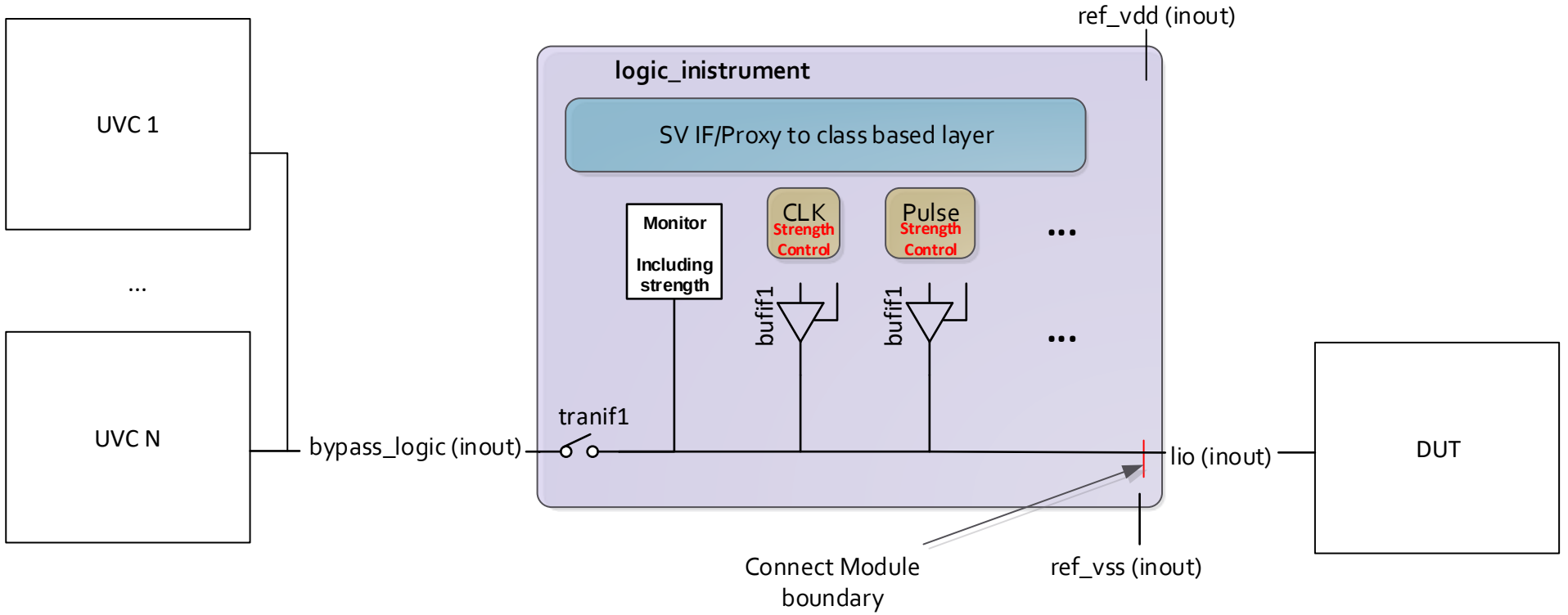
PHYSICAL TESTBENCH COMPONENTS

- Logic drivers/monitors
- Electrical drivers/monitors
- Passives Components/Networks
- Analog load models
- Digital UVC's



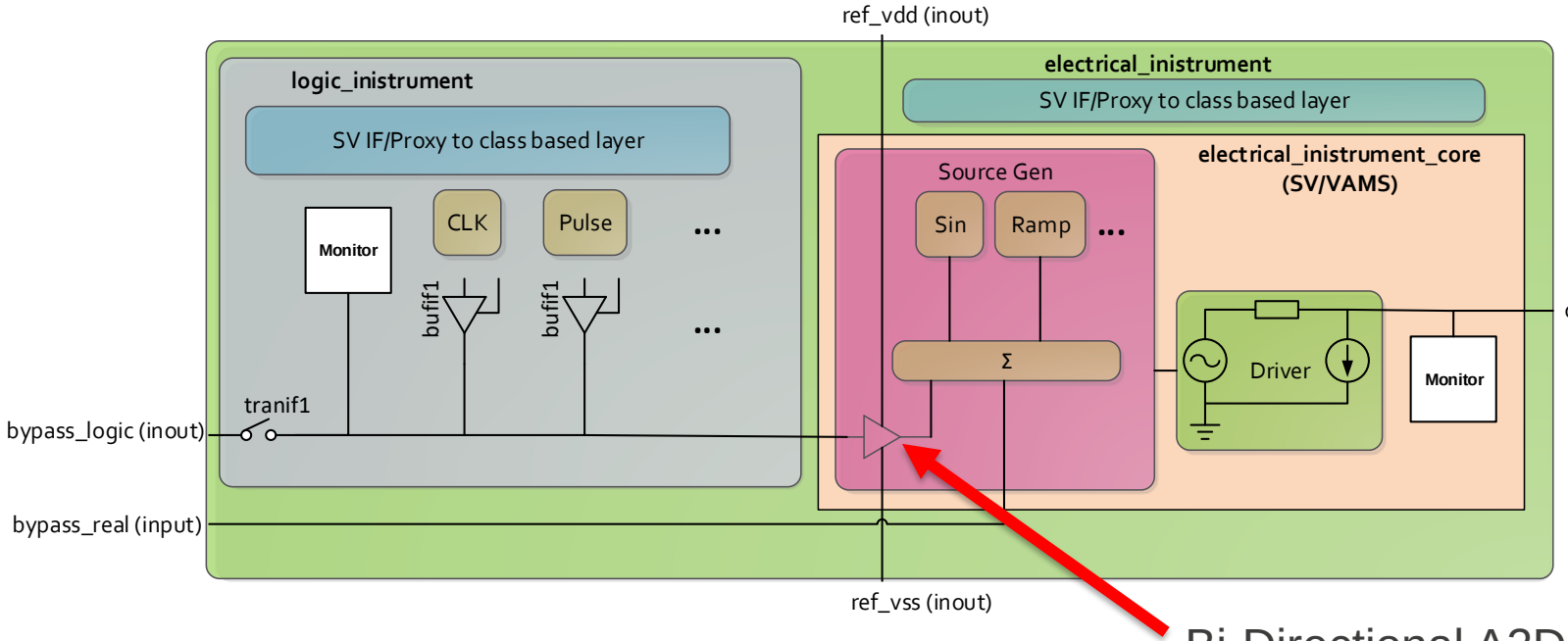
To share the testbench between DMS and AMS these components need multiple views

