



JUNE 23-27, 2024

MOSCONE WEST CENTER
SAN FRANCISCO, CA, USA



Use UVM for Analog Mixed Signal (AMS) DFT through IEEE 1687 Procedural Description Language (PDL)

Geert Seuren, Hitu Sharma, Rahul Lodwal
Chief Technology Office, NXP Semiconductors

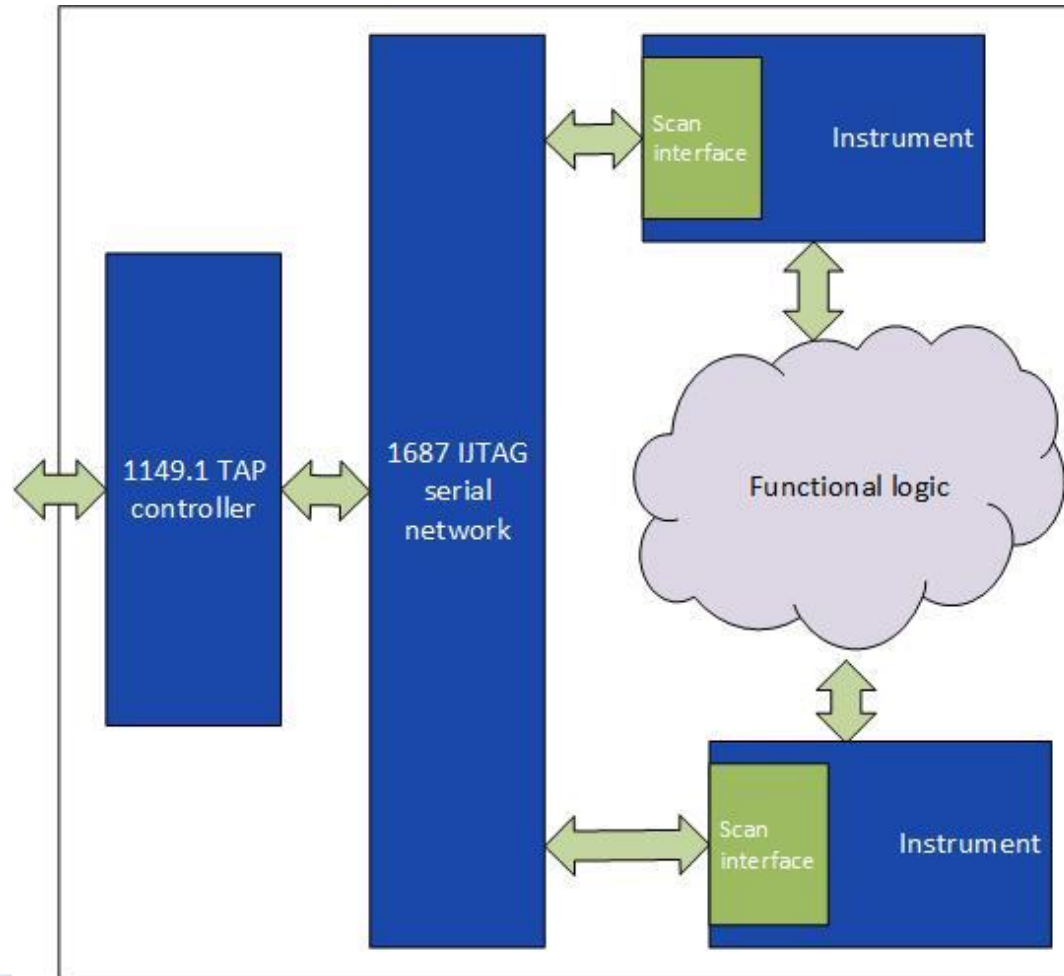


Agenda

- Current state of IEEE1687 standard and limitation
- Existing flow for development of DFT test cases
- Challenges for AMS DFT verification
- Scope of AMS DFT innovation Flow
- Proposed Verification flow for AMS DFT testcases
- Digital Verification environment brief introduction
- Experiments/Results from SoC
- Conclusion

