SYNOPSYS®
Silicon to Software™

Streamlining Low Power Verification: From UPF to Signoff

March 2024

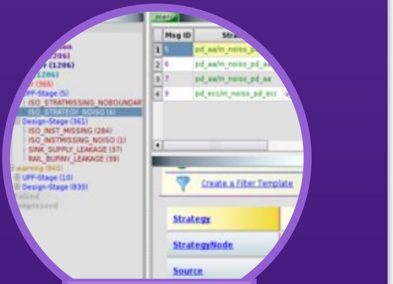
Synopsys' Low Power Verification

Find Power Bugs Pre-Silicon

Common UPF Front-End

UPF Architect

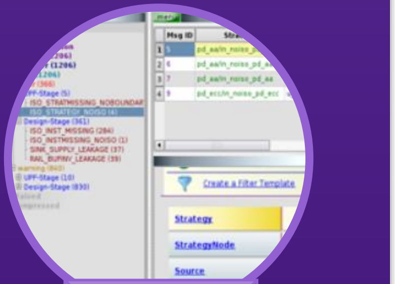
- Structural checks
- Functional checks
- Architectural checks
- PG checks



STATIC

VC LP

- Structural checks
- Functional checks
- Architectural checks
- PG checks



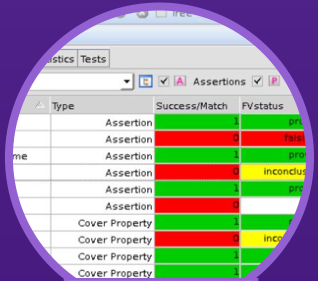
STATIC

VC SpyGlass

- Structural checks
- Functional checks
- Architectural checks
- PG checks

VC Formal

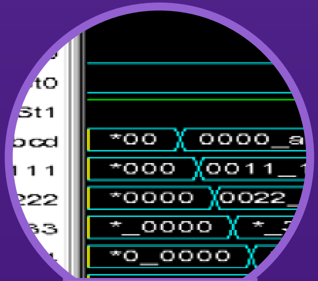
- LP Connectivity checks
- Formal LP property checks



FORMAL

VCS

- Power sequence
- Low Power Coverage
- Low Power Assertions
- PST verification



SIMUL

Integrated Verdi Low Power Debug

SYNOPSYS®
Silicon to Software™

Godwin Maben

Synopsys Fellow

Market Needs Driving E2E Power Flow

ULTRA LOW ENERGY

Power a differentiator

Battery life

Power-Perf tradeoff

Apps/features

Form factor

Competitiveness

“Every mW counts”

ENERGY EFFICIENCY

Power a limiting factor

Perf-Power trade-off

Carbon footprint

Thermal

Reliability

Cost – cooling, package

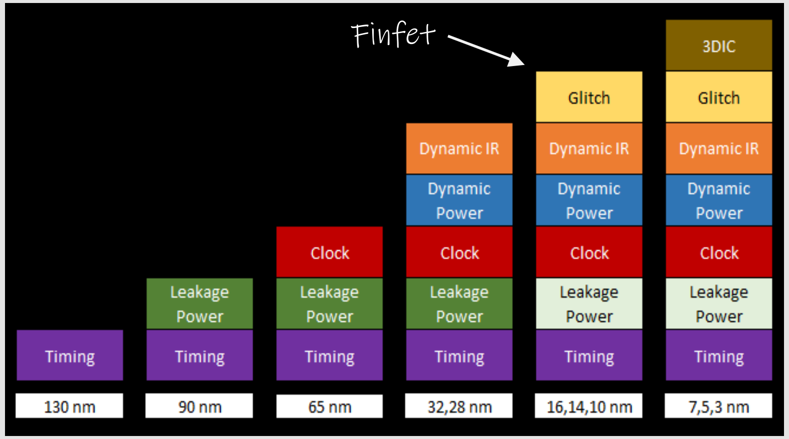
“Must manage power”

Power – The Next Generation Challenges

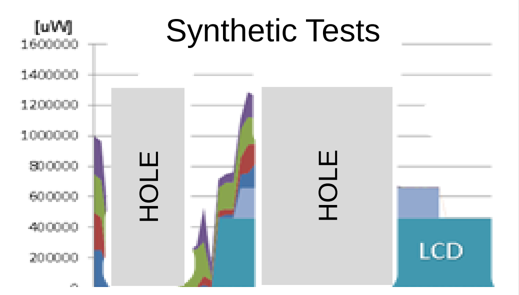
Shrinking technology nodes have increased chip power density, IR drop problem, and a significant jump in glitch power.

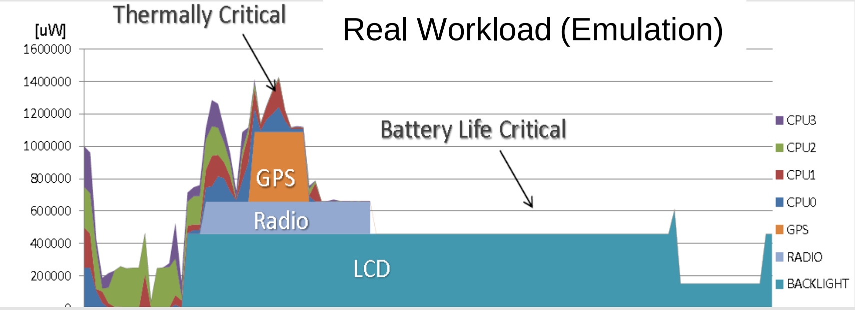
Synthetic simulation vectors often miss cross IP power bugs, firmware bugs, and overestimate peak power. This necessitates use of real workloads for power analysis and signoff.

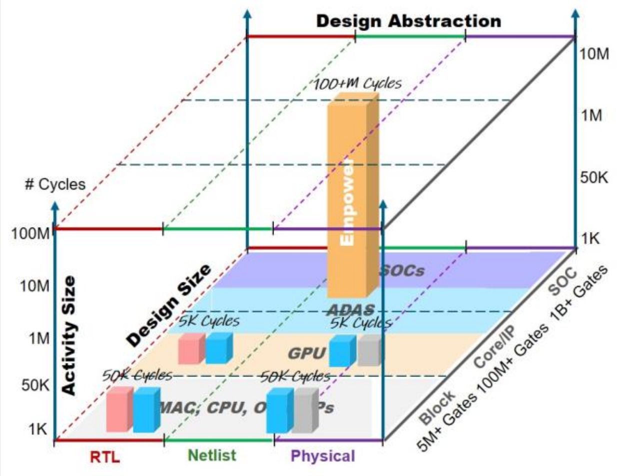
Growth in design and workload sizes makes cycle power analysis very memory and compute intensive, necessitating scalable distributed Power Analysis solution



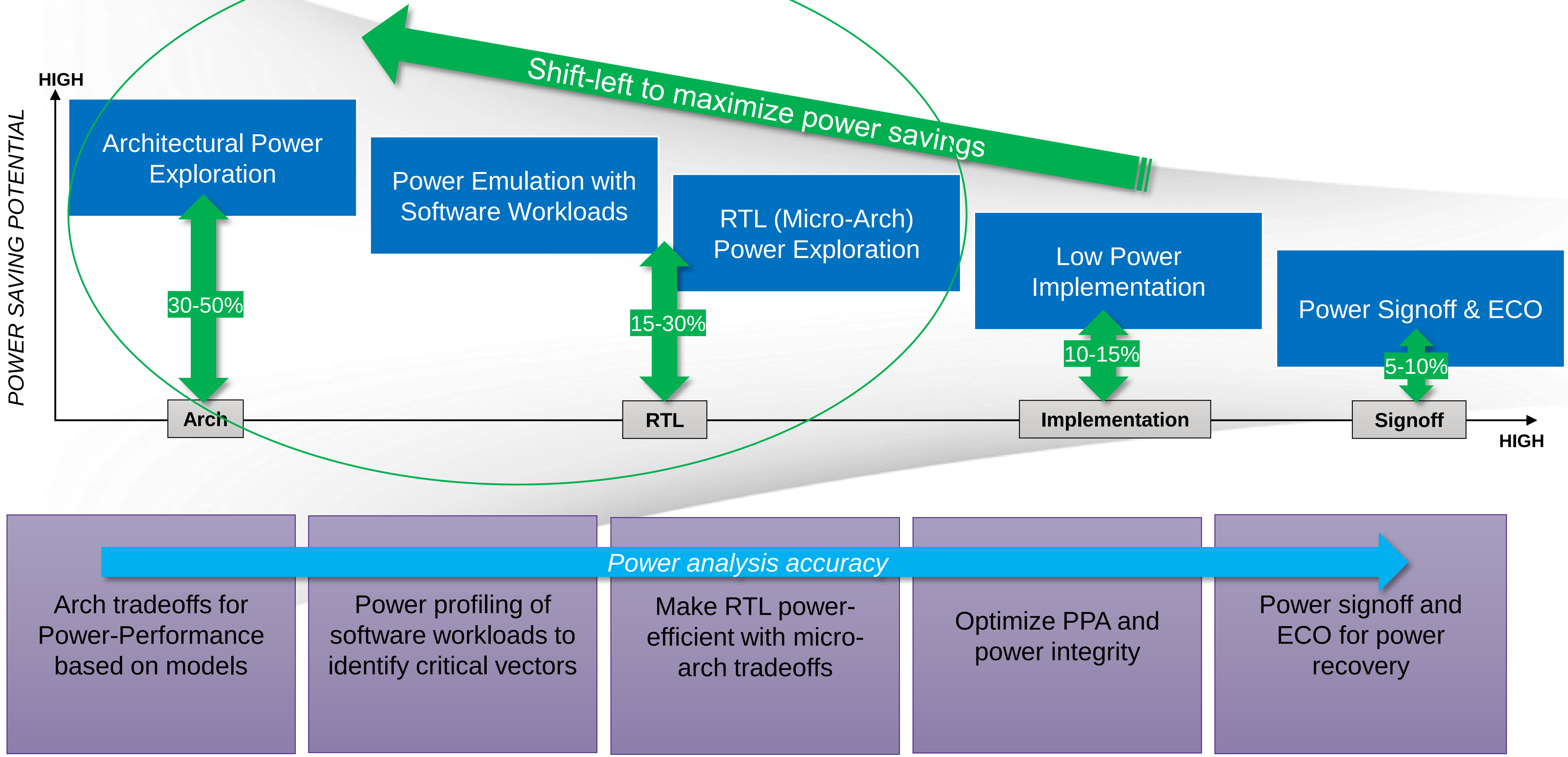
Technology Nodes →







Typical Low Power ASIC Design Flow



Shift-left to maximize power savings

HIGH

POWER SAVING POTENTIAL

Architectural Power Exploration

Power Emulation with Software Workloads

RTL (Micro-Arch) Power Exploration

Low Power Implementation

Power Signoff & ECO

30-50%

15-30%

10-15%

5-10%

Arch

RTL

Implementation

Signoff

HIGH

Power analysis accuracy

Arch tradeoffs for Power-Performance based on models

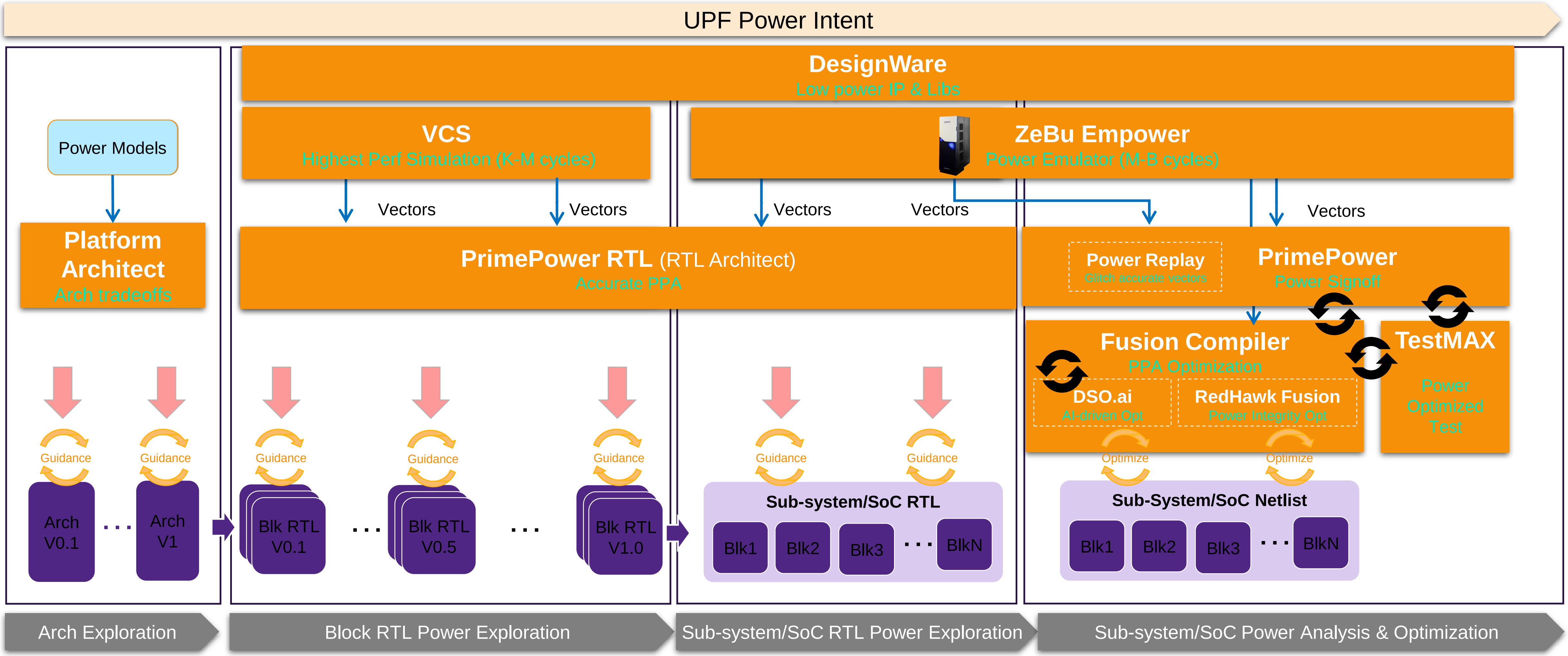
Power profiling of software workloads to identify critical vectors

Make RTL power-efficient with micro-arch tradeoffs

Optimize PPA and power integrity

Power signoff and ECO for power recovery

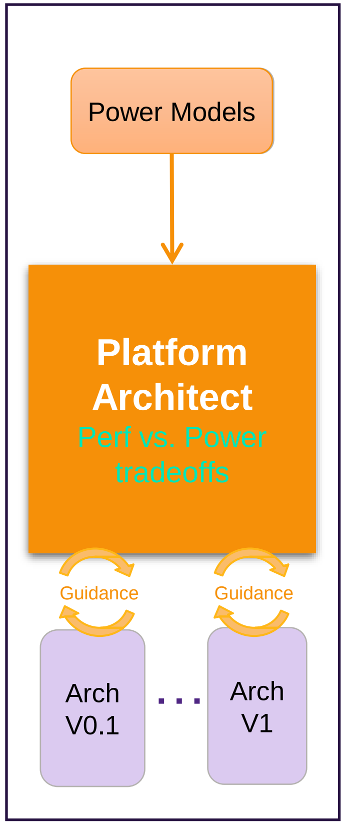
Software-Driven Power Exploration, Analysis & Optimization



Key Areas for Low Power Design

- Architectural Analysis and Power Efficient Macro Architecture Selection
- Workload Analysis/Profiling/Selection
- RTL Power Analysis and Optimization Guidance
- Power Optimization during Synthesis/P&R and ECO
- Power Signoff and Design Closure

Architectural Performance/Power Exploration



Power Models

Platform Architect

Perf vs. Power tradeoffs

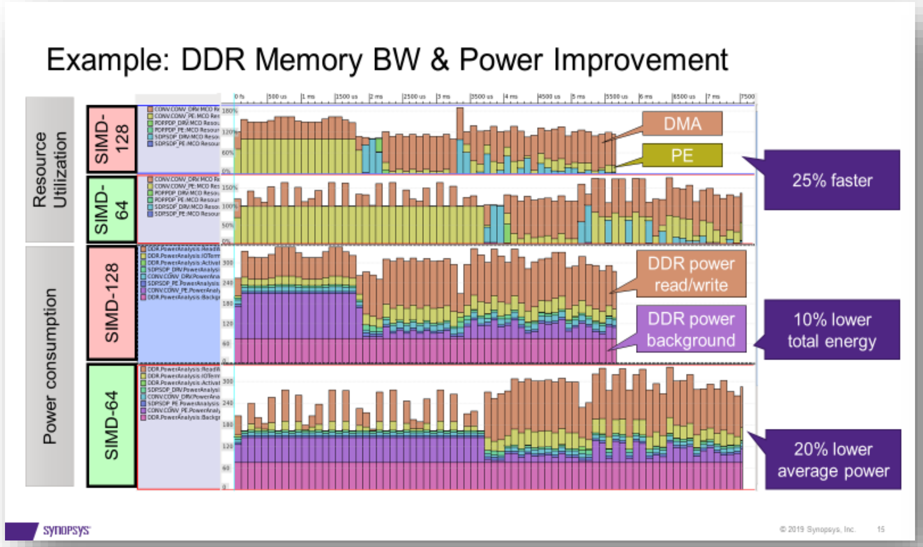
Guidance

Guidance

Arch V0.1

. . .