POSSIBLE LOGIC INSTRUMENT
DRIVER/MONITOR FOR LOGIC PINS IN DMS/AMS



- **lio** pin connects to DUT logic pin
- **ref_vdd/ref_vss** connect to an 'electrical' node for Connect Modules in AMS, unused in DMS/DV.
- **bypass_logic** – used as bidirectional connection to lio other stimulus to DUT pin that is parametrised.
  - Multiple UVC's can be connected to this node.
- SystemVerilog Interface (**SV IF**) / Proxy to enable class based interaction.

POSSIBLE ELECTRICAL INSTRUMENT
EXTENDS LOGIC INSTRUMENT



Source Gen is a real number passed to the driver stage.

In **AMS** Source Gen, Driver, and Monitor is in an analog statement.

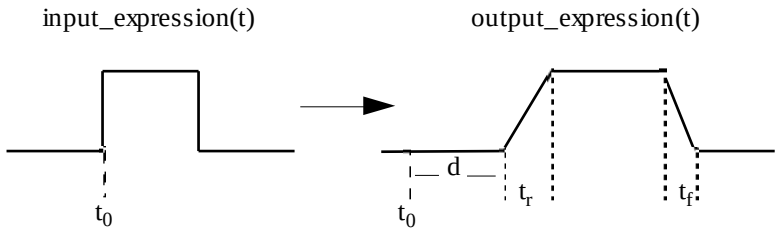
Bi-Directional A2D/D2A.

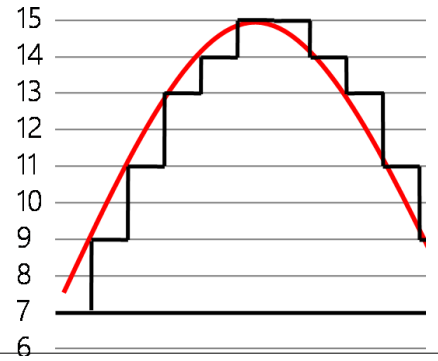
- **bypass_real** pin is a unidirectional input stimulus that is parameterised.
- **out** pin is the 'Electrical' output**.**
  - Support of Voltage, Current or Tristate output, with optional compliance.
- **ref_vdd/ref_vss** – used to level shift logic to Voltage level in DMS and AMS.
- SystemVerilog Interface (**SV IF**) / Proxy to enable class based interaction.

ANALOG SIGNAL GENERATION

- Analog Signal generation **MUST** be done in electric_instrument_core, not the UVM sequencer. (*See UVM-AMS*)
  - Ensure the right domain is used for 'electrical' signal generation ensuring *best simulation runtime performance*.
  - Signal instructions for sine wave, like frequency, phase and amplitude sent as a request to the drivers core.
    - Any change to variables used in analog statement is seen as a D2A event generating an analog matrix evaluation. **$$$**
    - Enables the analog solver to choose the best simulation timestep to use. (dynamic timestep algorithm)
    - DMS can choose the number of samples to create per period, for accuracy/performance

- **Surely I can do this digitally (only if you can be sure the hidden issues don't cause verification issues)**
  - AMS *transition* filter statement must have a tolerance of 0 and tr/tf must be exactly the same as the digital period for each sample.
    - Analog solvers rely on tolerances so a setting of 0 is bad and probably ignored. (Issue 1)
  - D2A event may not be exactly when digital happens but femto seconds later depending on the analog solver. Nasty harmonics! (Issue 2)
  - D2A event could create an unnecessary analog matrix solution to get good enough signal quality. (Issue 3)