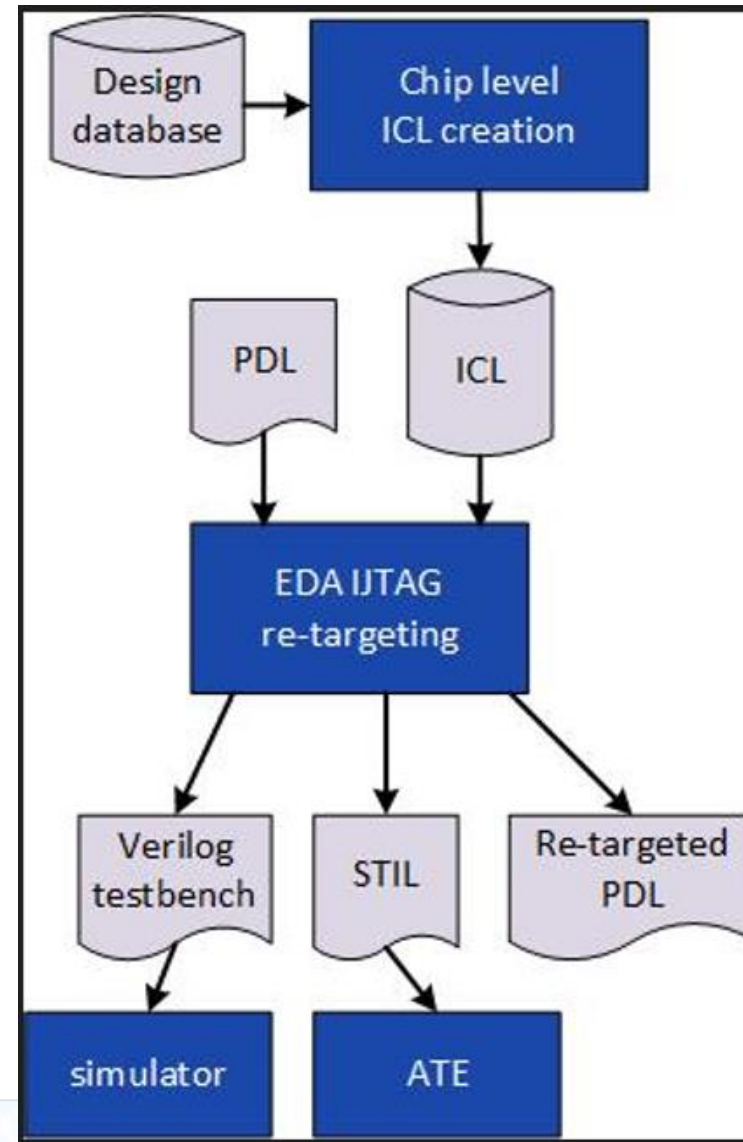
Current State of IEEE1687 Standard & Limitations

- IEEE 1687-2014 standard released in 2014, often referred to as IJTAG
 - instrument with scan interface
 - connected via IEEE1687 serial access network
 - chip level interface
- Formalized language for
 - test hardware description: Instrument Connection Language (ICL)
 - documenting operation of instrument: Procedural Description Language (PDL)
- An instrument level PDL test description can be retargeted to the chip level boundary
- Limitations:
 - only **digital signals** can be described and retargeted
 - **IEEE1149.1 controller** is the only supported chip level interface



Existing IEEE1687 DFT Test Development Flow

- Existing flow consist of three parts
 - Creation of chip level ICL
 - propriety tool which traces the design netlist
 - using DfT abstraction of DfT infrastructure in propriety format
 - translates result into ICL
 - Re-targeting
 - design rule checking
 - re-targeting to chip level
 - write out results as
 - retargeting result in STIL format
 - (retargeted) PDL at JTAG transaction level
 - Verilog test bench



