RENESAS'S CONTRIBUTION TO ACCELLERA UVM-(A)MS

14TH SEPTEMBER 2023

PETER GROVE / STEVEN HOLLOWAY
MEMBERS OF TECHNICAL STAFF
RENESAS ELECTRONICS CORPORATION

Agenda

- Why UVM-MS
- Verilog-AMS Simulator DC OP / Transient behavior
- UVM MS Bridge to analog resource (UVM->AMS/DMS Connection)
- UVM-MS Phasing Requirement
- UVM messaging from AMS files and $root cells

WHY UVM-MS

- UVM is the industry standard methodology for reusable *metric driven* verification
- **UVM-MS** is the standardisation of analogue/mixed signal extensions for UVM
  - Allows UVM to be more mixed-signal aware
  - Improved verification of analogue/mixed-signal designs
  - Same degree of thoroughness for both analogue and digital parts
- Originally named UVM-AMS but focus is to support any MS system; DMS, RNM, Spice or a mixture
- Metric-driven verification suits following objectives due to verification space size
  - Verifying analogue performance under large set of digital configurations
  - Digital control system transitions interacting with analogue functions
  - Dynamic control between analogue & digital circuits under wide range of conditions
  - Finding problems with A/D interaction in unexpected corner cases
- Randomisation is not mandatory and benefits are gained even when using directed tests
  - Standard methodology
  - Plug & play reuse of existing UVM components
  - Rich debug & messaging scheme integrated with simulator

UVM

accellera SYSTEMS INITIATIVE

UVM-MS REQUIREMENTS

- Apply UVM methods and techniques to AMS circuits and systems while allowing DMS/RNM.
- Enable a **single environment** to work whether it is DMS/RNM or AMS by changing the abstraction of the DUT.
- Extend the use of UVM components, and extensions thereof, into the physical layer enabling AMS verification.
- Allow predictable coordination of stimulating/measuring a signal
- Adhere to the sequence/sequence-item mechanism used by UVM
- Independent of the abstraction level of the AMS signals (electrical, RNM, UDT, etc.)
- Eliminate the need to rely on conversion elements to change the abstraction level of the DUT signals.
- Use existing language standards; SV and Verilog-AMS
  - Changes take years to get agreement.

Agenda

- Why UVM-MS
- Verilog-AMS Simulator DC OP / Transient behavior
- UVM MS Bridge to analog resource (UVM->AMS/DMS Connection)
- UVM-MS Phasing Requirement
- UVM messaging from AMS files and $root cells

VERILOG-AMS SIMULATOR DC OP

DC Op – Steady State operating point of all the nodes/branch currents

- Understanding of UVM-MS DC OP is important;
  - Knowing the sequence when variable assignment/initial block(s) all execute
    - To avoid race conditions between digital blocks.
    - To avoid odd issues at initialization of analog/digital constructs.
    - To avoid process initialization issues.
    - To make sure the correct value is captured between the analog/digital engines.
  - Knowing the effects of DC Op on certain AMS filters as they are different to the transient response.
    - e.g. transition, absdelta, above, cross, absdelay, Laplace.
  - Skipping DC op will cause odd results and vendor specific.
  - Enable UVM DUT configuration prior to analog circuit initialization (DC Op).
    - E.g. Make a Cap open for a particular test before DC op or short/open a path saving multiple testbenches of configurations.
  - Assist in debug when DC OP fails as there is often nothing in the waveform files to debug.
    - Use @(initial_step) $strobe() statement to print out values via ifdef
  - Using #0 is not good practice as it shows poor coding and understanding of the simulator(s) schedular.
- Must raise a UVM objection before DC OP otherwise the simulation finishes.

VERILOG-AMS SIMULATOR SCHEDULING – TIME 0

All defined in LRM's so nothing new!

All variables apply declaration initial values and class constructors called. e.g.
real my_var = 1.2;
ams_class my_class=new();

Variable Initialization

Analog initial block(s)

Executed before analogue matrix formation. Allows `$analog_node_alias` and `$analog_port_alias` commands plus analog variable initialization.

