University of Chinese Academy of Sciences

Verification & Implementation of SoC Design

System-Level Design and Verification

Chun-Zhang Chen, Ph.D.

June 25-29, 2018

中国科学院大学2018年夏季

SoC Design and Verification

- **SoC Design and Verification**
- SoC Verification Flow
- ABV/TBV and TLM/UVM
- Advanced Verification Methods
- Discussion

Verification Challenges

System Design and Verification

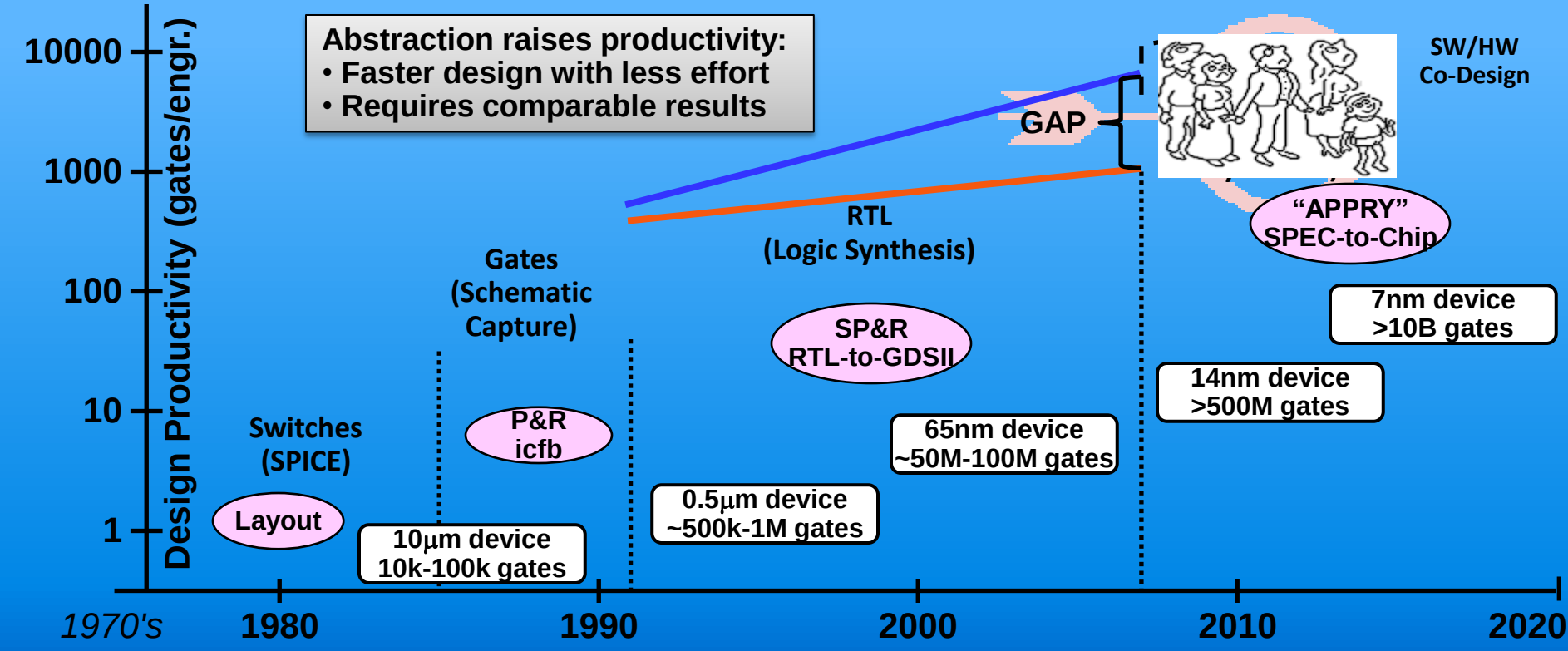
Methodology and Tools

- Architecture and Algorithm
- SW and HW Co-design
- Behavioral Modeling: TLM, UVM
- Languages: C, C++, SystemC, SystemVerilog, e
- Verification and Simulation
  - Testbench, DUV, ABV, TBV

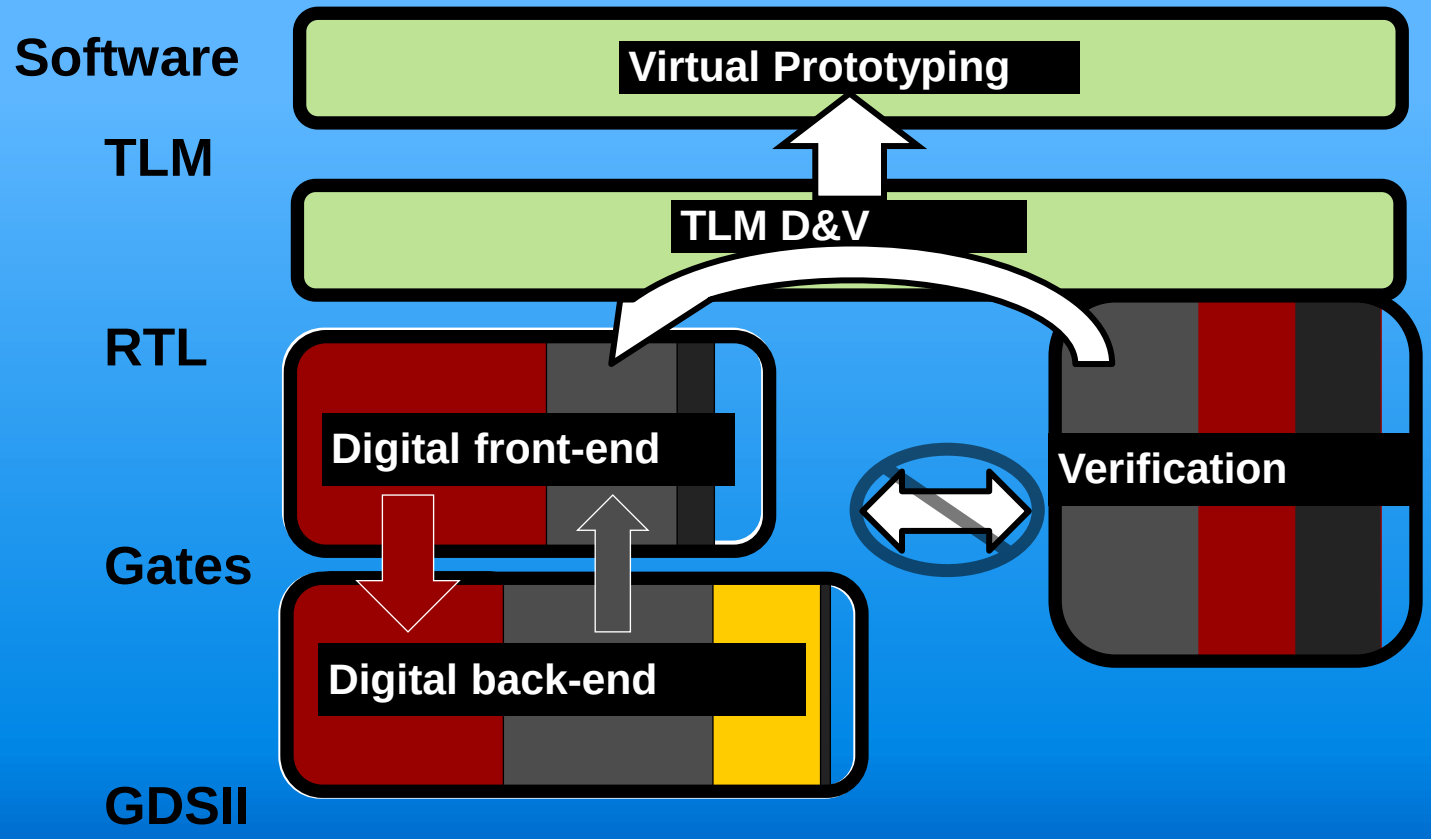
- Functional Verification
- Low Power Verification
- IP Verification (VIP)

Abstraction Must Be Raised To Keep Up!