Challenges with AMS DFT Verification Flow

Limitations of IEEE1687 A.k.a IJTAG

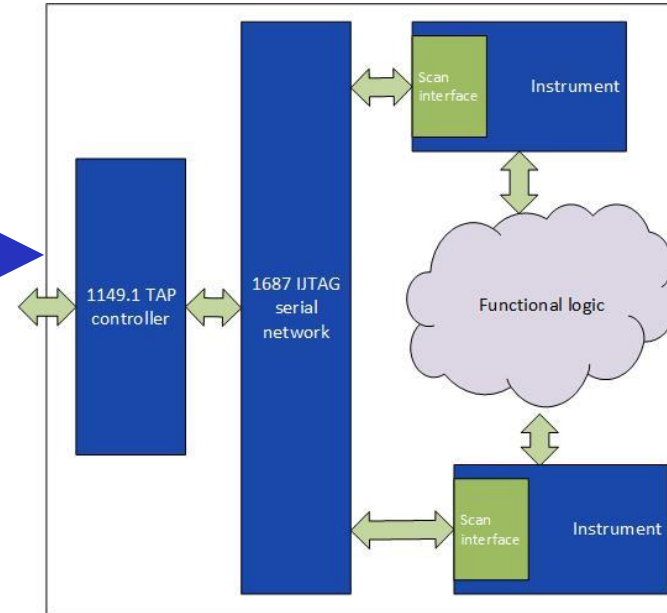
- IJTAG supports JTAG as a chip level test interface and can only describe control and observe signals in digital format.
- Extension towards analog interface signals, IEEE1687.2 is under development

Lack of EDA support /automation for AMS DFT

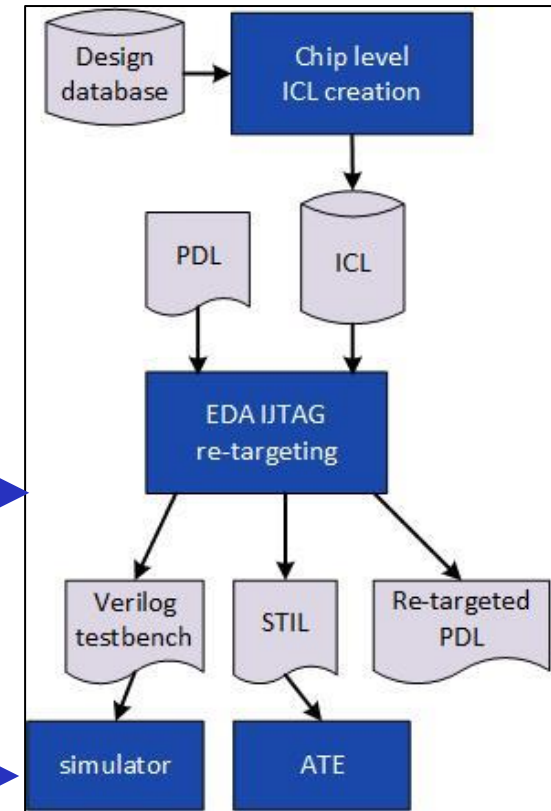
- AMS testing is often done in a functional way although many research and efforts are made to move it to a fault model-based testing as it is already done for digital domain since years
- AMS DFT engineers create their own verification environment therefore AMS DFT verification is prone to human errors.

Uniform verification methodology for AMS DFT

- Existing EDA support is for generation of Verilog test cases for the Digital DFT logic
- Traditional verification technique using Verilog lacks flexibility of reusability and fast Time to Market (TTM)



