Assertion-based Verification for Analog and Mixed Signal Designs

Srinivas Aluri
Mixed Signal DV Manager.
Introduction

• Assertion-based verification approaches have been making great strides into mixed signal verification environments.

• By using vunit, verification engineers can bind assertions to abstract models of analog or interactions between digital and analog models.

• There are scenarios where an analog block is not modelled and is run in schematic form and verification checks needs to be performed in a complete analog domain.

• The work shown in this paper specifically targets such scenarios and shows how assertion techniques can be used for verifying analog parameters.
Block diagram showing binding assertions
```
// Vunit declaration with module name based binding to PM_TOP
vunit pm_top_vunit(PM_TOP) {
    // declaring ports of PM_TOP explicitly of type electrical on which assertions are coded
    electrical REF_1p2, REF_2p5, REF_2p5_DAC, REF1p2_HV, HV_REF_0p6, DVDD_OUT;
    // internal nets can also be included in assertions but they need to be ports of internal hierarchial cells
    electrical IREF_GEN.VREF_2p5V, IREG_DVDD.EN, IVREF_IB_GEN.SD, IREF_GEN.IPTAT_1U, IVREF_IB_GEN.IPTAT_1p6u;

    // Checking if Analog signal is in expected range and converting it into a digital signal
    // Converting REF_1p2 -> ref_1p2_in_range which can be used in assertion.
    'V_IN_RANGE(REF_1p2, high_thr_limit, low_thr_limit, vdelta, ttol, vtol, enable_v_assert_trigger, ref_1p2_in_range)

    // Checking if Analog signal is in expected range and converting it into a digital signal
    // Converting curr_iztc_500n -> curr_iztc_500n_in_range which can be used in assertion.
    'I_IN_RANGE(curr_iztc_500n, high_thr_limit, low_thr_limit, idelta, ittol, ittol, enable_i_assert_trigger, curr_iztc_500n_in_range)

#define curr_iztc_500n
#define get_abs($cgav("<Hierarchical_OUT_path>.IZTC_500n","flow"))
```
Domain conversion macro code

Voltage Conversion

`define V_IN_RANGE(Node, hthr, lthr, vdelta, ttol, vtol, enable, digout) \ 
always @(absdelta(V(Node), vdelta, ttol, vtol, enable)) begin \ 
if((V(Node) > hthr) || (V(Node) < lthr)) digout = 0; \ 
else digout = 1; \ 
end

Current Conversion

`define I_IN_RANGE(exp, hthr, lthr, edelta, ttol, etol, enable, digout) \ 
always @(absdelta(exp, edelta, ttol, etol, enable)) begin \ 
if((exp > hthr) || (exp < lthr)) digout = 0; \ 
else digout = 1; \

Example -1 Assertion Code

Voltage Assertion

Current Assertion
Example-2 Assertion Code

```verilog
// Checking Decode Test mux control signal is Asserted and is in required voltage range
`V_IN_RANGE(EN_TOUT_2p5_BUF,top.vddp5v_lith,top.vddp5v_lith,top.rddp5v_typ + `v tol),enable_v_assert_trigger,mux_out_en_1)

// Create a tmuxout valid signals based on condition relevant to the design
always @(*)
begin
  check_tmux_out_valid = vddp5v_in_range && porb_release && out_of_sd && tmux_out_en;
end

// Digitizing expected tmux out value for specific mux selection which will be used in Assertion
always (TOUT_MUX_CTRL) begin
  ...
  ...
  case(tmux_ctrl)
    5'd0 : ...
    5'd16 : tmux_out_ref_2p5 = `get_abs(V(TOUT_REF_2p5_BUF) - 2.5) < 0.01) ? 1:0;
    5'd3 : ...
  endcase

// TMUX out Assertion
// CHECKER: when SD is released and TMUX_OUT is enabled with TMUX_OUT ctrl 10000 then TMUX_OUT gets REF_2p5_BUF with TMUX_OUT_EN<1> high
// (!) tmux_out_en_i asserted: assert always ((check_tmux_out_valid && (tmux_ctrl == 5'd16)) -> (tmux_out_en_i))
@posedge(nux_ctrl_change_dly);
// (!) tmux_out_i asserted: assert always ((check_tmux_out_valid && (tmux_ctrl == 5'd16)) -> (tmux_out_i_ref_2p5))
@posedge(nux_ctrl_change_dly);
```

Example code showing Analog Output Testmux Assertion.
Analog value fetch PSL V/s SVA

Example code showing analog value fetch and usage in PSL and SVA assertions

Topology for using SVA assertions

SVA Assertions topology

Top.vams:
Module top ()

//Instantiate SVA module
sva_module sva_module_i (....);
endmodule

//SVA assertions module
module sva_module(....);
real analog_v, analog_i;

analog_v = $cgav("signal_path", "potential");
analog_i = $cgav("signal_path", "flow");

// SVA assertion using above real variables
assert property (@(trig) (cond a) |-> (analog_v > vth));
endmodule
Integration of PSL assertions into virtuoso environment

AMS option form in ADE showing arguments to pull in assertion files

Code showing how to pull in multiple assertion files

// To pull in multiple assertion files with one common file
-assert-propfile_vlog cell1.psl
-assert-propfile_vlog cell2.psl
-assert-propfile_vlog cell2.psl
Integration of SV assertions into virtuoso environment

Pulling in top.vams which contains sva module

Declaring parallel top level hierarchy
Simulation Results

Simulation results showing current and voltage assertion checks
Assertion browser showing assertion statistics.