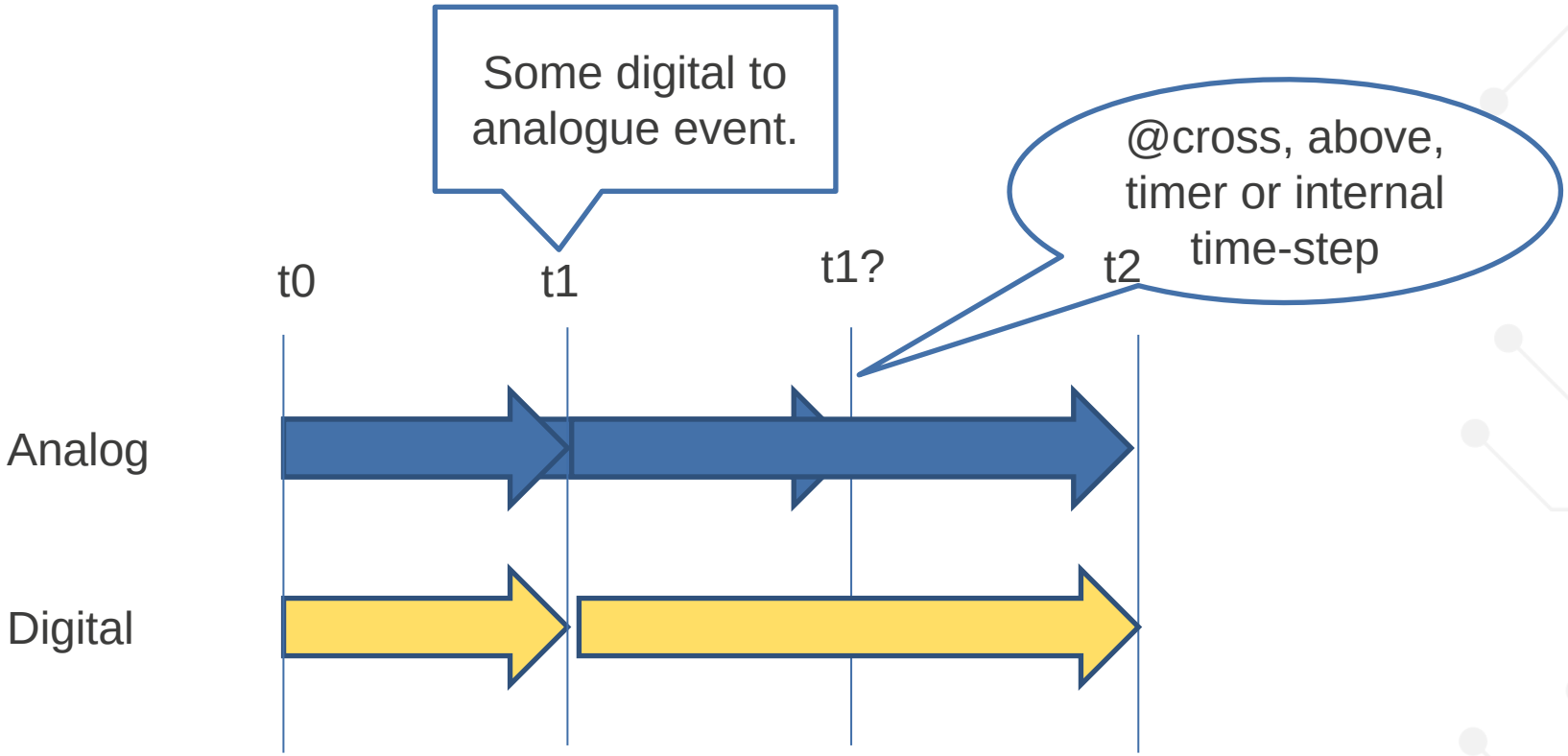
All initial blocks executed till they consume time. Order of initial blocks non-deterministic. UVM phases (< run_phase) included .

Digital initial block(s)

DC Operating Point at time Zero

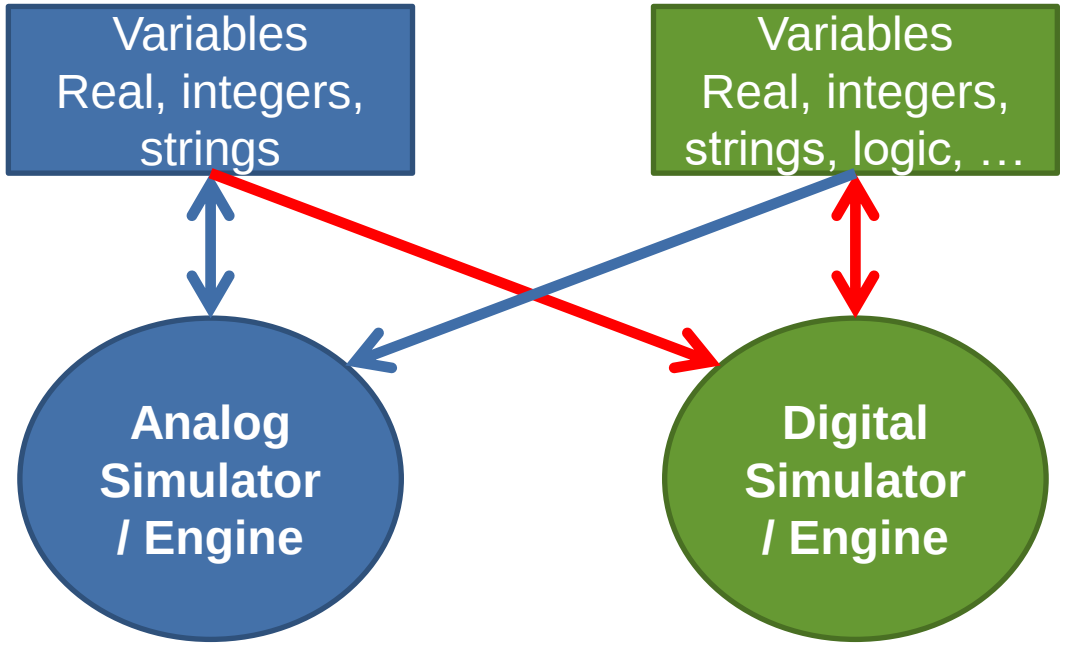
Iterative process to find stable operating point.

VERILOG-AMS SIMULATOR SCHEDULING - TRANSIENT



- Analogue engine always leads.
- Digital to analogue events cause matrix re-evaluation and timestep backtrack.
  - Most simulators see any digital var in the analog block as a D2A to monitor.

VERILOG-AMS BEST PRACTICES



- Variables are ‘owned’ by one engine, but can be read by another.
- AMS can’t access digital variables that are dynamic. (Everything in the matrix is fixed at time 0)
- Generally avoid ‘string’ datatypes in Verilog-AMS as support is flaky and the LRM is not clear.
- OOMR to analog owned variables not allowed – they are not part of the analog matrix.

Agenda

- Why UVM-MS
- Verilog-AMS Simulator DC OP / Transient behavior
- UVM MS Bridge to analog resource (UVM->AMS/DMS Connection)
- UVM-MS Phasing Requirement
- UVM messaging from AMS files and $root cells

MIXED SIGNAL BRIDGE & ANALOG RESOURCE

- MS Bridge (SystemVerilog) to connect the UVM layer to the analog resource.
  - The analog resource could be DMS/RNM/AMS based on DUT pin abstraction.