Current state of IEEE 1687



Current state of DFT Verification Flow with EDA

Scope of AMS DFT Verification Flow



Script

IEEE 1687 lacks support for analog DFT

Approach :- *In house Analog extension of IEEE1687*



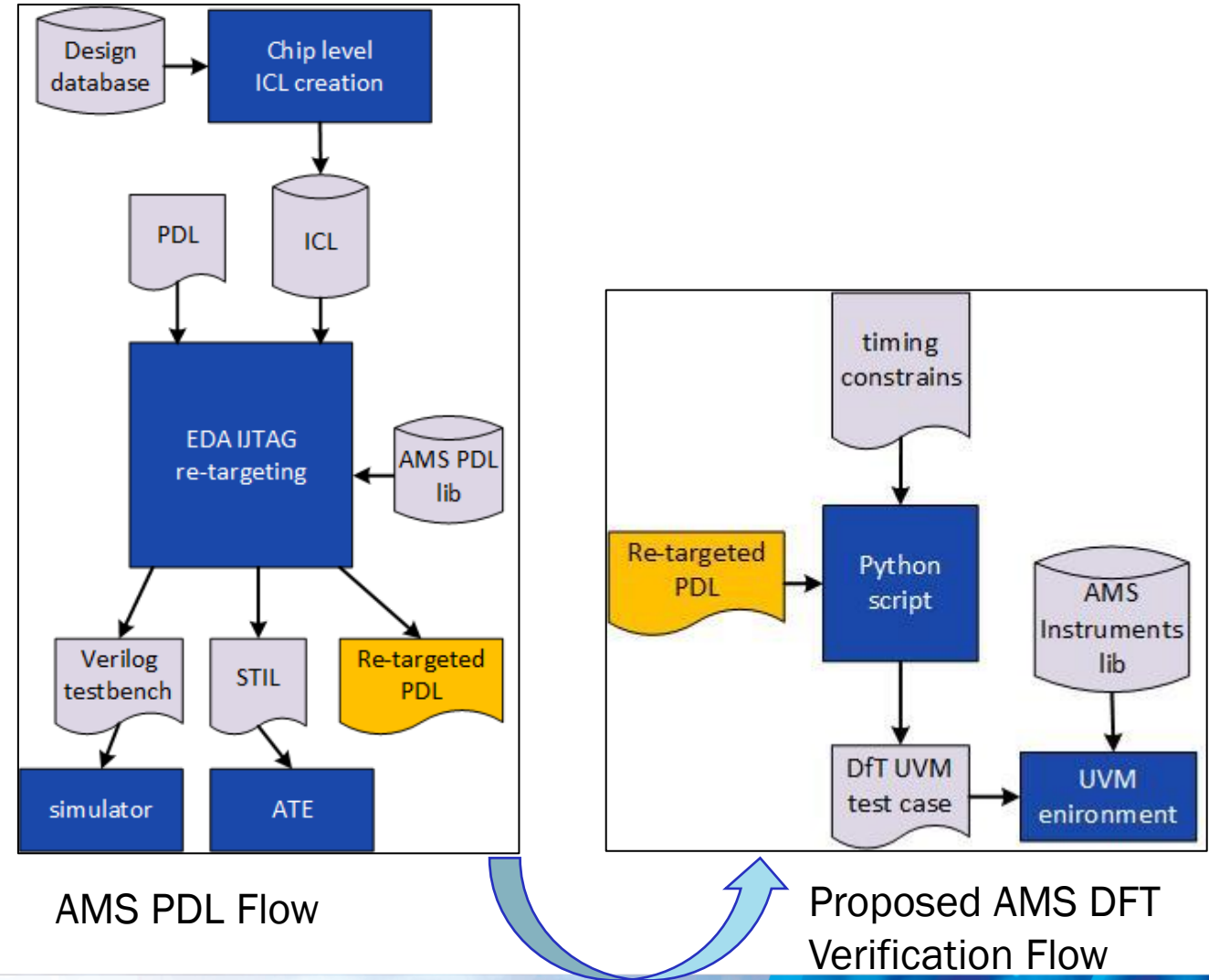
UVM

Unified verification flow for AMS DFT test cases

Approach :- *Reuse retargeted PDL for creating UVM test cases*

Main Idea- AMS DFT Verification Flow

- JTAG can describe DFT control and observe signals in ICL which is a digital only format.
- AMS PDL library has been implemented to enable analog signal source and measurement functions shown in Fig 3.
- The test case described in PDL is at IP level and shall be retargeted to chip level by EDA tool.
- Reuse retargeted PDL's to generate DFT UVM test cases using a python script (in house development), shown in Fig4.
- The script executes JTAG FSM based on the timing constraints fed to script.
- With the script we incorporated all the digital test case information through JTAG transactions and Analog part is fed through AMS Instrument library using analog pragma.
- The analog pragma is interpreted as events/tasks by the script.
- UVM environment uses AMS DFT library for the AMS DFT test cases.