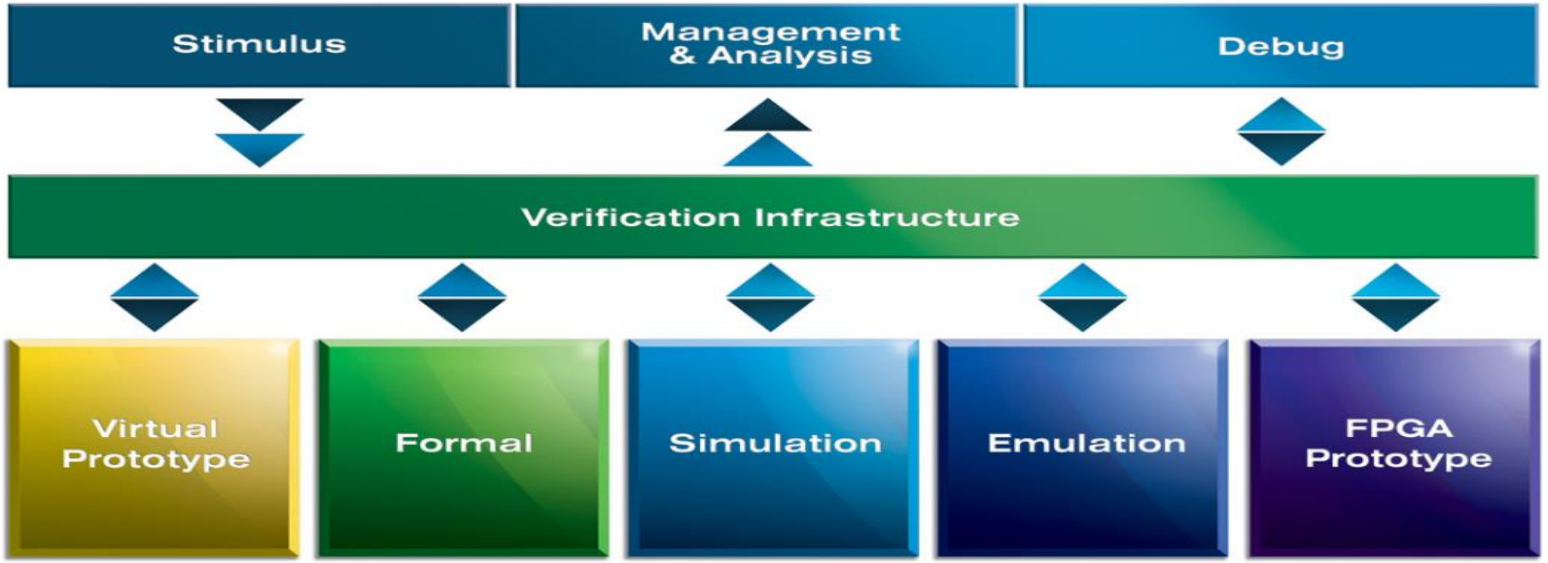
Challenges of Adopting the TLM Methodology



TLM, Transaction Level Modeling



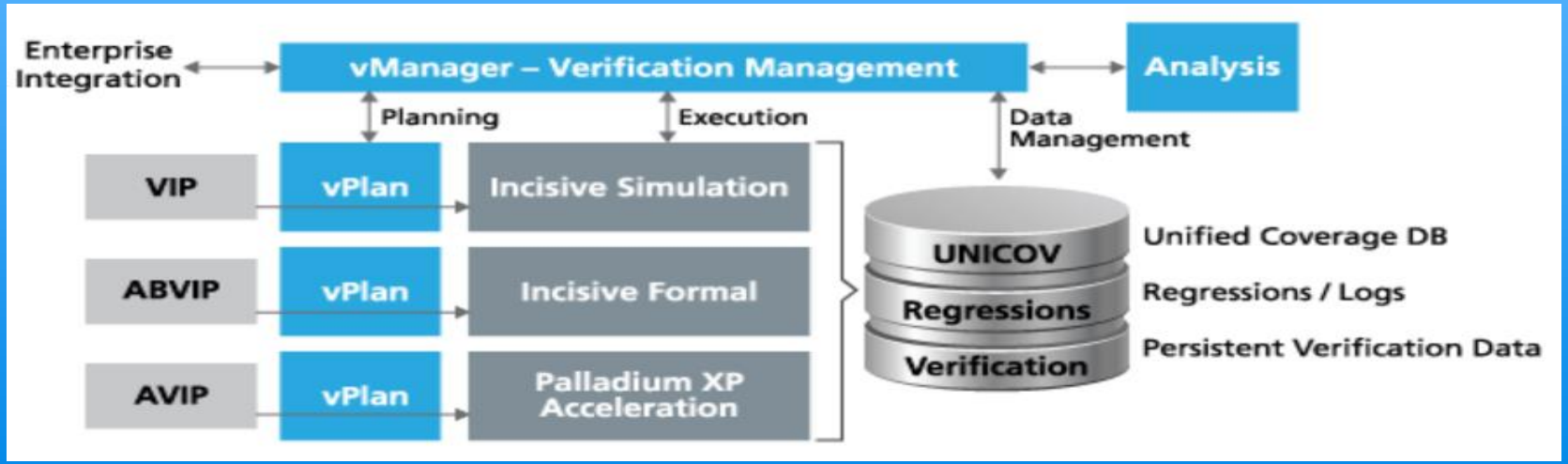
Functional Verification with M/Questa Verification Platform



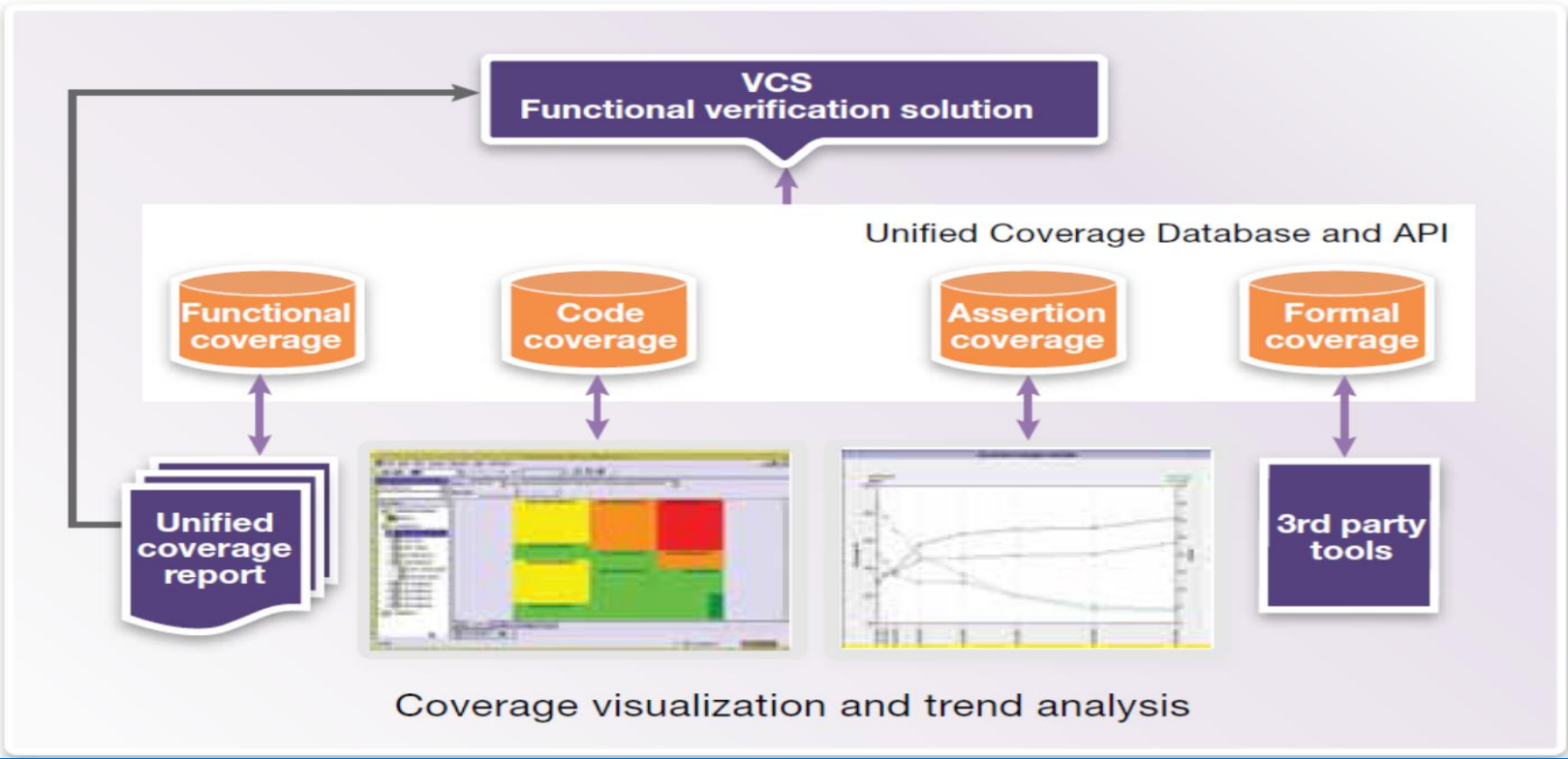
The Mentor Graphics Enterprise Verification Platform (EVP) is transforming verification by bringing powerful engines and platforms together to deliver high performance verification from prototype to silicon.