Arch V1

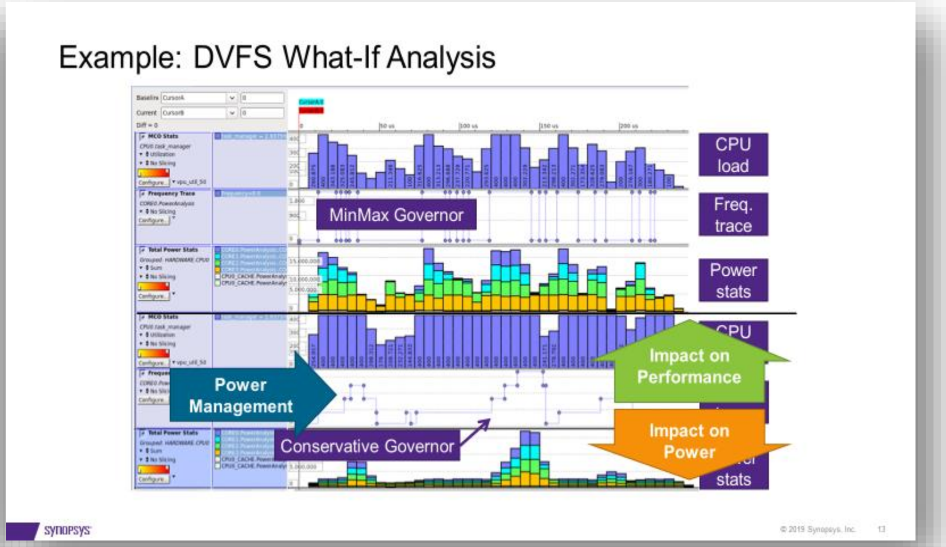


Example: DDR Memory BW & Power Improvement

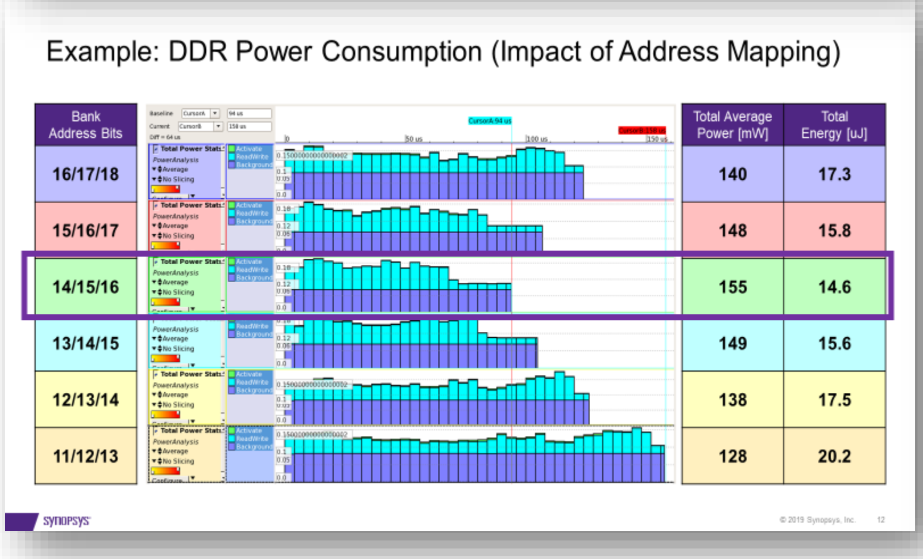
25% faster

10% lower total energy

20% lower average power

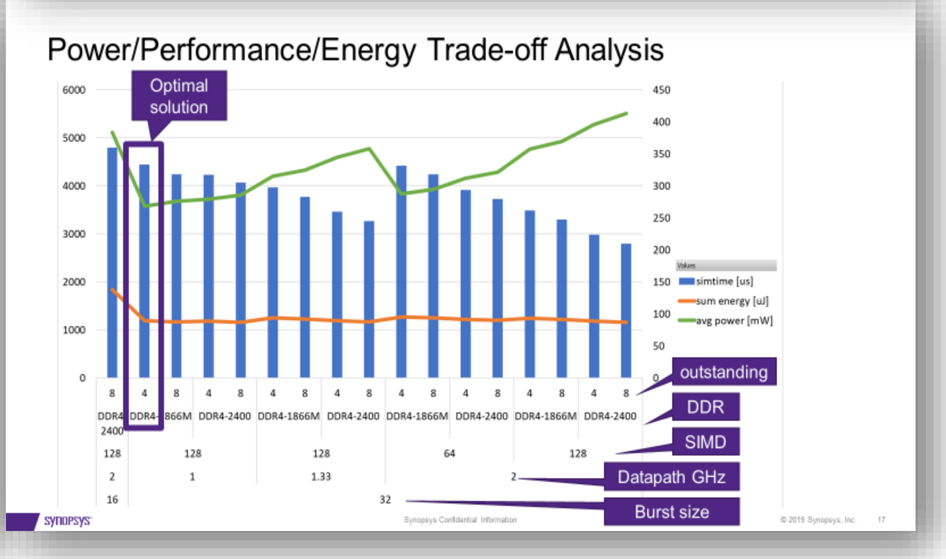


Example: DVFS What-If Analysis



Example: DDR Power Consumption (Impact of Address Mapping)

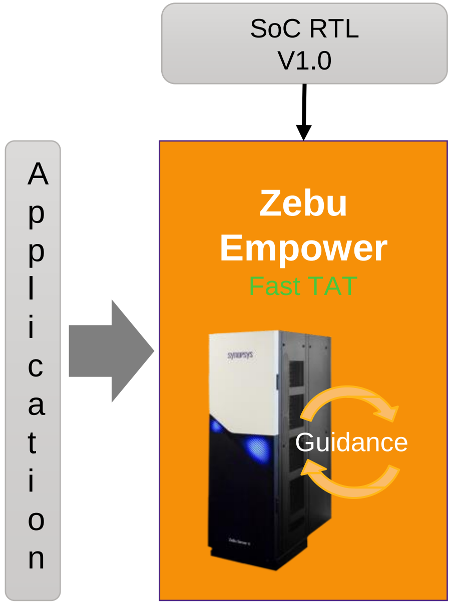
Bank Address Bits	Total Average Power [mW]	Total Energy [uJ]
16/17/18	140	17.3
15/16/17	148	15.8
14/15/16	155	14.6
13/14/15	149	15.6
12/13/14	138	17.5
11/12/13	128	20.2