AMS PDL File Example

```

iProc bandgap_voltage_typ {} {;

iWrite reg_map_top.testchip_ctrl.TEST_CTRL.TEST_BANDGAP_DEM_EN_CTAT 1
iWrite reg_map_top.testchip_ctrl.TEST_CTRL.TEST_BANDGAP_DISABLE 0
iWrite reg_map_top.mtp.BANDGAP_LDO_TRIM.BANDGAP_TRIM      0x1F
iWrite reg_map_top.mtp.BANDGAP_LDO_TRIM.BANDGAP_TRIM_TC    7
iWrite reg_map_top.testchip_ctrl.ATB_SELECT.ATBP_SELECT     0x0E
iWrite reg_map_top.testchip_ctrl.ATB_SELECT.ATBN_SELECT      0x00
iApply;
iRunLoop 1000 -tck;

#           pin_pos pin_neg num_averages  high_limit low_limit
MeasureVoltage ( ATBP   VSS           1           0.718   0.717 )

}

```

AMS PDL

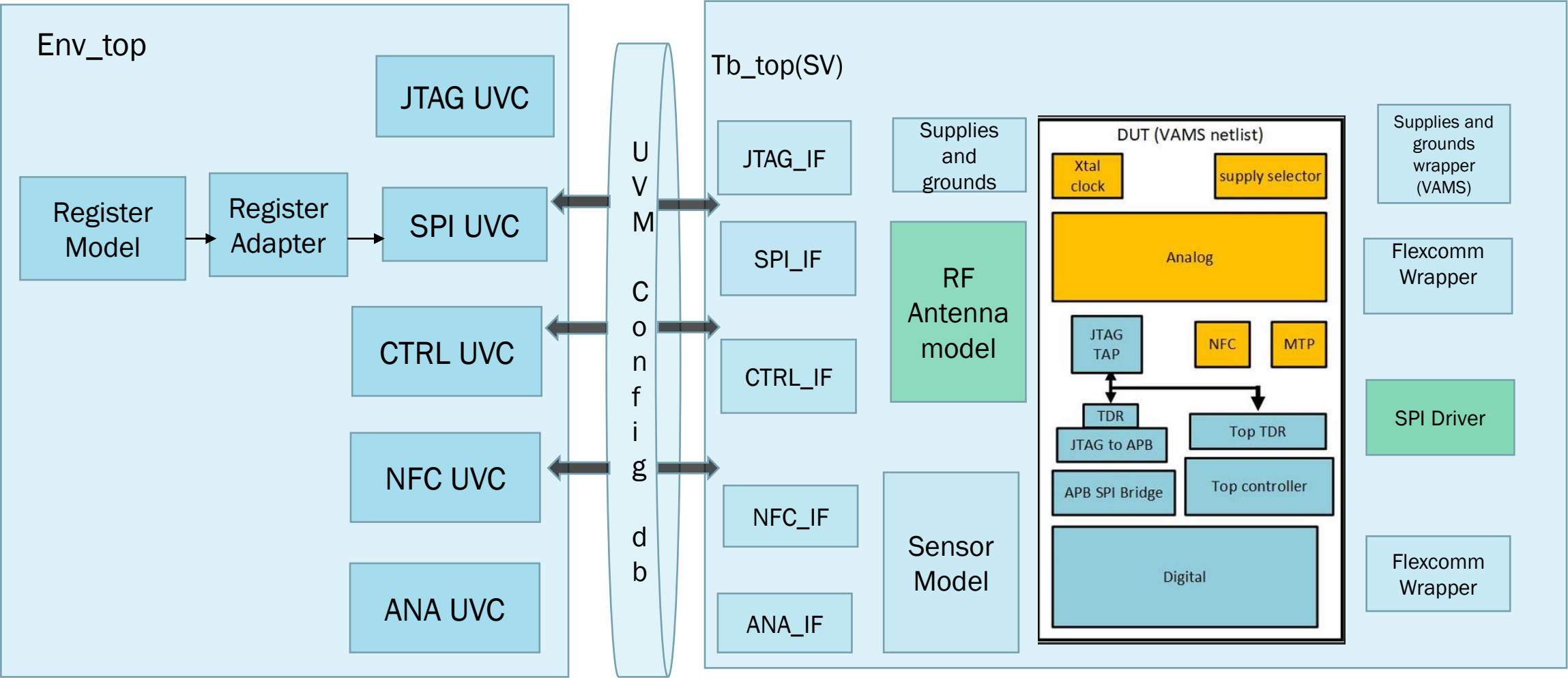
[illegible]