Verification with C/Incisive vManager Solution

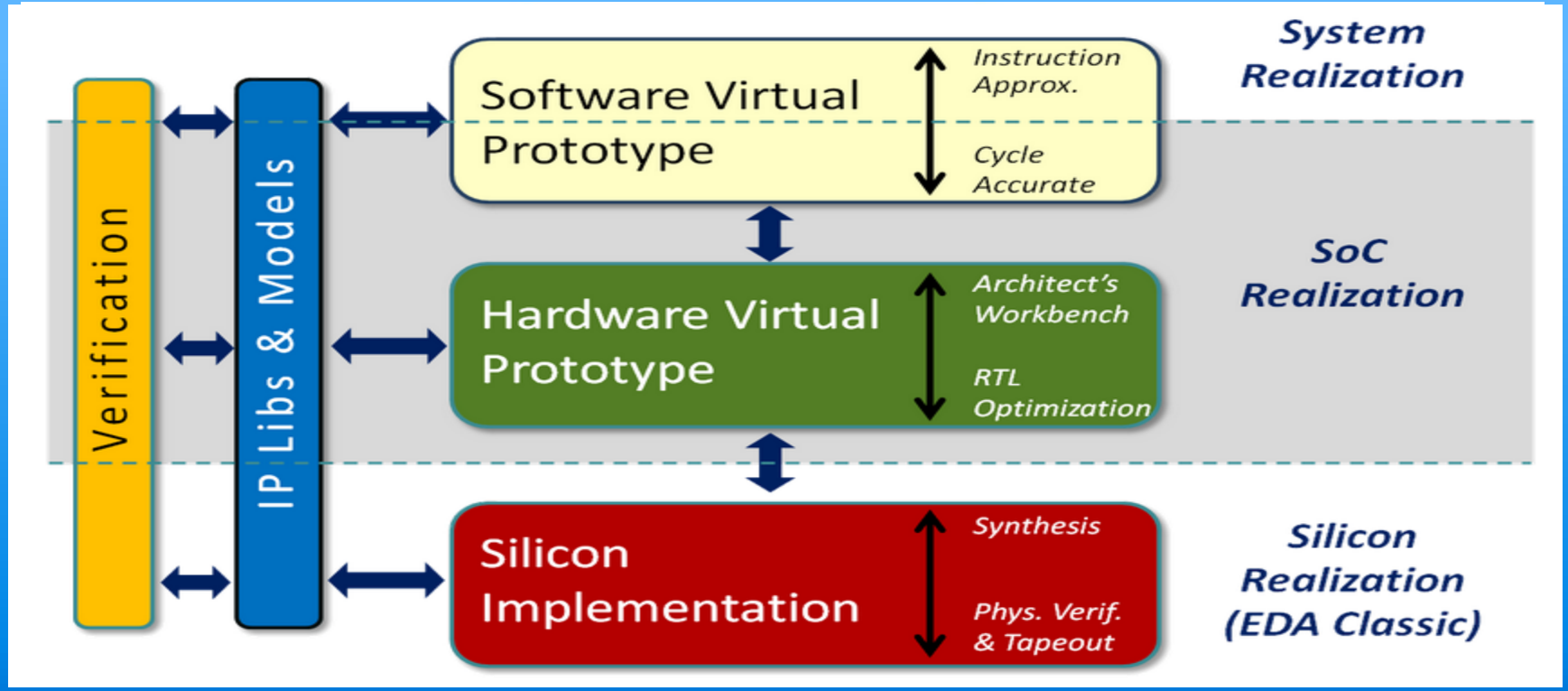
Multi-engine environment (and Multi-user environment)



Unified Coverage with S/VCS



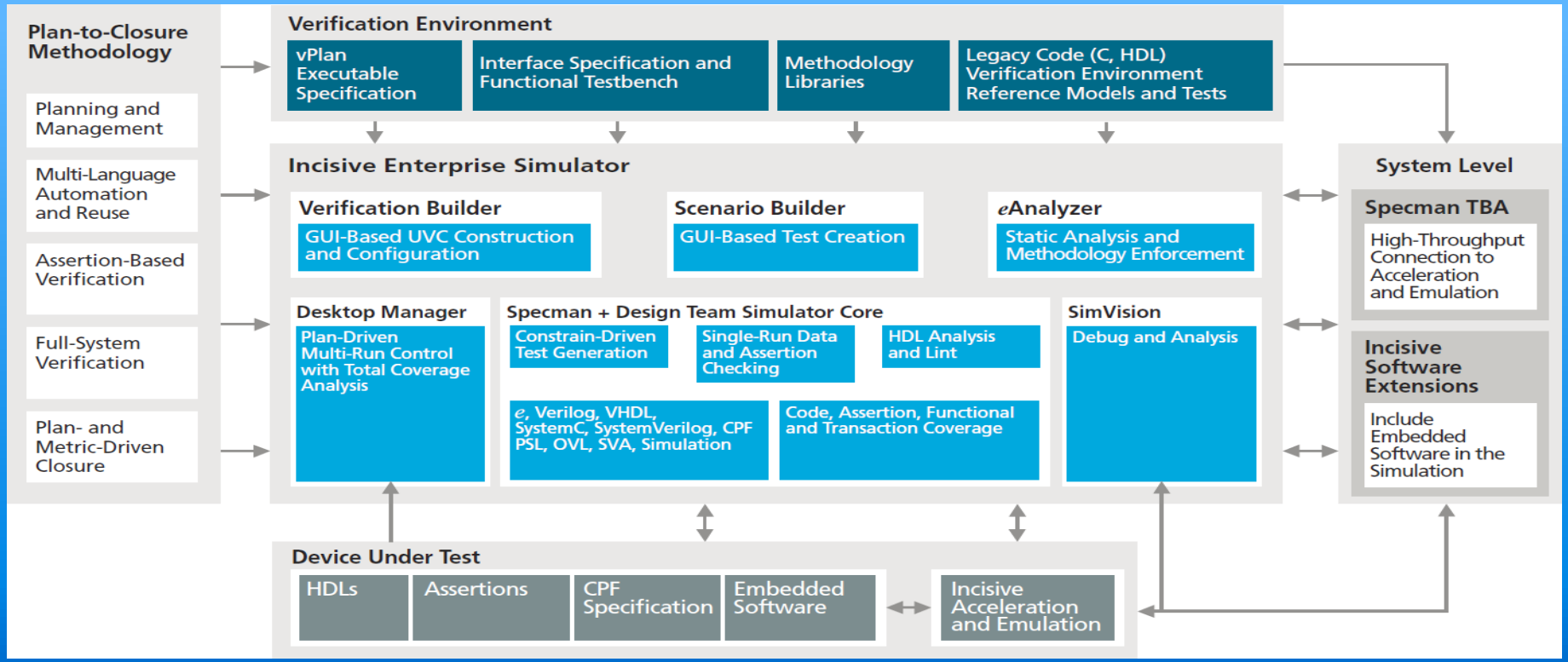
Synopsys - Atrenta SpyGlass Solution



IDM & IC Chain

- **SoC Design and Verification:** HW/SW Co-design
- **SoC Verification Flow:** module, system integration
- **Verification Methodology:** ABV/TBV and TLM/UVM
- **Advanced Verification Methods:** MS-SoC
- **Discussion**