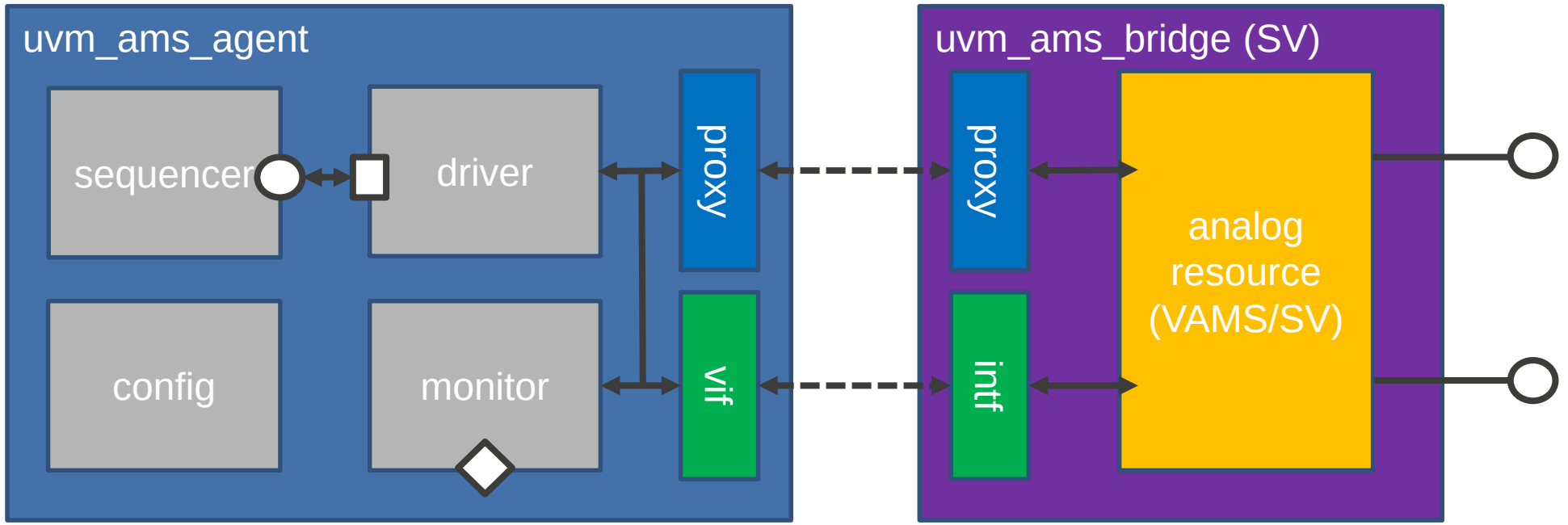


ANALOG SIGNAL MONITORING

- Analog Signal monitoring is **BEST** be done in electric_instrument_core, not the UVM sequencer. (*See UVM-AMS*)
  - Verilog-AMS has only the absdelta function to take a analog value and convert to a digital variable.
    - Can cause excessive A2D events slowing down simulations.
    - Can miss key values due to tolerances.
    - No one size fits all set of values, it depends on the signal been monitored.
  - Use analog statement to implement checkers and pass result to digital.

PASSIVES OR OTHER COMPONENTS

- Use a SV wrapper module around a core element.
  - Core element is swapped from DMS, AMS or other views based on abstraction.
  - SV wrapper to enable software connection to test_case module via intf/Proxy.
  - Renesas donation, including code/example, to create **UVM-AMS.**

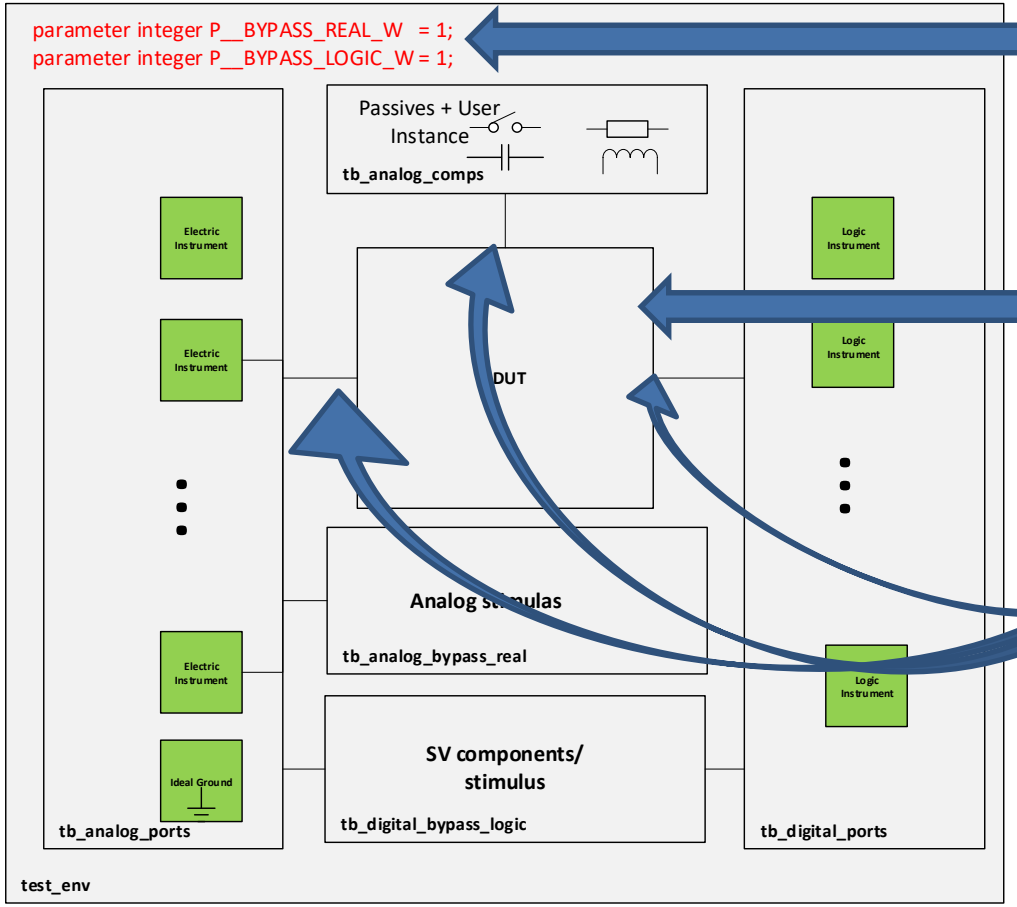


TESTBENCH PHYSICAL SPECIFICATION

TEST_ENV

- Each pin has a Logic or Electrical Instrument
- Users have the ability to add electrical models/components to electrical pins.
  - Cap, Res, Ind and maybe other electrical instruments on these networks, or instance a schematic for example.
- Support parameters on instances to set initial values for UVM config DB.
- Users can easily hook-up a bespoke logic driver(s) to the **bypass_logic**, e.g. I2C, SPI,SPMI etc.
- Users can easily create arbitrary reals signals via **bypass_real**
- Limit OOMR by using ‘interconnect’ for connectivity. (Except where datatype is fixed whether run in DMS or AMS.)
  - For DMS/AMS everything used interconnect other than bypass_logic connections.
- Limit user code to tb_analog_comps, tb_analog_bypass_real and tb_digital_bypass_logic modules.
- Support many use cases:
  - System level mixed signal (chip) to support AMS and/or DMS simulations
  - System/subsystem level mixed signal to support DMS only
  - Analogue block level schematic/model to support AMS/DMS
  - Pure digital block level for DV and DMS

TESTBENCH PHYSICAL STRUCTURE



Allow bypass_* ports to be parameterised. (Not used.)