AMS retargeted PDL

Targeted SoC

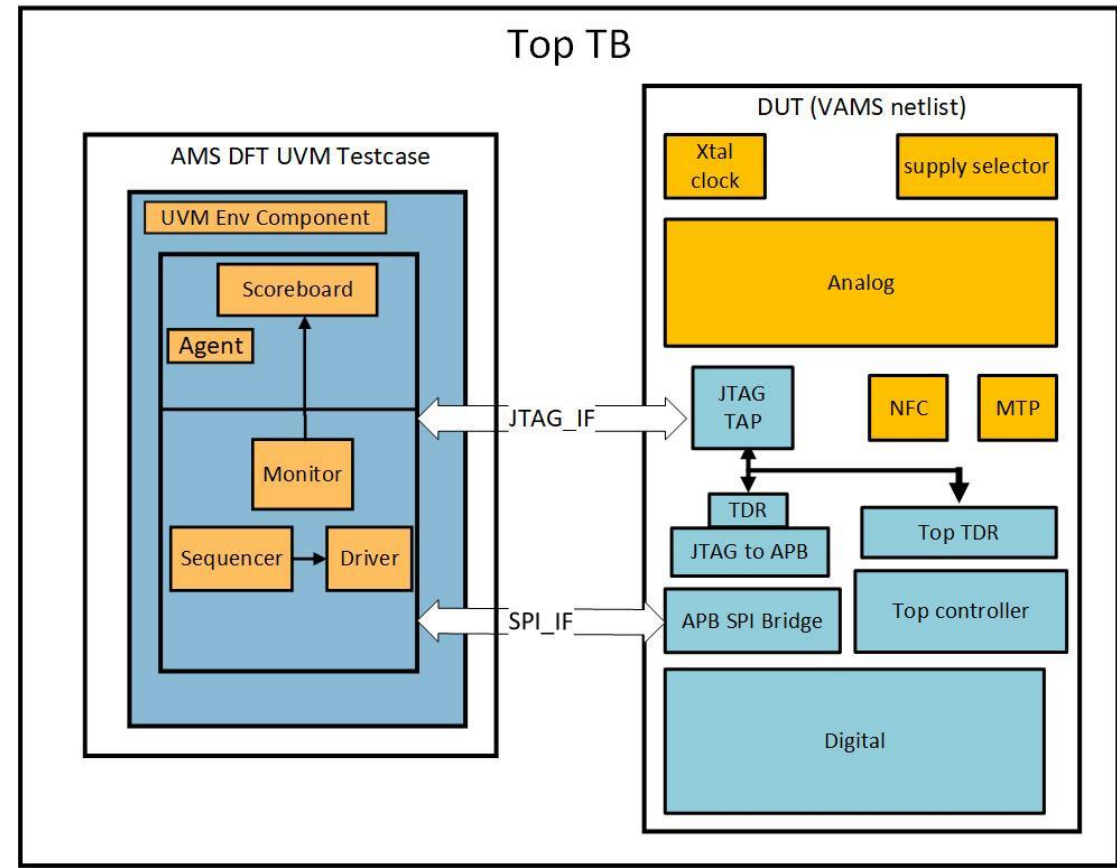
Tests

Sequence of Sequences(Virtual Sequencer)