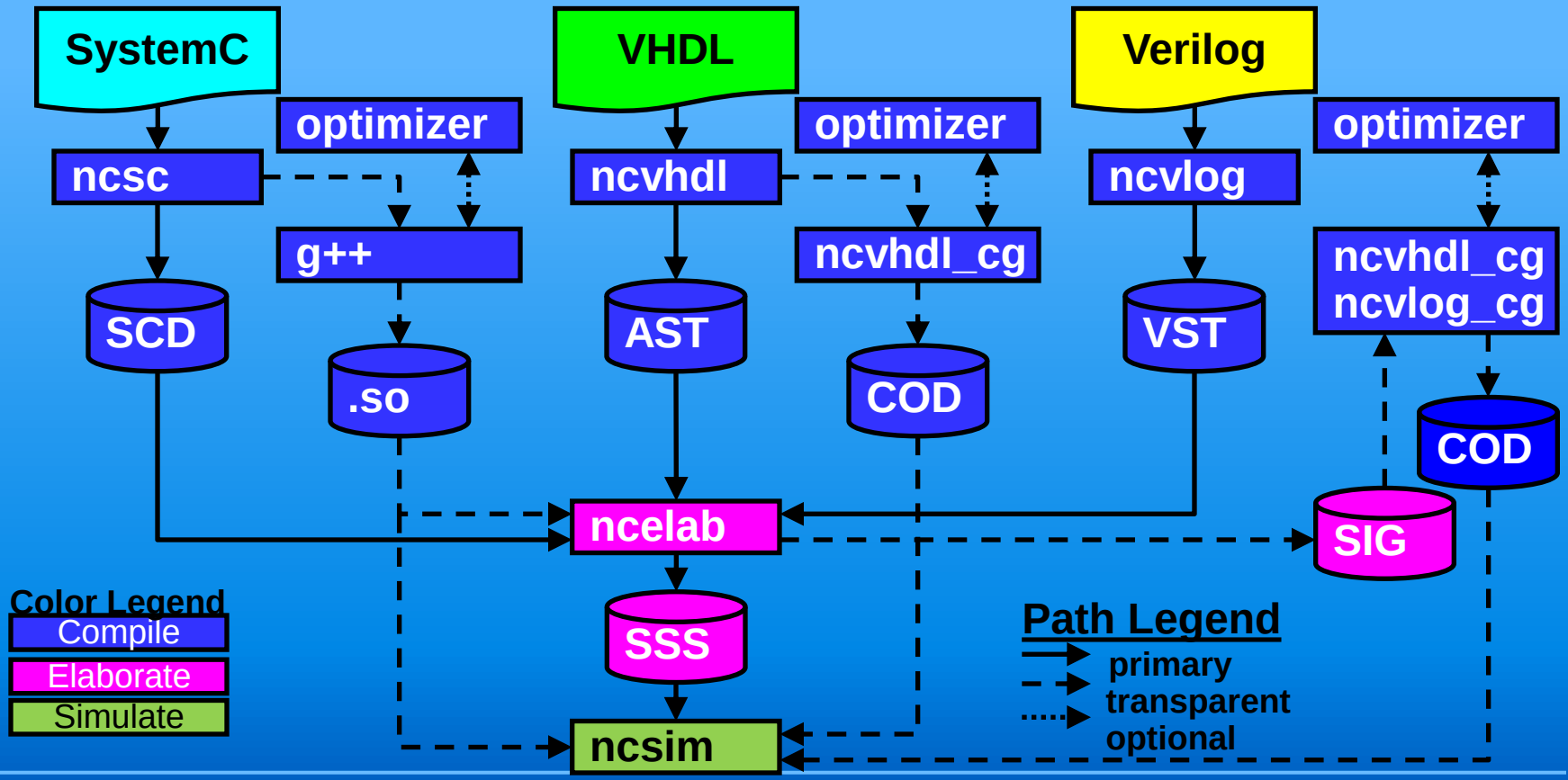
The Incisive Enterprise Simulator – XL



Course Prerequisites

- What you need to already have:
  - Basic computer literacy — you must know how to use a shell and editor of your choice and navigate the file system
  - A basic understanding of digital hardware design and verification
  - Knowledge of a hardware description language to facilitate your learning experience

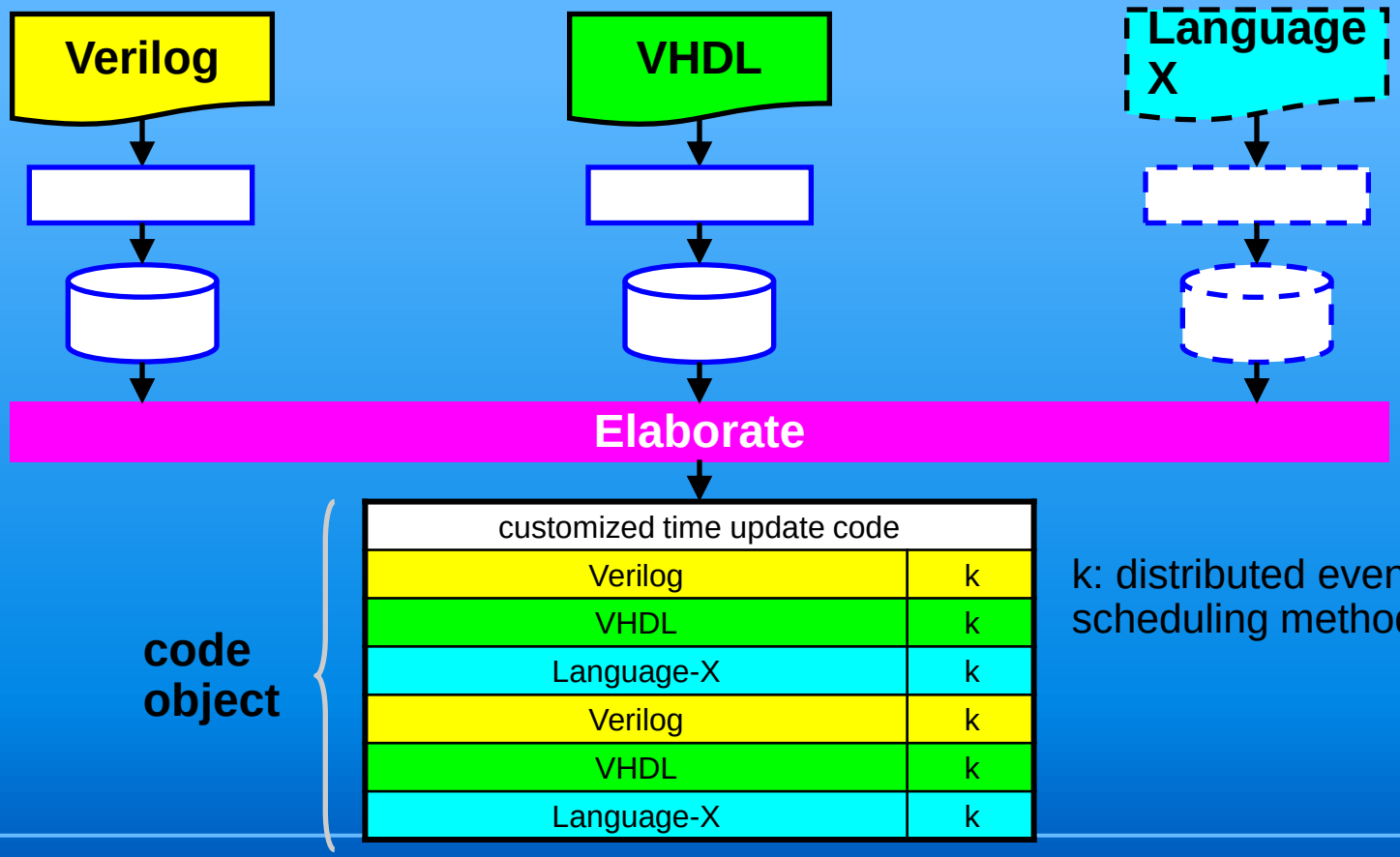
The Simulator Tool Flow



HDL Verification Methods

- SNPS:
  - VCS, SpyGlass; Verdi
- MENT:
  - ModelSim; Debussy (debug)/nLint (pragma check)
- CDNS:
  - Incisive [NC-Verilog]; HAL

How Native Compiled Code Is Interleaved



Running the Simulator

- Running the simulator is separated into three major steps:
  - Compilation with **ncsc**, **ncvhdl**, or **ncvlog**

    Checks syntax and semantics

    Creates design data objects (SCD, AST, VST)

    Creates SystemC and VHDL code objects (.o, COD)
  - Elaboration (expansion and linking) with **ncelab**

    Constructs design hierarchy and connects signals

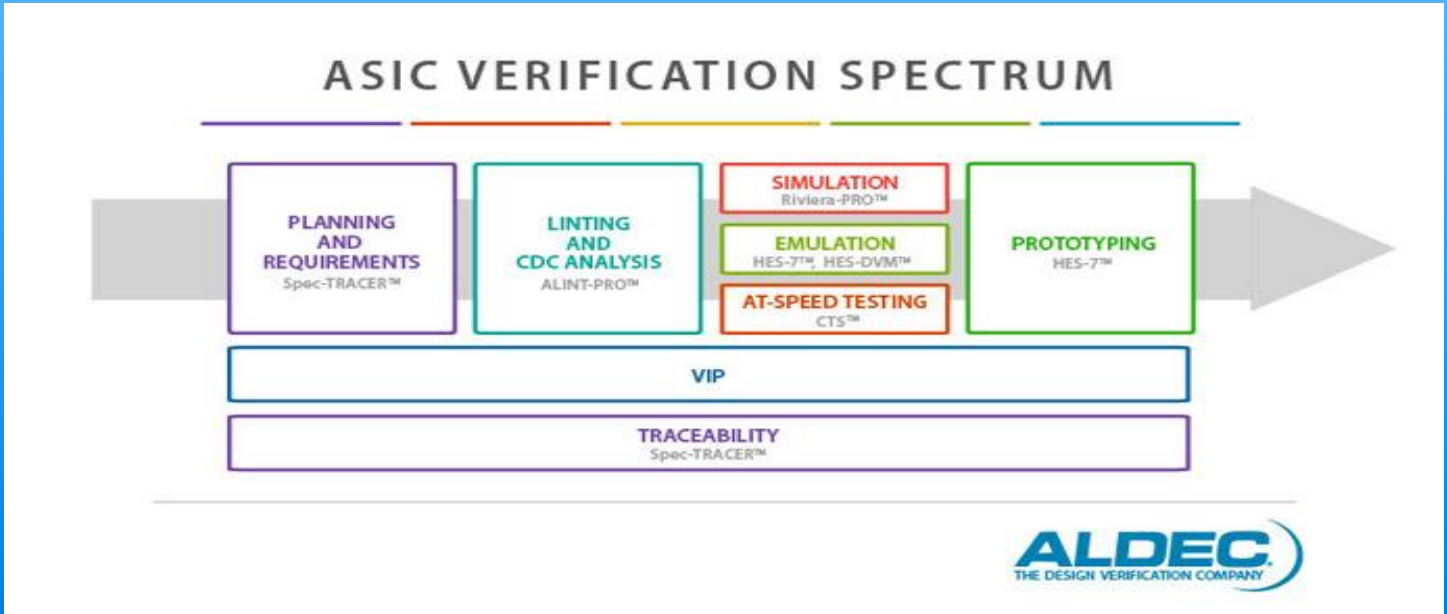
    Creates signature object (SIG) and Verilog code object (COD)

