The How To's of Advanced Mixed-Signal Verification

John Brennan, Thomas Ziller, Kawe Fotouhi, Ahmed Osman

Cadence Design Systems

2015 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

Agenda

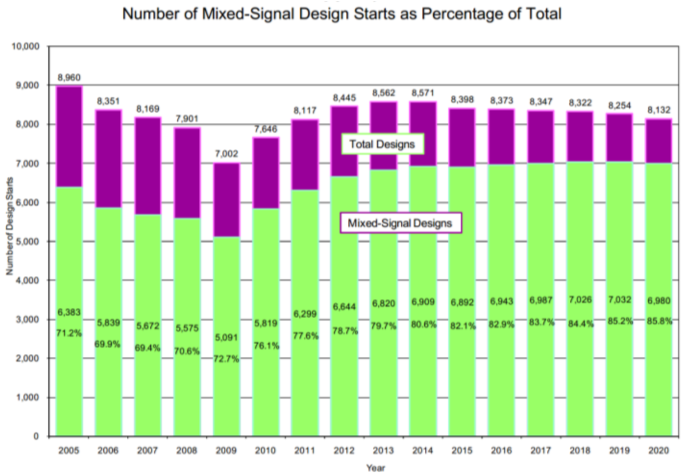
1. **Metric-Driven Verification for MS**
2. Verification Planning and Management in MS
3. Universal-Verification Methodology for MS
4. Real-number Modeling Capabilities
5. Analog and MS Assertions
6. Q&A

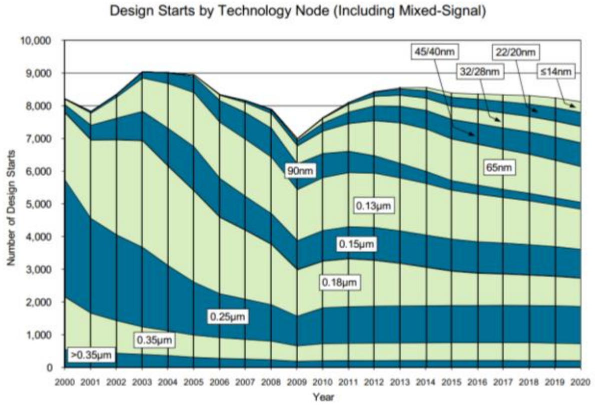
Metric-Driven Verification for Mixed-signal

John Brennan

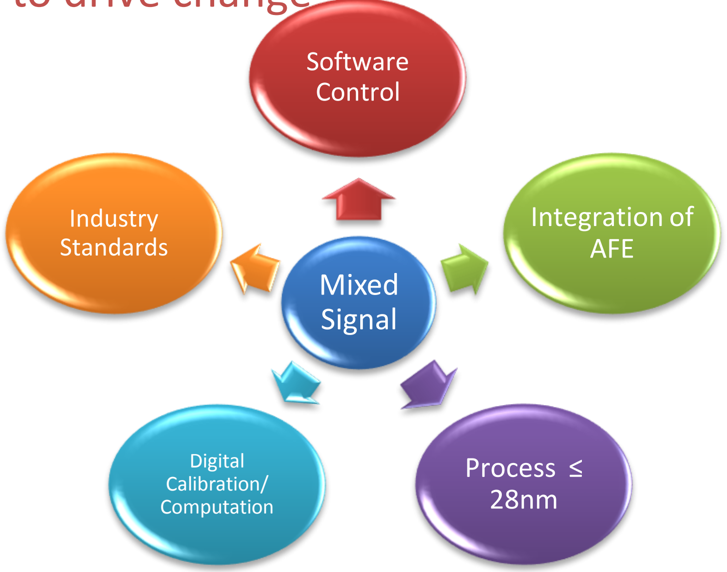
The Winds of Change

Many forces at work to drive change


Number of Mixed-Signal Design Starts as Percentage of Total


Design Starts by Technology Node (Including Mixed-Signal)

Source: IBS



TIPPING POINT INDICATORS

- Digitally-calibrated, compensated
- Feedback between D and A
- Software controlled
- Power management
- 28nm and below
- Long Verification Cycles

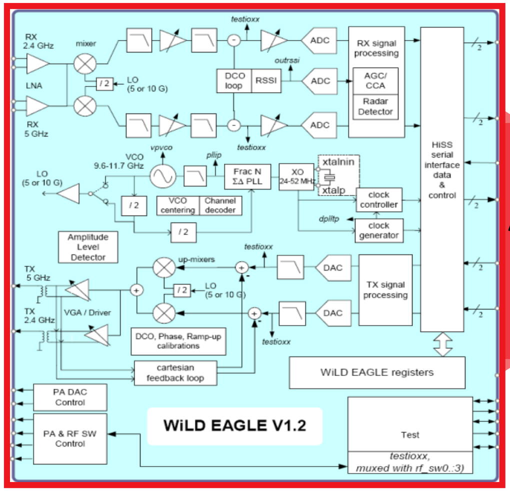
Mixed-signal Verification: Complexity Issues

How do I build consistency between digital and analog teams?

Mixed-signal DUT &
Verification test bench environment

How do I verify the digital content in this SoC?

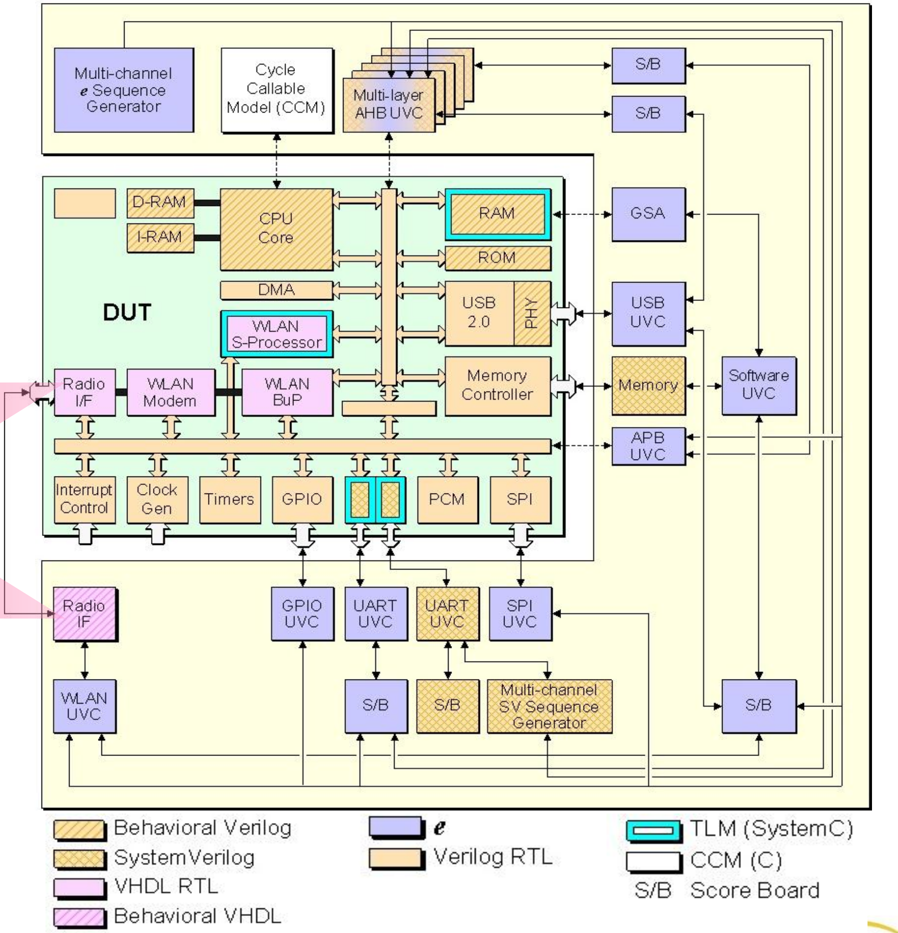
How do I verify the mixed-signal interconnects?



Analog Design

Design

Measure



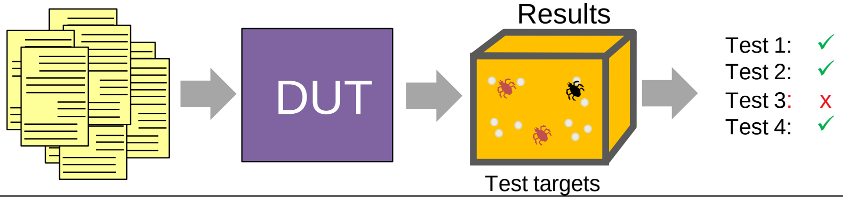
How do I abstract analog behavior?

How do I verify the mixed-signal IP?

Advanced Verification Methodology

Functional Verification Approaches

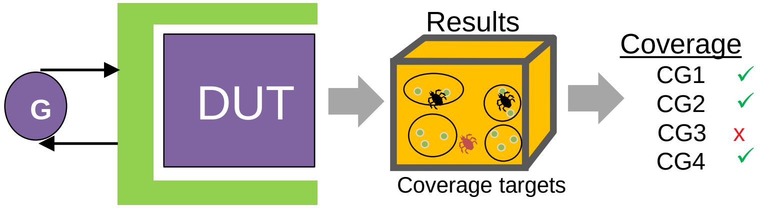
Directed Tests Driven



Test 1: ✓
Test 2: ✓
Test 3: ✗
Test 4: ✓

Least effective in finding the hidden bugs

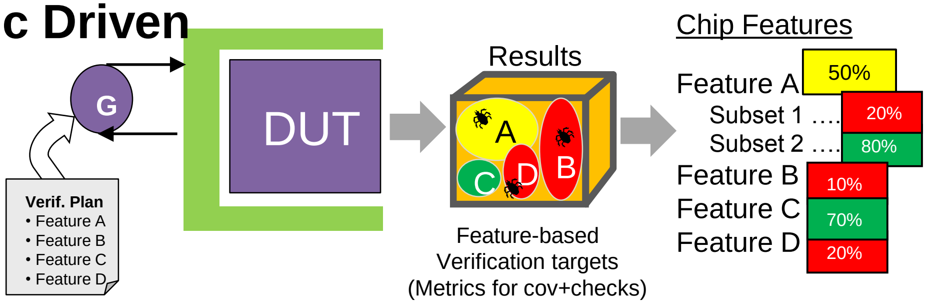
Coverage Driven



Coverage
CG1 ✓
CG2 ✓
CG3 ✗
CG4 ✓

Adds quality & productivity, but difficult to estimate completion

Metric Driven



Verif. Plan
- Feature A
- Feature B
- Feature C
- Feature D

Feature-based Verification targets (Metrics for cov+checks)

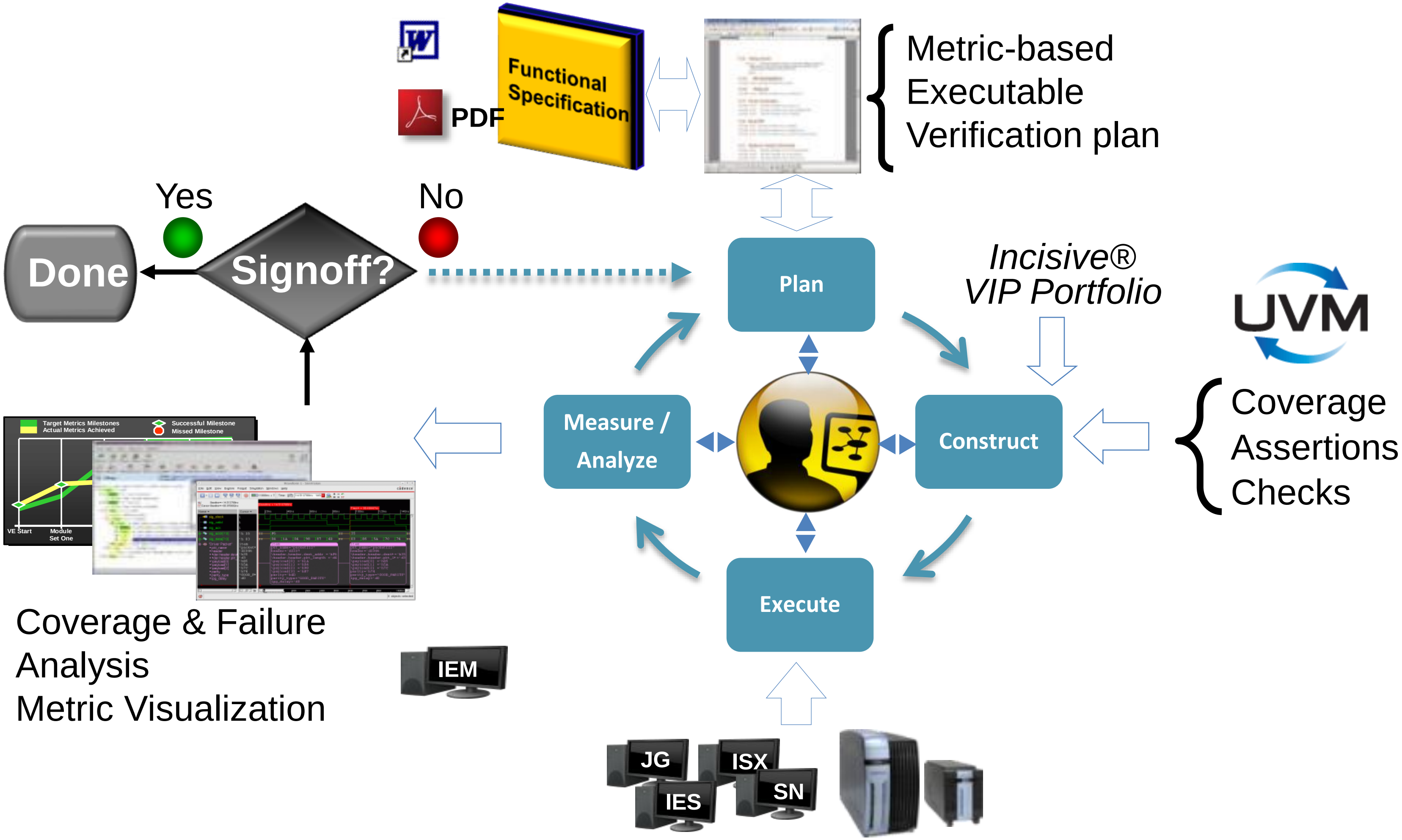
Chip Features

Feature A	50%
Subset 1 ….	20%
Subset 2 ….	80%
Feature B	10%
Feature C	70%
Feature D	20%