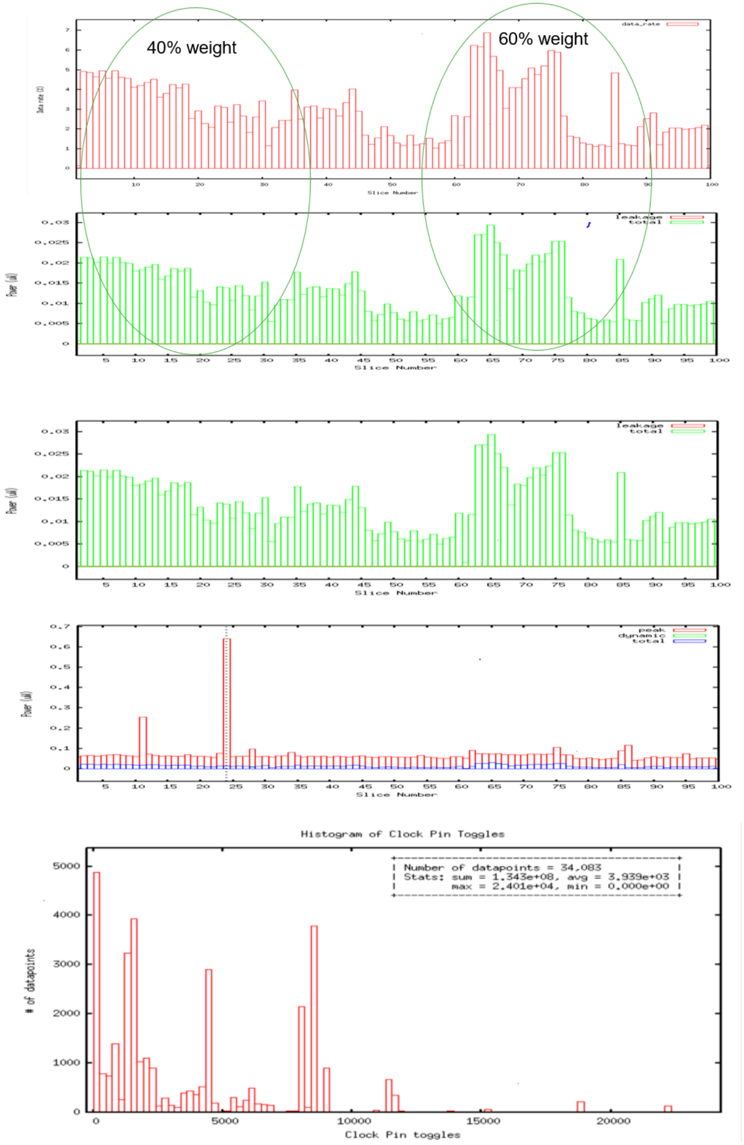


Power/Performance/Energy Trade-off Analysis

Optimal solution

Workload Analysis/Profiling/Selection Through Emulation

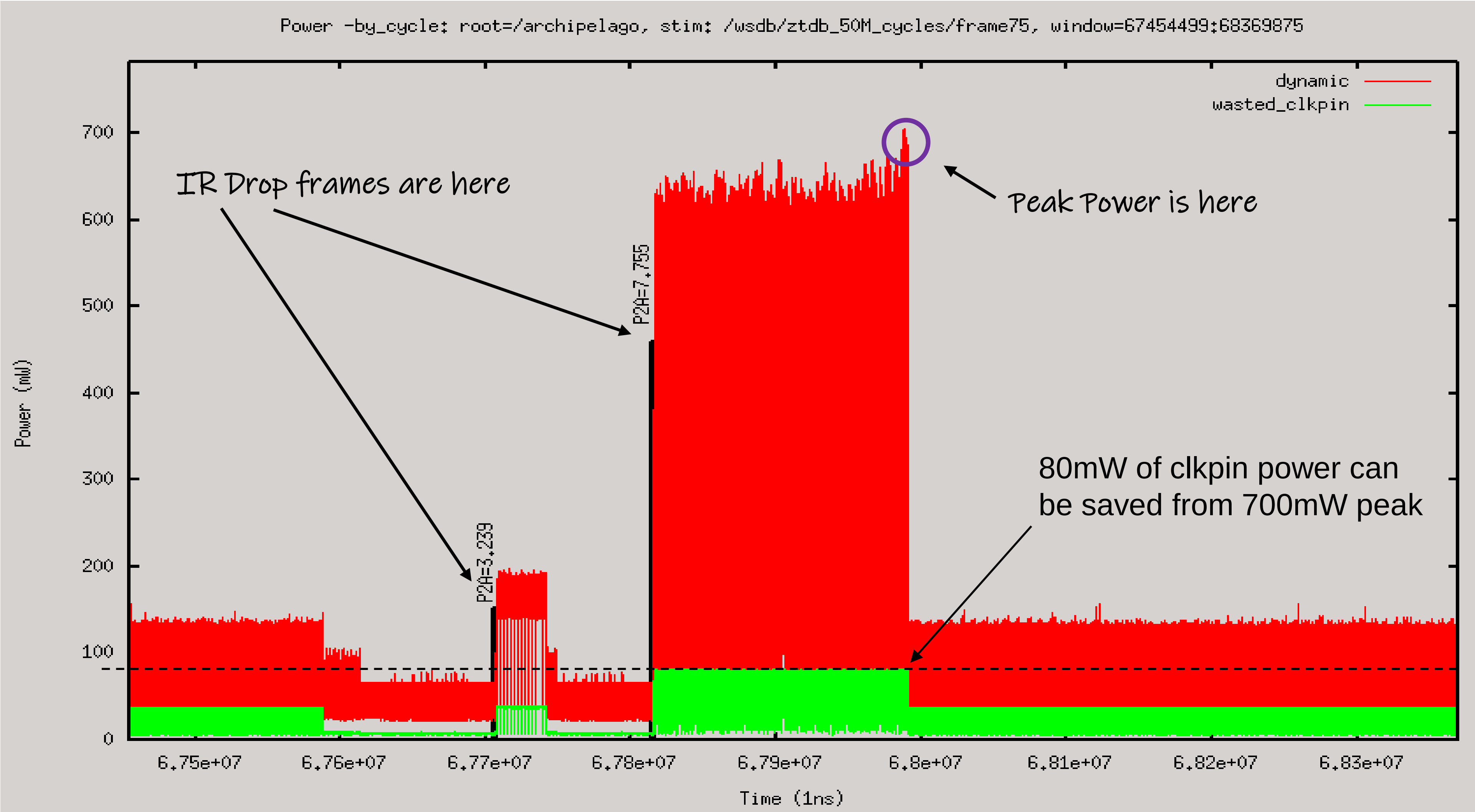




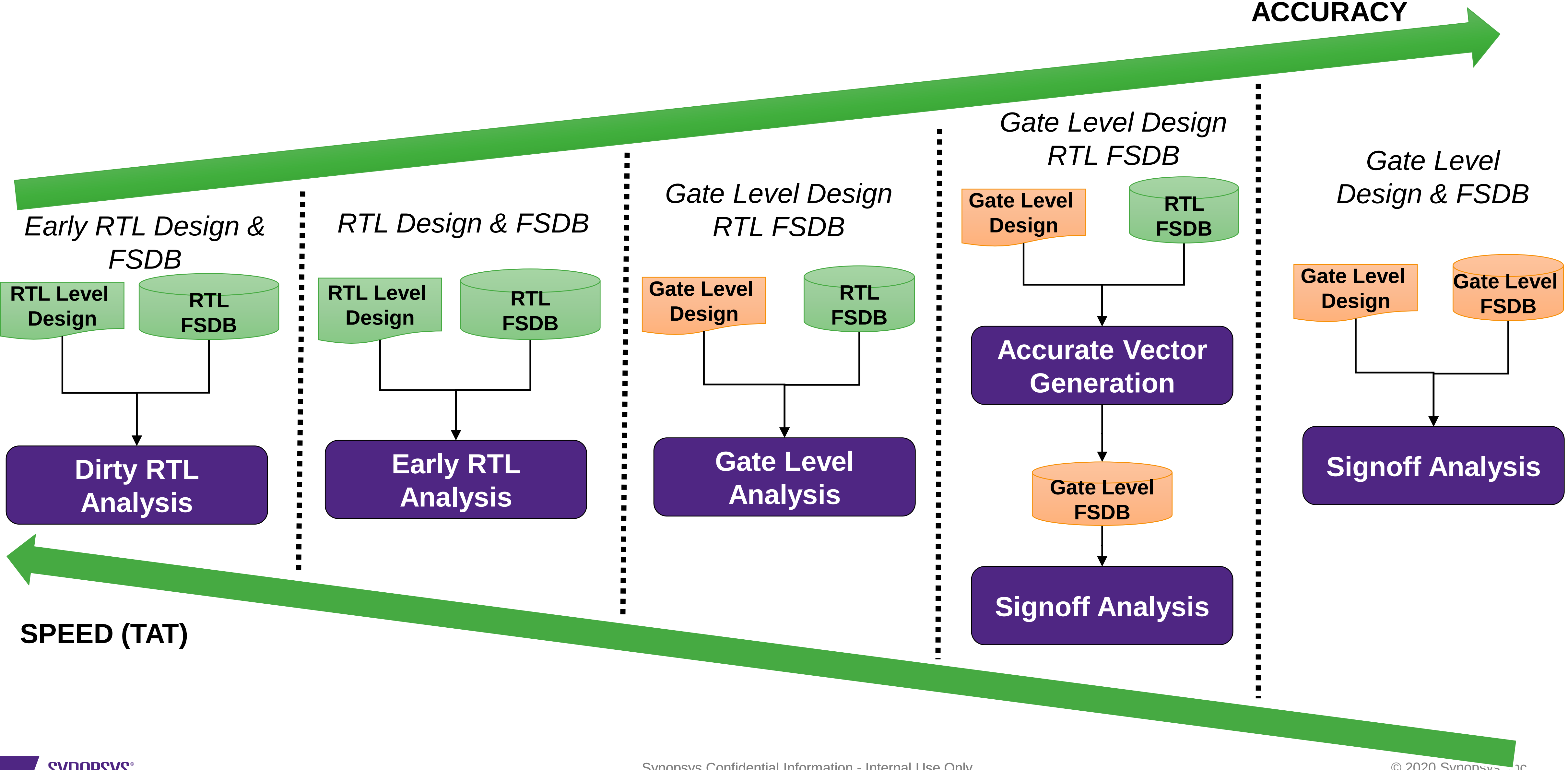
```
+---------------------------------------------------+
| Number of datapoints = 34,083                     |
| Stats: sum = 1.343e+08, avg = 3.939e+03           |
|        max = 2.401e+04, min = 0.000e+00           |
+---------------------------------------------------+
```

Vectors for Analysis and Optimization

Vector Profiling Using Emulation Workloads



Power Analysis Flows – Speed vs. Accuracy Tradeoffs



ACCURACY

Early RTL Design & FSDB

RTL Level Design

RTL FSDB

Dirty RTL Analysis

RTL Design & FSDB

RTL Level Design

RTL FSDB

Early RTL Analysis

Gate Level Design RTL FSDB

Gate Level Design

RTL FSDB

Gate Level Analysis

Gate Level Design RTL FSDB

Gate Level Design

RTL FSDB

Accurate Vector Generation

Gate Level FSDB

Signoff Analysis

Gate Level Design & FSDB

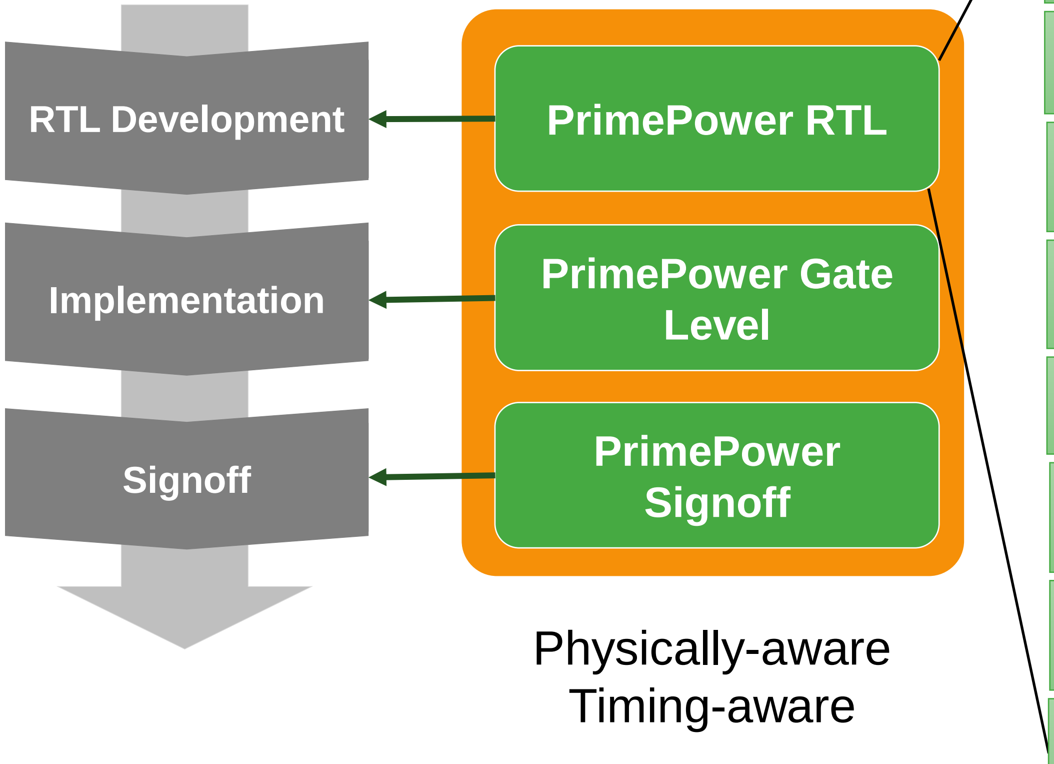
Gate Level Design

Gate Level FSDB

Signoff Analysis

SPEED (TAT)

RTL Power Analysis/Exploration



Physically-aware
Timing-aware

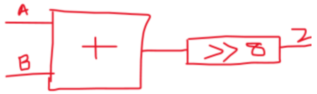
- **Multiple views to identify hot spots**
- **Power linting –** structural linting rules
- **Glitch power –** source identification & ranking
- **Clock gating efficiency** – per design hierarchy
- **Register gating efficiency** – CGE & Q/Clk ratios
- **Clock tree efficiency** – cascaded CGE
- **XOR, STC, ODC gating with power saving**
- **Memory profiling with power saving**
- **Micro-architectural explorations** (DW, FSM, Fmax/Vmin)

Timing aware

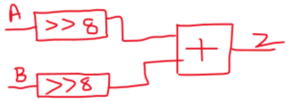
Power Efficient RTL Coding Guidelines

RTL Coding for Glitch : Example 1

```
module fft(A,B,Z)

assign Z = (A+B)/8;

endmodule
```

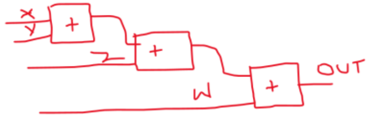


```
module fft(A,B,Z)

assign Z = A/8 + A/8;

endmodule
```

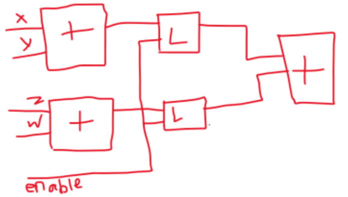


RTL Coding for Glitch : Example 4

```
module lft(X,Y,Z,W, out)

assign temp1 = X+Y;
assign temp2 = temp1 + Z
assign out = temp2+W;

endmodule
```

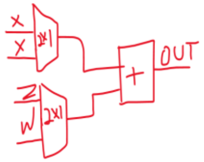


```
module lft(X,Y,Z,W, out)

assign temp1 = X+Y;
assign temp2 = Z+W
assign out = temp1_l+temp2_l;

always @ (enable)
  begin
    temp1_l = temp1;
    temp2_l = temp2;
  end
endmodule
```

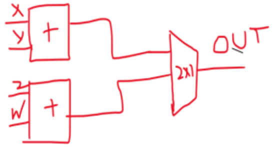


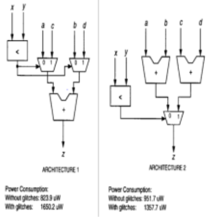
RTL Coding for Glitch : Example 5 (resource sharing)

```
module lft(X,Y,Z,W,sel_data, out)

assign temp1 = (sel_data)?Y:X;
assign temp2 = (!sel_data)? Z:W;
assign out = temp1+temp2;

endmodule
```



```
module lft(X,Y,Z,W, out)

assign temp1 = X+Y;
assign temp2 = Z+W
assign out = (sel_data)?temp1_l:temp2_l;

endmodule
```





Power Consumption:
Without glitches: 823.9 uW
With glitches: 1650.2 uW

Power Consumption:
Without glitches: 951.7 uW
With glitches: 1357.7 uW

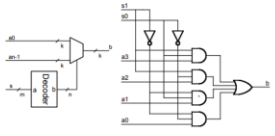
RTL Coding for Multiplexer : Example 6 (One Hot/Gray/Binary)

```
input [1:0] select; input a0, a1, a2, a3; output out;

// Example 1: if/else statements
always @(*) begin if (select == 2'b00) out = a0; else
if (select == 2'b01) out = a1; else
if (select == 2'b10) out = a2; else
out = a3; end
// Example 2: case
always @(*) begin case(select):
2'b00: out = a0;
2'b01: out = a1;
2'b10: out = a2;
2'b11: out = a3;
default: out = a0;
endcase end
// Example 3: Assignment with ?:
assign out = select[1] ? (select[0] ? a0 : a1) : (select[0] ? a2 : a3);
```

One-hot encoding

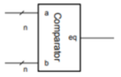
	cat	mat	on	sat	the
the =>	0	0	0	0	1
cat =>	1	0	0	0	0
sat =>	0	0	0	1	0

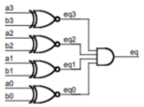
???



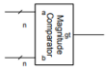
```
module muxb3(a2, a1, a0, sb, b) ;
parameter k = 1 ;
input [k-1:0] a0, a1, a2 ;
input [1:0] sb ;
output[k-1:0] b ;
wire [2:0] s ;
dec #(2,3) d(sb,s) ; // Decoder converts binary to one-hot
mux3 #(k) m(a2, a1, a0, s, b) ; // multiplexer selects input
endmodule
```

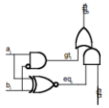
RTL Coding for Comparator : Example 7 (Magnitude/Equality)



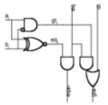


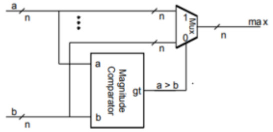
```
// equality comparator
module EqComp(a, b, eq) ;
  parameter k=8;
  input  [k-1:0] a,b;
  output eq ;
  wire   eq;

  assign eq = (a==b) ;
endmodule
```



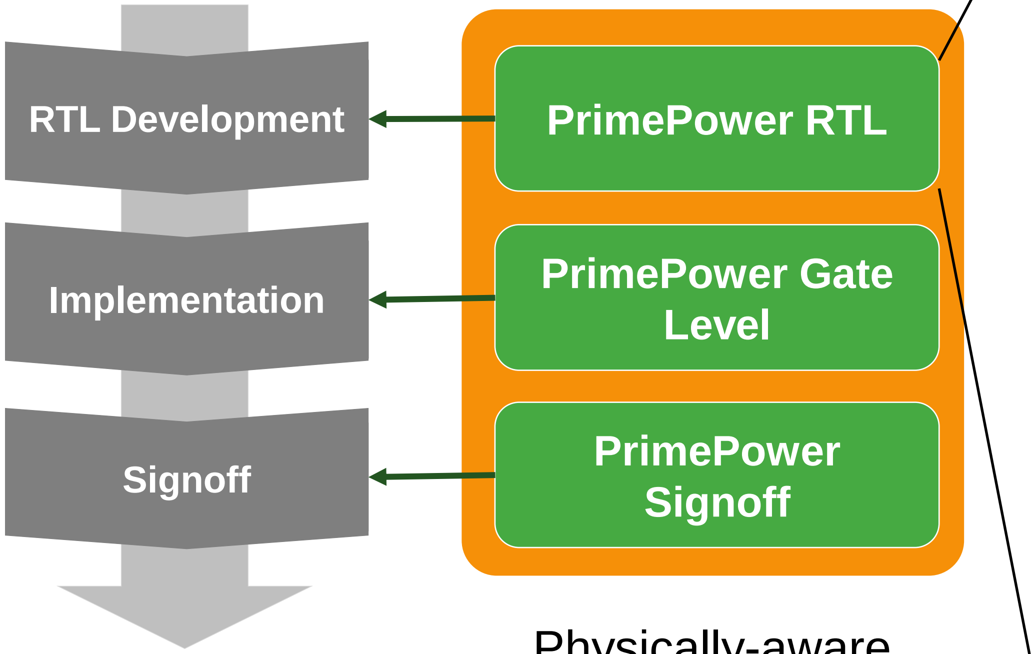


```
// magnitude comparator
module MagComp(a, b, gt) ;
  parameter k=8 ;
  input  [k-1:0] a, b ;
  output gt ;
  wire   [k-1:0] eqi = a ~^ b ;
  wire   [k-1:0] gti = a & ~b ;
  wire   [k:0]   gtb {((eqi[k-1:0] & gtb[k-1:0]) | gti[k-1:0]), 1'b0) ;
  wire   gt = gtb[k] ;
  endmodule
```





RTL Power Analysis/Exploration



Physically-aware
Timing-aware

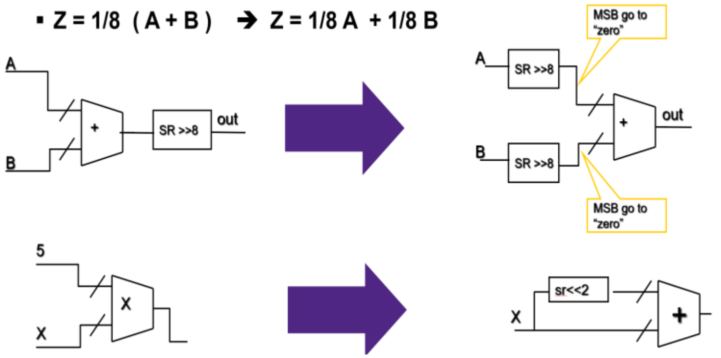
- **Multiple views to identify hot spots**
- **Power linting –** structural linting rules
- **Glitch power –** source identification & ranking
- **Clock gating efficiency** – per design hierarchy
- **Register gating efficiency** – CGE & Q/Clk ratios
- **Clock tree efficiency** – cascaded CGE
- **XOR, STC, ODC gating with power saving**
- **Memory profiling with power saving**
- **Micro-architectural explorations** (DW, FSM, Fmax/Vmin)