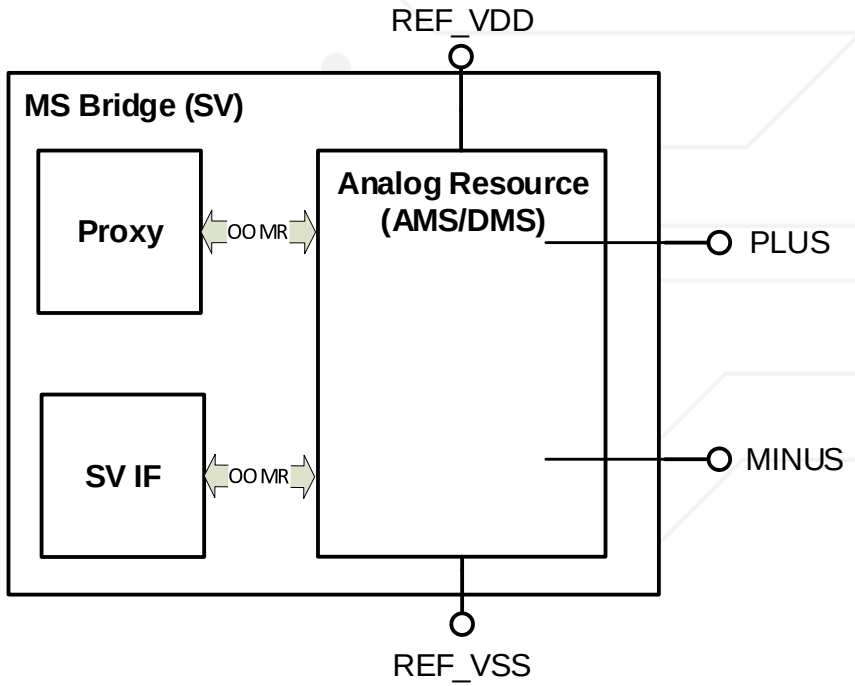
- Proxy Features (mandatory)
  - ❖ Can't contain wires needed for logic strength by some digital type IO's only logic/reg are valid.
  - Push analog resource controls using function calls.
  - Push-Sync contain registers for end of transition detection or other synchronisation from resource control.
  - Pull analog resource values via functions calls.
  - Monitored 'reals' for monitored continuous signals.

- SystemVerilog Interface (optional) digital signals as they are currently
  - Enable logic strength/ports on wires.
  - More suited to allow reuse of existing IF with a MS Bridge.

- PLUS/MINUS/REF_VDD/REF_VSS set as '*interconnect*' in MS Bridge.
  - Allow shared DMS/AMS/RNM analog resource.

- REF_VDD/REF_VSS for SV IF logic level to electrical conversion.

- OOMR's work around datatype limitations on AMS modules IO's.



ANALOG RESOURCE – WHAT DOES IT LOOK LIKE

Ensure OOMR, port, paramaters from MS Bridge to analog_resource abstractions are the same!

SV

```
module ms_bridge #(parameter real res_val = 1.0,
                   parameter real res_tf = 1.0e-9,
                   parameter real res_tf = 1.0e-9
                  )(inout interconnect PLUS, inout interconnect MINUS);

   import   uvm_pkg::*;
   import   uvm_ms_pkg::*;
   `include "uvm_macros.svh"
   `include "uvm_ms.svh"

    //Proxy declaration + proxy instance + SV IF + Instance of analog_resource
```

AMS

```
module analog_resource (PLUS, MINUS);
   output PLUS, MINUS;
   electrical PLUS, MINUS;

   parameter real res_val = 1.0;
   parameter real res_tr = 1.0e-9;
   parameter real res_tf = 1.0e-9;

   `include "uvm_ms.vamsh"


   function real getVoltage(input dummy);
       begin
           getVoltage = V(PLUS,MINUS);
       end
   endfunction

   ...
```

DMS

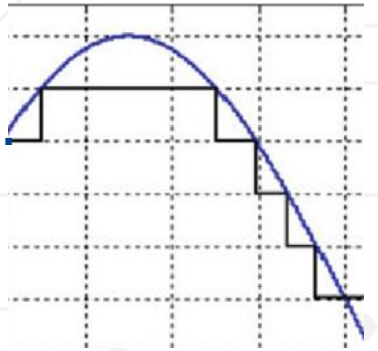
```
module analog_resource (PLUS, MINUS);
   output PLUS, MINUS;
   real PLUS, MINUS;

    parameter real res_val = 1.0;
    parameter real res_tr = 1.0e-9;
    parameter real res_tf = 1.0e-9;

    import uvm_pkg::*;
    `include "uvm_ms.dmsh"

    function real getVoltage(input dummy);
       return PLUS-MINUS;
    endfunction
```

Required for UVM messaging

Verilog-AMS functions must have input!

ANALOG RESOURCE – DOES WHAT?