Block to be tested. Ports could be parameterised

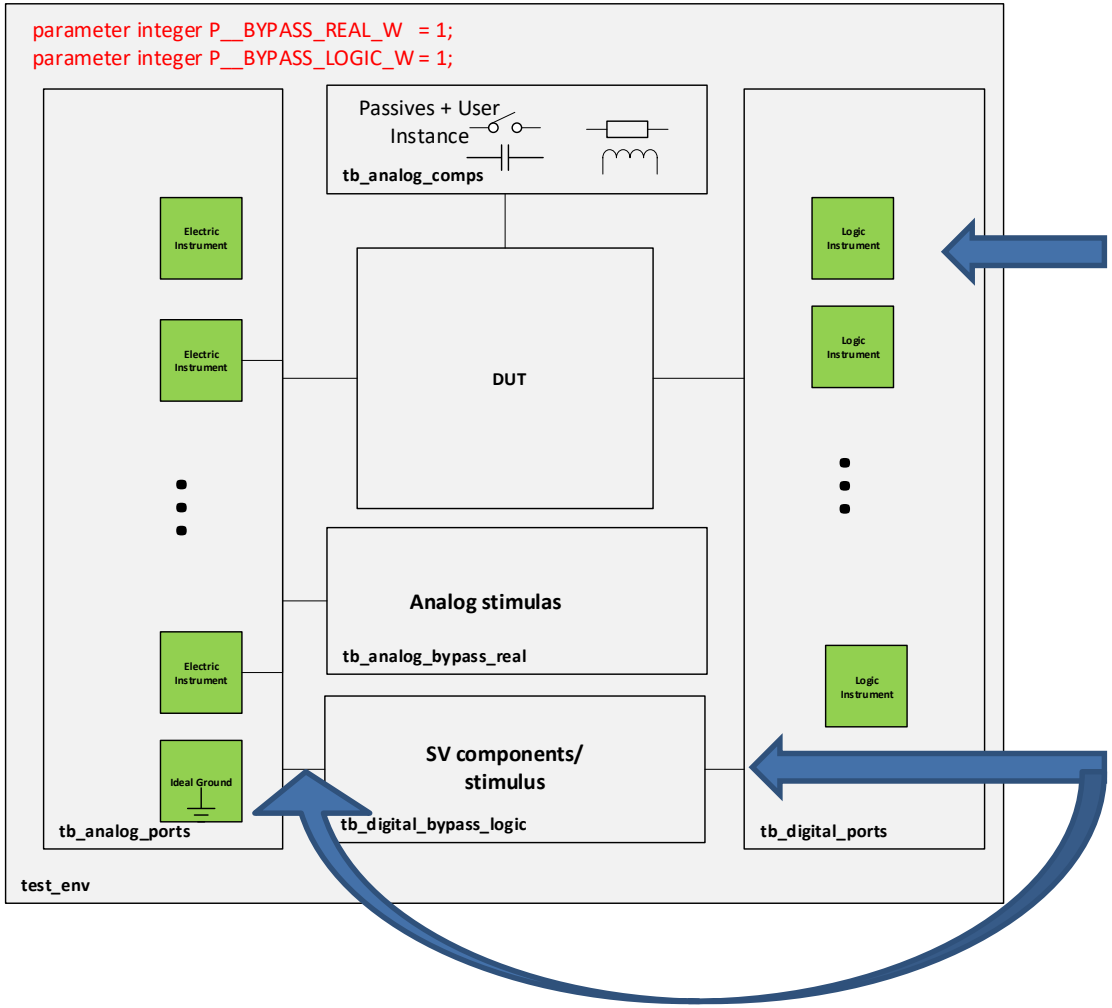
- Multidimensional
- SV Interface
- Packed/Unpacked arrays
- … (easy to support more.)

All DUT pins go to the modules:

tb_analog_ports
tb_digital_ports
tb_analog_comps

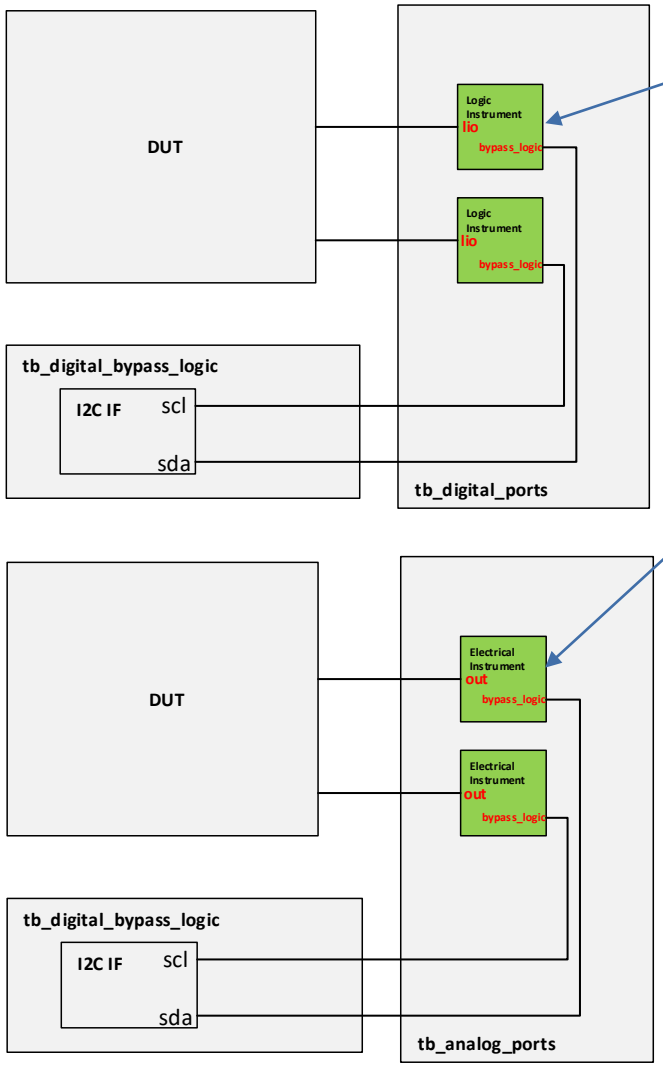
All structural using SystemVerilog constructs avoiding the limitations of schematics. E.g. Instance parameters, non-DUT parameterised ports.

TESTBENCH PHYSICAL STRUCTURE



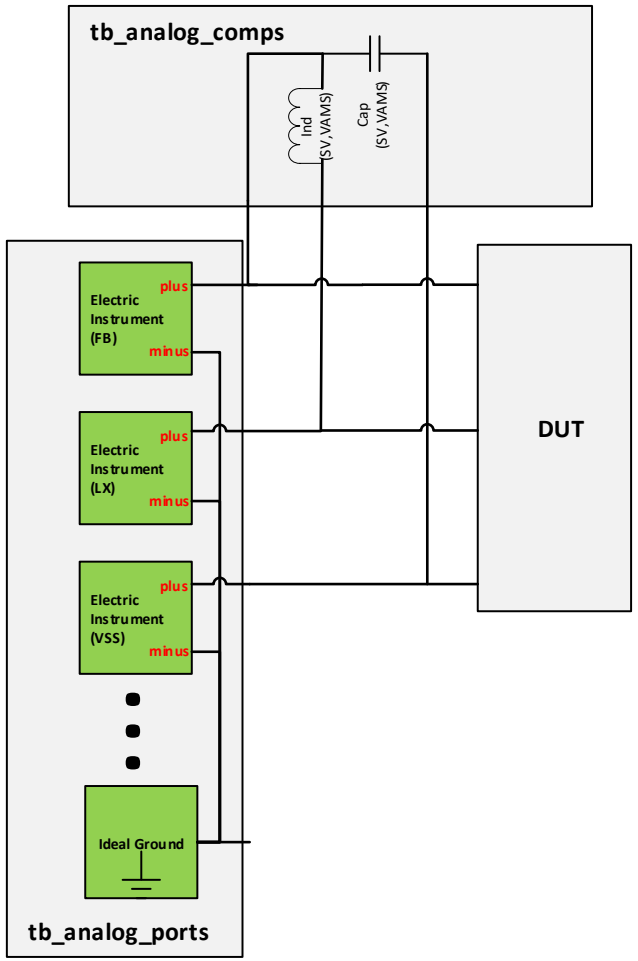
- Instance logic instruments for ‘digital’ ports
  - Ports unravelled so one instrument per bit
    - SVIF unravelled to ***lio*** port per bit.
  - Parameterised ports utilize generate loops.
- ref_vdd/vss usually connect to one of DUT pins for AMS.
  - Option have this as a TB only net.
- Ideal_Ground also ported in from tb_analog_ports, which is the node 0 for a reference.
- bypass_logic constructed to align with DUT port.
  - SVIF recreated per bit connection.
  - Parameterised ports supported.

ADDING DIGITAL UVC'S



- **Logic Instruments** set bypass_logic path
  - Simple tranif1 statement between lio and bypass_logic pin
  - Net collapsing means there is little to no runtime cost.
- **Electrical Instruments** set bypass_logic path
  - Bidirectional level shifter in the path. (A2D and D2A)
  - Rest comes from logic instrument instanced in electrical instrument.
- Instance I2C SV Interface in tb_digital_bypass_logic
- Use existing I2C Interface and API to do I2C reads/writes whether DUT is DMS or AMS.
- Internals to tb_digital_bypass_logic can use any SV constructs as always treated as digital in all use cases.
- Default is tb_digital_bypass_logic ports are driven with 1'bz.

ATTACHING A DCDC BUCK OUTPUT FILTER



- Components default values can be on the instance.
- Utilize Simulator scheduling to reprogram the DUT components via the test case rather than via view binding.
  - In AMS this can be before a DC-OP has happened but after the matrix is formed.
- Normal Usage
  - FB instrument can act as load.
  - LX instrument is left as tristate.
  - Control ind/cap settings controlled by API.
- ATE testing
  - FB instrument can act as source/load.
  - LX instrument can act as source/load.
  - ind/cap act as open via API.
- Optional DMS (based on DCDC model)
  - Views of components are empty modules.