Timing aware

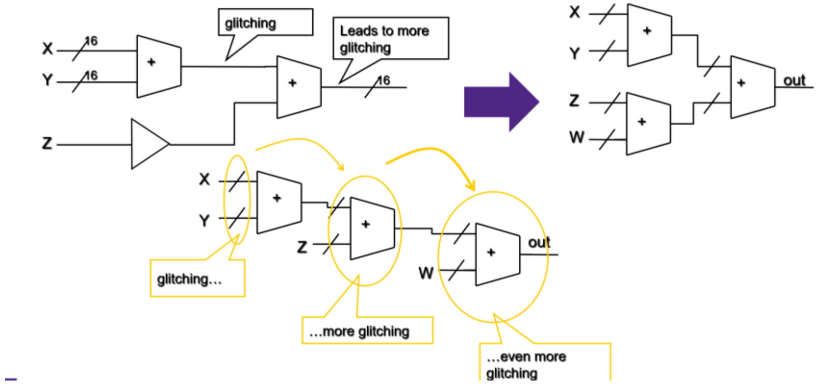
Glitch Power Optimization, by changing Architecture

Reduce Toggles by Reordering operands

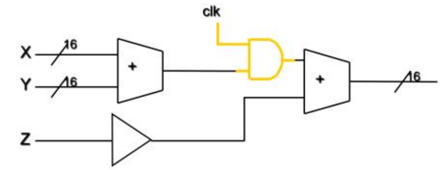
- Reduce number of bits in operations
  - Z = 1/8 (A + B) ➔ Z = 1/8 A + 1/8 B



Datapath Glitching



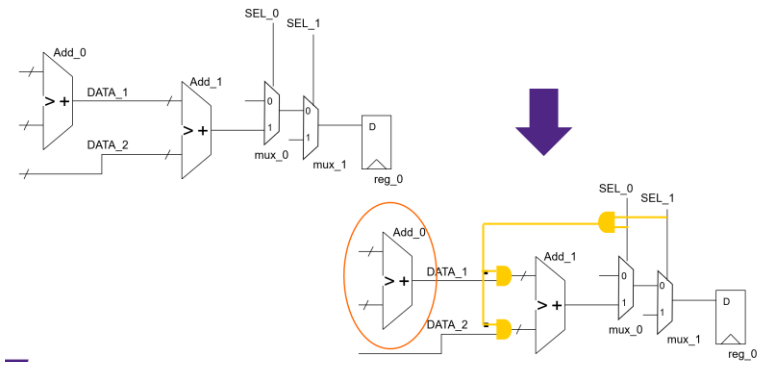
Absorb the Glitches



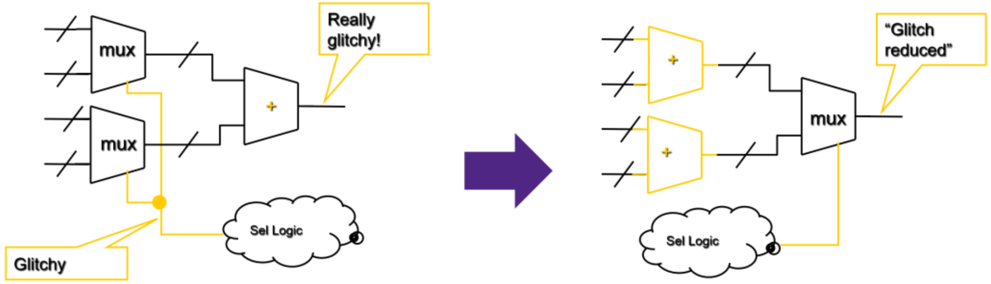
- What to gate with?

Gate type	*Glitch protection*	*Fanout*	*Caveats*
Simple Gate	Good	Small	Need suitable enable signal
Latches	Better	Medium	Clock duty cycle must be appropriate
Registers	Best	large	Adds stage in logic

Operand Isolation/Data Gating

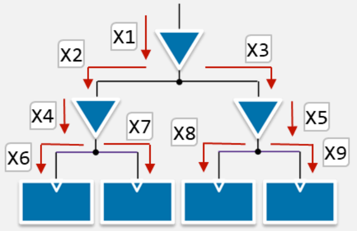


Resource Sharing



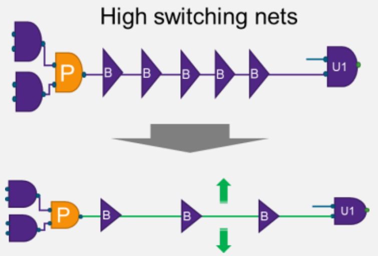
Low Power Technologies in Fusion Compiler

Power-Aware CCD Everywhere



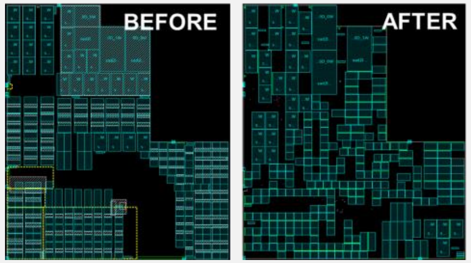
Arc-based Useful Skew Models Physical/Structural Effects

Global Route-driven Optimization



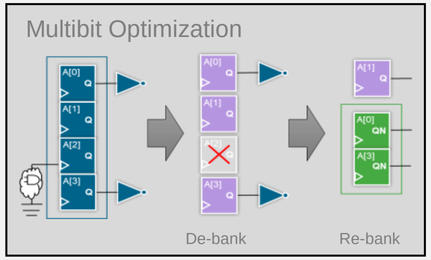
Double spacing, low C layer, re-buffer

Freeform Floorplanning



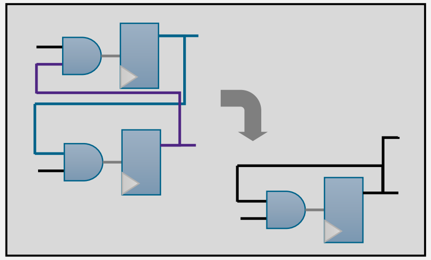
Targeted Power and Wirelength Reduction

Multibit



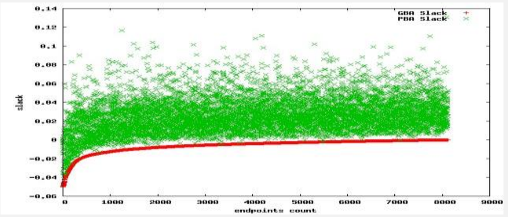
Unique WordView architecture for physically-aware multibit support

Advanced Register Merging



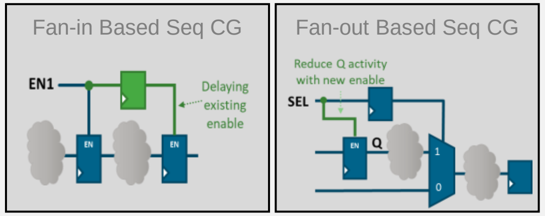
Merges registers in same equivalence class

PBA-Based Optimization



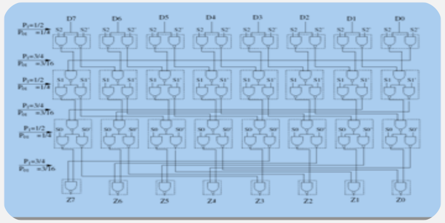
Post Route Optimization for recovering comb/seq power

Sequential Clock-Gating



Multi-cycle dynamic power savings with equivalence checking support

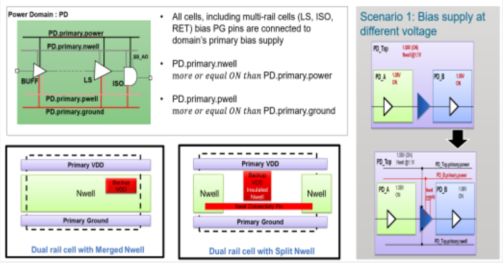
DesignWare minPower



Low Power Data-path Architectures

UPF Technologies in Fusion Compiler

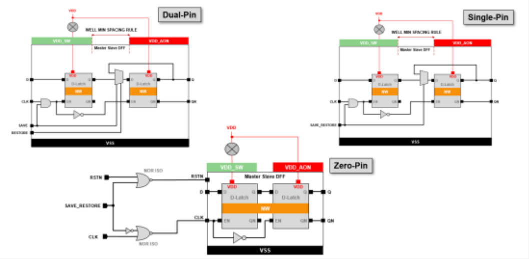
Merge/Split Nwell Implementation



- All cells, including multi-rail cells (LS, ISO, RET) bias PG pins are connected to domain's primary bias supply
- PD.primary.nwell *more or equal ON than* PD.primary.power
- PD.primary.pwell *more or equal ON than* PD.primary.ground

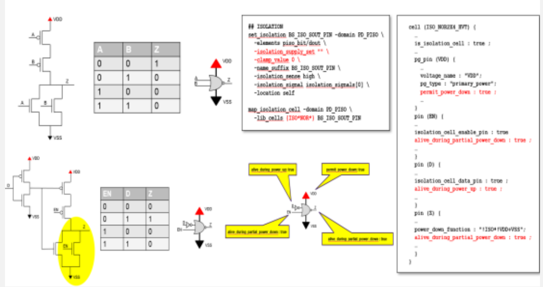
Optimization Using Insulated cells

Various Retention Styles



Complete Support for Zero Pin Retention flops

Complex Isolation Cell Styles


VDD
VSS
A
B
Z
EN
alive_during_power_up : true
permit_power_down : true
alive_during_partial_power_down : true
alive_during_partial_power_down : true

A	B	Z
0	0	1
0	1	0
1	0	0
1	1	0

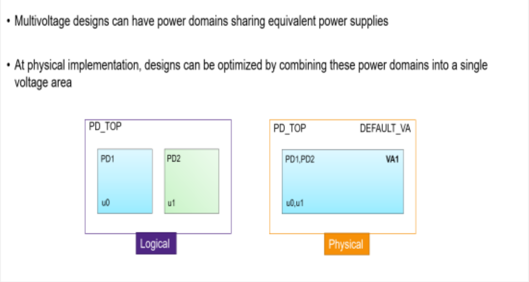
EN	D	Z
0	0	0
0	1	1
1	0	0
1	1	0

```
## ISOLATION
set_isolation BS_ISO_SOUT_PIN -domain PD_PISO \
  -elements piso_bit/dout \
  -isolation_supply_set "" \
  -clamp_value 0 \
  -name_suffix BS_ISO_SOUT_PIN \
  -isolation_sense high \
  -isolation_signal isolation_signals[0] \
  -location self

map_isolation_cell -domain PD_PISO \
  -lib_cells [ISO*NOR*] BS_ISO_SOUT_PIN
```

```
cell (ISO_NOR2X4_HVT) {
  …
  is_isolation_cell : true ;
  …
  pg_pin (VDD) {
   …
   voltage_name : "VDD";
   pg_type : "primary_power";
   permit_power_down : true ;
   …
  }
  pin (EN) {
  …
  isolation_cell_enable_pin : true
  alive_during_partial_power_down : true ;
  …
  }
  pin (D) {
  …
  isolation_cell_data_pin : true ;
  alive_during_power_up : true ;
  …
  }
  pin (Z) {
  …
  power_down_function : "!ISO*!VDD+VSS";
  alive_during_partial_power_down : true ;
  …
  }
}
```

NOR Isolation cell for reduced PG resources

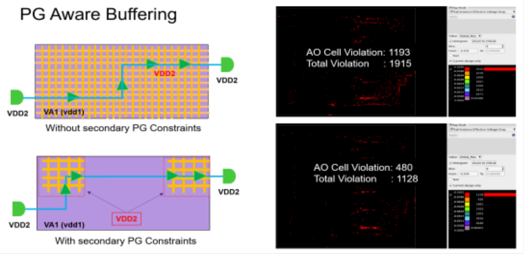
Multiple Power Domains Per Voltage Area



- Multivoltage designs can have power domains sharing equivalent power supplies
- At physical implementation, designs can be optimized by combining these power domains into a single voltage area

Logical/Physical Alignment for VA

PG Aware Buffering

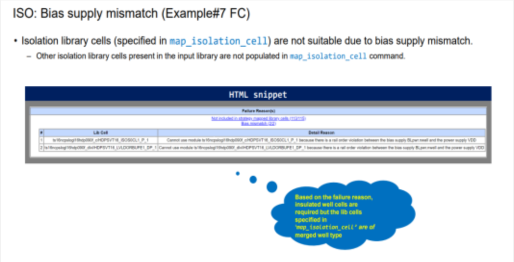


AO Cell Violation: 1193
Total Violation : 1915

AO Cell Violation: 480
Total Violation : 1128

Physical Power Strap Aware Buffering

Advanced Voltage Aware Debugging



ISO: Bias supply mismatch (Example#7 FC)

- Isolation library cells (specified in `map_isolation_cell`) are not suitable due to bias supply mismatch.
  - Other isolation library cells present in the input library are not populated in `map_isolation_cell` command.

HTML snippet

Failure Reason(s)

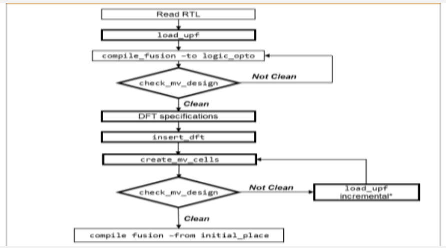
Not included in strategy mapped library cells (11/215)
Bias mismatch (2/2)

#	Lib Cell	Detail Reason
1	[illegible]_ISOS0CL1_P_1	Cannot use module [illegible]_ISOS0CL1_P_1 because there is a rail order violation between the bias supply [illegible] and the power supply VDD
2	[illegible]_LVLDORBUFE1_DP_1	Cannot use module [illegible]_LVLDORBUFE1_DP_1 because there is a rail order violation between the bias supply [illegible] and the power supply VDD

Based on the failure reason, insulated well cells are required but the lib cells specified in 'map_isolation_cell' are of merged well type