- For logic signal the analog_resource *must* convert the logic to the DUT pin abstraction.
  - Proxy can be used to control the properties of the conversion element. Logic to UDN/Real/Electrical
  - For logic DUT pin abstraction it is a short. `alias in = out;`
- Classical RNM would drive real numbers from UVC sequence/driver within the agent.
  - **In AMS this would generate to many D2A events or not give enough finesse to the signal.**
  - Place the signal generator is located in the domain of the DUT I/O it will connect to.
  - A generator could be made of many components; ramp noise, sinewave, logic conversion.
- A sine wave is made up of 4 properties; *frequency*, *phase*, *amplitude*, and *DC bias*.
  - UVM transaction encodes the properties of the sine wave as real values in the uvm_sequence_item
  - Properties passed to analog_resource to generate the sine wave.
  - Still honors the UVM paradigm of having a relatively simple interface for the test writer

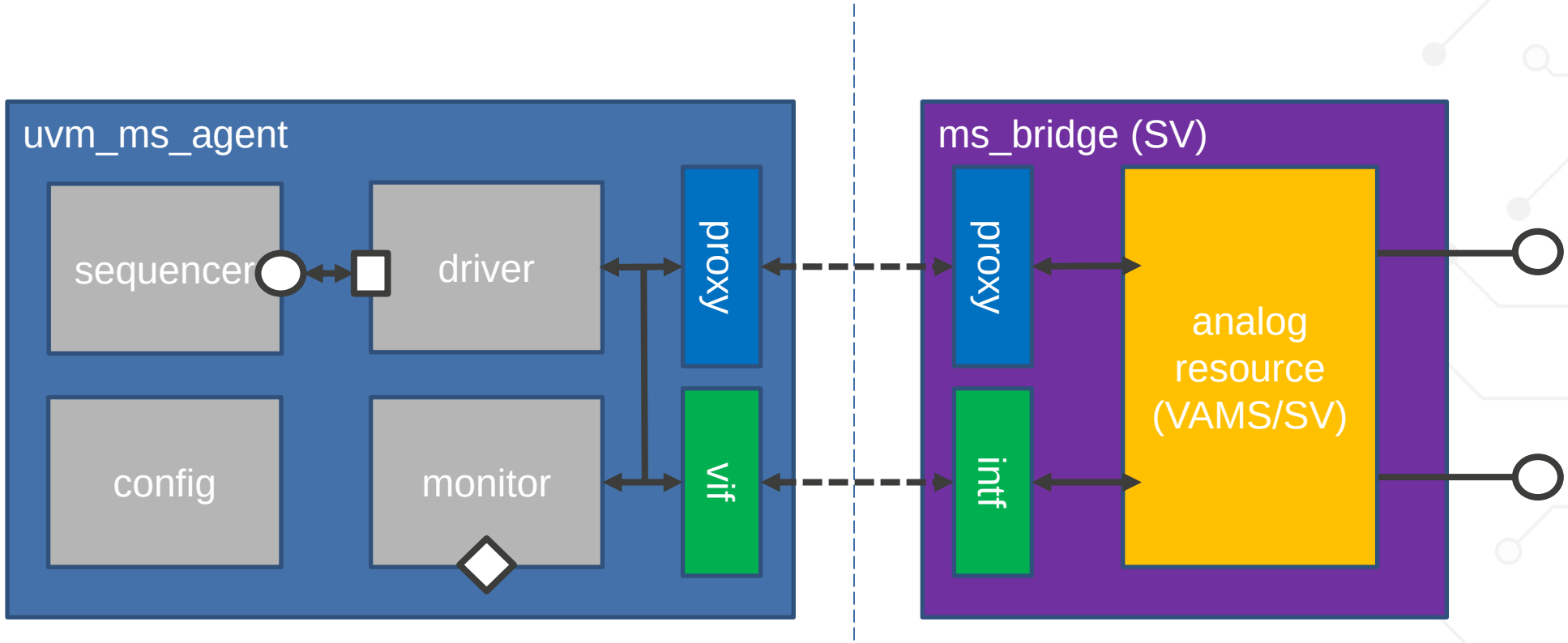


- Change from classical UVM sequencer/drivers for UVM-MS⚠

PROXY CLASS

- Proxy is designed to be a “thin layer” between UVM and the analog resource implementation
- Alternative style of connection between UVM classes and SV static hierarchy
- Proxy class derived from *uvm_ms_proxy* which is derived from *uvm_report_object*
- Embedded class definition placed inside SV bridge module – called “MSProxy” – concrete class
  - **Class instance must be called __uvm_ms_proxy for messaging to work.**
- A handle to the embedded proxy class is obtained by hierarchical reference and placed in the uvm_config_db for access by UVM components. Same as SV Virtual IF!
- Implementation of proxy API methods in bridge module in turn execute analog resource “core” methods – hence “proxy” pattern.

UVM-MS AGENT BLOCK DIAGRAM



- Proxy MUST always be present and instanced as __uvm_ms_proxy – more later on this!
- SV IF is optional but must be placed in ms_bridge.
- Agent could use override using the UVM factory to extend the pure digital solution to a mixed signal one.

MS PROXY “HOOK-UP”

Proxy Template (API)

```
class cap_proxy extends uvm_ms_proxy;
…
  virtual function void setCapacitance(…);
  virtual function real getCapacitance(…);
…
endclass
```

Implement

Proxy instance in bridge module

```
module cap_bridge(…);
…
cap_core #(…) i_core (…); // AMS model
…
class MSProxy extends cap_proxy;
…
   function void setCapacitance(real val, real tr, real tf);
        i_core.setCapacitance(val, tr, tf);
        …
   endfunction
    …
endclass

MSProxy __uvm_ms_proxy= new(“__uvm_ms_proxy”);
…
endmodule
```

Instance of analog resource

Calls function in analog resource to ensure no race/synchronization issues.