Feature-based plan with extended metrics enables efficient and accurate project closure

Metric Driven Verification (MDV): Overview

Planning with unified verification metrics



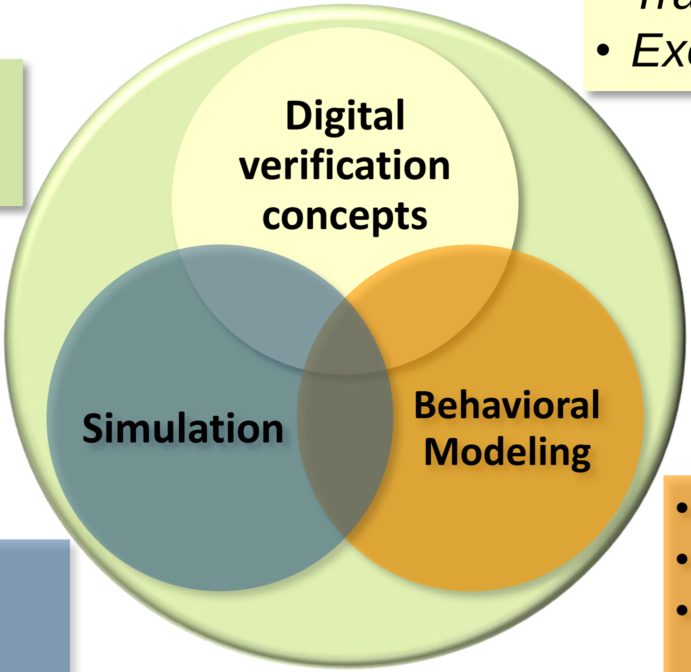
Metric-based
Executable
Verification plan

Coverage
Assertions
Checks

Coverage & Failure
Analysis
Metric Visualization

Testbench Simulation,
Formal,
HW/SW Co-Sim, LPV, MSV,
Sim-Acceleration, Emulation

Key Elements of MS Verification Solution



- *Integrated Environment*

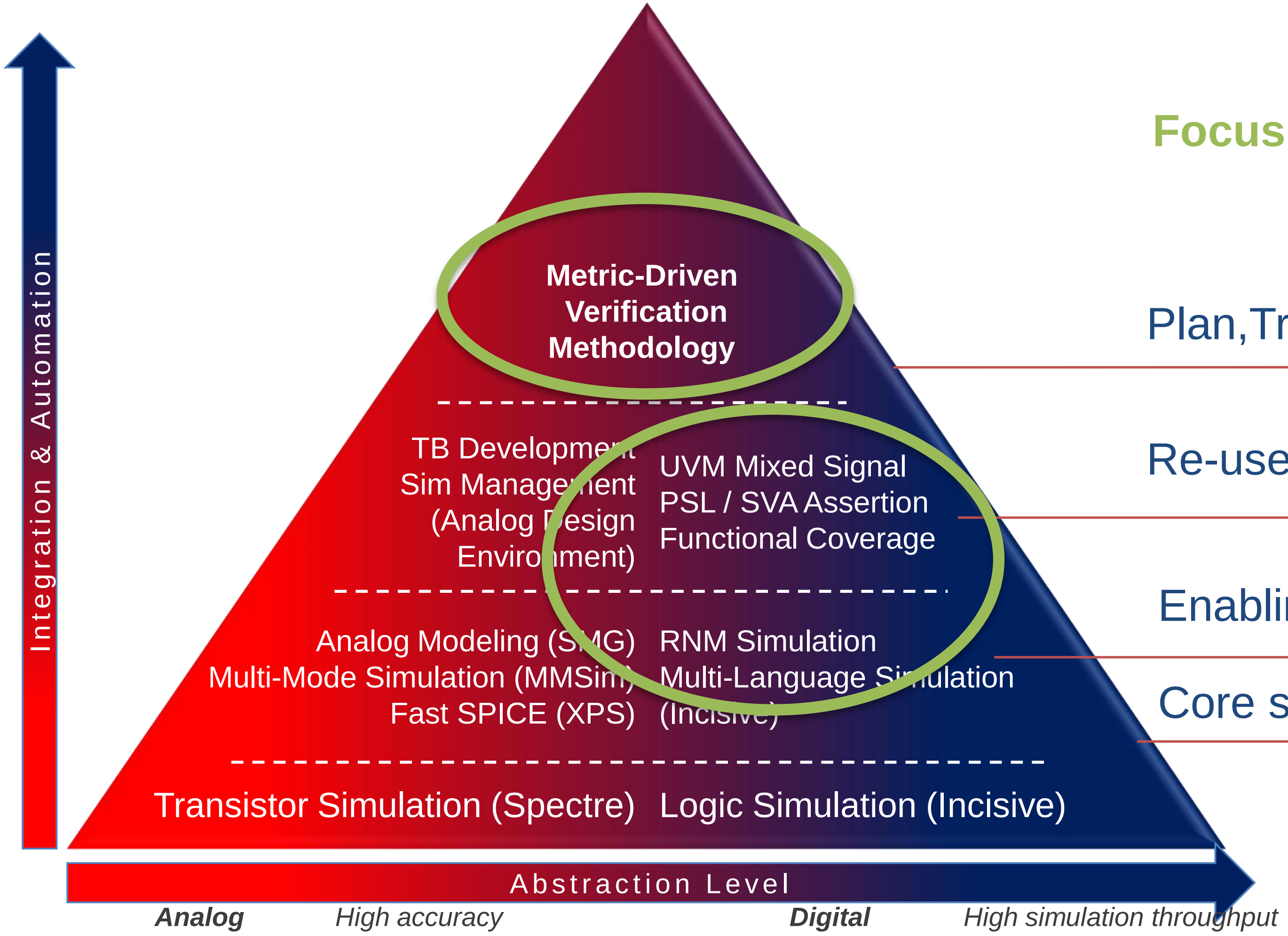
- *Planning*
- *Tracking to closure*
- *Execution and debugging*

- *Performance*
- *Features*

- *Methodology*
- *Library*
- *Tools abstracting analog and mixed-signal functionality*

Cadence mixed-signal verification solution

Bridging the GAP, addressing complexity



Agenda

1. Metric-Driven Verification for MS
2. **Verification Planning and Management in MS**
3. Universal-Verification Methodology for MS
4. Real-number Modeling Capabilities
5. Analog and MS Assertions
6. Q&A

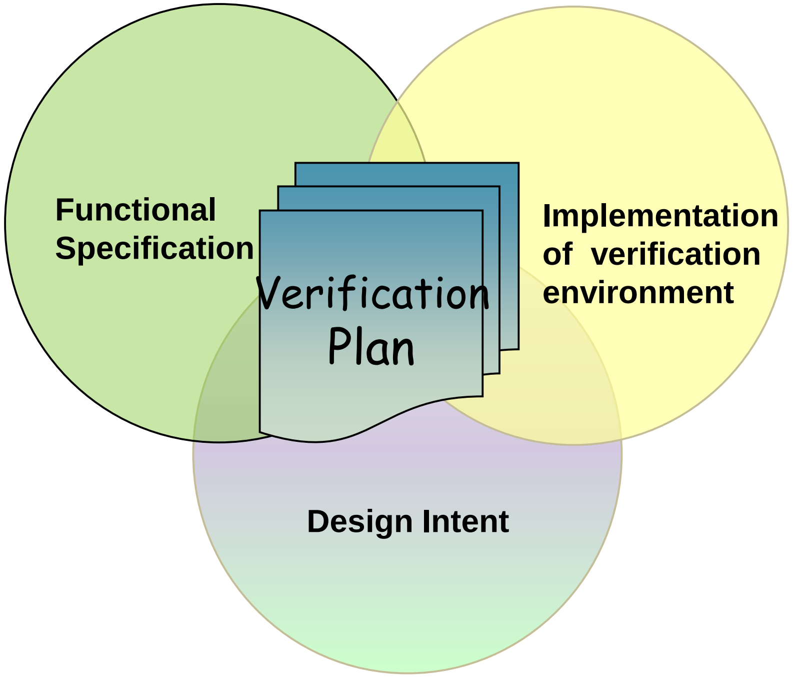
Verification Planning in MS

Kawe Fotouhi

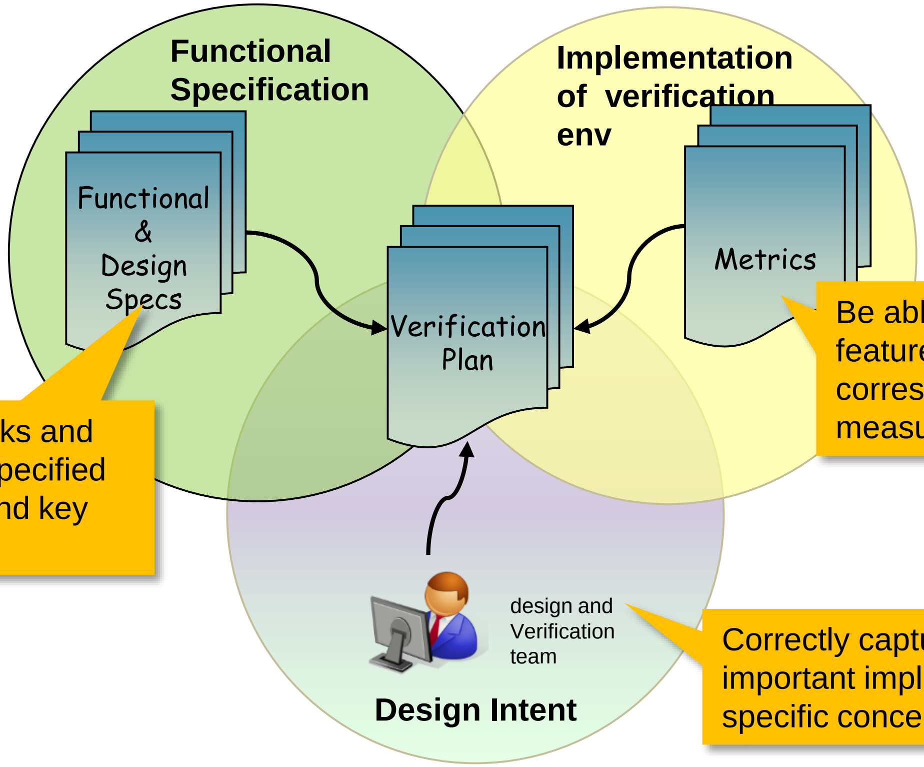
What is a Meaningful Verification plan?