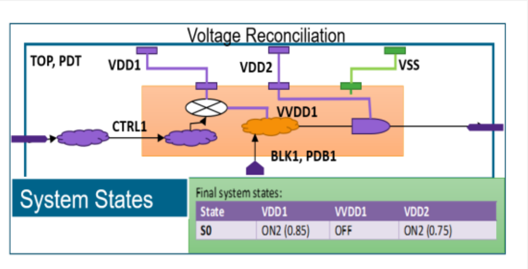
Automated Why/What Analysis

UPF Aware DFT RTL/Gate Level Flow



Automatic Isolation Derivation for DFT paths

Voltage Reconciliation

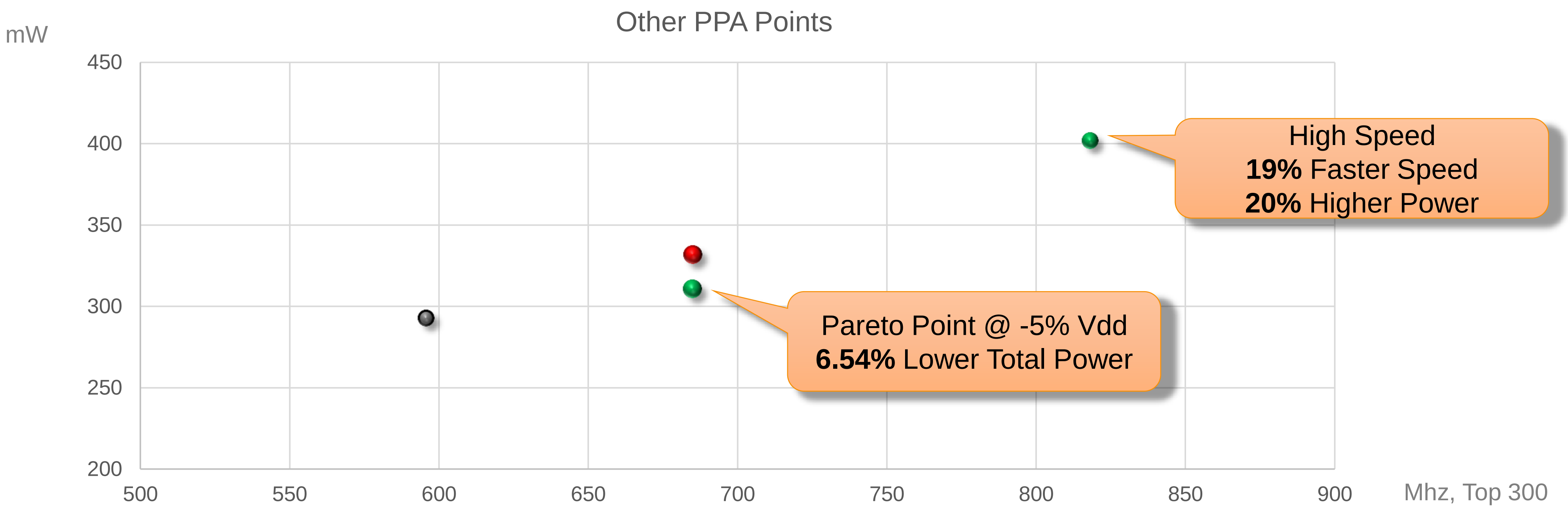


Final system states:

State	VDD1	VVDD1	VDD2
S0	ON2 (0.85)	OFF	ON2 (0.75)

Voltage Resolution at SOC level

Finding Optimal Voltage for Timing/Power



High Speed
19% Faster Speed
20% Higher Power

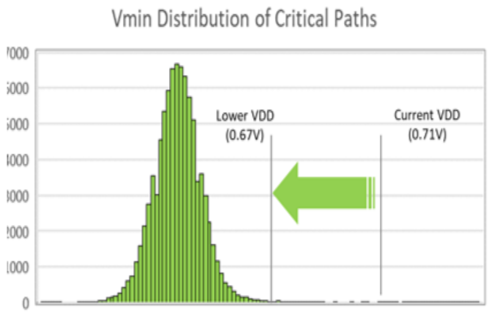
Pareto Point @ -5% Vdd
6.54% Lower Total Power

Alternative solutions with timing/power tradeoffs, depending on designer goals

PrimeShield Improves Power on Low-Power Designs

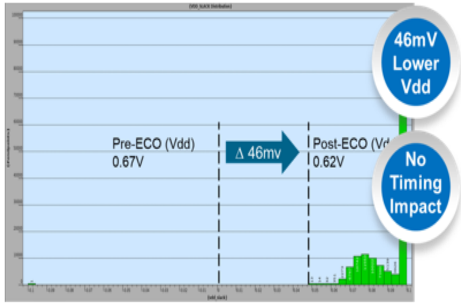
Voltage Slack Analysis (VSA) & Robustness Optimization reduces Vdd to improve power

Improved Power & Robustness maintaining performance



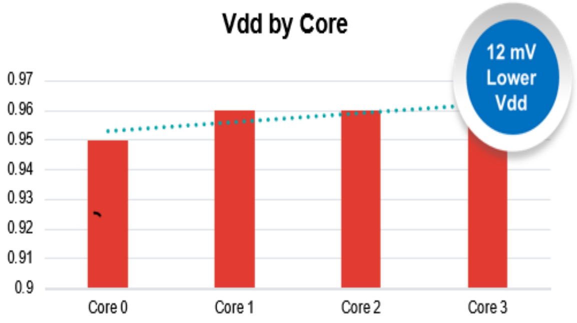
Silicon Accurate

voltage slack predicted with VSA

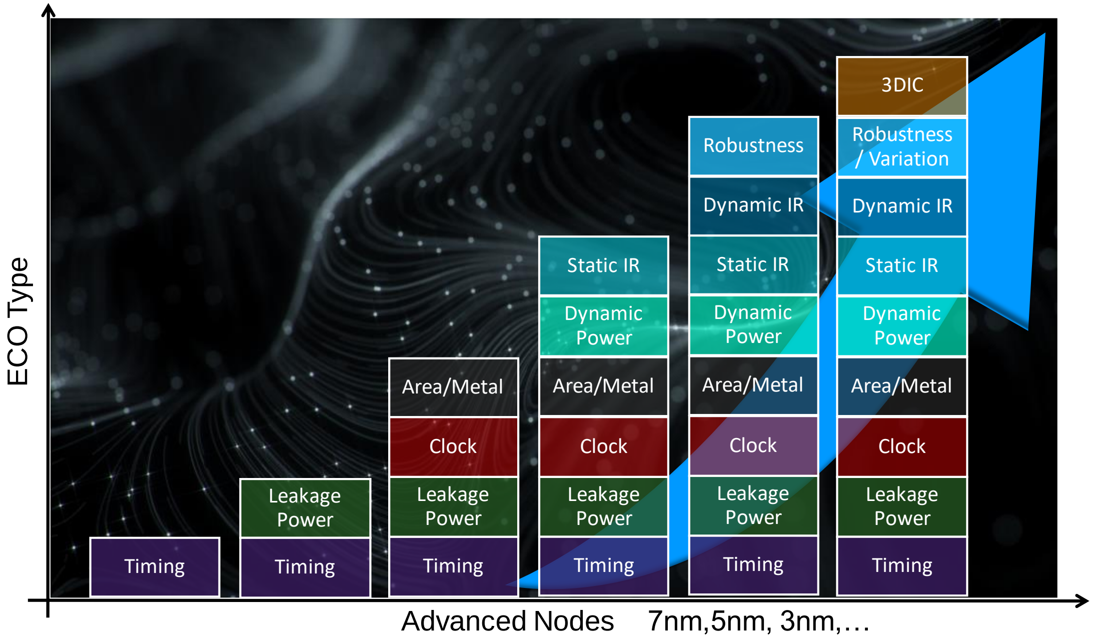


Lower Dynamic Power

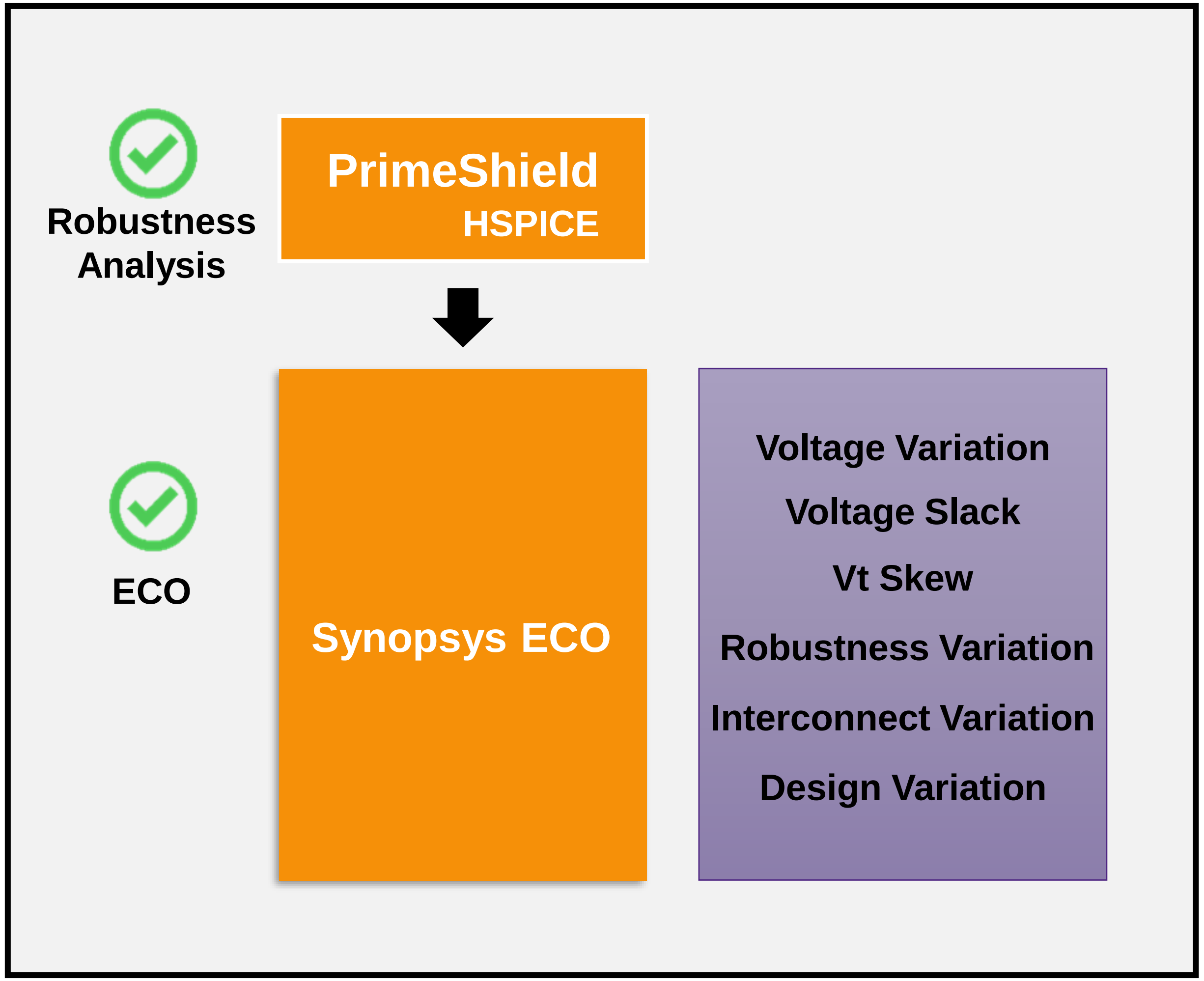
on PrimeShield optimized AI core



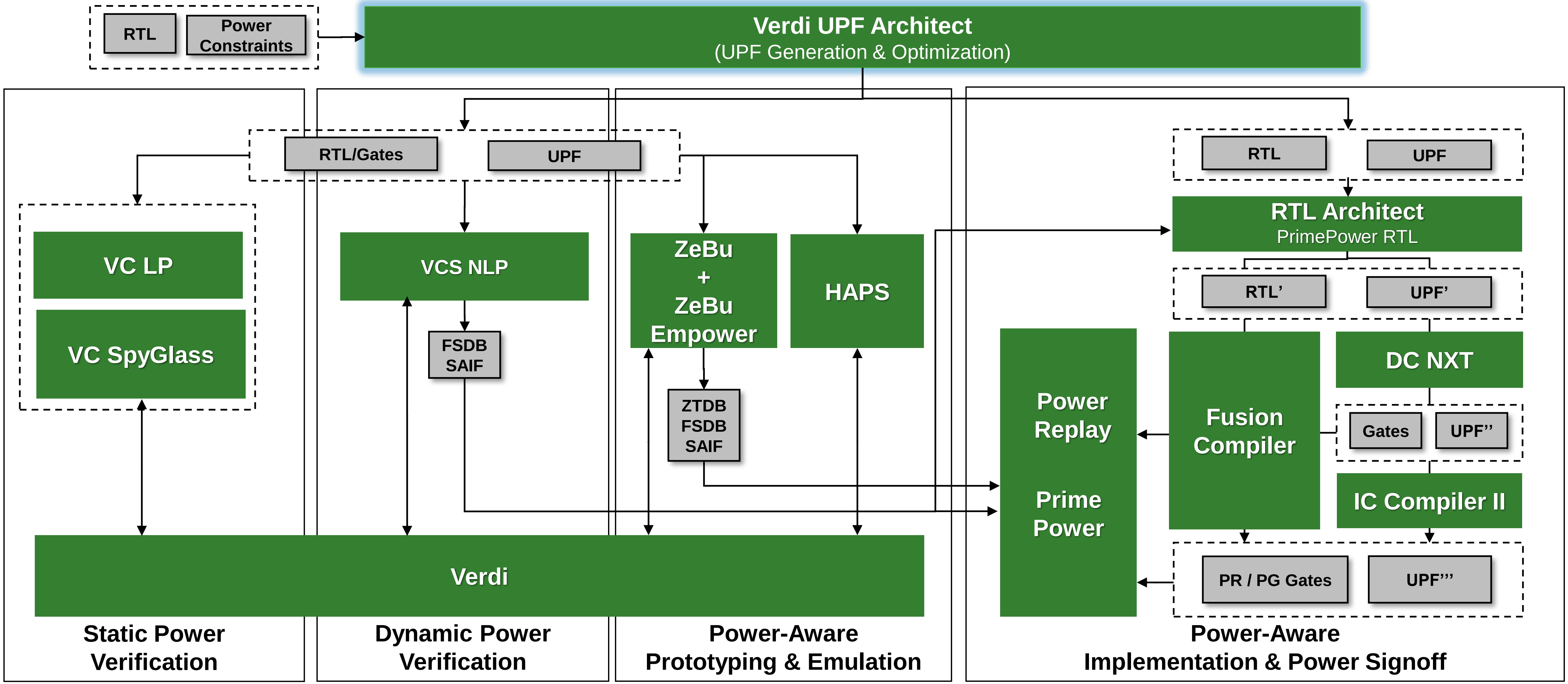
Power Closure using Synopsys ECO



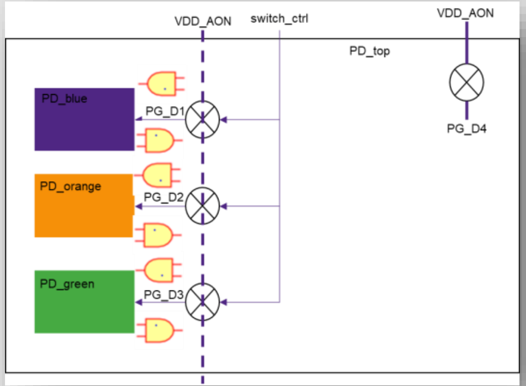
ECO tools must understand growing physical effects



UPF-Based Power Verification, Implementation & Signoff

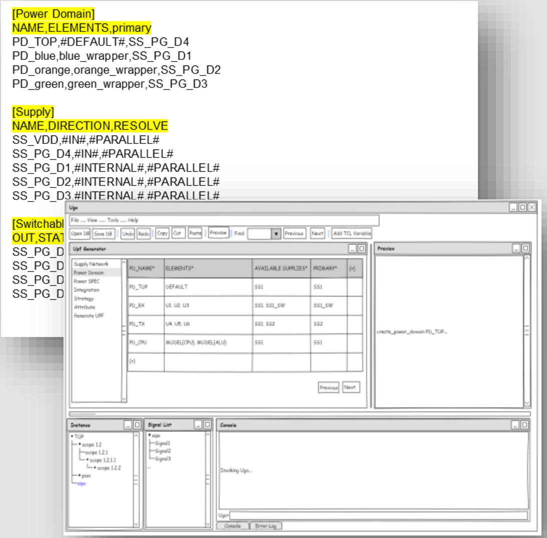


Verdi UPF Architect for Automated UPF Generation



Design

Text input (csv)



```
[Power Domain]
NAME,ELEMENTS,primary
PD_TOP,#DEFAULT#,SS_PG_D4
PD_blue,blue_wrapper,SS_PG_D1
PD_orange,orange_wrapper,SS_PG_D2
PD_green,green_wrapper,SS_PG_D3

[Supply]
NAME,DIRECTION,RESOLVE
SS_VDD,#IN#,#PARALLEL#
SS_PG_D4,#IN#,#PARALLEL#
SS_PG_D1,#INTERNAL#,#PARALLEL#
SS_PG_D2,#INTERNAL#,#PARALLEL#
SS_PG_D3,#INTERNAL#,#PARALLEL#

[Switchabl
OUT,STAT
SS_PG_D
SS_PG_D
SS_PG_D
SS_PG_D
```