Must be called__uvm_ms_proxy*

UVM config setting

```
module tb;
…
  cap_bridge  i_cap_bridge (.PLUS(cap_node),.MINUS(gnd));
…
  initial begin
    uvm_config_db#(cap_proxy)::set(null, "uvm_test_top.env", "c_proxy", i_cap_bridge.__uvm_ms_proxy);
    …
    run_test("uvm_ams_test");
  end
endmodule
```

Register proxy in config_db

PROXY ←→ ANALOG RESOURCE

MS Proxy Class

```
function void setRamp(input real val,tx);
Push void'(i_core.setRamp(val,tx));
endfunction

function real getVoltage();
Pull   return i_core.getVotlage(0);
endfunction

real V_PLUS_MINUS;
```

Analog Resource (AMS/DMS)

```
real tf,tR, target;

function integer setRamp(input real val,tx);
  begin
      tf = tx; tr = tx;  target = val;
      setRamp = 1;
  end
endfunction

analog
   V(PLUS,MINUS) <+ transition(target,0,tr,tf);

function real getVoltage(input dummy);
  begin
     getVoltage = V(PLUS,MINUS);
  end
endfunction

real V_PLUS_MINUS;
always@(absdelta(V(PLUS,MINUS),...)) V_PLUS_MINUS = V(PLUS,MINUS);
```

If target is different its seen as a D2A event

Interpolated value

Analog generates update

REF_VDD

PLUS

MINUS

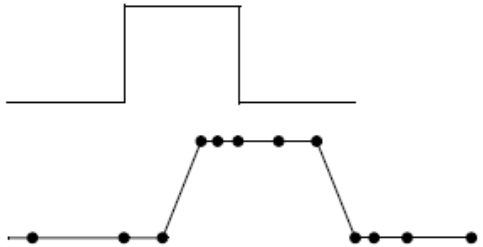
REF_VSS

Monitored

```
assign i_proxy.V_PLUS_MINUS = i_core.V_PLUS_MINUS;
```

PROXY – PUSH ANALOG RESOURCE WITH SYNCHRONIZATION (PUSH-SYNC)

- Transition filter in AMS is used to convert discrete signals to a continuous time one.



Start/Stop have tolerances!

  - Use some AMS code to detect if the transition filter has completed. Useful feedback to agent.

```
vdc_tran = transition(vdc, 0,vdc_tr, vdc_tf );

...

eot_vdc = (abs(vdc_tran - vdc) < tol) ? 1:0;
```

tolerance

- Use function calls from UVM to the Analog Resource to ensure stack has completed.
  - Updated eot_vdc in proxy as part of this function stack
  - Use @(eot_vdc) to block further agent execution until request has completed.
- Can't use #(delay) in UVM agent as it requires analogue timestep to update eot_vdc!
  - Might not happen if the new value of vdc was the same as the last or transition tolerance is big.
- UVM agents can use sync items in the proxy to implement reactive behaviour to AMS conditions
  - e.g. blocking call to voltage ramp which returns when target is reached

PROXY – PUSH ANALOG RESOURCE WITH SYNCHRONIZATION (PUSH-SYNC)

```
proxy.setCapacitance(txn.farads, txn.trise, txn.tfall);
wait(proxy.cap_eot); // VAMS handshake in proxy
```

MS Bridge

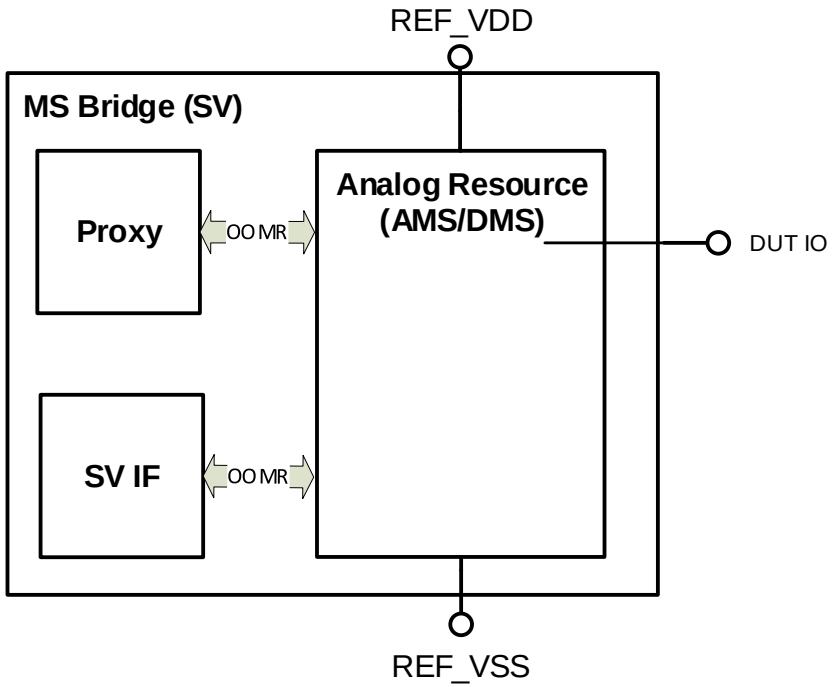
```
  reg cap_eot;
  ...
  function void setCapacitance(real val, real tr, real tf);
    void'(i_core.setCapacitance(val, tr, tf));
    cap_eot = i_core.cap_eot; //Ensure value is updated before function returns
  endfunction
  ...
endclass
...
AMSProxy i_proxy = new();
...
always begin i_proxy.cap_eot = i_core.cap_eot; @(i_core.cap_eot); end
```

Analog Resource

```
function automatic integer setCapacitance(input real val, tr, tf);
   begin
     cap = val; cap_tr = tr; cap_tf = tf;
     cap_eot = (abs(cap_tran-cap) < 1e-7) ? 1:0;
     setCapacitance = 1; //Always return 1.
    end
endfunction
```

```
reg cap_eot;
analog begin
  ...
  cap_tran = transition(cap, 0,cap_tr, cap_tf );
  analog_clk = 1 - analog_clk;
   ...
end
always@(absdelta(analog_clk,1,0,0, 1))
  cap_eot = (abs(cap_tran-cap) < 1e-7) ? 1:0;
```

MS BRIDGE FOR LOGIC SIGNAL



- Abstraction of analog_resource to match DUT IO
- OOMR, IO, Parameters align.
- More powerful than using conversion elements.
  - Dynamic or static supplies.
  - Control all aspect.
- Examples to be provided!