    Creates initial simulation snapshot object (SSS)
  - Simulation with **ncsim**

    Executes simulation code

FPGA vs. ASIC Verification

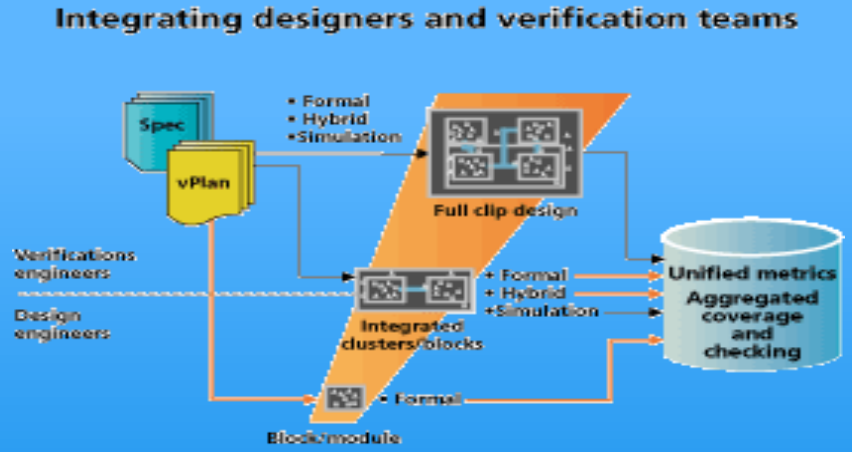
- From FPGA verification to ASIC one

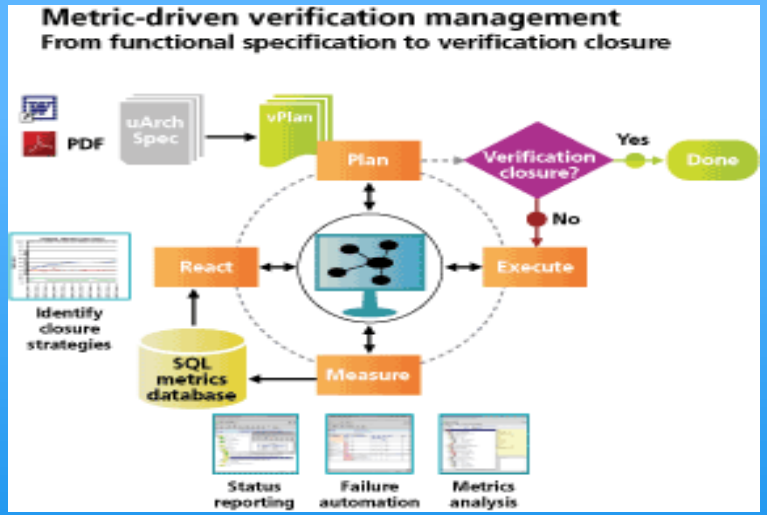


IDM & IC Chain

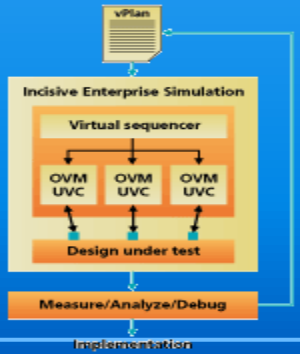
- **SoC Design and Verification:** HW/SW Co-design
- **SoC Verification Flow:** module, system integration
- **Verification Methodology:** ABV/TBV and TLM/UVM
- **Advanced Verification Methods:** MS-SoC
- **Discussion**

Incisive Functional Verification Flows

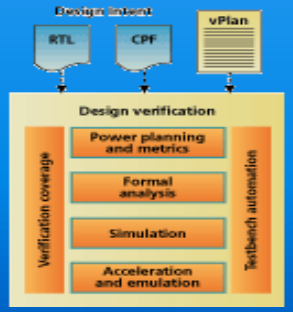




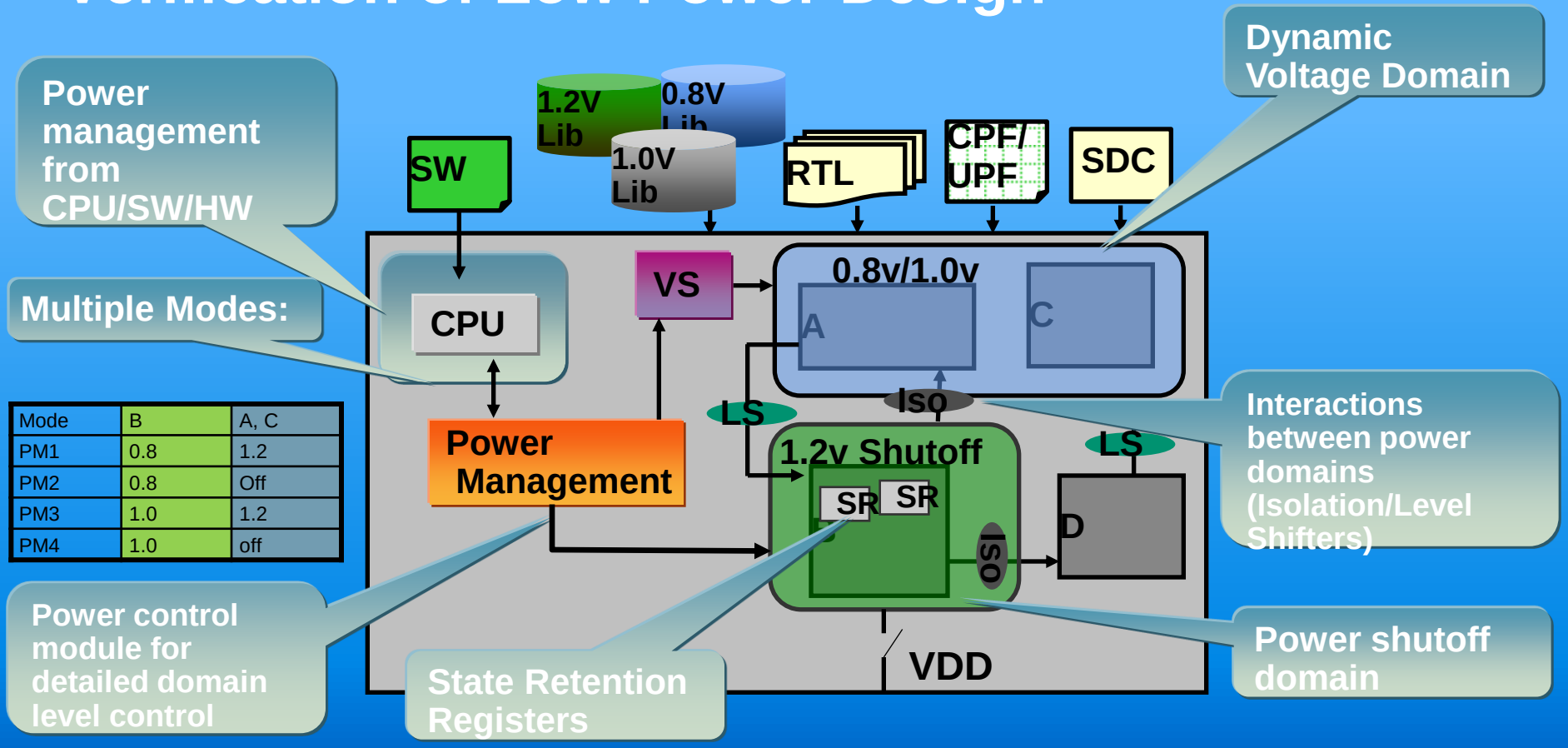
Open Verification Methodology flow with Incisive Enterprise Simulator