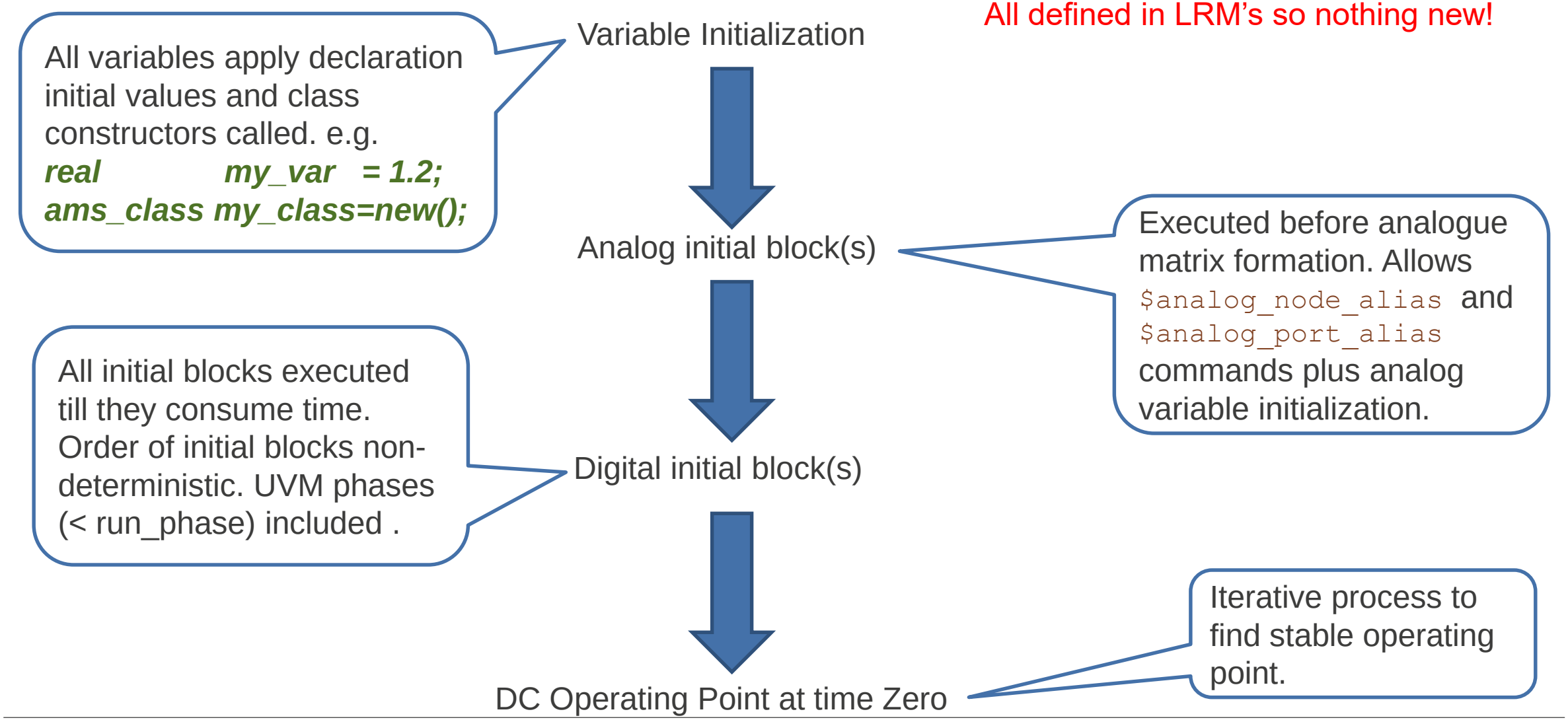
TESTCASE

- test_case kept as separate top level module for the software layer.
  - Imports UVM
  - Include file for all the user packages that need importing.
  - Import for base package
  - Include file for base environment that extend uvm_env
  - Base test class that extend uvm_test
    - Contains all test case classes for run time selection.
    - Could be autogenerated when a simulation is run, allowing filtering to tests to include/exclude.
  - One Initial block that calls run_test() for UVM.

- Template test case class sets the instruments and other components at time 0 pre DC-OP.
  - Allows reconfiguration of the electrical instrument and other electrical components before DC-OP
    - Step change cap values or instrument mode without convergence issues. Save numerous DUT configurations.
  - Electrical instruments default to 0V or 0A so DC-OP is easily found!
    - Value changes requested after at least #(1step) has happened.

VERILOG-AMS SIMULATOR SCHEDULING – TIME 0

All defined in LRM's so nothing new!



AMS/DMS START-UP AND PHASING

connect → end_of_elaboration → start_of_simulation → run

Read instance params into UVM cfg

```
virtual function void
my_driver::connect_phase(…);
  cfg.copy(proxy.getParameters());
endfunction : connect_phase
```

Set initial values before t=0

```
virtual function void
my_driver::start_of_simulation_phase(…);
  proxy.setParameters(cfg);
endfunction : start_of_simulation_phase
```

(Optional) Modify params from UVM test

```
virtual function void my_test::end_of_elaboration_phase(…);
  env.agent.cfg.rseries = 1e4; // 10k rseries in this test
endfunction : end_of_elaboration_phase
```

AMS/DMS STARTUP IN UVM RUN_PHASE()

Raise test objection → Wait for DC-OP → Launch sequences

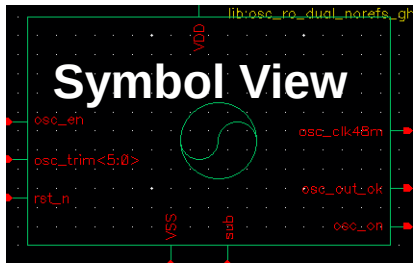
Ensures time is consumed

```
virtual task my_ams_test::run_phase(uvm_phase phase);
  …
  phase.raise_objection(this); // Prevent termination in DC OP

  if ($realtime <= 0.0) #1step;

  `uvm_info("TEST", "AMS DC-OP finished", UVM_MEDIUM)

  my_seq.start(my_seqr); // Launch sequence(s)
  …
  phase.drop_objection(this); // Test termination
endtask: my_ams_test
```