UVM Verification Environment

- UVM offers flexibility to reuse test bench environment and fast TTM.
- UVM based functional verification tries to achieve maximum code coverage by adding more test scenarios.
- Objective is to integrate AMS DFT test cases in UVM environment and get the higher coverage results.
- UVM wrapper provides a standard methodology for the JTAG controller and connects it to the UVM testbench using the UVM configuration database.
- At the top-level TB, the DUT is instantiated along with other UVM related packages, and verification components.
- The JTAG sequencer generates different sequences that are passed through the sequencer. The components driver, sequencer, monitor are encapsulated under agent.
- Added JTAG agent to run the DFT test cases in the UVM environment.
- The UVM environment is dominantly digital in nature and the analog information of the test case is available in AMS instrument library, which we have included in the UVM environment.

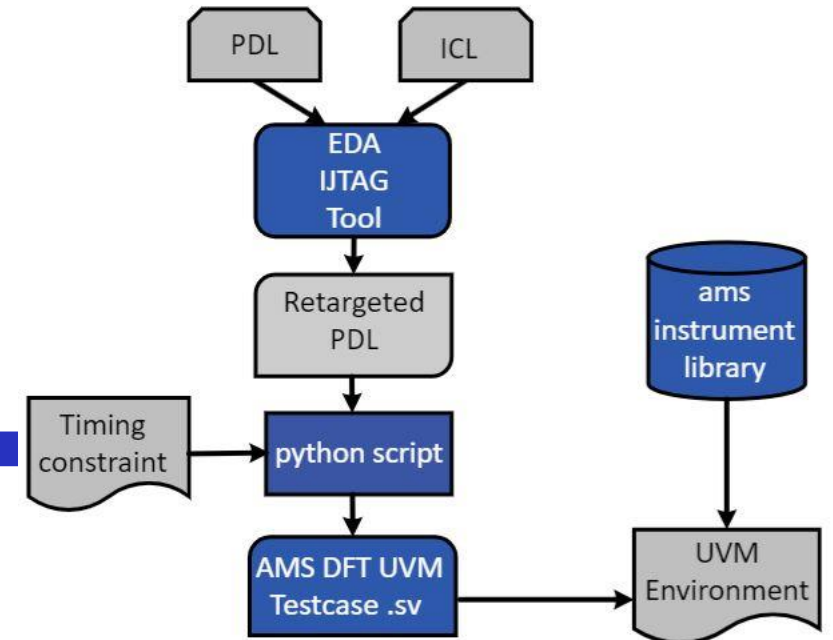


UVM Verification Environment

Final AMS DFT UVM Test Case

```
// #####  
// t_dft_bandgap_bandgap_voltage_min  
  
// -  
//  
// #####  
`ifdef AMS_SIM  
`else  
`define dft_wreal_wrapper_path    uhp_sapphire_top_tb.dut_wrapper_u.dft_wreal_wrapper_u0  
`endif  
  
class t_dft_bandgap_bandgap_voltage_min extends test_base;  
    `uvm_component_utils(t_dft_bandgap_bandgap_voltage_min)  
  
    //=====  
    // Constructor.  
    //=====  
    function new(string name, uvm_component parent);  
        super.new(name, parent);  
    endfunction : new  
  
    //=====  
    // Run phase.  
    //=====  
    task run_phase(uvm_phase phase);  
  
        `ifdef AMS_SIM  
        `else  
            uvm_status_e    status;  
            uvm_reg_data_t  read_reg_data;  
            uvm_reg_data_t  write_reg_data;  
            reg[1000:0] bits_to_write;  
            reg[1000:0] bits_to_read;  
            int iter;  
  
            phase.raise_objection(this);  
  
            //Code to run test:bandgap_bandgap_voltage_min  
  
            $timeformat(-9,0,"");  
            #1000 `uvm_info(get_type_name(),$sformatf("Starting with test setup"),UVM_NONE)  
            ijtag_enable();
```