Low-power verification flow

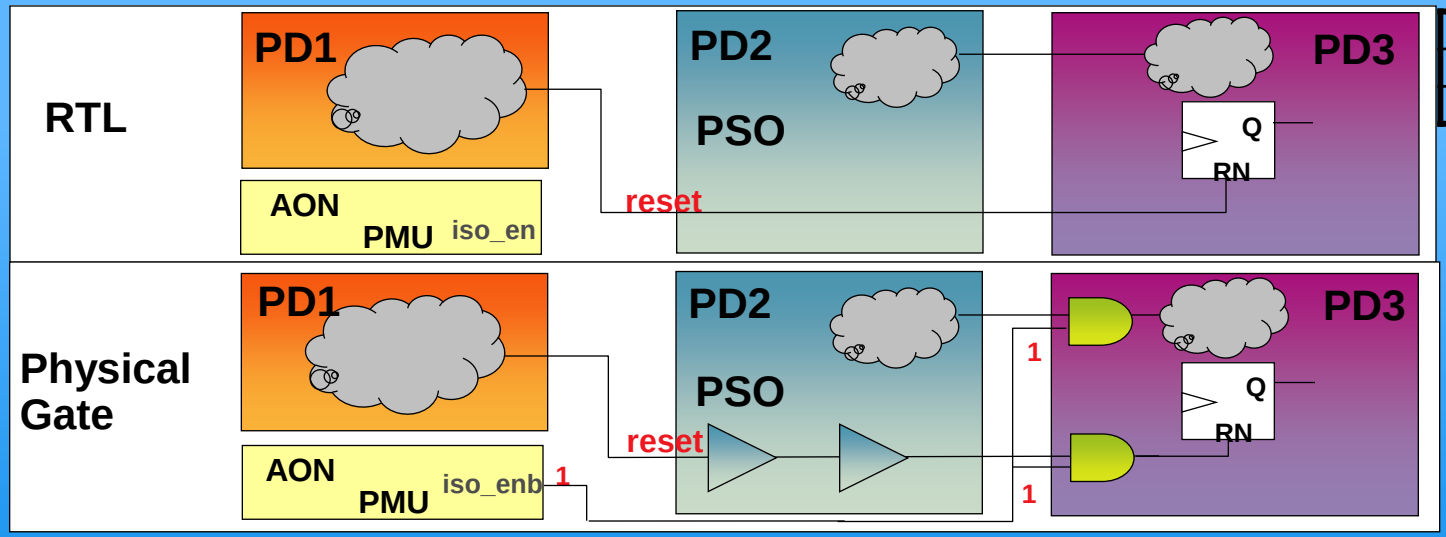


Verification of Low Power Design



Mode	B	A, C
PM1	0.8	1.2
PM2	0.8	Off
PM3	1.0	1.2
PM4	1.0	off

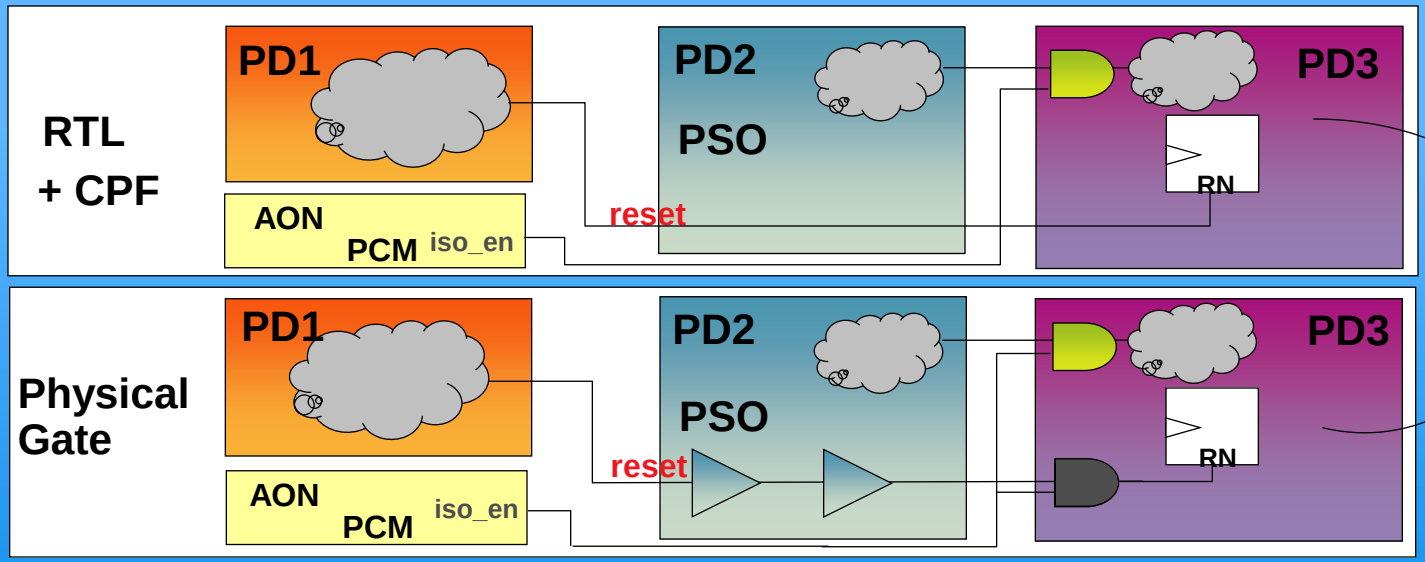
Ad-hoc Equivalence Checking



Mode	PD1	PD2	PD3
PM1	1.0	1.0	1.0
PM2	1.0	0.0	1.0

- RTL does not have low power logic inserted
  - *create_isolation_rule r1 -from PD2 -to PD3 -isolation_condition {i_pmu/iso_enb} -isolation_output low*
- Place and Route Step
  - Buffers inserted on reset line in PD2
  - Isolation inserted in PD3 based on CPF isolation rule
- Ad-hoc EC
  - iso_en='1' signal to disable isolation logic
  - Reports design is equivalent
- In fact, it is not equivalent because reset behavior changes when PD2 is off and PD3 is on
  - Feedthrough reset signal should not be isolated

Power Aware Equivalence Checking



Mode	PD1	PD2	PD3
PM1	1.0	1.0	1.0
PM2	1.0	0.0	1.0

NEQ

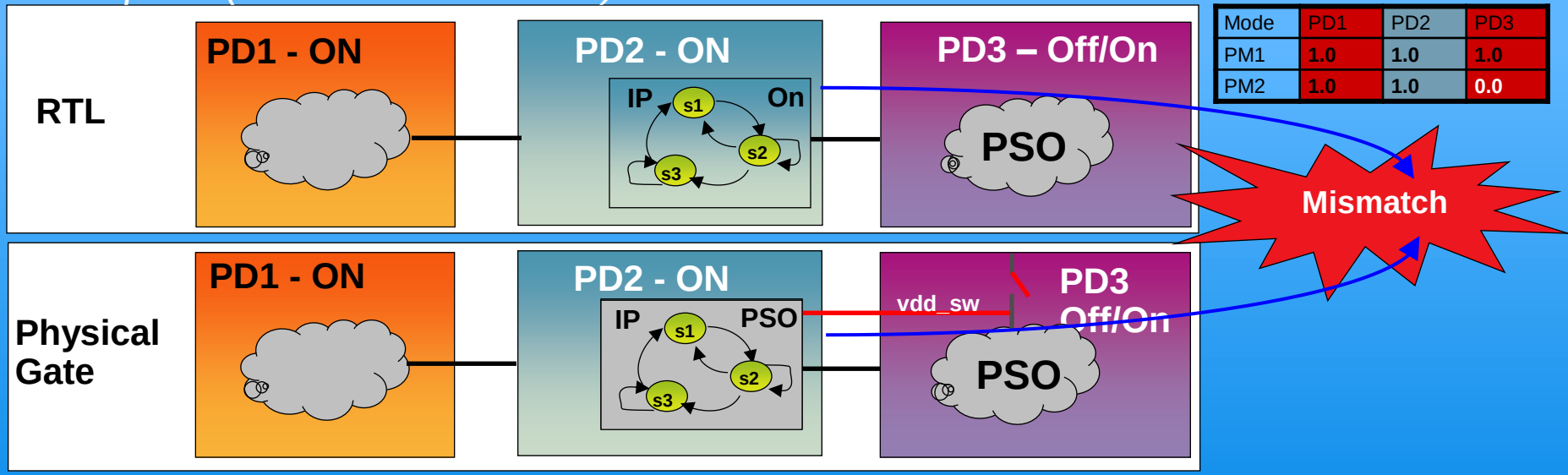
- EC tool inserts isolation logic per isolation rule

```
create_isolation_rule r1 -from PD2 -to PD3 -isolation_condition {i_pmu/iso_enb} \
-isolation_output low
```

- Place and Route Step
  - Buffers inserted on reset line in PD2
  - Isolation inserted in PD3 based on CPF isolation rule
- RTL2Gate CLP EC reports design is non-equivalent

The Need For Domain Aware Equivalence Checking

Example 2 (Actual Silicon failure)



Mode	PD1	PD2	PD3
PM1	1.0	1.0	1.0
PM2	1.0	1.0	0.0

- IP block in PD2 was connected incorrectly to PD3 switchable power net *vdd_sw* during PnR instead of PD2 primary power net
- When PD3 is shut off, registers in IP block will lose state in silicon
- Domain Aware equivalence checking finds this problem
  - Customer used Ad-hoc equivalence checking flow did not run CLP EC