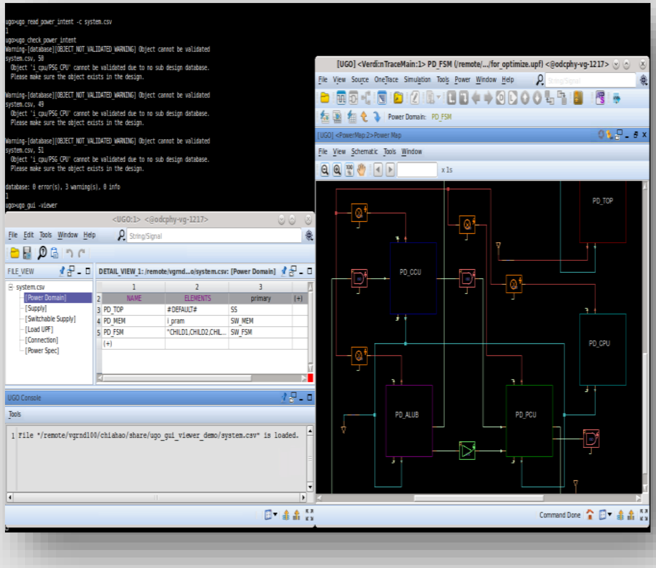
PD_NAME*	ELEMENTS*	AVAILABLE SUPPLIES*	PRIMARY*	(+)
PD_TOP	DEFAULT	SS1	SS1	
PD_RX	U1, U2, U3	SS1, SS1_SW	SS1_SW	
PD_TX	U4, U5, U6	SS1, SS2	SS2	
PD_CPU	MODEL(CPU), MODEL(ALU)	SS1	SS1	
(+)				

create_power_domain PD_TOP...

Invoking Ugo...

GUI

Visualization & Debug



```
ugo>ugo_read_power_intent -c system.csv
1
ugo>ugo_check_power_intent
Warning-[database][OBJECT_NOT_VALIDATED_WARNING] Object cannot be validated
system.csv, 50
  Object 'i_cpu/PSG_CPU' cannot be validated due to no sub design database.
  Please make sure the object exists in the design.

Warning-[database][OBJECT_NOT_VALIDATED_WARNING] Object cannot be validated
system.csv, 49
  Object 'i_cpu/PSG_CPU' cannot be validated due to no sub design database.
  Please make sure the object exists in the design.

Warning-[database][OBJECT_NOT_VALIDATED_WARNING] Object cannot be validated
system.csv, 51
  Object 'i_cpu/PSG_CPU' cannot be validated due to no sub design database.
  Please make sure the object exists in the design.

database: 0 error(s), 3 warning(s), 0 info
1
ugo>ugo_gui -viewer
```

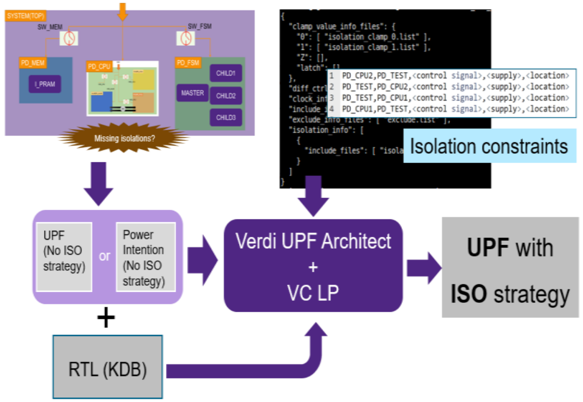
	1	2	3	
2	NAME	ELEMENTS	primary	(+)
3	PD_TOP	#DEFAULT#	SS	
4	PD_MEM	i_pram	SW_MEM	
5	PD_FSM	"CHILD1,CHILD2,CHIL...	SW_FSM	
	(+)			

```
1 File "/remote/vgrnd100/chiahao/share/ugo_gui_viewer_demo/system.csv" is loaded.
```

UPF Generation, Integration and Optimization

UPF Optimization using Verdi's UPF Architect

Optimizer - Generate Isolation Strategies

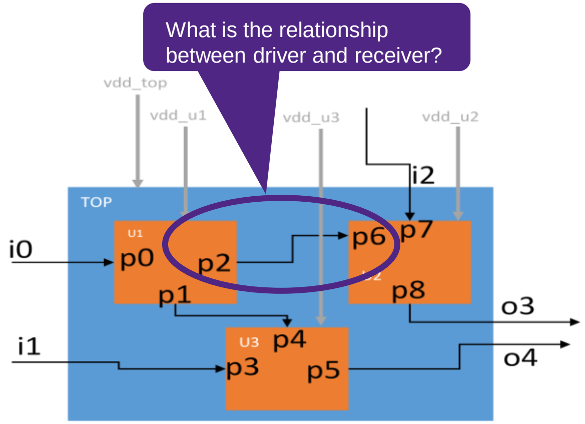


```
{
  "clamp_value_info_files": {
    "0": [ "isolation_clamp_0.list" ],
    "1": [ "isolation_clamp_1.list" ],
    "Z": [],
    "latch": []
  },
  "diff_ctrl
  "clock_inf
  "include_i
  "exclude_info_files": [ "exclude.list" ],
  "isolation_info": [
    {
      "include_files": [ "isola
    }
  ]
}
```

```
1 PD_CPU2,PD_TEST,<control signal>,<supply>,<location>
2 PD_TEST,PD_CPU2,<control signal>,<supply>,<location>
3 PD_TEST,PD_CPU1,<control signal>,<supply>,<location>
4 PD_CPU1,PD_TEST,<control signal>,<supply>,<location>
```

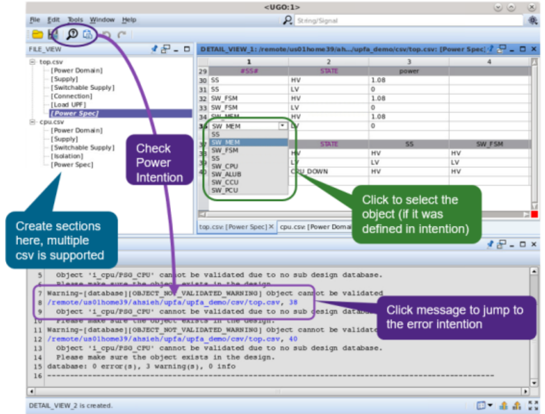
- Leveraging **VC LP** technology
- Analyze RTL design to generate proper Isolation Strategy automatically

Optimizer – UPF Budgeting



- Budgeting sub blocks in top UPF
- Existing UPF for blocks, generate **SPA** (**set_port_attribute**) in top level

GUI - The Power Intent Editor



	1	2	3	4
29	#SS#	STATE	power	
30	SS	HV	1.08	
31	SS	LV	0	
32	SW_FSM	HV	1.08	
33	SW_FSM	LV	0	
34	SW_MEM	HV	1.08	
35	SW_MEM	LV	0	
37		STATE	SS	SW_FSM
38		HV	HV	HV
39		LV	LV	LV
40		CPU DOWN	HV	HV

Check Power Intention

Create sections here, multiple csv is supported

Click to select the object (if it was defined in intention)

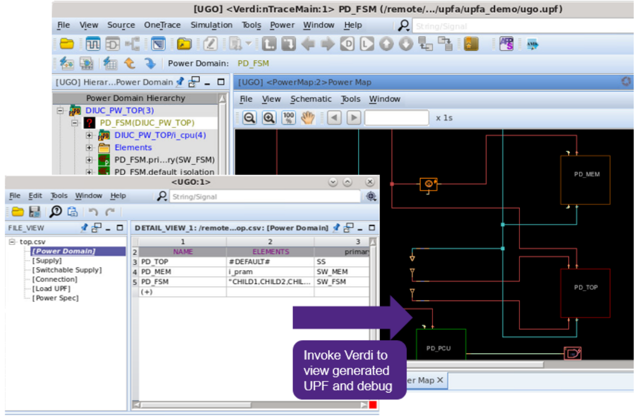
Click message to jump to the error intention

```
  Object 'i_cpu/PSG_CPU' cannot be validated due to no sub design database.
  Please make sure the object exists in the design.
Warning-[database][OBJECT_NOT_VALIDATED_WARNING] Object cannot be validated
/remote/us01home39/ahsieh/upfa/upfa_demo/csv/top.csv, 38
  Object 'i_cpu/PSG_CPU' cannot be validated due to no sub design database.
  Please make sure the object exists in the design.
Warning-[database][OBJECT_NOT_VALIDATED_WARNING] Object cannot be validated
/remote/us01home39/ahsieh/upfa/upfa_demo/csv/top.csv, 40
  Object 'i_cpu/PSG_CPU' cannot be validated due to no sub design database.
  Please make sure the object exists in the design.
database: 0 error(s), 3 warning(s), 0 info
-----------------------------------------------------------------------------------------
```

DETAIL_VIEW_2 is created.

- Input intention in xls-like GUI
  - Intention is syntax/version free
- Check power intention
  - Mismatch objects across the intentions
  - Mismatch objects between intention and design

GUI - Interactive with Verdi

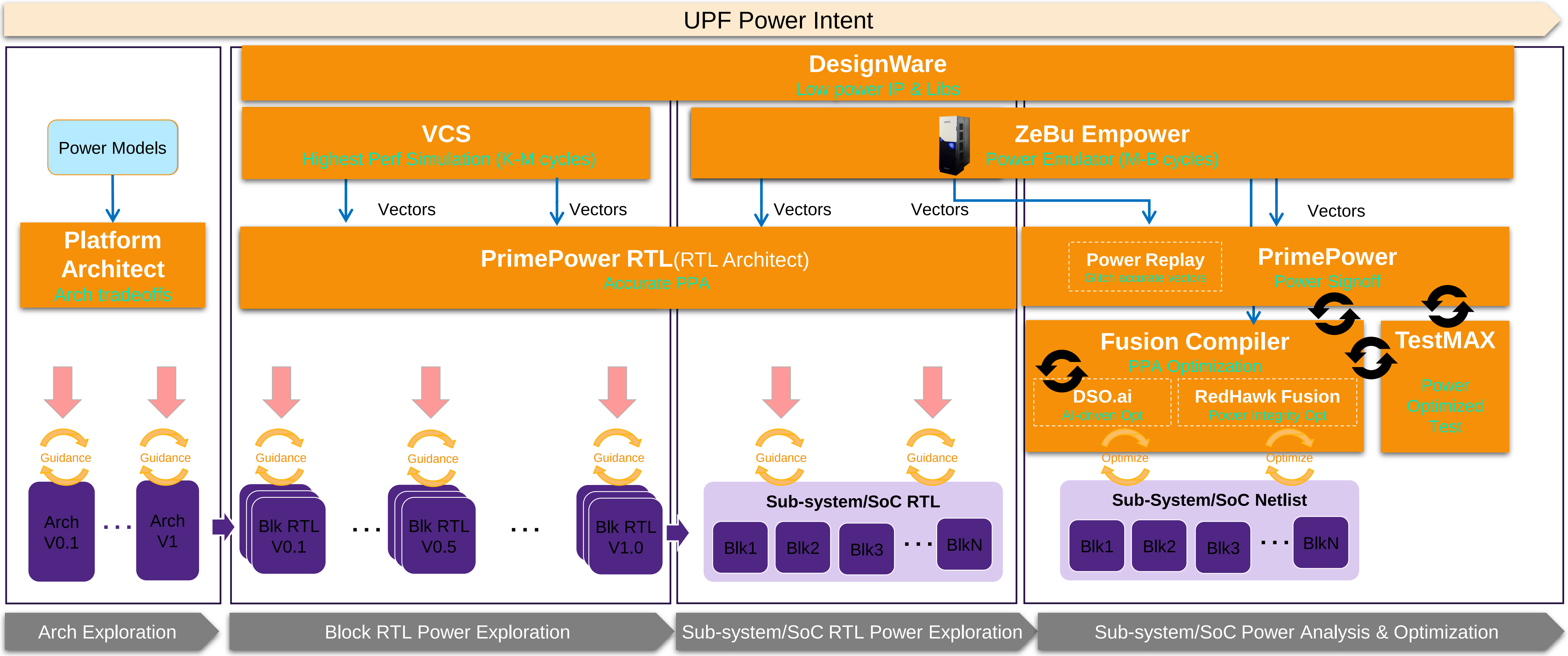


	1	2	3
2	NAME	ELEMENTS	primar
3	PD_TOP	#DEFAULT#	SS
4	PD_MEM	i_pram	SW_MEM
5	PD_FSM	"CHILD1,CHILD2,CHIL...	SW_FSM
	(+)		

Invoke Verdi to view generated UPF and debug

- View and debug intentions in **Verdi**
  - Hierarchical Power Domain Tree
  - Power Map: Power domains, ISO, level-shifters…

Synopsys Low Power Solution Summary



UPF Power Intent

DesignWare
Low power IP & Libs

Power Models

Platform Architect
Arch tradeoffs

VCS
Highest Perf Simulation (K-M cycles)

ZeBu Empower
Power Emulator (M-B cycles)

Vectors

Vectors

Vectors

Vectors

Vectors

PrimePower RTL(RTL Architect)
Accurate PPA

Power Replay
Glitch accurate vectors

PrimePower
Power Signoff

Fusion Compiler
PPA Optimization

DSO.ai
AI-driven Opt

RedHawk Fusion
Power Integrity Opt

TestMAX
Power Optimized Test

Guidance

Guidance

Guidance

Guidance

Guidance

Guidance

Guidance

Optimize

Optimize

Arch V0.1 . . . Arch V1

Blk RTL V0.1 . . . Blk RTL V0.5 . . . Blk RTL V1.0

Sub-system/SoC RTL
Blk1 Blk2 Blk3 . . . BlkN

Sub-System/SoC Netlist
Blk1 Blk2 Blk3 . . . BlkN

Arch Exploration

Block RTL Power Exploration

Sub-system/SoC RTL Power Exploration

Sub-system/SoC Power Analysis & Optimization

Automating and Optimizing UPF generation from Higher Level Power Intent through UPF Architect

Santhana Krishnan
Meta

Introduction

GOAL: Socially Acceptable and All-Day Wearable

- Reality Labs silicon team works in developing Virtual and Augmented reality hardware.
- Driving state-of-the-art forward.
- AR devices require extreme computing power.
- Optimize power starting from the transistor through architecture & software.
- **Power Gating is the key technology for silicon power benefits.**

Problem Statement:

Translating Power Intent from Architecture spec.
- Capturing Power Intent in a user-friendly format

UPF versions
- Maintaining the UPF that works across all EDA tools.

Modeling power domain Interfaces
- Associating interfaces with supply attributes based on connectivity & functionality.

Analog Macros
- Deriving the PG pin information from the datasheet

Power State Table
- Solving complex power states and deriving isolation strategies.

Motivation:

Addressing Power Management Challenges: Build or Buy?