- Functional Verification is the process of proving the convergence of the functional specification, the design intent, and the Test environment implementation

- A good and meaningful verification plan will prove that convergence



Fundamentals of a Good Verification Plan


Functional Specification
Implementation of verification env
Functional & Design Specs
Verification Plan
Metrics
design and Verification team
Design Intent

Directly links and maps all specified features and key details

Be able to correlate features with corresponding measured metrics

Correctly captures important implementation specific concerns

Creating a Feature based Verification Plan I

Feature Identification

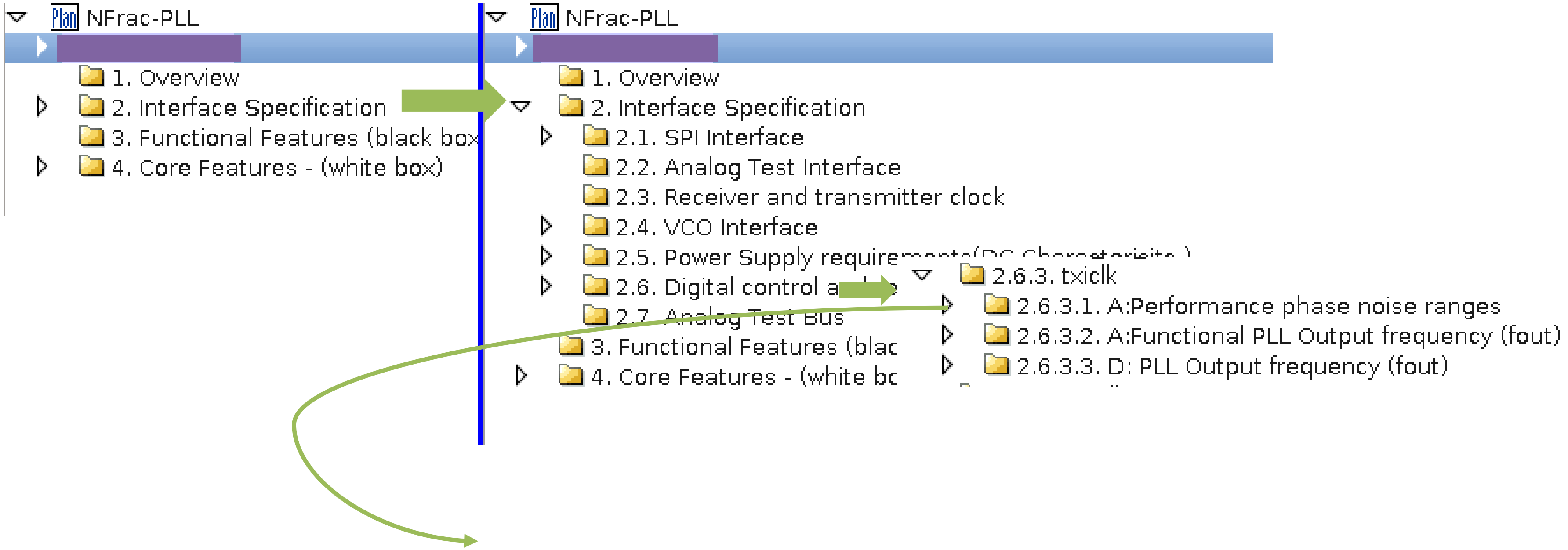
- Get all project related people together
  - Analog designer, analog and digital verification engineer, Marketing, Concept, Software, ...
- „Brainstorm“ plan hierarchy and features based on
  - Specification
  - KnowHow, experience & gut feeling
- Feature analysis focuses on :
  - "What" to verify
  - Which domain (analog/digital) to verify
  - "How” to verify
- Feature Examples
  - Device mode and configuration options
  - Traffic or protocol handling
  - Protocol or device exception handling
  - Performance specification
  - Operation conditions (PVT)
    - Process variations
    - Voltage supply
    - Temperature
  - Application modes
  - External connections
  - Typical and critical use and corner cases (duty cycle, phase noise ratio etc.)

Creating Feature based Plan II

Attribute Elaboration

- Translating Feature requirements into concrete metric goals
- Ask HOW features will be measured
- Identify required testcases, coverage and checks metrics
- Which attributes and values are important?
  - Driven by the spec
- Where should each value be observed?
  - At boundary or involving internal signals
- When are the values valid to be sampled?
  - reaching a certain voltage in a given time window after power-up

PLL output (txi_clk) analog performance and functional feature



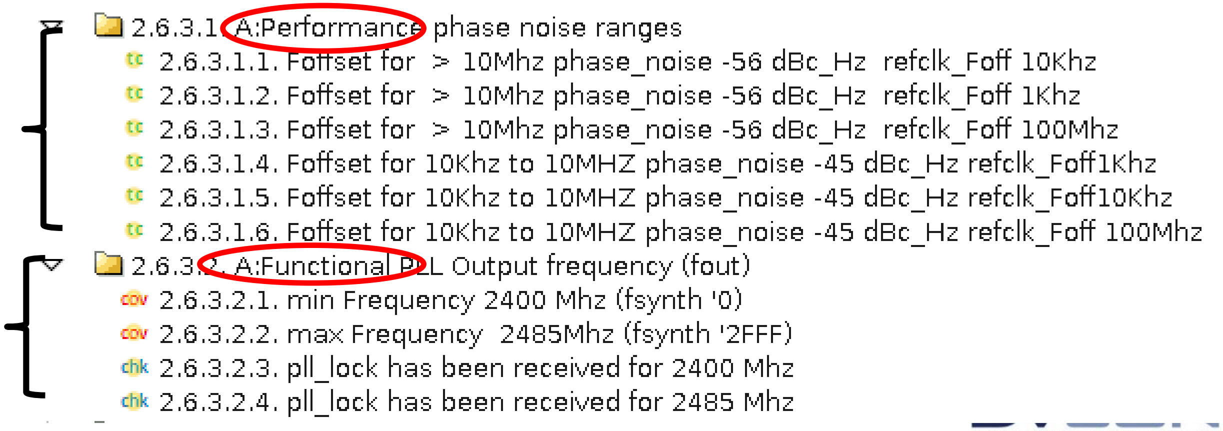
NFrac-PLL
1. Overview
2. Interface Specification
3. Functional Features (black box
4. Core Features - (white box)

NFrac-PLL
1. Overview
2. Interface Specification
2.1. SPI Interface
2.2. Analog Test Interface
2.3. Receiver and transmitter clock
2.4. VCO Interface
2.5. Power Supply requirements(DC Characterisitc)
2.6. Digital control a
2.7. Analog Test Bus
3. Functional Features (blac
4. Core Features - (white bc

2.6.3. txiclk
2.6.3.1. A:Performance phase noise ranges
2.6.3.2. A:Functional PLL Output frequency (fout)
2.6.3.3. D: PLL Output frequency (fout)

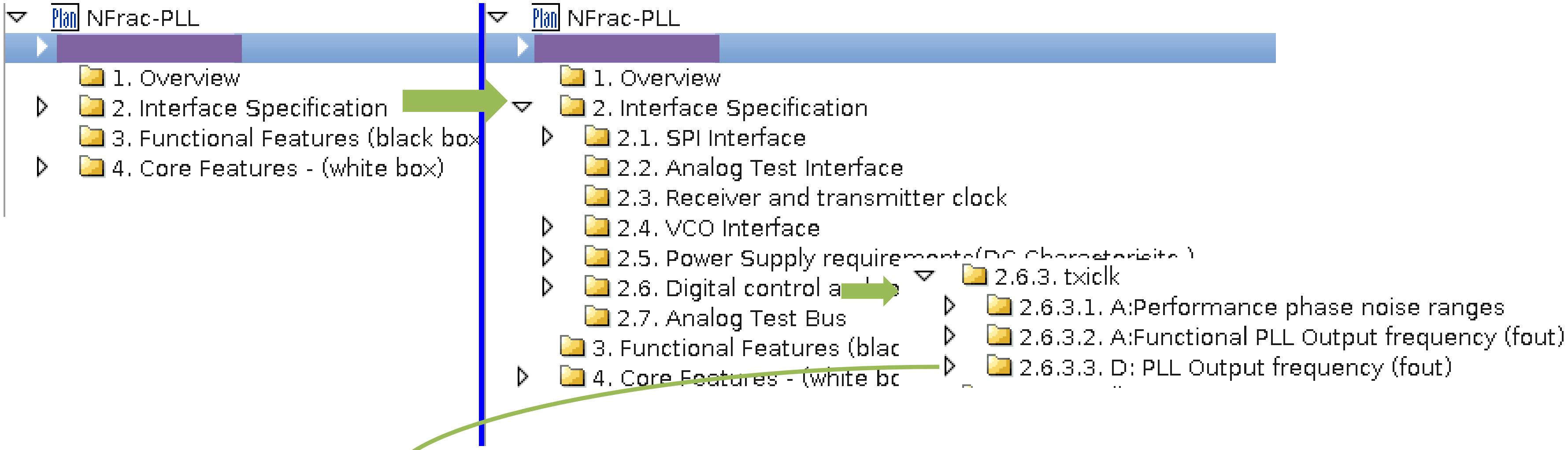
Test the SNR in combination with ref_clk offset

- Cover corner case frequency
- Check PLL locks



2.6.3.1. A:Performance phase noise ranges
2.6.3.1.1. Foffset for > 10Mhz phase_noise -56 dBc_Hz refclk_Foff 10Khz
2.6.3.1.2. Foffset for > 10Mhz phase_noise -56 dBc_Hz refclk_Foff 1Khz
2.6.3.1.3. Foffset for > 10Mhz phase_noise -56 dBc_Hz refclk_Foff 100Mhz
2.6.3.1.4. Foffset for 10Khz to 10MHZ phase_noise -45 dBc_Hz refclk_Foff1Khz
2.6.3.1.5. Foffset for 10Khz to 10MHZ phase_noise -45 dBc_Hz refclk_Foff10Khz
2.6.3.1.6. Foffset for 10Khz to 10MHZ phase_noise -45 dBc_Hz refclk_Foff 100Mhz
2.6.3.2. A:Functional PLL Output frequency (fout)
2.6.3.2.1. min Frequency 2400 Mhz (fsynth '0)
2.6.3.2.2. max Frequency 2485Mhz (fsynth '2FFF)
2.6.3.2.3. pll_lock has been received for 2400 Mhz
2.6.3.2.4. pll_lock has been received for 2485 Mhz

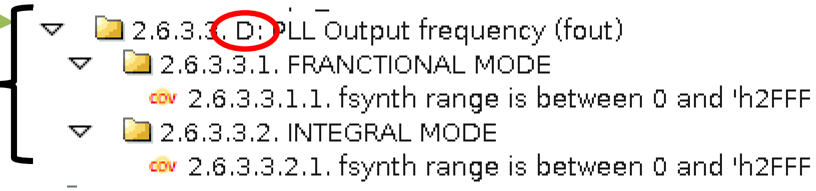
PLL (txi_clk) output Digital



- NFrac-PLL
  - 1. Overview
  - 2. Interface Specification
  - 3. Functional Features (black box
  - 4. Core Features - (white box)

- NFrac-PLL
  - 1. Overview
  - 2. Interface Specification
    - 2.1. SPI Interface
    - 2.2. Analog Test Interface
    - 2.3. Receiver and transmitter clock
    - 2.4. VCO Interface
    - 2.5. Power Supply requirements(DC Characterisits)
    - 2.6. Digital control a
    - 2.7. Analog Test Bus
  - 3. Functional Features (blac
  - 4. Core Features - (white bc

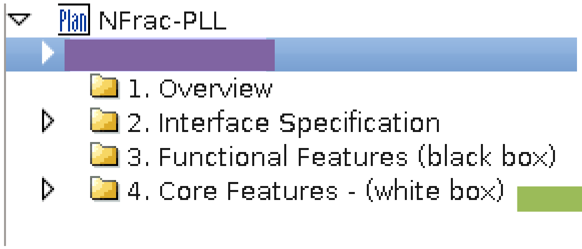
- 2.6.3. txiclk
  - 2.6.3.1. A:Performance phase noise ranges
  - 2.6.3.2. A:Functional PLL Output frequency (fout)
  - 2.6.3.3. D: PLL Output frequency (fout)



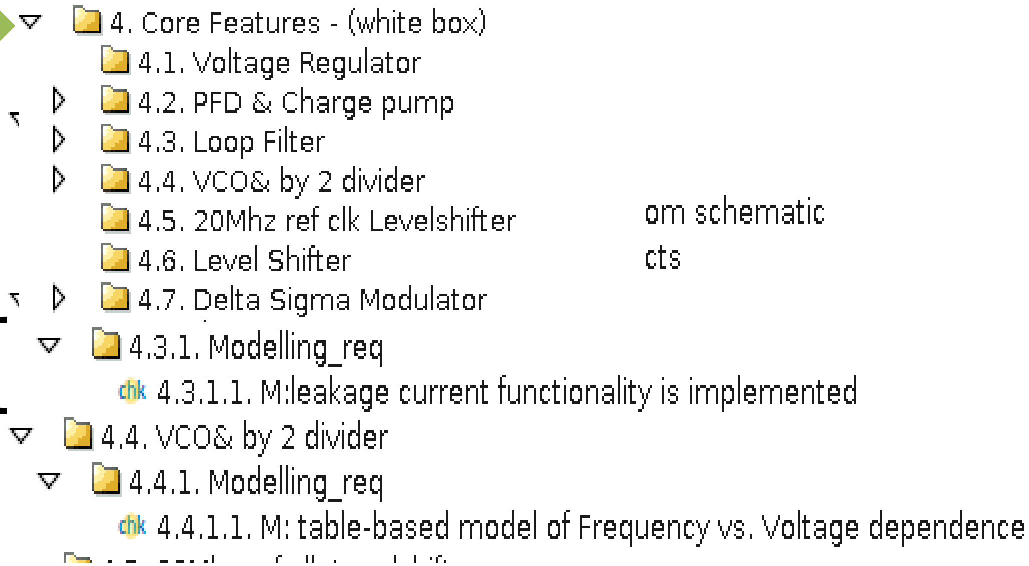
- 2.6.3.3. D: PLL Output frequency (fout)
  - 2.6.3.3.1. FRANCTIONAL MODE
    - 2.6.3.3.1.1. fsynth range is between 0 and 'h2FFF
  - 2.6.3.3.2. INTEGRAL MODE
    - 2.6.3.3.2.1. fsynth range is between 0 and 'h2FFF