IDM & IC Chain

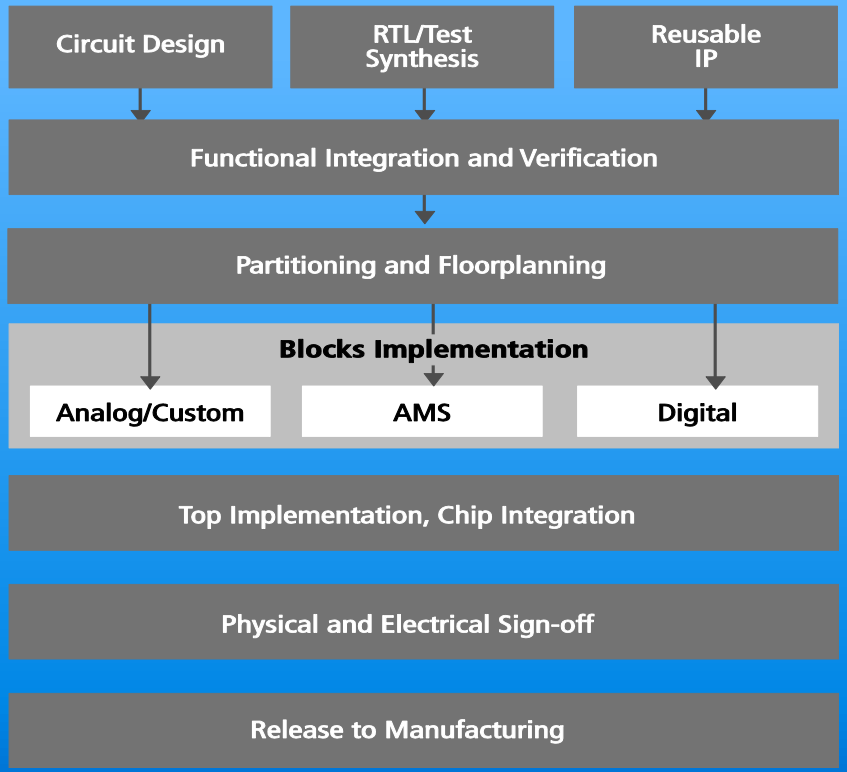
- SoC Design and Verification: HW/SW Co-design
- SoC Verification Flow: module, system integration
- Verification Methodology: ABV/TBV and TLM/UVM
- Advanced Verification Methods: MS-SoC
- Discussion

Features of SoC Implementation

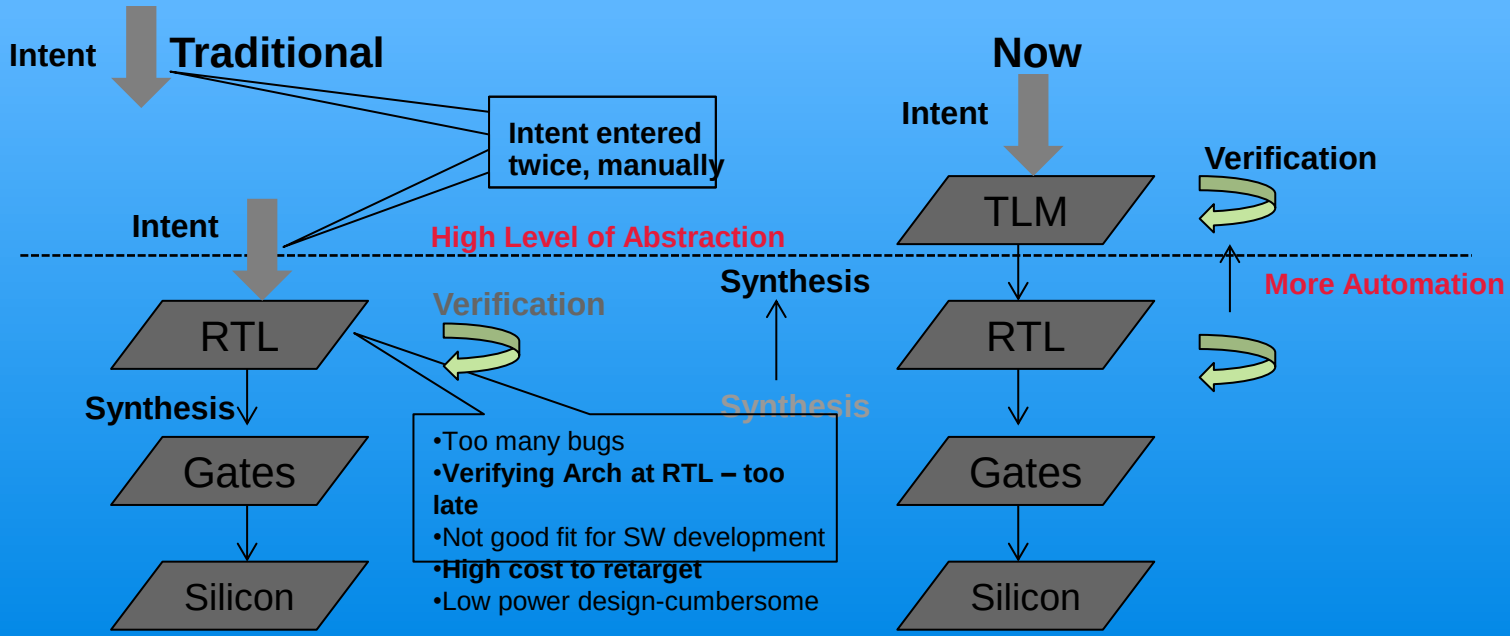
Analog & Circuit
SPICE Level Focused

Digital and DFT
Gate Level Focused

IP Integration and Reuse
Block Level Focused

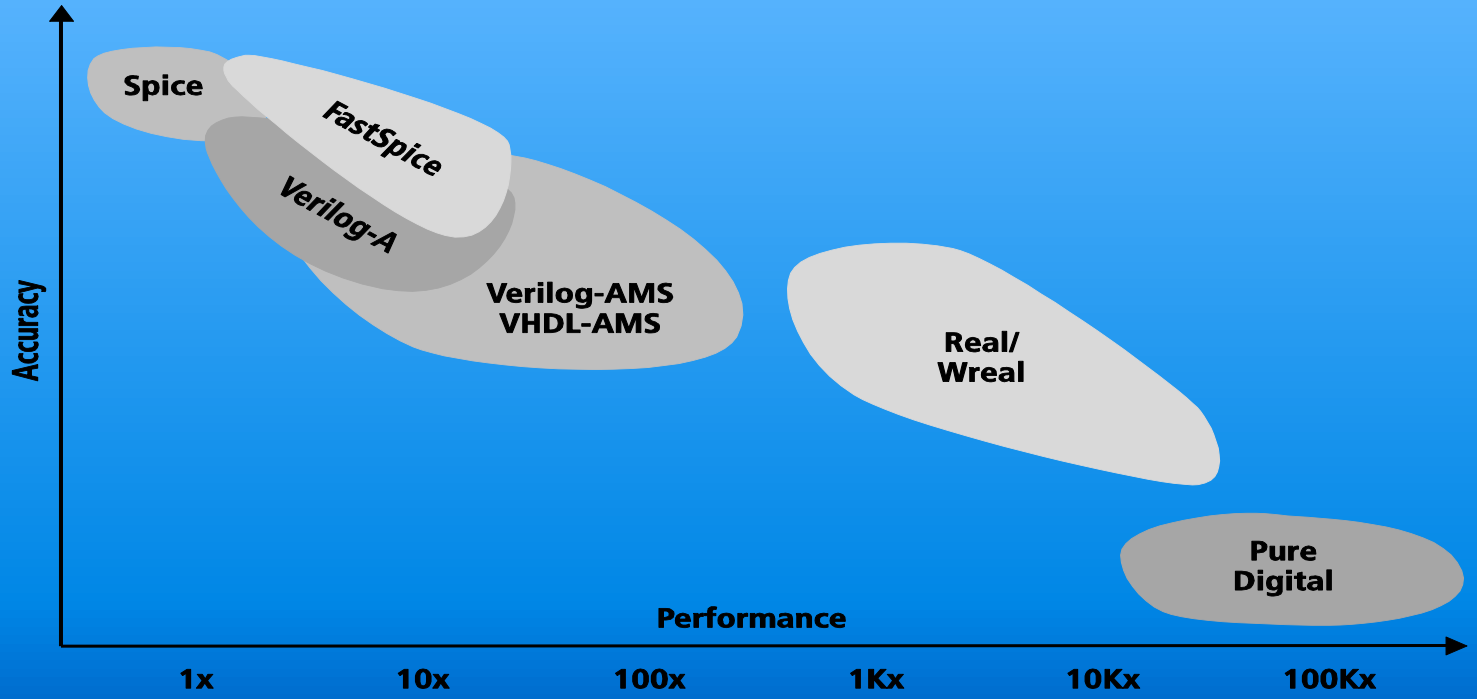


Verification Flow in SoC Designs



Redrawn after Bailey et al 2010 (TLM-Driven Design …)

Accuracy of Verification Methods



Development and Unified Verification Methodology

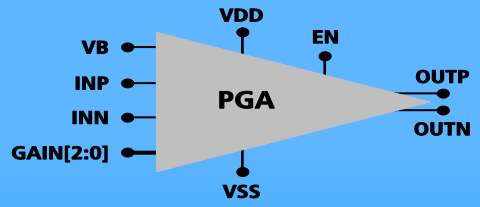
- 2000, Verisity Design (now Cadence Design System) introduced Verification Advisor, using *e* language, which included generation of stimuli, strategy of auto comparison, and covering models. E language is an object language - V Advisor started measuring platform with an object language
- 2002, Verisity published a first verif. library – e reuse methodology, eRM
- 2003, Synopsys published reusable verif. meth., RVM, using its internal Vera langauage
- 2006, Mentor announced advanced verif. meth., AVM. This methodology adopted TLM standard of OSCI SystemC, AVM used both SystemVerilog and SystemC
- 2006, Synopsys promoted a verif. meth. manual, VMM, which transformed RVM in Vera to SystemVerilog
- 2007, Cadence promoted universal reusable meth., URM, mainly to transfer eRM in e lang. to SystemVerilog, and added TLM interface, factory model replacement, matching mechanism, and other features
- 2008, Cadence and Mentor worked together to promote open verif. meth., OVM
- 2010, ACCELLERA adopted OVM and promoted UVM, and introduced 'callbacks' concept of VMM, established an unified industry prototype methodology
- **UVM: Universal Verification Methodology**