AMS DFT logic





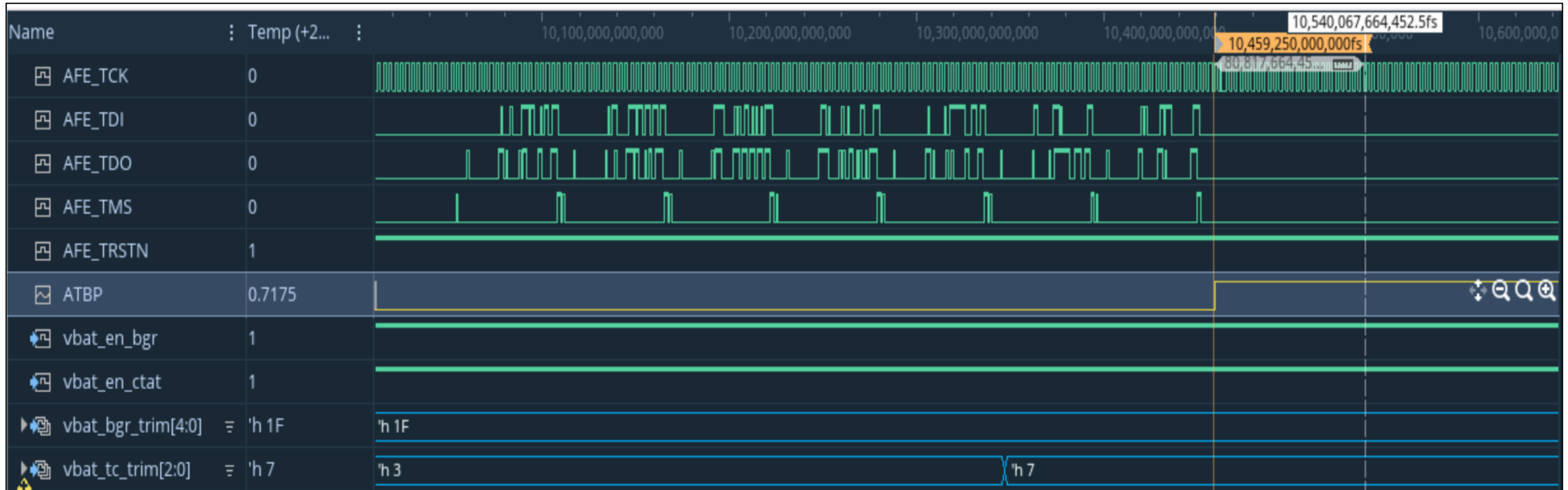
JUNE 23-27, 2024

MOSCONE WEST CENTER
SAN FRANCISCO, CA, USA

Evidence and Results



Evidence - Simulation Output - AMS UVM DFT Test Case



Code Coverage Results from SoC

- Table 1 shows few design blocks where row 1 and 2 are specific DFT circuitry
- Adding the DFT test cases to the code coverage shows improvement in the code coverage results.
- DFT test cases of SoC were developed to check whether a certain test could be performed or to check that in test mode, user has the required access to embedded (test) nodes.
- DFT test cases **were not** developed to gain max RTL code coverage as this kind of code coverage analysis is not common for most DFT engineers.
- With the recommended flow the quality of the Functional as well as DFT Verification activity can greatly improve.

Sr.no	Blocks	Coverage before(in %)	Coverage After(in %)
1	Scan_top_tdr	17.93	35.15
2	Scan Mux	29.83	45.32

Table 1 - Code coverage results of SoC

Conclusion

- With the proposed AMS DFT Verification flow we have included AMS DFT test cases to UVM Verification environment using IEEE 1687 Procedural Description Language. This flow has clear benefits:
 - We could reuse the existing UVM verification environment to simulate AMS DFT test cases.
 - It has bridged the gap between DFT and Verification flow and will further improve the collaboration between DFT and Verification disciplines.
 - It will improve the overall code coverage figure as AMS DFT circuitry is included in the analysis.
- Adding AMS DFT test cases inside the UVM environment opened possibilities to improve overall quality of the pre-silicon chip verification.
- This is a collaborative work done by AMS DFT verification engineer and Digital verification teams which has given us edge with TTM.
- As IEEE-P1687.2 soon will be available we hope EDA vendors will enable such flow