Cover all possible output frequency (randomize fsynth) FRACTIONAL and INTEGRAL MODE

PLL Core feature – modelling requirements



- Plan NFrac-PLL
  - 1. Overview
  - 2. Interface Specification
  - 3. Functional Features (black box)
  - 4. Core Features - (white box)



- 4. Core Features - (white box)
  - 4.1. Voltage Regulator
  - 4.2. PFD & Charge pump
  - 4.3. Loop Filter
  - 4.4. VCO& by 2 divider
  - 4.5. 20Mhz ref clk Levelshifter
  - 4.6. Level Shifter
  - 4.7. Delta Sigma Modulator

om schematic

cts

- 4.3.1. Modelling_req
  - chk 4.3.1.1. M:leakage current functionality is implemented
- 4.4. VCO& by 2 divider
  - 4.4.1. Modelling_req
    - chk 4.4.1.1. M: table-based model of Frequency vs. Voltage dependence

PLL locks even if it shouldn't if the dutycycle of the ref clock is too large

Agenda

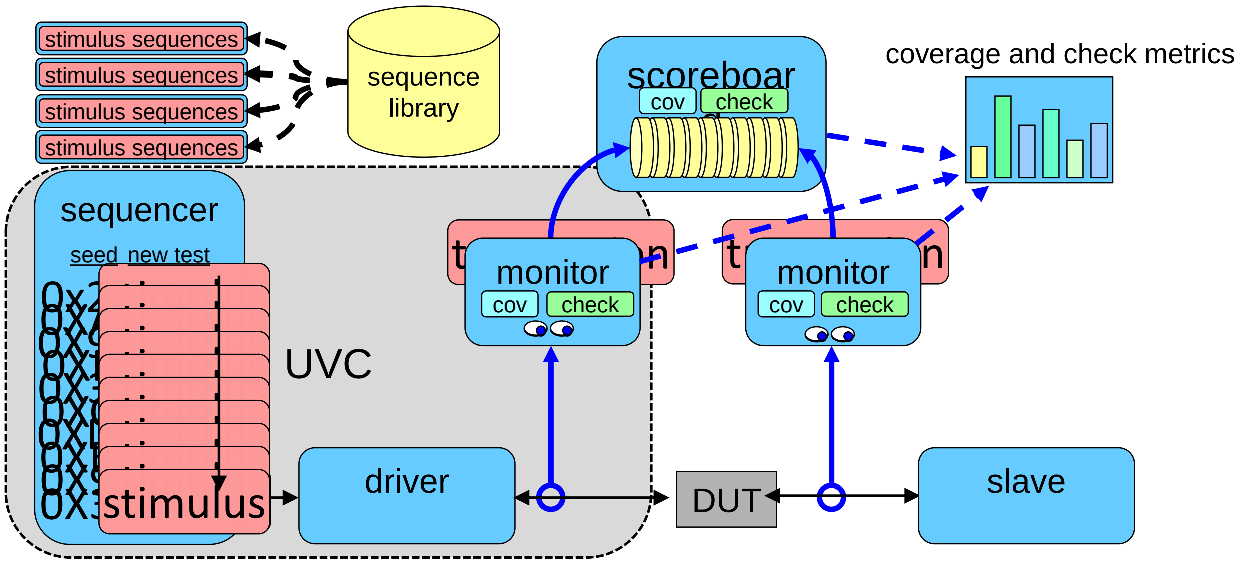
1. Metric-Driven Verification for MS
2. Verification Planning and Management in MS
3. **Universal-Verification Methodology for MS**
4. Real-number Modeling Capabilities
5. Analog and MS Assertions
6. Q&A

UVM for Mixed-signal

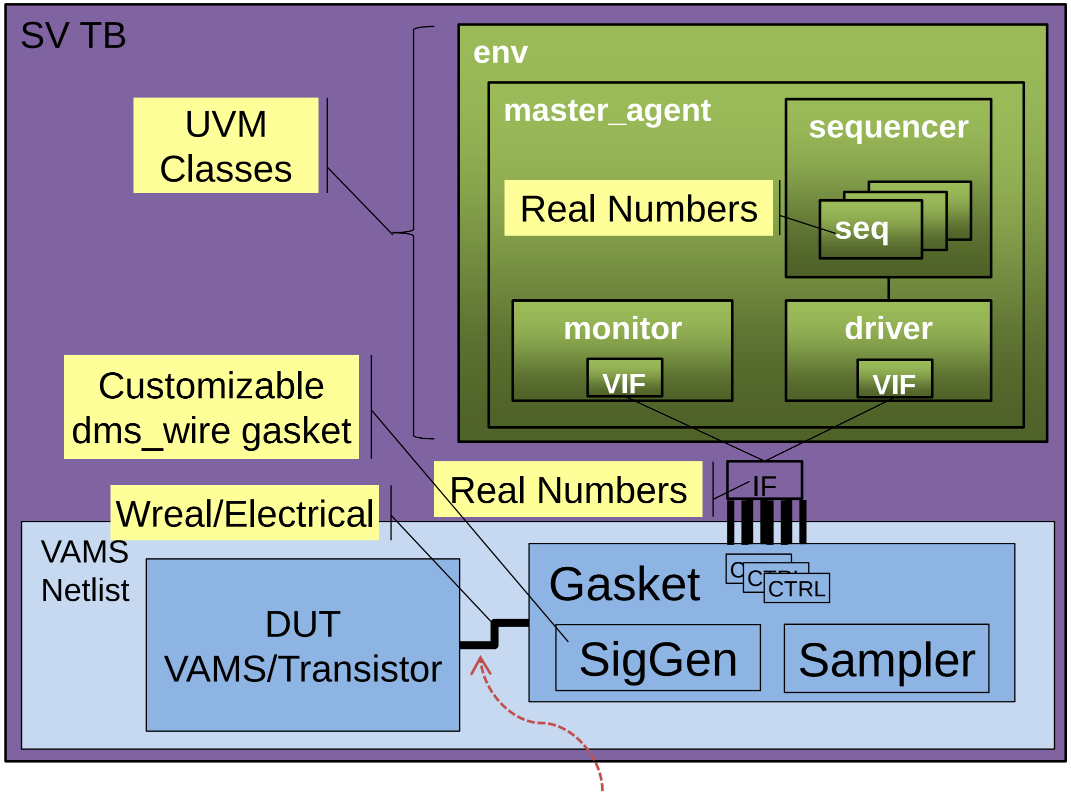
Thomas Ziller

Using UVM to Apply MDV

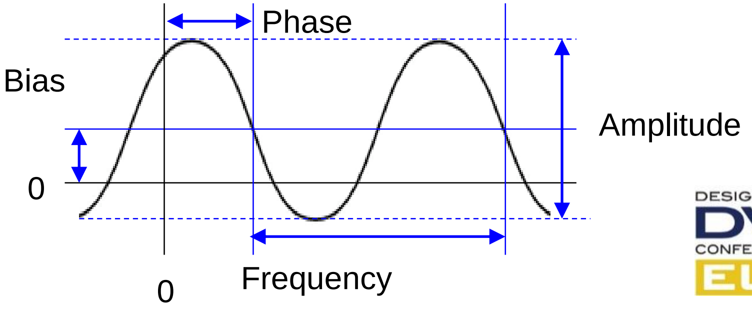
- Components of a MDV environment
  - Automated Stimulus Generation
  - Independent Checking
  - Coverage Collection



MS-MDV Block Diagram (dms_wire, SV top)



```
`uvm_do_with ana1_wire_seq {
        clk_period == 0.5; // sample clk
        ampl == 0.001;     // 1 mV
        bias == 1.1;
        freq == 100e6;     // 100 MHz
        phase == 0.0;
}
```



SV RNM: Coverage/Randomization

- Coverage/Randomization of reals
- Cadence provides full coverage/randomization support
  - Full compliment of real variable usage in randomization

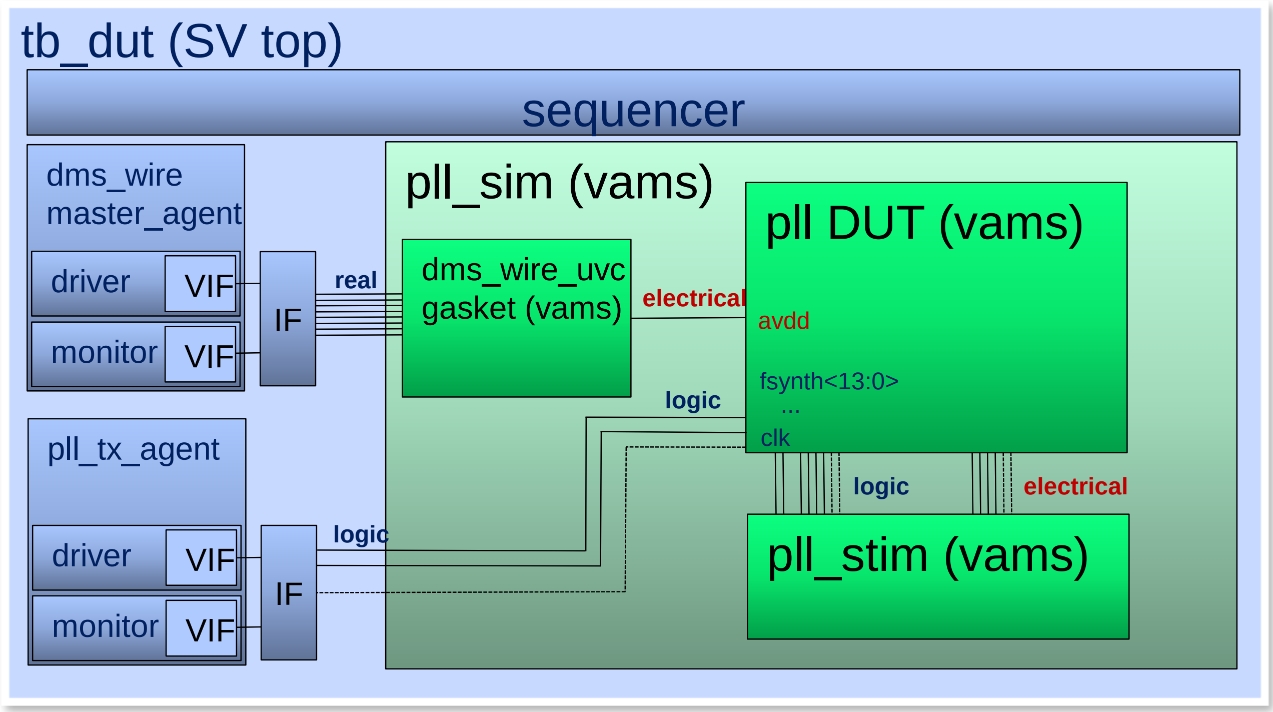
```
// Vector bins with precision
class my_tb_cls;
  rand real voltage;
  constraint my_constr {voltage dist
    { [1.0 :1.25] := 1,
      [1.25:1.5 ] := 10,
      [1.5 :2.0 ] := 1 };
  }
  covergroup cg {
    my_voltage : coverpoint voltage {
      type_option.real_interval = 0.1;
      bins b1[] = {[1.0:2.0]};
    }
  endgroup : cg
endclass
```