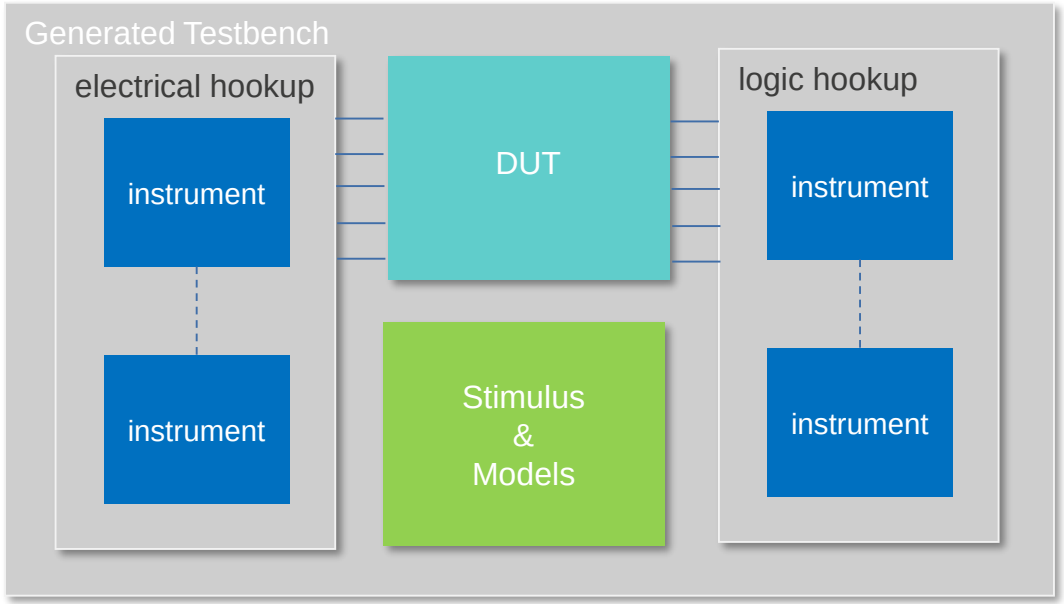
MAKETB SCRIPT

dia_maketb.xlsx

config.ini
or
filelist.f
or
*.sv



maketb.pl -cfg config.ini -top <blockname>




```
config.ini
rtl
verif
    ccov
    dia_maketb.xlsx
    regression
        vplans
        vsifs
    sim
    tb
        cov_model
        filelists
        packages
        procedures
        scripts
        sequences
        tcl
        tests
    uvc
```

Support files
& flow scripts

DUT PIN CONFIGURATION

From DUT's definition

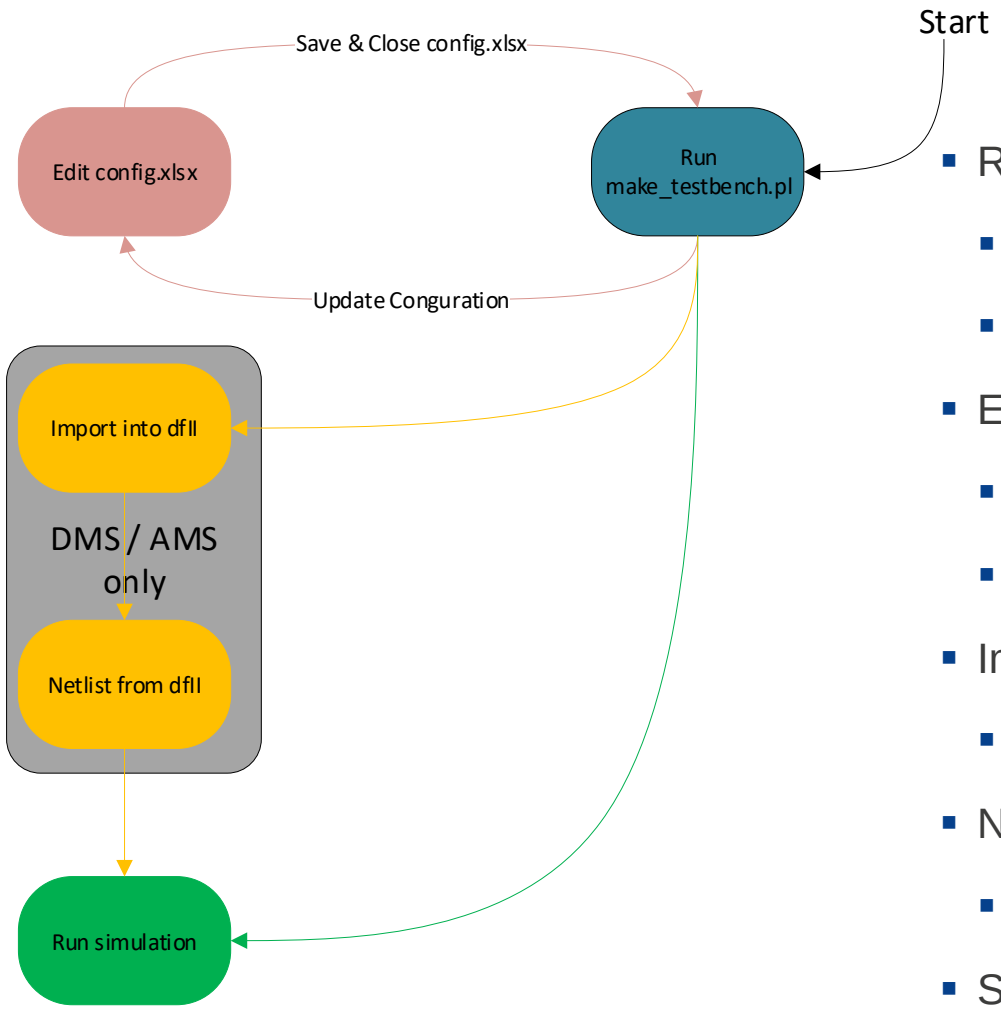
	A	B	C	D	E
1	Module	PinName	IO	PackedDims	UnpackedDims
2	loop_ctl	rst_n	input		
3	loop_ctl	clk	input		
4	loop_ctl	ticks	input	[P_TICK_WIDTH-1:0]	
5	loop_ctl	set_vsys_	input	[2:0]	
6	loop_ctl	set_vbat_	input	[6:0]	
7	loop_ctl	vbat_chg_	input	[9:0]	
8	loop_ctl	set_iprec	input		
9	loop_ctl	set_ilim_	input	[7:0]	
10	loop_ctl	set_itopo	input	[4:0]	
11	loop_ctl	ibat_chg	input	[7:0]	