❑ Challenges with Building your solution:

- ➢ Maintaining UPF syntax
- ➢ Deriving power control signals
- ➢ Managing analog macro connections
- ➢ Handling multiple voltage corner cases
- ➢ Implementing level shifters and retention registers

❑ Buying off the shelf:

- ➢ Limited EDA tool options for UPF generation

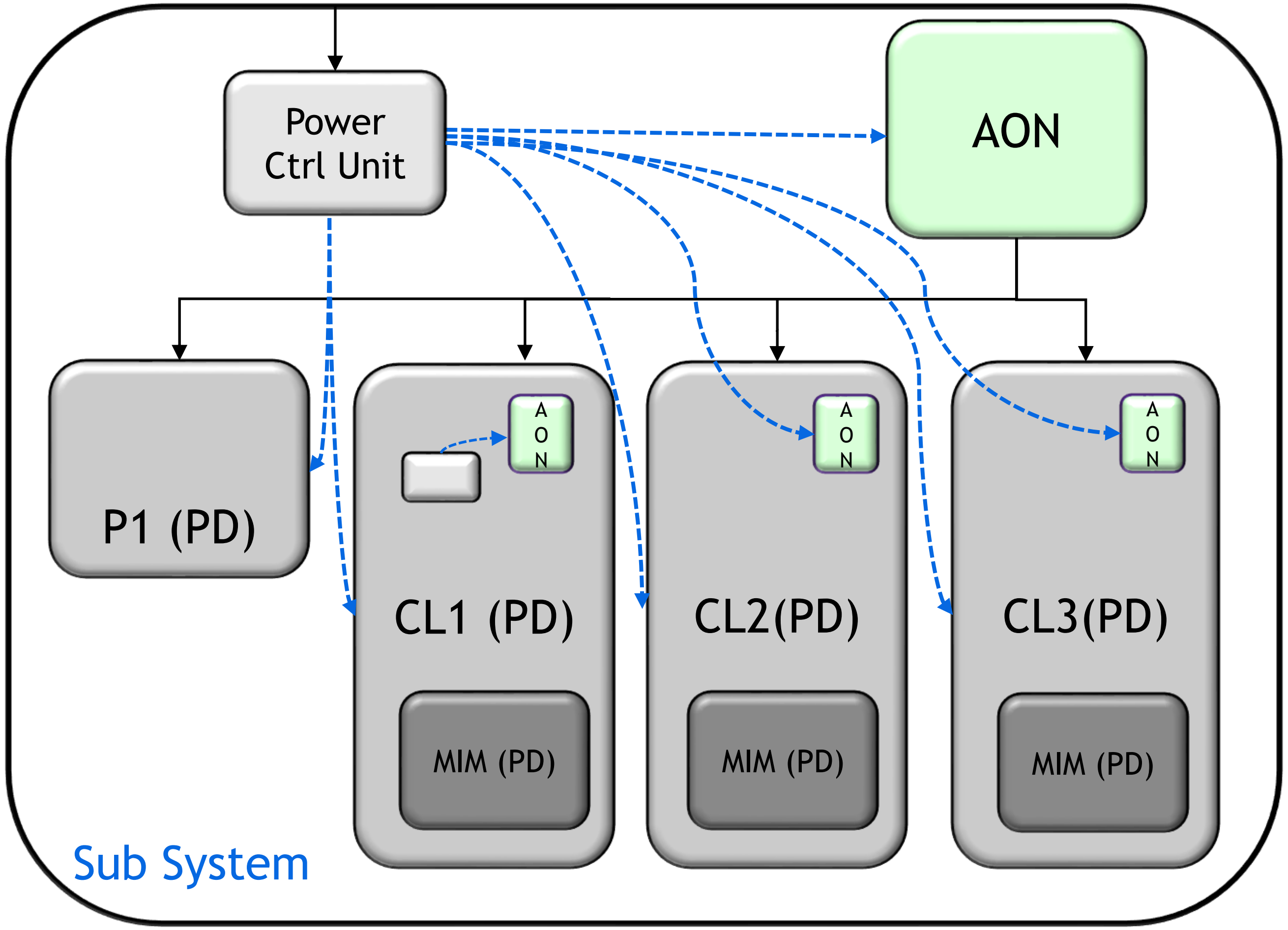
❑ Partnering with Synopsys for UPF generation:

- ➢ Offers **Verdi UPF Architect**
- ➢ Engagement from early concept to production release.

Design (Sub-System)

Logical View

- Sub-System with three levels of nested hierarchy
- RTL Module for each power domain
- Gated Power Control Unit of Sub-System
- Gated MIM Partition instantiated in the cluster
- Each power domain can be turned ON/OFF individually

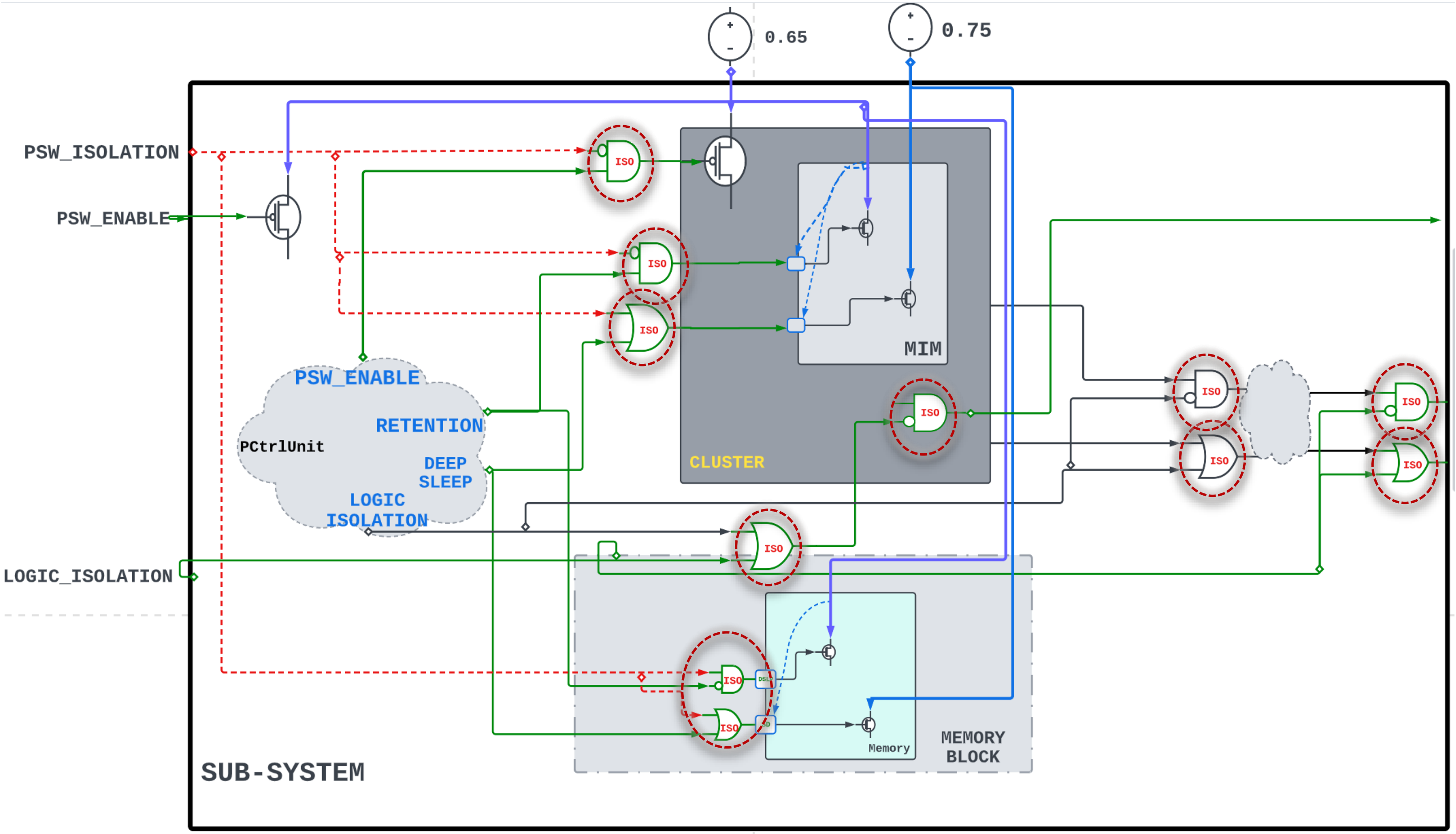


P* -> Partition
CL* -> Cluster
PD -> Power Domain
⬜ -> Power Gated
🟩 -> Always On

Design (Sub-System)

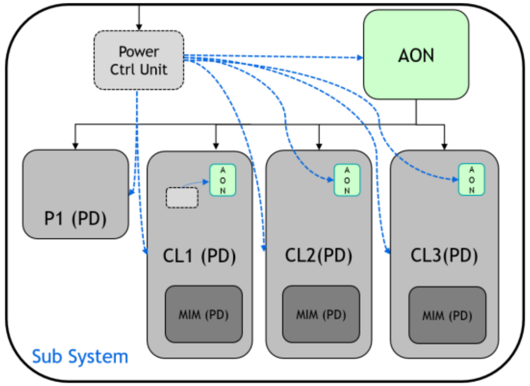
Physical Power Domain View

- A physical floorplan view of the power domains
- Each power domain represented as a voltage area
- Power states of the design require Isolation cells at multiple levels
- Gated Power Ctrl Unit demands Isolation on Isolation and power-enable signals.



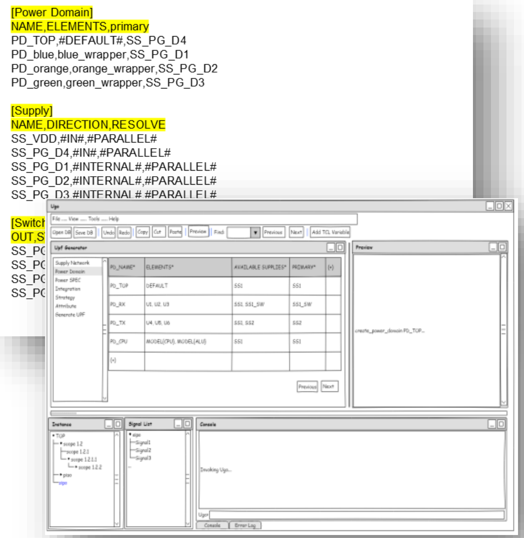
Numerous Isolations at various levels of hierarchy!
How do I solve this strategy ??

Verdi UPF Architect flow steps



Design

Text input (csv)



```
[Power Domain]
NAME,ELEMENTS,primary
PD_TOP,#DEFAULT#,SS_PG_D4
PD_blue,blue_wrapper,SS_PG_D1
PD_orange,orange_wrapper,SS_PG_D2
PD_green,green_wrapper,SS_PG_D3

[Supply]
NAME,DIRECTION,RESOLVE
SS_VDD,#IN#,#PARALLEL#
SS_PG_D4,#IN#,#PARALLEL#
SS_PG_D1,#INTERNAL#,#PARALLEL#
SS_PG_D2,#INTERNAL#,#PARALLEL#
SS_PG_D3,#INTERNAL#,#PARALLEL#

[Switch
OUT,S
SS_PG
SS_PG
SS_PG
SS_PG
```

PD_NAME*	ELEMENTS*	AVAILABLE SUPPLIES*	PRIMARY*	(+)
PD_TOP	DEFAULT	SS1	SS1	
PD_RX	U1, U2, U3	SS1, SS1_SW	SS1_SW	
PD_TX	U4, U5, U6	SS1, SS2	SS2	
PD_CPU	MODEL(CPU), MODEL(ALU)	SS1	SS1	
(+)				

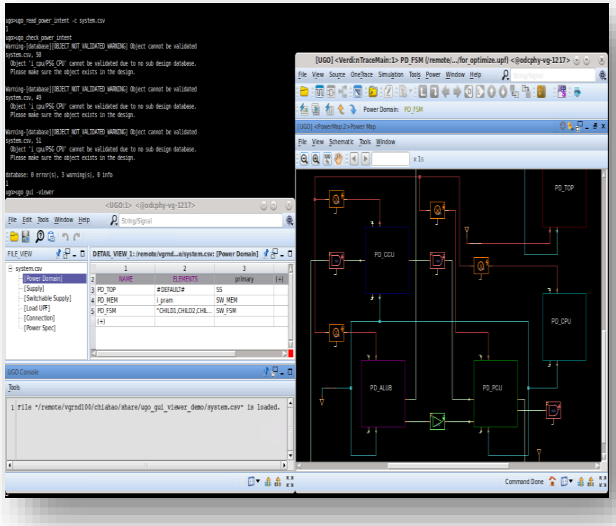
create_power_domain PD_TOP...

GUI

UPFA

STEP 1

Visualization & Debug



```
ugo>ugo_read_power_intent -c system.csv
1
ugo>ugo_check_power_intent
Warning-[database][OBJECT_NOT_VALIDATED_WARNING] Object cannot be validated
system.csv, 50
  Object 'i_cpu/PSG_CPU' cannot be validated due to no sub design database.
  Please make sure the object exists in the design.

Warning-[database][OBJECT_NOT_VALIDATED_WARNING] Object cannot be validated
system.csv, 49
  Object 'i_cpu/PSG_CPU' cannot be validated due to no sub design database.
  Please make sure the object exists in the design.

Warning-[database][OBJECT_NOT_VALIDATED_WARNING] Object cannot be validated
system.csv, 51
  Object 'i_cpu/PSG_CPU' cannot be validated due to no sub design database.
  Please make sure the object exists in the design.

database: 0 error(s), 3 warning(s), 0 info
1
ugo>ugo_gui -viewer
```

	1	2	3	
2	NAME	ELEMENTS	primary	(+)
3	PD_TOP	#DEFAULT#	SS	
4	PD_MEM	i_pram	SW_MEM	
5	PD_FSM	"CHILD1,CHILD2,CHIL...	SW_FSM	
	(+)			

```
1 File "/remote/vgrnd100/chiahao/share/ugo_gui_viewer_demo/system.csv" is loaded.
```

UPF Generation, Integration, and Optimization

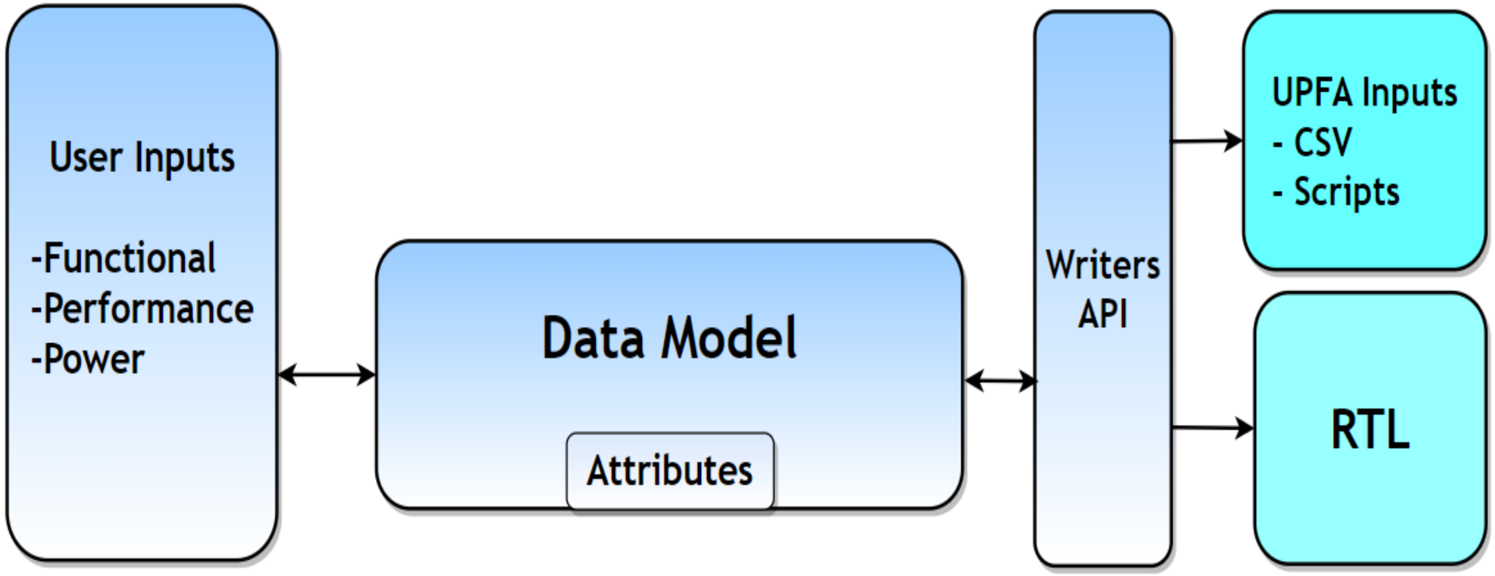
STEP 2

Generates UPF compliant with EDA/SNPS Tools, simplifies UPF integration from block to sub-system to chip