Randomization of the voltage

Coverage of what voltage values were generated

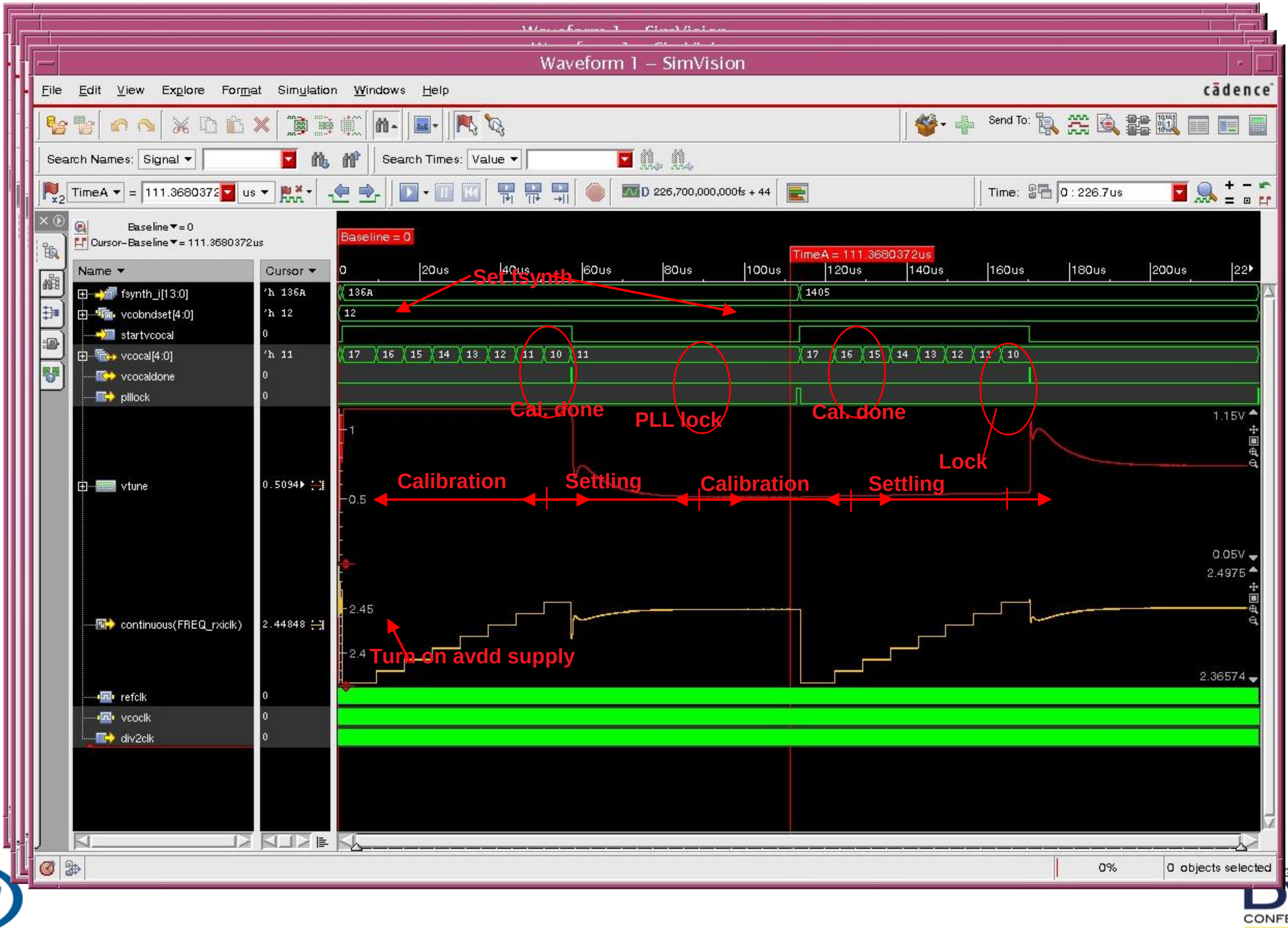
N-Fractional PLL Mixed-Signal

UVM-MS Testbench Hierarchy Structure



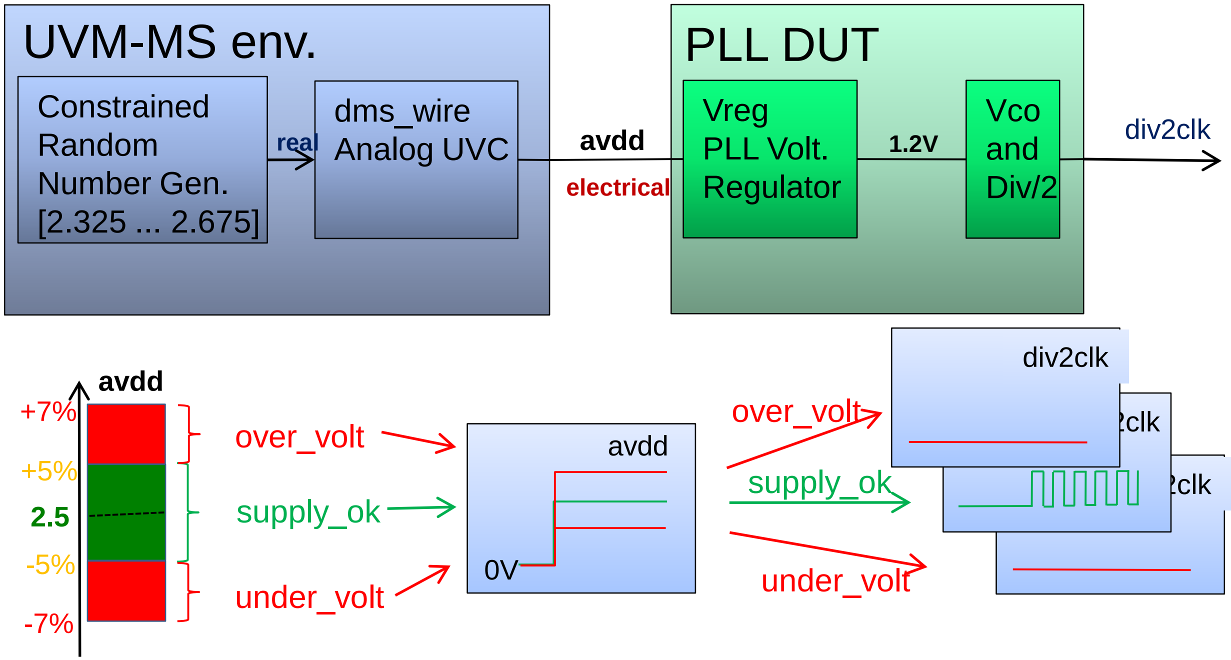
N-Fractional PLL Mixed-Signal

Constrained Random Simulation Results



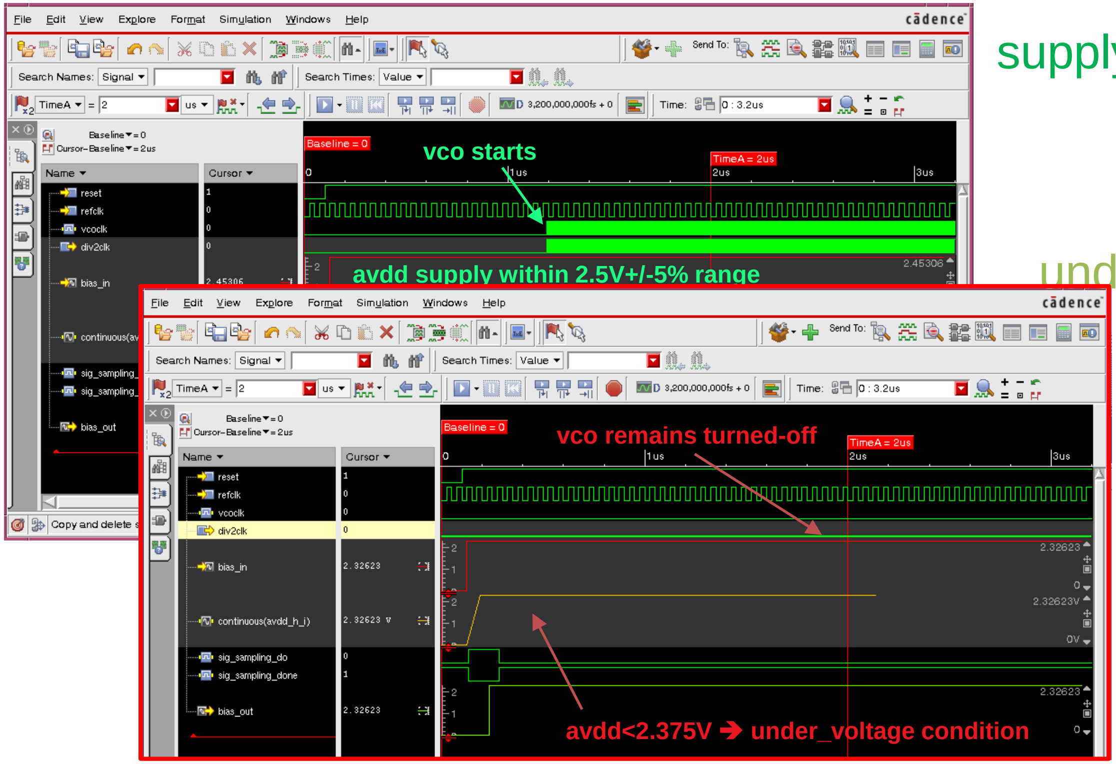
Constrained random variations

N-Fractional PLL Mixed-Signal
”avdd” Supply Range Checking



N-Fractional PLL Mixed-Signal

”avdd” Supply Range Checking



vco starts

avdd supply within 2.5V+/-5% range

vco remains turned-off

avdd<2.375V ➔ under_voltage condition

supply_ok

under_voltage

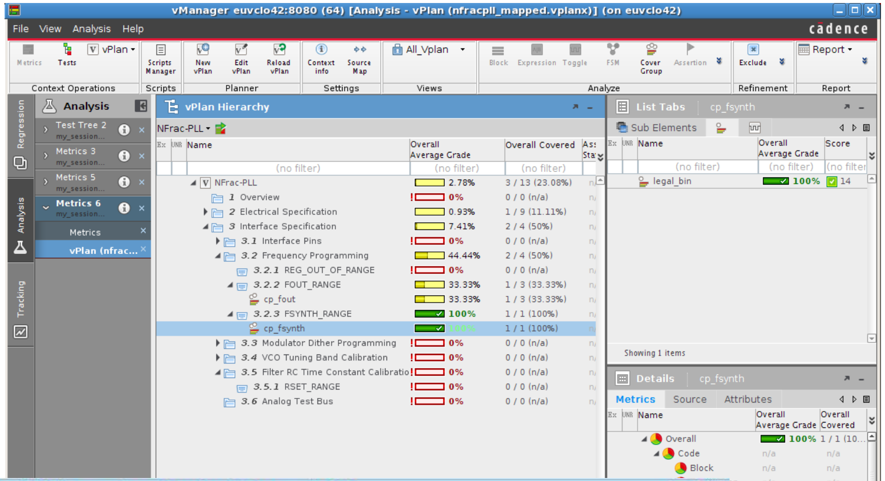
MS Regression Control & Analysis

Functional Coverage Results Example (20 runs)

Covergroup definitions:

```
covergroup bias_cg;
  bias_cp : coverpoint bias {
    bins over_volt = {[2.625:10]};
    bins supply_ok =
{[2.375:2.625]};
    bins under_volt ={[0: 2.375]}; }
endgroup // bias_cg
```

```
covergroup cg_fsynth;
  cp_fsynth: coverpoint fsynth{
        illegal_bins a =
          {[14'h2201:14'h3fff]};
        option.auto_bin_max = 25; }
endgroup : cg_fsynth
```



vManager euvclo42:8080 (64) [Analysis - vPlan (nfracpll_mapped.vplanx)] (on euvclo42)

File View Analysis Help
cadence
Metrics Tests vPlan Scripts Manager New vPlan Edit vPlan Reload vPlan Context info Source Map All_Vplan Block Expression Toggle FSM Cover Group Assertion Exclude Report
Context Operations Scripts Planner Settings Views Analyze Refinement Report
Regression Analysis Tracking
Test Tree 2
Metrics 3
Metrics 5
Metrics 6
Metrics
vPlan (nfrac...

vPlan Hierarchy

NFrac-PLL

Name	Overall Average Grade	Overall Covered
NFrac-PLL	2.78%	3 / 13 (23.08%)
1 Overview	0%	0 / 0 (n/a)
2 Electrical Specification	0.93%	1 / 9 (11.11%)
3 Interface Specification	7.41%	2 / 4 (50%)
3.1 Interface Pins	0%	0 / 0 (n/a)
3.2 Frequency Programming	44.44%	2 / 4 (50%)
3.2.1 REG_OUT_OF_RANGE	0%	0 / 0 (n/a)
3.2.2 FOUT_RANGE	33.33%	1 / 3 (33.33%)
cp_fout	33.33%	1 / 3 (33.33%)
3.2.3 FSYNTH_RANGE	100%	1 / 1 (100%)
cp_fsynth	100%	1 / 1 (100%)
3.3 Modulator Dither Programming	0%	0 / 0 (n/a)
3.4 VCO Tuning Band Calibration	0%	0 / 0 (n/a)
3.5 Filter RC Time Constant Calibratio	0%	0 / 0 (n/a)
3.5.1 RSET_RANGE	0%	0 / 0 (n/a)
3.6 Analog Test Bus	0%	0 / 0 (n/a)