DUT is logic

```
module analog_resource (inout interconnect VDD, VSS, output wire dout);
   wire din; //OOMR
   alias din = dout; //short dout to be din
   //OOMR Vars
   real tr = 0.4e-9; //Voltage ramp time
   ...
   //OOMR Dummy functions to replicate whats in the AMS/DMS abstraction
   function automatic integer setLogicSupply(...);
      setLogicSupply = 1'b1;
   endfunction
   ...
```

DUT is RNM/DMS

```
module analog_resource (input real VDD, VSS, output real dout);
   wire din; //OOMR
   //OOMR Vars
   real tr = 0.4e-9; //Voltage ramp time
   ...
   //functions to control conversion
   function automatic integer setLogicSupply(...);
      ...
      setLogicSupply = 1'b1;
   endfunction
   ... Some assignment to dout as a real number
```

DUT is AMS

```
module analog_resource (inout electrical VDD, VSS, dout);
   wire din; //OOMR
   //OOMR Vars
   real tr = 0.4e-9; //Voltage ramp time
   ...
   //functions to control conversion
   function automatic integer setLogicSupply(...);
      ...
      setLogicSupply = 1'b1;
   endfunction
   ... Analog block to contribute onto the electrical dout pin
```

Agenda

- Why UVM-MS
- Verilog-AMS Simulator DC OP / Transient behavior
- UVM MS Bridge to analog resource (UVM->AMS/DMS Connection)
- UVM-MS Phasing Requirement
- UVM messaging from AMS files and $root cells

UVM PHASING REQUIREMENTS FOR AMS

- MS Bridges will have parameters.
- UVM should have a means to read/modify/write params before simulation consuming time
- Implement methods getParameters() / setParameters() in proxy
- Use existing UVM phases to guarantee read/modify/write order

UVM Phase	What should happen for AMS resources
build	
connect	Read parameters values from ‘SV+VAMS’ module (Instrument/Passive) into the agent’s configuration.
end_of_elaboration	Modify agents parameters based on test requirements
start_of_simulation	Apply agents parameters to ‘SV+VAMS’ module (Instrument/Passive)
run	Must consume some time to allow DC OP to happen before agents drive sequence items so that synchronization system works. Recommend run_phase() in the base test to have a if($realtime <= 0.0) #1step; to cause a DC OP to happen.
extract	
check	
report	
final	

ANALOG RESOURCE CONFIGURATION

- Analog components tend to be placed with initial values as parameters. E.g. A decoupling cap on a LDO output.
- Allow the MS Bridge to have parameters that are copied UVM configuration in connect_phase.
- Test cases can override the configuration, which are then set in the analog resource in start_of_simulation_phase.
  - This is pre DC OP so you can do step changes to analog values!

```
module ms_bridge #(parameter real res_val = 1.0, …)
                  (inout interconnect PLUS, MINUS );
import uvm_pkg::*;
import uvm_ms_pkg::*;
`include "uvm_macros.svh"
`include "uvm_ms.svh"

import res_pkg::*;

class MSProxy extends template_proxy;
   …
   function res_config getParameters();
      res_config cfg = new();
      cfg.res_val = i_core.rseries_val;
      …
      return(cfg);
   endfunction: getParameters

   function void setParameters(res_config cfg);
      i_core.rseries = cfg.res_val;
      …
   endfunction: setParameters
…
analog_resource i_core #(.res_val(res_val),...)(.PLUS (PLUS),.MINUS(MINUS));
```

read path

write path

```
module analog_resource (PLUS, MINUS);
   inout PLUS, MINUS;
   electrical PLUS, MINUS; //Values read by getParamters in MS Bridge
   parameter real res_val = 1.0;
   parameter real res_tr = 1.0e-9;
   parameter real res_tf = 1.0e-9;

   //Initial values set from parameter, then set
   //by setParameter in MS Bridge
   real rseries_val = res_val;
   real rseries_tr = res_tr;
   real rseries_tf = res_tf; ...
```

For DMS this could have empty functionality!

Parameter assigned to variable so setParameters can override them. Variable used in rest of code not parameters

AMS START-UP & UVM PHASING

- AMS models will have parameters!
- UVM should have a means to read/modify/write params before simulation consuming time
- Use UVM phasing to guarantee read/modify/write order