User Controls

	F	G	H	I	J	K	L	M	N
1	Input_Class	Input_Subclass	Output_Class	Output_Subclass	Default	Vddsrc	Vsssrc	ActH	ActL
2	Digital	Digital_Input			1'b0	IDEAL_GN	IDEAL_GND		
3	Digital	Digital_Input			1'b0	IDEAL_GN	IDEAL_GND		
4	Digital	Digital_Input			{22{1'b0}}	IDEAL_GN	IDEAL_GND		
5	Digital	Digital_Input			{3{1'b0}}	IDEAL_GN	IDEAL_GND		
6	Digital	Digital_Input			{7{1'b0}}	IDEAL_GN	IDEAL_GND		
7	Digital	Digital_Input			{10{1'b0}}	IDEAL_GN	IDEAL_GND		
8	Digital	Digital_Input			1'b0	IDEAL_GN	IDEAL_GND		
9	Digital	Digital_Input			{8{1'b0}}	IDEAL_GN	IDEAL_GND		
10	Digital	Digital_Input			{5{1'b0}}	IDEAL_GN	IDEAL_GND		
11	Digital	Digital_Input			{8{1'b0}}	IDEAL_GN	IDEAL_GND		

- DUT's IO defines port direction input/output/inout/SV IF, name and diamensions.
- User Controls
  - Class is Digital/Analog
    - Sub Class defines it default configuration. E.g. Analog class, has subclass options like V_Analog_Input.
  - Default is the default drive value if an input.
  - Vddsrc/Vssrc define connections to **ref_vdd/ref_vss on instruments** to set logic levels

WORKFLOW



- RTL passed to make_testbench.pl for DUT IO details
  - Cadence's oa2verilog used if DUT is a schematic in dfII
  - Ports/Parameters used to create/update config.xlsx
- Editing of config.xlxs
  - Control pins' function/parameter values
  - Auto-insertion of some digital components into tb_digital_bypass_logic
- Import into dfII for DUT's for a schematic.
  - Unix script for those 'digital' guys
- Netlisting can use Cadence's VSVN or UNL-AMS.
  - Unix script for those 'digital' guys
- Simulation will run out the box!

FILE LIST PARTITIONING

- Source code and command line arguments order is important.
  - SV Packages must be parsed before imported. E.g. UVM often is the first to be required.
  - Leaf modules should be parsed before instanced.
  - Make use of –v/-y for libraries.
- Enable ease of control
  - Whether an feature is required or not. E.g. Code Coverage, GLS or RTL.
  - Switching from running a DMS simulation to an AMS one.
- Suggested order;
  - Defines
  - UVM
  - Other Packages based on order dependencies.
  - Interfaces
  - Leaf modules

FILELIST PARTITIONING EXAMPLE

Cadance	Synopsys	Description/Contents	Order #
xrunargs.c.f	vcsargs.c.f	C code command line switches.	
xrunargs.com.f	vcsargs.com.f	General command line switches.	3
xrunargs.cov.f	vcsargs.cov.f	Code Coverage command line switches.	.
xrunargs.uvc.f	vcsargs.uvc.f	UVC's	5
xrunargs.last.f	vcsargs.last.f	Last *.f file in generated file list if there are specific options that must be last.	.
xrunargs.msg.f	vcsargs.msg.f	Command switch to control simulator messages.	4
xrunargs.rtl.f	vcsargs.rtl.f	RTL level command line switches.	.
xrunargs.sdf_fast.f	vcsargs.sdf_fast.f	Settings for SDF fast corner files	.
xrunargs.sdf_slow.f	vcsargs.sdf_slow.f	Settings for SDF slow corner files	.
xrunargs.uvm.f	vcsargs.uvm.f	UVM command line switches.	1
xrunargs.vhdl.f	vcsargs.vhdl.f	VHDL code command line switches.	.
sdf.fast.cmd	-	Cadence SDF control file for the fast corner	.
sdf.slow.cmd	-	Cadence SDF control file for the slow corner	.
sdf.tfile	sdf.vcs.cmd	File to control timing checks, such as disabling them on sync cells.	.
files.com.f		Files, incdir, defines and libraries that are needed for the test_env and test_case hierarchy. E.g. packages/interfaces	2
files.hw.f		Files for the test_env hierarchy excluding the DUT netlist.	.
files.sw.f		Files for the test_case hierarchy.	6

CURRENT LIMITATIONS

- Good open source SystemVerilog Parsers/Elaborators are hard to find. Some recent ones look promising.
- Getting IO details can require pre and post elaboration details. E.g. Width of Enum’s defined in package file.
  - Order of package file search is required.
- Currently using the Verilog-Perl parser the Renesas implementation can’t handle
  - Nested SV interfaces.
  - Ports using packages to define the datatype. (Requires elaboration)

Renesas.com