List Tabs cp_fsynth

Sub Elements

Name	Overall Average Grade	Score
legal_bin	100%	✓ 14

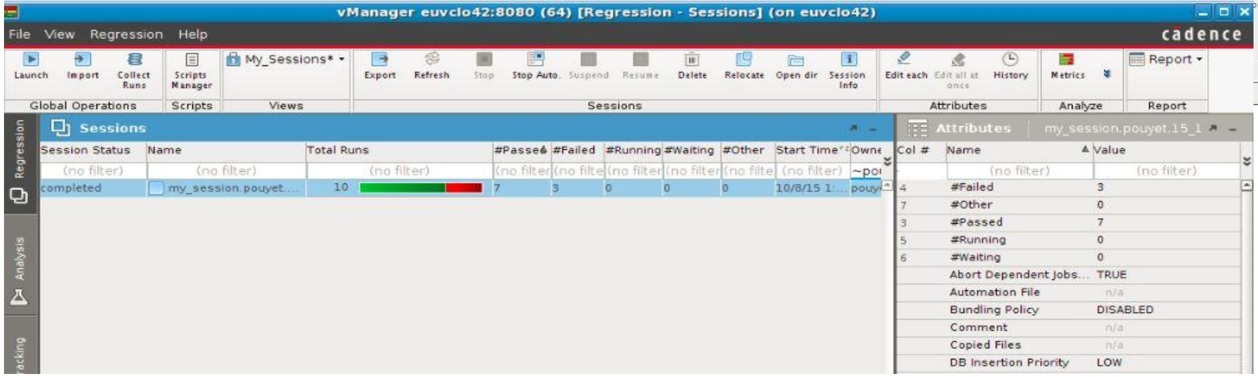
Showing 1 items

Details cp_fsynth

Metrics Source Attributes

Name	Overall Average Grade	Overall Covered
Overall	100%	1 / 1 (10...
Code	n/a	n/a
Block	n/a	n/a
...on	n/a	n/a
	n/a	n/a
	n/a	n/a
	100%	1 / 1 (10...

Messages



vManager euvclo42:8080 (64) [Regression - Sessions] (on euvclo42)

File View Regression Help
cadence
Launch Import Collect Runs Scripts Manager My_Sessions* Export Refresh Stop Stop Auto. Suspend Resume Delete Relocate Open dir Session Info Edit each Edit all at once History Metrics Report
Global Operations Scripts Views Sessions Attributes Analyze Report
Regression Analysis Tracking

Sessions

Session Status	Name	Total Runs	#Passed	#Failed	#Running	#Waiting	#Other	Start Time	Owne
completed	my_session.pouyet....	10	7	3	0	0	0	10/8/15 1:...	pouy

Attributes my_session.pouyet.15_1

Col #	Name	Value
4	#Failed	3
7	#Other	0
3	#Passed	7
5	#Running	0
6	#Waiting	0
	Abort Dependent Jobs...	TRUE
	Automation File	n/a
	Bundling Policy	DISABLED
	Comment	n/a
	Copied Files	n/a
	DB Insertion Priority	LOW

accellera SYSTEMS INITIATIVE

2015 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

Agenda

1. Metric-Driven Verification for MS

2. Verification Planning and Management in MS

3. Universal-Verification Methodology for MS

4. Real-number Modeling Capabilities

5. Analog and MS Assertions

6. Q&A

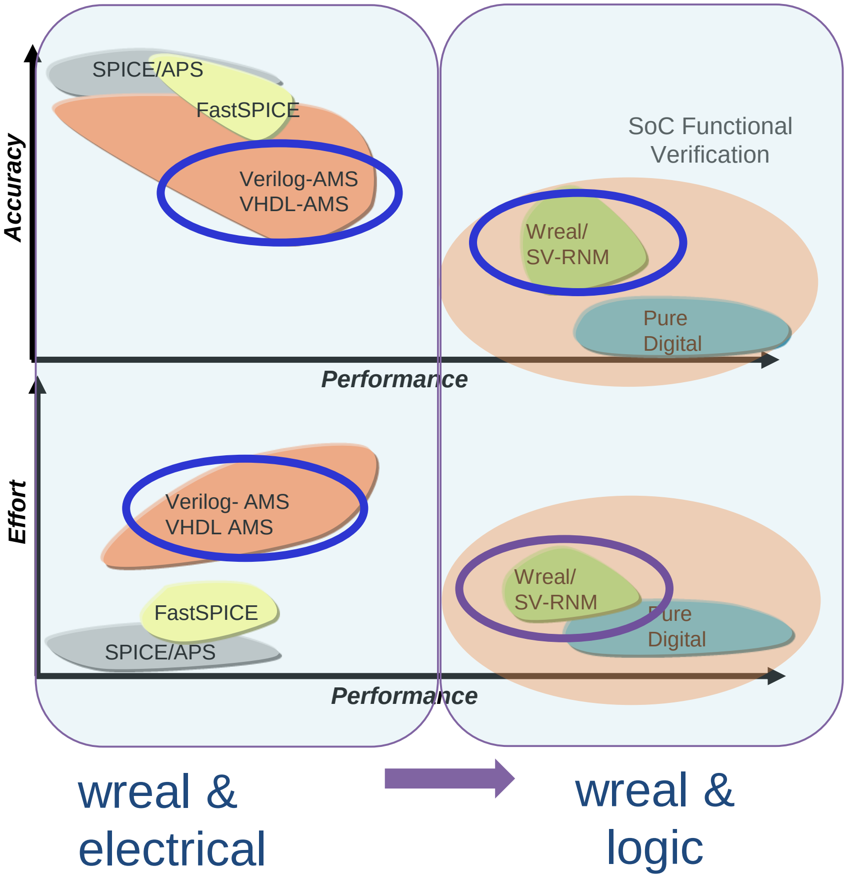
Real-number Modeling Capabilities

Ahmed Osman

Performance : Simulation throughput

Behavioral Modeling DMS vs AMS

- **Model analog block operation as discrete real data**
  - Signal flow-based modeling approach
- **Key advantages of RNM**
  - *Discrete* solver only
  - Very high simulation *performance*
  - *Event driven* or sampled data modeling of analog operation
  - No analog solver, *no convergence problems*!
  - Can be written by analog designers and/or digital verification engineers
- **RNM languages include**
  - Verilog-AMS (wreal)
  - VHDL
  - SystemVerilog
  - ***e***



Analog or Real Modeling: What is the Difference?

Analog Modeling

- Describes **current** vs. **voltage** relationship between nodes in model
- Newton-Raphson iteration process performs **matrix inversion** to solve all voltage and currents
- **Timestep** until next solution is selected based on accuracy criteria

Real Modeling

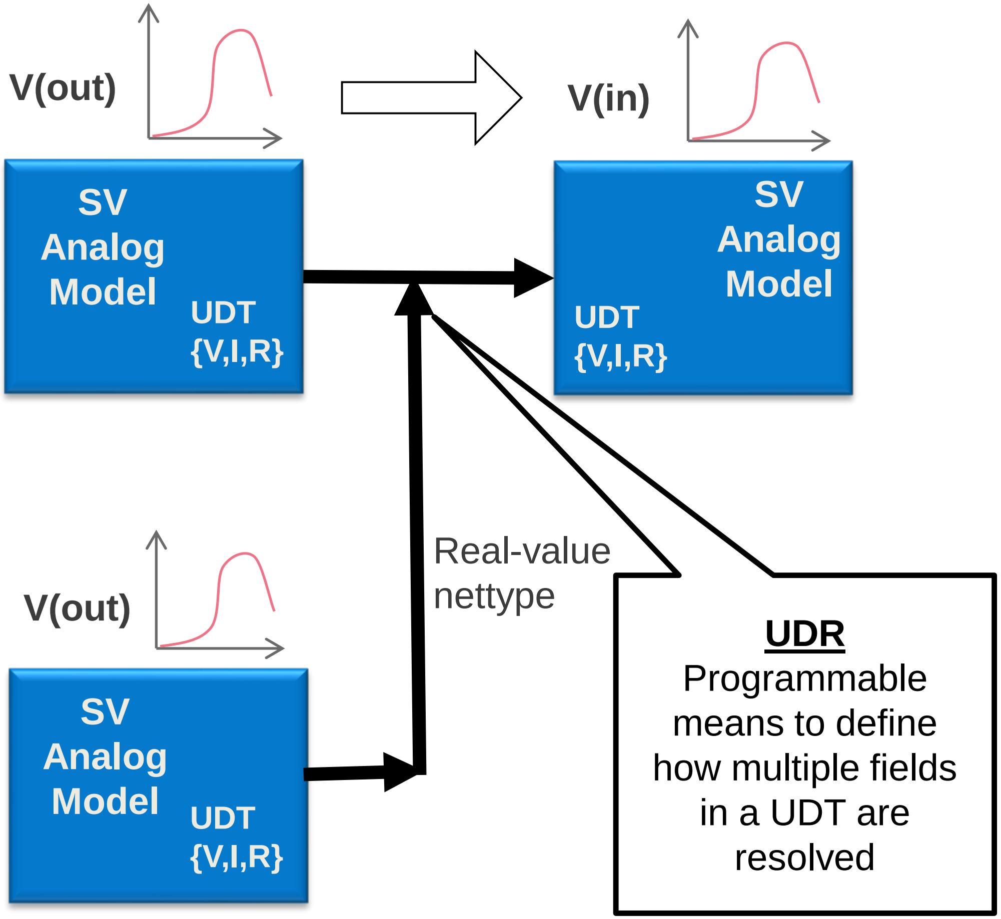
- **No matrix solution** – output computed directly from input & internal state. Model defines when to perform each internal computational segment
- No continuous time operation – only **sampled**, **clocked**, and/or **event-driven operations**. Updates can be performed when inputs change and/or at specific time increments
- Same format for digital and real modelling – difference is data type

SystemVerilog IEEE 1800-2012 LRM

- **User-Defined Types (UDTs)**
  - Allows for single-value real nettypes
  - Keyword used: **nettype**
  - Allows for multi-value nets (multi-field record style)
  - It can hold one or more values (such as voltage, current, impedance) in a single complex data type that can be sent over a wire
- **User-Defined Resolution (UDRs)**
  - Functions to resolve user-defined types using keyword: **with**
  - Specifies how to combine user defined types
- **Interconnect Nets**
  - Types
    - Explicit: Type-less/Generic nets with keyword: **interconnect**
    - Implied: A Verilog(-AMS) net with keywords: **wire, tri, wand, triand, wor,** or **trior**
  - Used only for a net or port declarations

SystemVerilog User-Defined Nets

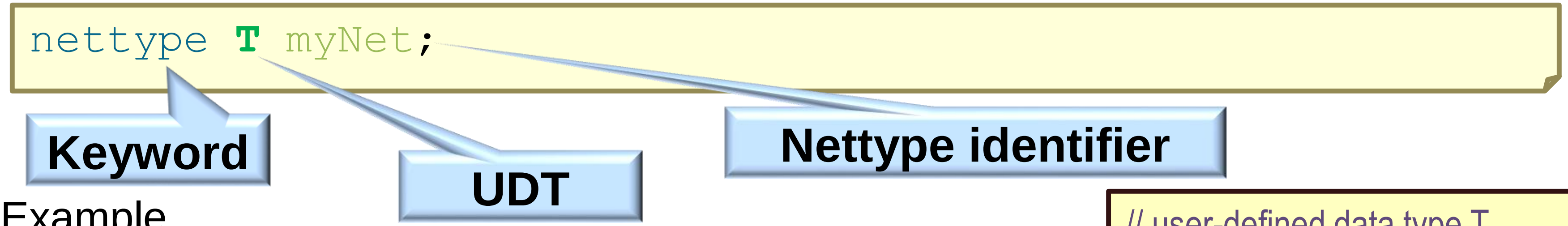
- User-Defined Nets can carry one or more values over a single net.
- Real values can be used to communicate voltage, current and other values between design blocks
- User-Defined Resolutions (UDR) functions are used to combine multiple outputs together.