Read VAMS params into UVM cfg

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

Set VAMS 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 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
```

Agenda

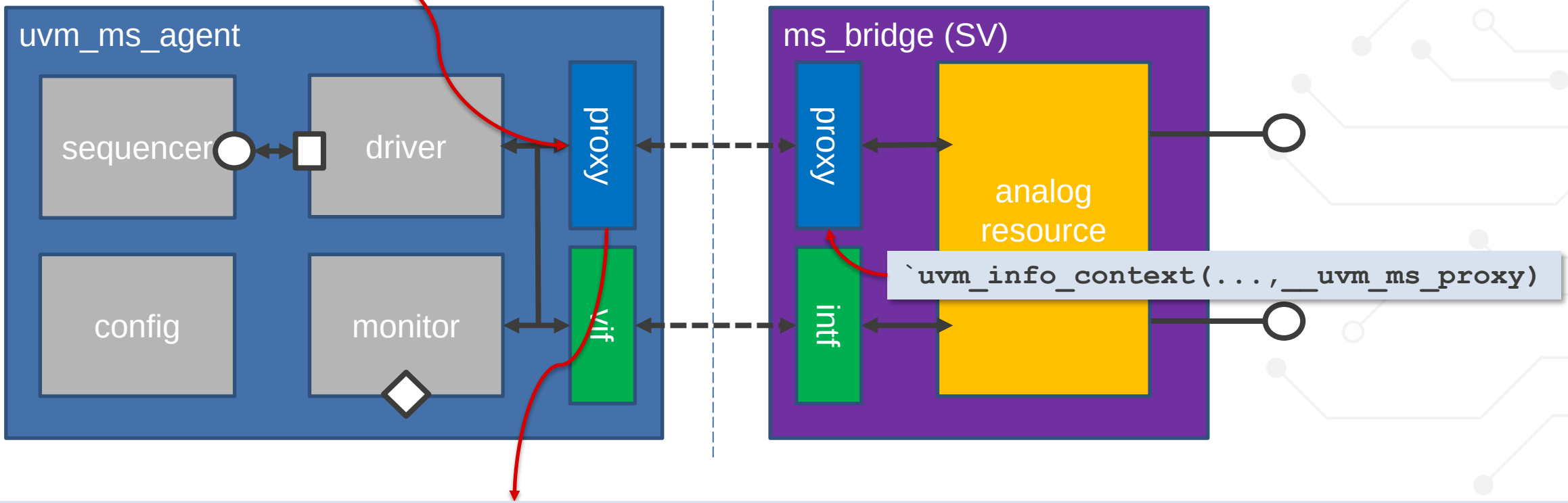
- Why UVM-MS
- Verilog-AMS Simulator DC OP / Transient behavior
- UVM MS Bridge to analog resource (UVM->AMS/DMS Connection)
- UVM-MS Phasing Requirement
- UVM messaging from AMS files and $root cells

UVM MESSAGING REQUIREMENT

- Need to filter and control generation of messages from analog resource
- UVM offers this control for components in UVM hierarchy
- But analog resource is not part of the UVM component hierarchy. It's a module!
- However, if we extend the MS proxy from *uvm_report_object*
  - set_report_handler() can redirect handling to the enclosing UVM MS monitor
  - messages from MS bridge (and below) can use the proxy context
  - `uvm_info_context(. , . , . , ***ro***) takes reporting object to provide context
  - Messaging macros called from analog resource can use upscoping (see later)
  - Recommend to include %m in the UVM message body to get a physical path.

MS MESSAGING CONTEXT

```
virtual function void uvm_ms_monitor::connect_phase(uvm_phase phase);
  ...
  proxy.set_report_handler(get_report_handler);
endfunction : connect_phase
```




```
UVM_INFO @ 0.000 ns: uvm_test_top.env.ms_agent.monitor [MS_MONITOR] Incorrect bias voltage: 0.125V
```

UVM MESSAGE REPORTING FROM ANALOG RESOURCE.

- UVM reporting system designed for class based structure registered with uvm_report_object
- UVM reporting macros not supported in Verilog-AMS modules.
  - Lets use the up-scoping system to solve this for us. (LRM 6.8)

- *`include "uvm_ms.vamsh"* in Verilog-AMS file (analog resource)
  - localparams to define UVM Verbosity levels as integers to match UVM enum's
  - Provide macro's *`uvm_ms_[info|warning|error|fatal](…)* to call function in MS bridge
- *`include "uvm_ms.vdmsh"* in SystemVerilog file (analog resource) don't forget to import uvm_pkg!
  - Provide macro's *`uvm_ms_[info|warning|error|fatal](…)* to call function in MS bridge
- *`include "uvm_ms.svh"* in SV file (MS Bridge)
  - Void functions that wrap `uvm_ms_*() reporting macros into functions of the same name.
  - Provide macro's *`uvm_ms_[info|warning|error|fatal](…)*

- Within analog block, many solutions so here is one (calling of digital functions not allowed.)
  - Use absdelta to trigger on toggle and read string to call up-scoping function.

UVM MESSAGE – VERILOG-AMS ANALOG BLOCK

Example – many other ways

```
analog begin
   if((I_PLUS > 1.0) && !I_thr_triggered) I_thr_triggered = 1;
   else if(I_PLUS < 0.9) I_thr_triggered = 0;
end

//Convert the detection in the analog block to a UVM report.
string message;
always@(absdelta(I_thr_triggered,1,0,0,1)) begin
   $sformat(message,"The Current is above the thresholds @ %e",I_PLUS);
   if(I_thr_triggered) `uvm_ms_info(P__TYPE,message,UVM_MEDIUM)
end
```

Upscope function call

- Use analog domain to detect the issue and toggle a integer.
- Integer is detected by absdelta to then report the message via the Digital Engine.
- Note this will not be reported if there is a convergence failure!

UVM MESSAGE

Analog Resource

```
string message;
...
$sformat(message,"The Current is above the threshold @ %eA",I_PLUS);
`uvm_ms_info(P__TYPE,message,UVM_MEDIUM);
```

SV Bridge

Hence the proxy name requirement!

```
function void uvm_ms_info(string id, string message, int verbosity_level, string uvm_path);
`uvm_info_context(id,message,uvm_verbosity'(verbosity_level),__uvm_ms_proxy)
endfunction: uvm_ms_info
```

- Use *_context reporting macros to direct message to relevant component