IDM & IC Chain

- **SoC Design and Verification:** HW/SW Co-design
- **SoC Verification Flow:** module, system integration
- **Verification Methodology:** ABV/TBV and TLM/UVM
- **Advanced Verification Methods:** MS-SoC
- **Discussion**

Verilog-AMS Model for PGA

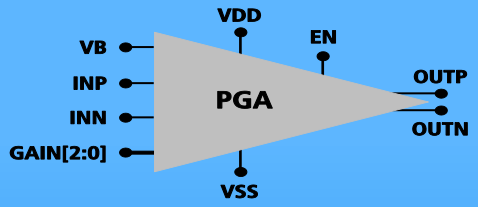
Since the Programmable-Gain Amplifier (PGA) is, by its nature, a mixed-signal system, the most natural approach to modeling it is using the AMS modeling approach. Here is the header of a model for the PGA using the Verilog-AMS language:




```
`include “disciplines.vams”
module PGA ( OUTP,OUTN, INP,INN, GAIN, VB, VDD,VSS, EN );
output OUTP,OUTN;       // differential output
input INP,INN;          // differential input
input [2:0] GAIN;       // digital control bus
input VDD,VSS,VB;       // power supplies & bias voltage input
input EN;               // output enable
electrical OUTP,OUTN, INP,INN, VB, VDD,VSS;
logic EN; logic [2:0] GAIN;
```

Note that the differential outputs and inputs, along with the supplies and bias input, are all defined to be of discipline electrical, while the gain bus and enable input are defined to be of logic. The disciplines themselves are actually described in the include file disciplines.vams, which is a standard part of the Verilog-AMS system. When a discipline of electrical is specified for a node, it indicates that it is a continuous analog node that has a potential of voltage by V() and can support …

Verilog-A Model for PGA



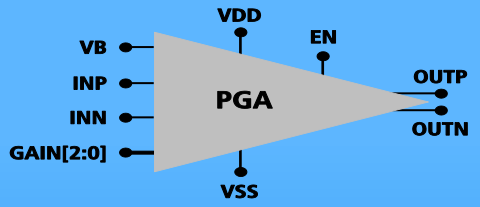
In Verilog-A format, the header information is nearly identical – the only difference is that the enable and gain pins are now declared to electrical as well:

```
`include "disciplines.vams"
module PGA ( OUTP,OUTN, INP,INN, GAIN, VB, VDD,VSS, EN );
output OUTP,OUTN;      // differential output
input INP,INN;         // differential input
input [2:0] GAIN;      // digital control bus
input VDD,VSS,VB;      // power supplies & bias voltage input
input EN;       // output enable
electrical OUTP,OUTN, INP,INN, VB, VDD,VSS, EN;
electrical [2:0] GAIN;
```

The parameters and variables are nearly identical, since they were all used for the analog modeling. There is only one additional parameter, Gint, which is the integer value (0 to 7) to be extracted from the gain bus input:

```
…
parameter real dbmin=-1,dbmax=20; // gains for VCVGA=000&111
…
integer Gint;  // integer value from gain bus
real Gout;     // output conductance
…
```

Wreal (Verilog-AMS) Model for PGA

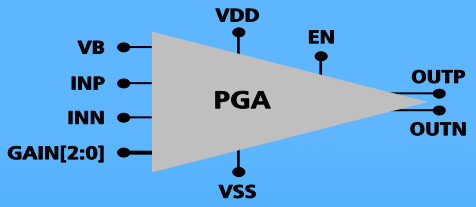


The Verilog-AMS language supports real number wires using the type of wreal. Internally the system passes the value as a 64-bit real number, and is equivalent to passing a digital value between blocks. The value is interpreted as a real number rather than as a bit or a bus. Here is the header for the Verilog-AMS/wreal implementation of the programmable gain amplifier:

```
module PGA ( OUTP,OUTN, INP,INN, GAIN, VB, VDD,VSS, EN );
output OUTP,OUTN;          // differential output
input INP,INN;           // differential input
input [2:0] GAIN;          // digital control bus
input VDD,VSS,VB;          // power supplies & bias voltage
input
input EN;              // output enable
wreal OUTP,OUTN, INP,INN, VB, VDD,VSS;
```

All of the pins that were previously electrical are now declared to have a type of wreal. Note that wreal is not a discipline – it is just a signal type, much like defining a wire to be a bus rather than a signal bit. wreal wires still, by default, have a discipline of logic associated with them. In this case we did not indicate the discipline for GAIN or EN either, so they would also pick up the defined discipline of logic.

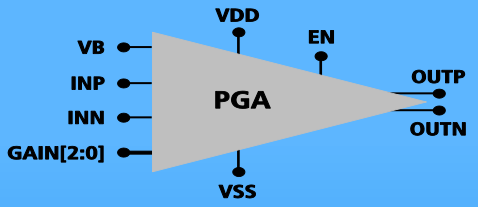
Digital Verilog Model for PGA



A typical approach to a digital model of the PGA may define that proper biasing requires a high input to the positive supply (VDD=1), a low input to the negative supply (VSS=0), and a high to indicate that the bias input is available (VB=1). … Here is a (digital Verilog) model that implements the discipline:

```
module PGA ( OUTP,OUTN, INP,INN, GAIN, VB, VDD,VSS, EN );
output OUTP,OUTN;          // differential output
input INP,INN;             // differential input
input [2:0] GAIN;          // digital control bus
input VDD,VSS,VB;          // power supplies & bias voltage input
input EN;                  // output enable
// note that all the signals are just digital here - no analog content.
parameter real dbmin=-1,dbmax=20;  // gains for VCVGA=000&111
real DBinc,Adb,Av;                 // terms in gain calculation
initial DBinc=(dbmax-dbmin)/7;     // compute per-bit change to gain
always begin
 if ((^GAIN)===1'bx) Adb=-40;      // low gain if invalid control
 else Adb=dbmin+DBinc*GAIN;        // compute gain in dB
 Av=pow(10,Adb/20.0);              // convert to V/V
 @(GAIN);                          // update on gain bus change
end
// Check if EN=1, VDD=1, VSS=0, VB=1, and GAIN bits are all valid:
wire Active = ({EN,VDD,VSS,VB}===4'b1101) && ((^GAIN)!==1'bx);
// Pass input to output if active; high impedance when disabled,
// any other case output goes to invalid:
assign OUTP = Active? INP : !EN? 1'bz : 1'bx;
assign OUTN = Active? INN : !EN? 1'bz : 1'bx;
endmodule
```

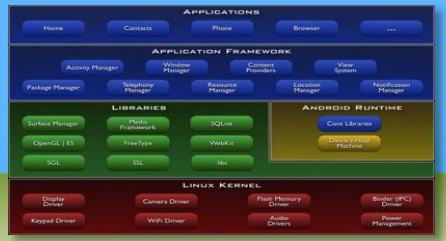
SPICE Model for PGA



SPICE cdl netlist

```
* File: PDIODE. netlist
* Created: Fri Sep 28 13:04:39 2002
* Program “xxx xRC"
* Version "v2009.xx“
*
.include "PDIODE.pex“
.subckt PDIODE  VSSIO VDDIO …
N_VSSA_d11_pos
N_VSSIO_d11_neg
PDIO18
AREA=3e-11
PJ=4.3e-05d8
N_VSSIO_d8_pos
N_VSSA_d8_neg
PDIO18
AREA=3e-11
PJ=4.3e-05d0
N_VDDA_d0_pos
N_VDDIO_d0_neg
… …
```

System Verification Platform



SOFTWARE

CONCEPT

PRODUCT

Architectural-level (Pre-RTL) development

SoC and System Development

Prototype-level (Post-RTL) development

CtoSilicon HLS

System Development Suite

Virtual System Platform *VSP*

Functional Verification Platform *Incisive®*

Verification Computing Platform *Palladium® XP*

Rapid Prototyping Platform *RPP*

Mixed TLM/RTL simulation, debug

Compile, hot-swap, debug

Compile, debug and adapters

HARDWARE

System Development Suite:

Elektra Awards Product of the Year Finalist

Electronic Products Magazine Product of the Year Finalist

Virtual System Platform:

Editors Choice by Journal of Embedded Signal Processing

ACE Awards Software Product of the Year Finalist

ACE AWARDS

Elektra2011 European Electronics Industry Awards FINALIST