UDR
Programmable means to define how multiple fields in a UDT are resolved

Declaring User-Defined Nettype

- A SystemVerilog user-defined nettype without any resolution function can be declared as:



```
nettype T myNet;
```

Example

```
module top;
 nettype  T  myNet;
 myNet w;
 assign w = T'{0.1, 0.2, 1'b1, 10};
 initial begin
     $display("Value of w -> %f => %p",$realtime, w);
  #1 $display("Value of w -> %f => %p",$realtime, w);
  #5 $display("Value of w -> %f => %p",$realtime, w);
 end
endmodule
```

```
// user-defined data type T
typedef struct {
  real    voltage;
  real    current;
  bit     field3;
  integer field4;
} T;
```

UDT

```
Value of w ->  0.000000 => '{voltage:0, current:0, field3:'h0, fileld4:x}
Value of w ->  1.000000 => '{voltage:0.1, current:0.2, field3:'h1, fileld4:10}
Value of w ->  6.000000 => '{voltage:0.1, current:0.2, field3:'h1, fileld4:10}
```

Declaring User-Defined Net with Resolution Function

- A user-defined SystemVerilog nettype with its resolution functions can be declared as:

```
nettype data_type  nettype_identifier with
                         [package_scope|class_scope] tf_identifier] ;
```

  - ***nettype_identifier*** is the identifier you specify for the nettype.
  - ***[package_scope|class_scope] tf_identifier]*** can be a Cadence built-in resolution function or any *typedef* to the built-in real type

```
//Declaring a UDT nettype with UDR
nettype T  wTsum with Tsum;
```

```
// user-defined data type T
typedef struct {
   real field1;
   real field2;
} T;
```

UDT

```
// user-defined resolution function Tsum
function automatic T Tsum (input T driver[]);
   Tsum.field1 = 0.0;
   Tsum.field2 = 0.0;
   foreach (driver[i]) begin
      Tsum.field1 += driver[i].field1;
      Tsum.field2 += driver[i].field2;
   end
endfunction
```

UDR

User-Defined Nettype Example

Data Type and Resolution Function (As a Package)

```
package temp_pkg;

// user-defined data type T

typedef struct {
    real field1;
    real field2;
}  T;

// user-defined resolution function Tsum

function automatic T Tsum  (input T driver[]);
  Tsum.field1 = 0.0;
  Tsum.field2 = 0.0;
  foreach (driver[i]) begin
     if (driver[i].field1 !== `wrealZState)
        Tsum.field1 += driver[i].field1;
     if (driver[i].field2 !== `wrealZState)
        Tsum.field2 += driver[i].field2;
  end
endfunction

// A nettype declaration with datatype and resolution function

nettype  T  wTsum  with Tsum;

endpackage
```

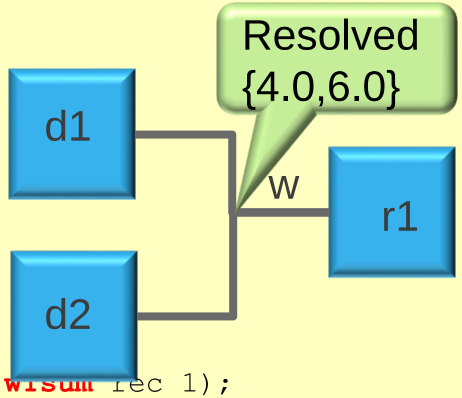
UDT

UDR

Nettype

Model

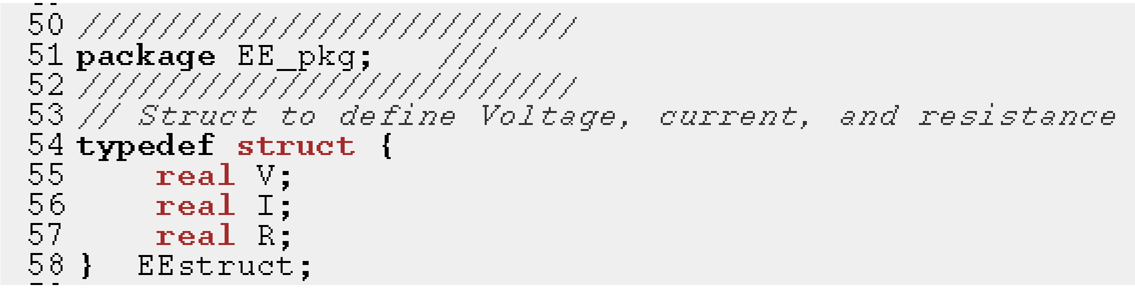
```
import temp_pkg::*;
module top;
  wTsum w;
  T  myvar;
  assign myvar = w;

  driver1 d1(w);
  driver2 d2(w);
  receiver1 r1(w);
endmodule

module receiver1 (input wTsum rec_1);
  always @(rec_1.field1, rec_1.field2)
 $display($time , ," sum = %f  flag = %f \n",
rec_1.field1, rec_1.field2);

endmodule

module driver1 (output wTsum dr_1);
  assign dr_1 = T'{1.0, 2.0};
endmodule

module driver2 (output wTsum dr_2);
  assign dr_2 = T'{3.0,4.0};
endmodule
```



Electrical Package in SystemVerilog

- An Electrical Package for Systemverilog (***EE_pkg.sv)*** defines an electrical equivalent net (V-I-R) for use in discrete analog behavioral models.
- You can use the new EE_pkg package to port existing wreal models to SV.

❑ Describes the structure ***EEstruct*** (UDT) which consists of three reals namely V, I and R.

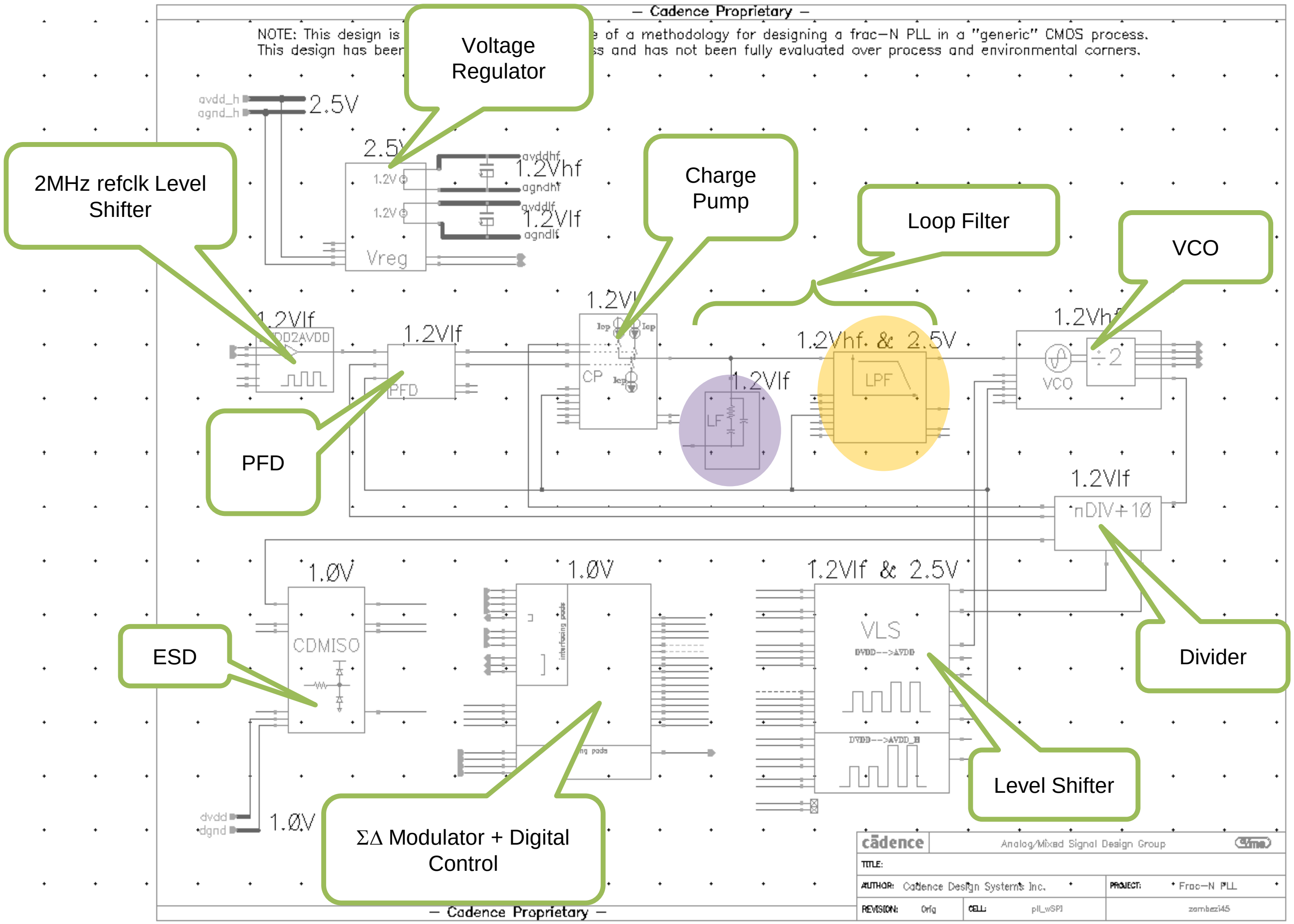


```
//////////////////////////////
package EE_pkg;    ///
//////////////////////////////
// Struct to define Voltage, current, and resistance
typedef struct {
     real V;
     real I;
     real R;
}  EEstruct;
```

- Has a UDR function that describes how the resolution of V, I and R are resolved, *res_EE*.
- This package ends with the nettype declaration statement:
- The ***EEnet*** will conform to Kirchoff's laws.

```
nettype EEstruct EEnet with res_EE;
```

Case Study 1: N-Fractional PLL Mixed Signal



Voltage Regulator

2MHz refclk Level Shifter

Charge Pump

Loop Filter

VCO

PFD

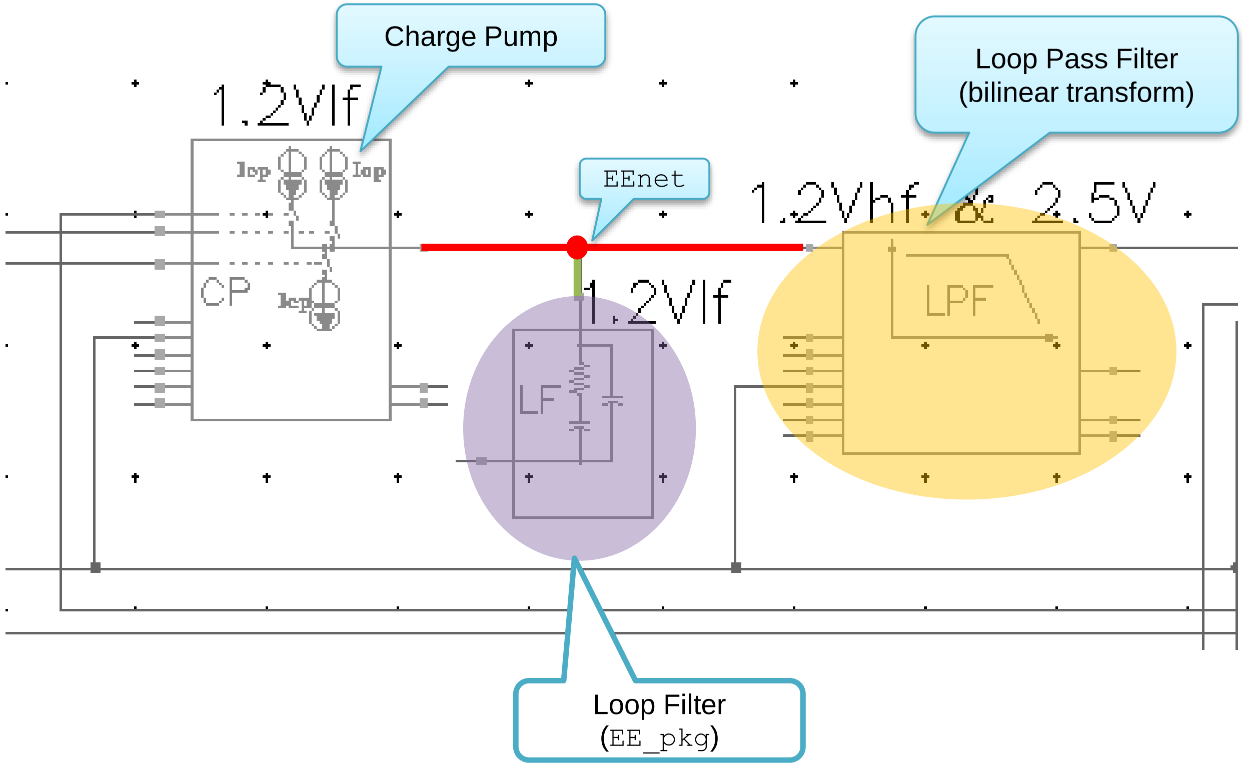
ESD

Divider

Level Shifter

ΣΔ Modulator + Digital Control

Case Study 1: N-Fractional PLL Mixed Signal



Charge Pump

Loop Pass Filter
(bilinear transform)