```
UVM_INFO ../../includes/uvm_ams.svh(26) @ 52001.098068ns: uvm_test_top.env.v_agent [vdriver]
The Current is above the threshold @ 1.178812e+00A
```

UVM AMS REPORTING ISSUES

- uvm_*_printer print_real() uses %f formatting which truncates very small values
- Propose change to UVM implementation otherwise override print_real() in chosen printer

%f

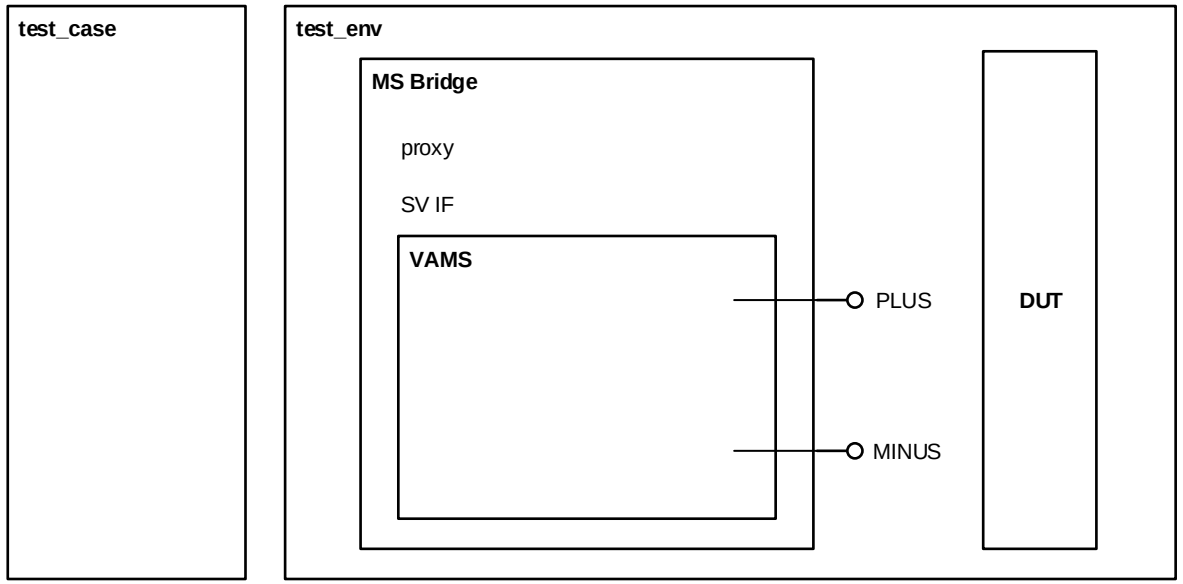
```
----------------------------------------------------
Name          Type      Size  Value
----------------------------------------------------
cap_txn       cap_txn   -     @4931
  volts       real      64    0.000000
  amps        real      64    0.000000
  rseries     real      64    100.000000
  rparallel   real      64    10000000000000000.000000
  farads      real      64    0.000000
  henrys      real      64    0.000000
----------------------------------------------------
```

%e

```
-----------------------------------
Name          Type      Size  Value
-----------------------------------
cap_txn       cap_txn   -     @4917
  volts       real      64    0
  amps        real      64    0
  rseries     real      64    100
  rparallel   real      64    1e+15
  farads      real      64    1e-09
  henrys      real      64    0
-----------------------------------
```

- Timestamps with limited precision in UVM-1.2 onwards
- compose_report_message() in uvm_report_server uses $time not $realtime
- Issue in UVM (MANTIS 0005807) workaround is to override report server

RECOMMENDED MS SETUP



- Two $root cells
  - test_case to encapsulate the stimulus generation SV Class based world.
  - test_env to encapsulate the physical environment of the PCB components/DUT.

- test_env is independent of how the test_case is setup enabling a DMS/AMS class based environment.
-
- test_env could be a schematic and analog resources drawn or text view
  - Proposed system allows parameters to be used.
  - Proposed system worth hardware accelerators for the physical design.

PROVIDED PACKAGES/INCLUDE FILES

Statement	Usage
import uvm_ms_pkg::*;	Within the MS Bridge and uvm_ms_agent.
`include “uvm_ms.vamsh”	For Verilog-AMS modules defined as the analog_resource or hierarchy.
`include “uvm_ms.dmsh”	For SV modules defined as the analog_resource or hierarchy. E.g. The Verilog-AMS file instance in SV module, this `include would allow the SV module to use the same messaging system.
`include “uvm_ms.svh”	For inclusion in the MS Bridge to enable the commincation from the analog_resources. It requires the MS Proxy instance is named __uvm_ms_proxy.

Renesas.com

LOGIC CONVERSION WITHIN ANALOG RESOURCE

- Logic Conversion needs reference VDD/VSS levels.
  1. Dynamic tracking
     1. Dedicated pins REF_VDD/REF_VSS in MS_BRIDGE/Analog Resource.
     2. OOMR using analog_node_alias() and parameters for AMS only.
        - Can only be parameters as this is setup pre DC OP.
        - Ideally it would use ref_vdd/vss but alias to port is not allowed. (Verilog-AMS LRM 9.20)
  2. real values set like other controls in the MS Bridge to the analog resource.

```
analog initial begin
  if((P__VDD_PATH != "NOT_VALID") && ($analog_node_alias(REF_VDD_INT, P__VDD_PATH) == 0))
          $error("Unable to resolve power supply: %s", P__VDD_PATH);
  if((P__VSS_PATH != "NOT_VALID") && ($analog_node_alias(REF_VSS_INT, P__VSS_PATH) == 0))
          $error("Unable to resolve ground supply: %s", P__VSS_PATH);
  ...
end
analog begin
  if(use_fixed_supply) logic_supply = a2d_supply * logic_tran;
  else if(P__VDD_PATH != "NOT_VALID") logic_supply = V(REF_VDD_INT,REF_VSS_INT);
  else logic_supply = V(REF_VDD,REF_VSS);
  ...
end
```

OOMR Paths

Voltage Selection

ACCELLERA

accellera SYSTEMS INITIATIVE

- Independent organization founded in 2000
- Mission to collaborate to innovate and deliver global standards that improve design and verification productivity
- Partnership with IEEE for formal standardisation & governance
- Renesas has representatives on many working groups, including UVM, UVM-MS, SV-AMS, SystemC