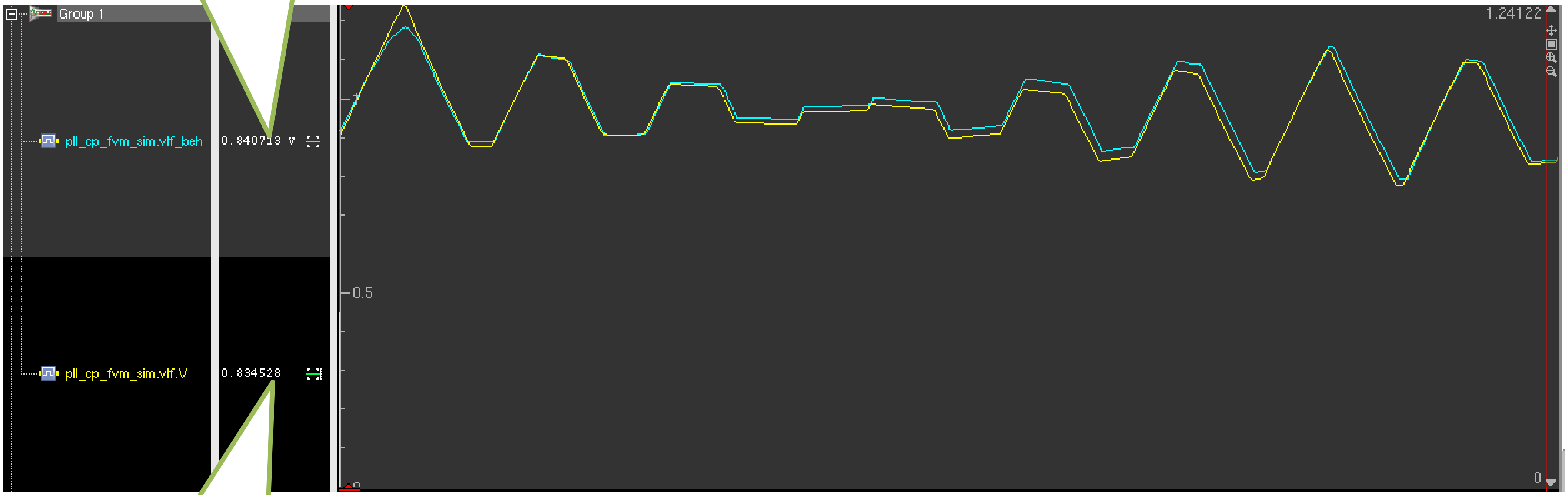
EEnet

Loop Filter
(EE_pkg)

Case Study 1: N-Fractional PLL Mixed Signal

- Loop Filter Voltage output (Verilog-AMS vs. SV EE_pkg)

Verilog-AMS



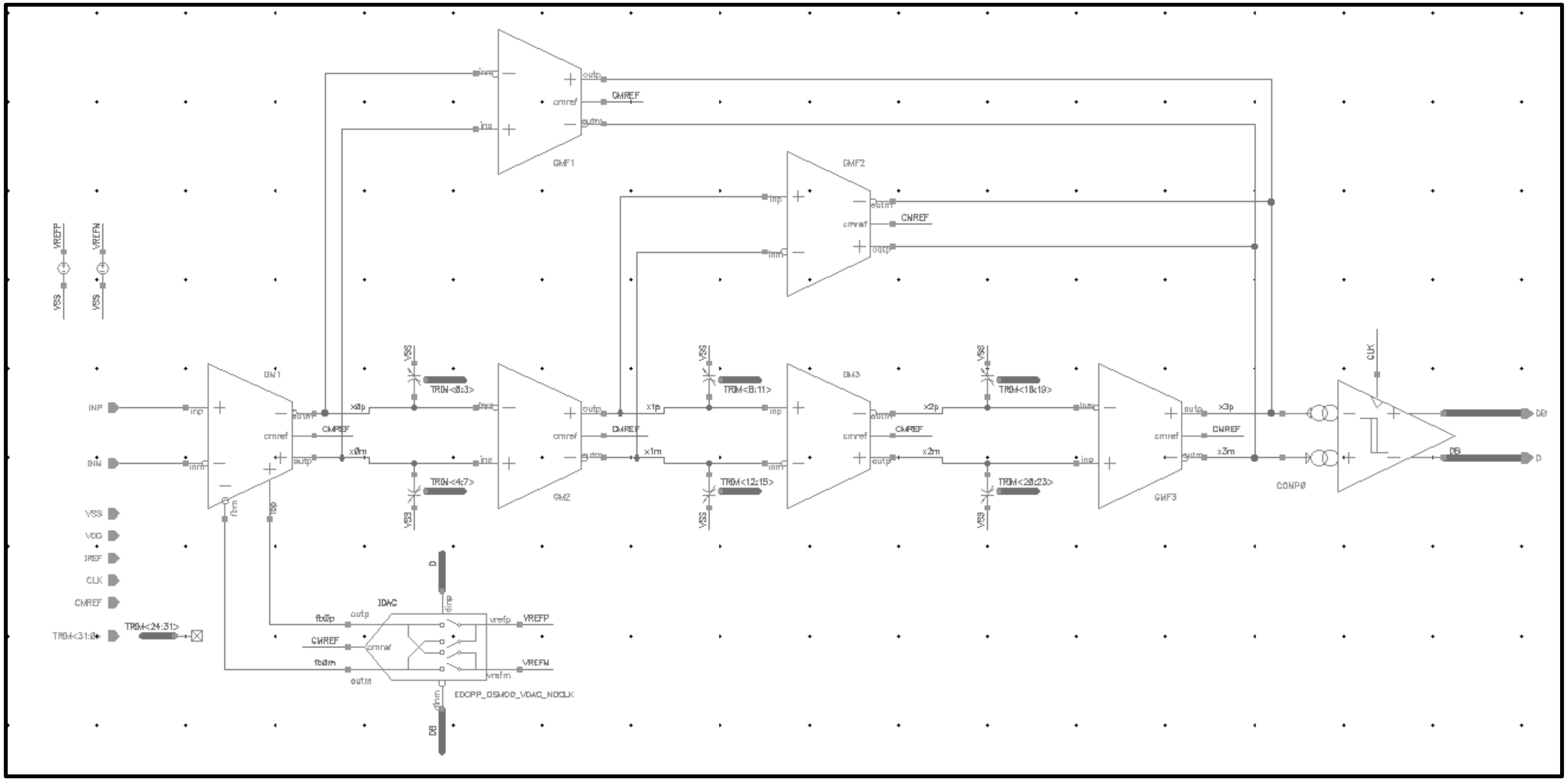
EE_pkg
SystemVerilog

	SV-RNM	VAMS
CPU Time	47 seconds	1 hr 8 min. 32 sec

A speed gain of 90x over mixed-signal Verilog-AMS

Case Study2: 3rd – order Feed-forward Gm-C **ΔΣ** ADC

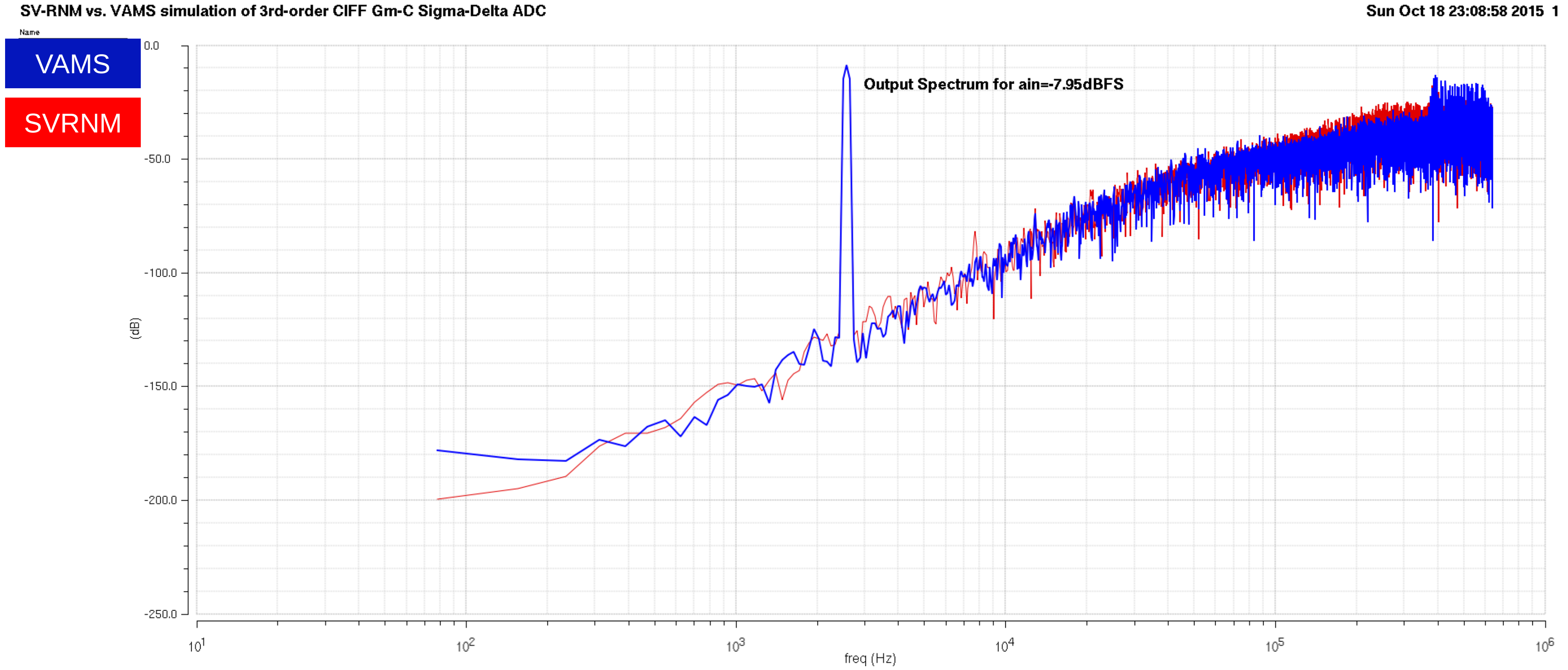
High-level Sizing and frequency scaling



Schematic of 3rd – order Gm-C **ΔΣ** ADC

Case Study2: 3rd-order Feed-forwardGm-C ΔΣ ADC

Simulation results for input signal = 80mV



Case Study2: 3rd-order CIFF Gm-C ΔΣ ADC

Simulation results for ain = 80mV

- **Spectrum Assistant** has been used in ViVA to evaluate various spectrum properties, e.g. SINAD, ENOB, THD, etc.

	SV-RNM	VAMS
SQNR	72.92 dB	72.33 dB
SINAD	71.06 dB	72.33 dB
ENOB	11.515	11.72
THD %	18.19m %	8.1m %
Noise Floor (per sqrt Hz)	-126 dB/sqrt Hz	-125.3 dB/sqrt Hz
CPU Time	0.4 seconds	92.5 seconds

A speed gain of 230x over mixed-signal Verilog-AMS

Agenda

1. Metric-Driven Verification for MS
2. Verification Planning and Management in MS
3. Universal-Verification Methodology for MS
4. Real-number Modeling Capabilities
5. **Analog and MS Assertions**
6. Q&A

Analog and MS Assertions

Ahmed Osman

Automation & re-use thru Assertions in Digital, Analog, and Mixed Signal

Why Assertions?

- Assume
- Assert
- Cover

Language Support

- SVA
- PSL

Not New for Analog

- Device checks
- Spectre MDL
- $cds_get_analog_value

Data converters	• e.g. Monotonicity,DNL, comparator meta-stability
Digitally-assisted analog	• e.g. Calibration / process variability compensation
Systems with Feedbcak	• PLL : e.g. PLL lock-in time, Output frequency tuning • Sigma-Delta : e.g. Integrator stability, presence of tones
Multiple modes	• Power modes, programmable gain, adaptive filters

Analog / Mixed-signal PSL Assertions

- Real Assertion (using RNM data type)
  - PSL with explicitly declared wreals
  - SVA using real variable

```
real vin;
// psl vin_check : assert always ( 1.2 < vin && vin < 1.3 )
// @(posedge clk);
```

- Analog Assertion (electrical domain behavior)
  - PSL or ***e*** containing analog objects or access functions or operators
  - (This is not possible in SVA since there is no analog object allowed in SV)

```
electrical vin;
// psl vin_check : assert always ( 1.2 < V(vin) && V(vin) < 1.3 )
// @( cross(V(clk)-1.25));
```

Analog PSL assertions: Verification Unit

- Verification units in PSL can contain analog objects
- Write your PSL statements/vunit into a file, e.g. inv_vams.pslvlog
- Example:

```
module INV_vams ( out1, in1 );
   output out1;
   input in1;
   electrical in1, out1;
   analog begin
     if (V(in1) >= 1.25)
       V(out1) <+ 0.0;
     else
       V(out1) <+ 2.5;
   end
endmodule
```

```
vunit inv_vams_inst_vunit(INV_vams)
{
  // psl assert
  // always ( V(out1) < 1.25 )
  // @( cross(V(in1)-1.25));
}
```

Demo

Questions