Meta’s collateral generation flow

Step 1: Generation

- Instantiates and configures third-party and proprietary IP to produce integrated designs
- User inputs include functional, performance, and power attributes
- Generates RTL + associated collateral for verification, FW & physical design.
- Input files for UPFA (CSV’s)



Input to the Generator: Python Dictionary

Power Domain Attributes

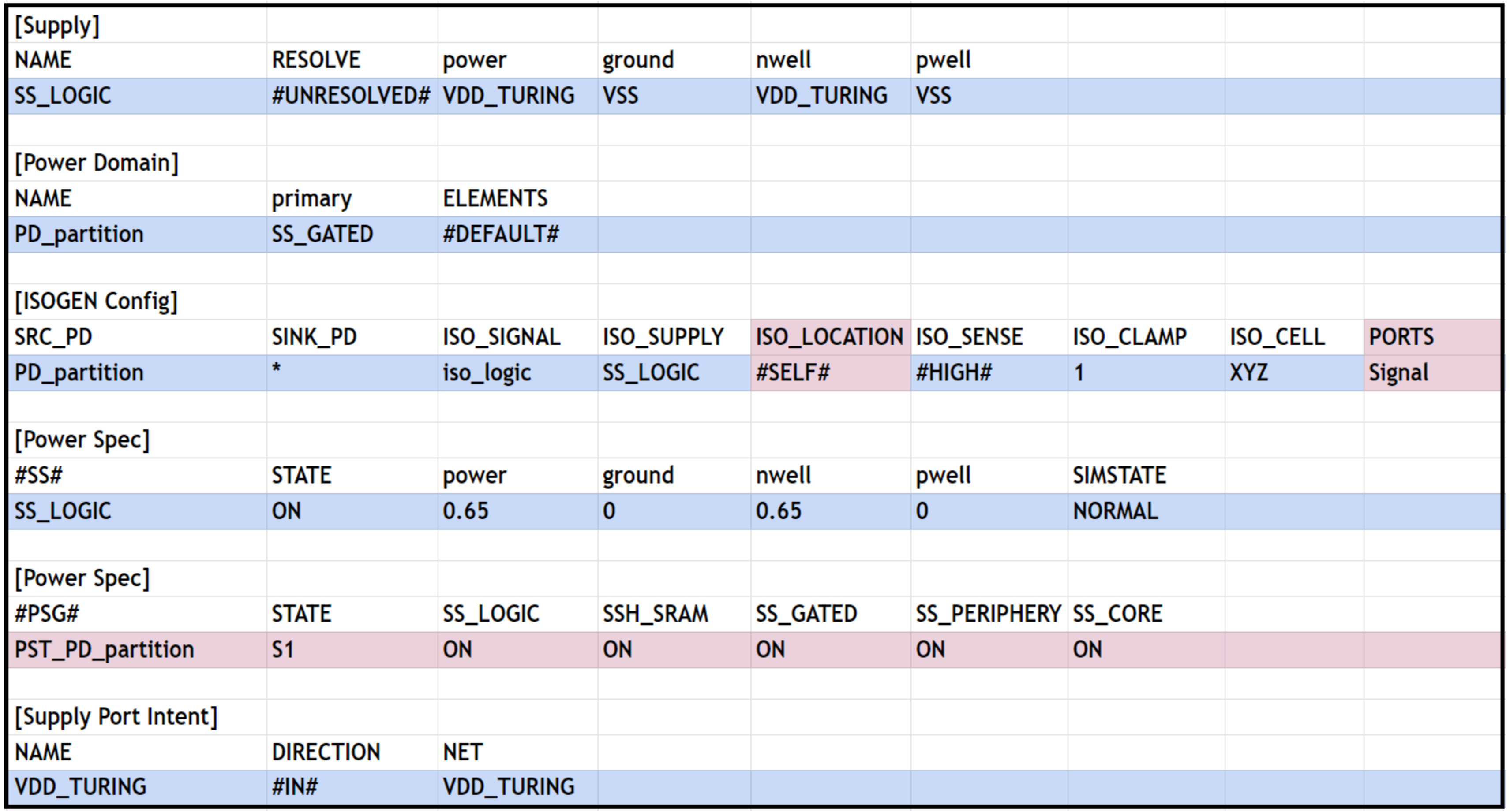
```
class UPFConfig:
     upfInfo = {
      'SubSystem’: {
                  'technology'                : 'ABC’,
                  'logic_rail_voltage'        : 1.0 ,
                  'gated'                     : False,
                  'contains_sram'             : False,
                   }

     'Cluster':   {
                 'gated'                      : True,
                 'abstract'                   : False,
                 'iso_location'               : `self’,
                 'iso_clamp1_ports'           : .*SD.*',
                 }
```

Power State info

```
     pstInfo = {
           'SubSystem' :   {
               'domains'           :   ['SubSystem', 'cluster_instance_name'],
               'states'   :   {
                 'SS_CL_ON'        :    '1           X',
                   'SS_OFF_CL_ON':    '1           pd18_pd17_OFF',
                   'ALL'           :    '1           pd18_pd17_ON',
               },
           },
     }
```

Output of the Generator: CSV format

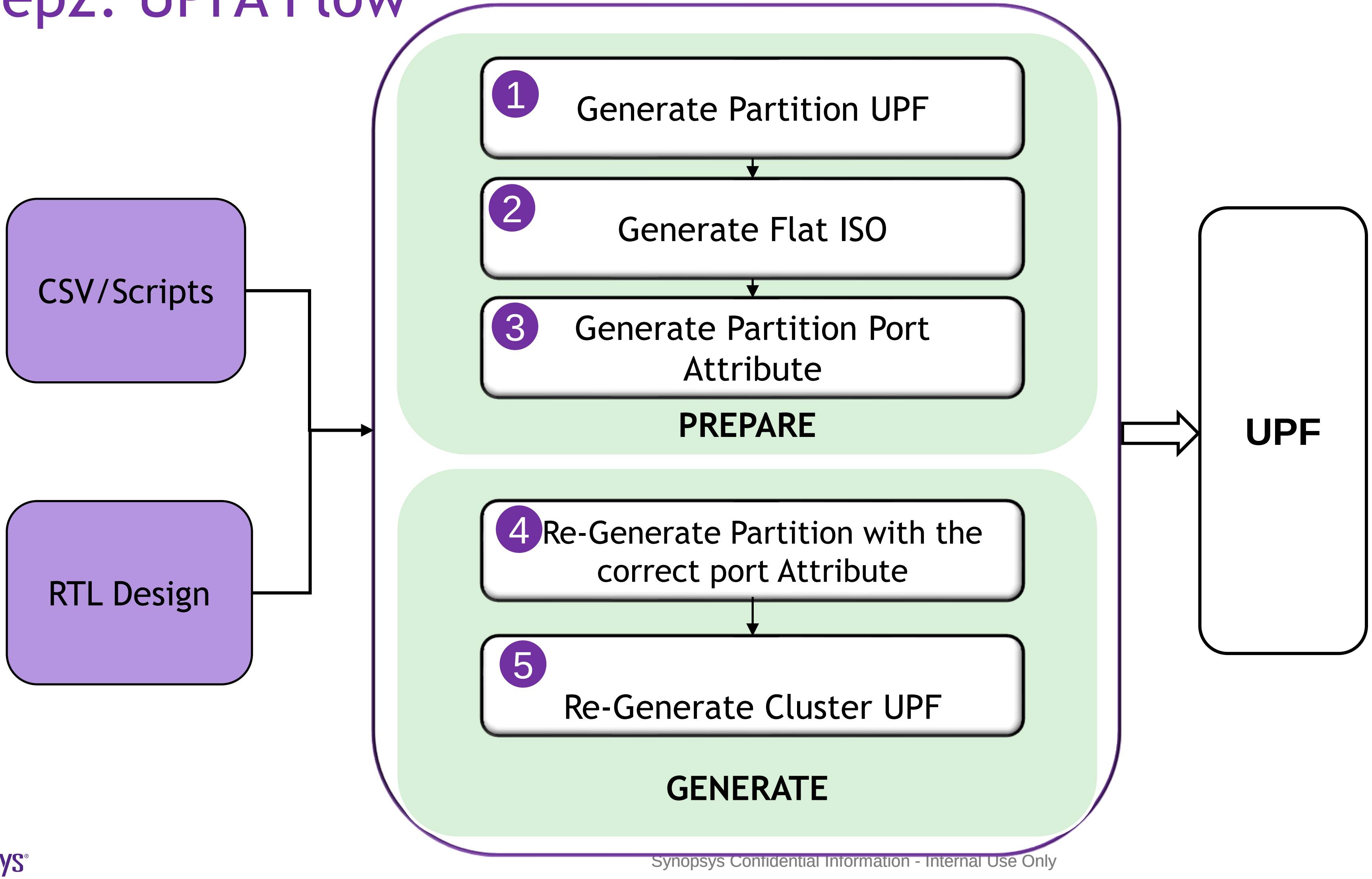
- Auto Derive
  - Control signals
  - Supply set
  - Power Switch Model
  - Isolation cell type based on location
  - Supply sets
- Translate
  - PST
  - Clamp Ports
  - Location
  - Technology



[Supply]								
NAME	RESOLVE	power	ground	nwell	pwell			
SS_LOGIC	#UNRESOLVED#	VDD_TURING	VSS	VDD_TURING	VSS			
[Power Domain]								
NAME	primary	ELEMENTS						
PD_partition	SS_GATED	#DEFAULT#						
[ISOGEN Config]								
SRC_PD	SINK_PD	ISO_SIGNAL	ISO_SUPPLY	ISO_LOCATION	ISO_SENSE	ISO_CLAMP	ISO_CELL	PORTS
PD_partition	*	iso_logic	SS_LOGIC	#SELF#	#HIGH#	1	XYZ	Signal
[Power Spec]								
#SS#	STATE	power	ground	nwell	pwell	SIMSTATE		
SS_LOGIC	ON	0.65	0	0.65	0	NORMAL		
[Power Spec]								
#PSG#	STATE	SS_LOGIC	SSH_SRAM	SS_GATED	SS_PERIPHERY	SS_CORE		
PST_PD_partition	S1	ON	ON	ON	ON	ON		
[Supply Port Intent]								
NAME	DIRECTION	NET						
VDD_TURING	#IN#	VDD_TURING						

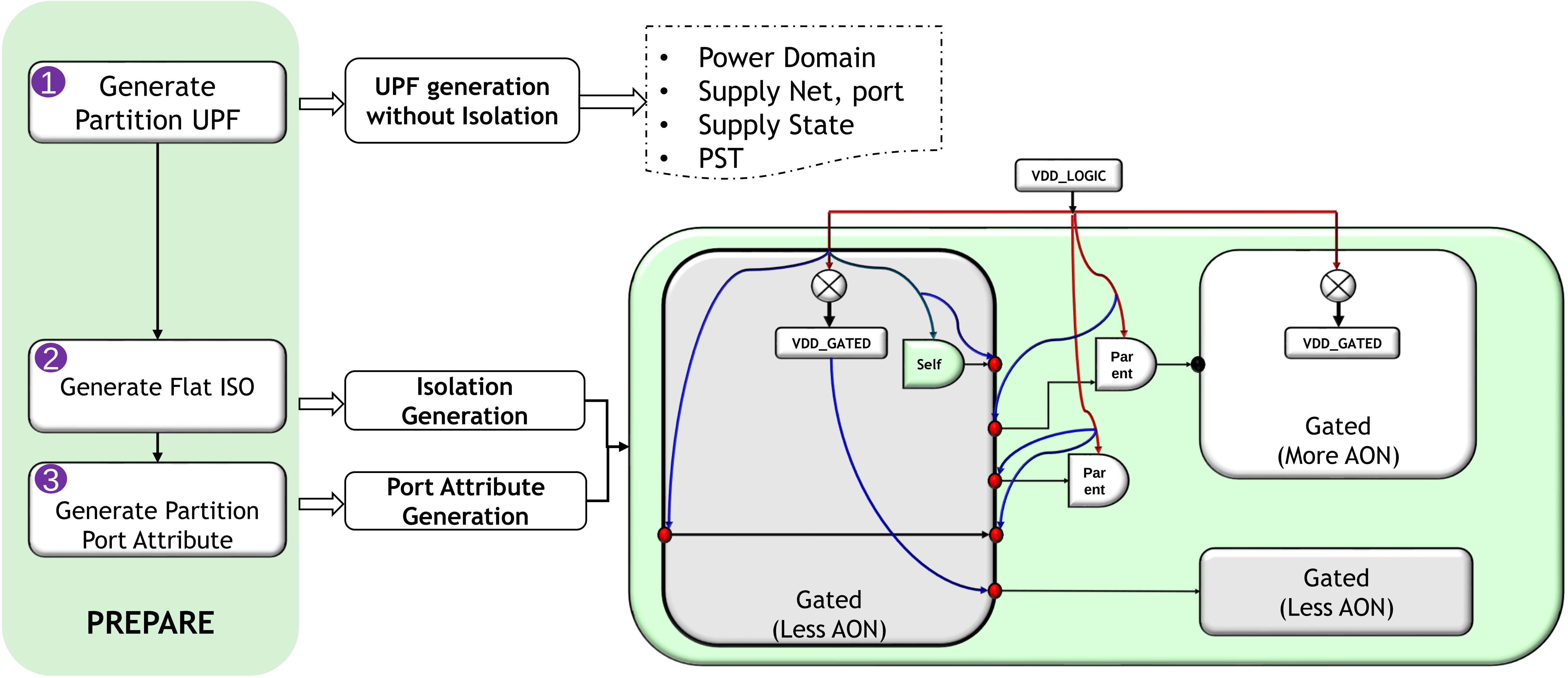
Integrating UPFA into Meta’s Flow

Step2: UPFA Flow



CSV/Scripts
RTL Design
1 Generate Partition UPF
2 Generate Flat ISO
3 Generate Partition Port Attribute
PREPARE
4 Re-Generate Partition with the correct port Attribute
5 Re-Generate Cluster UPF
GENERATE
UPF

PREPARE: Steps to derive Port Attributes



1 Generate Partition UPF

2 Generate Flat ISO

3 Generate Partition Port Attribute

PREPARE

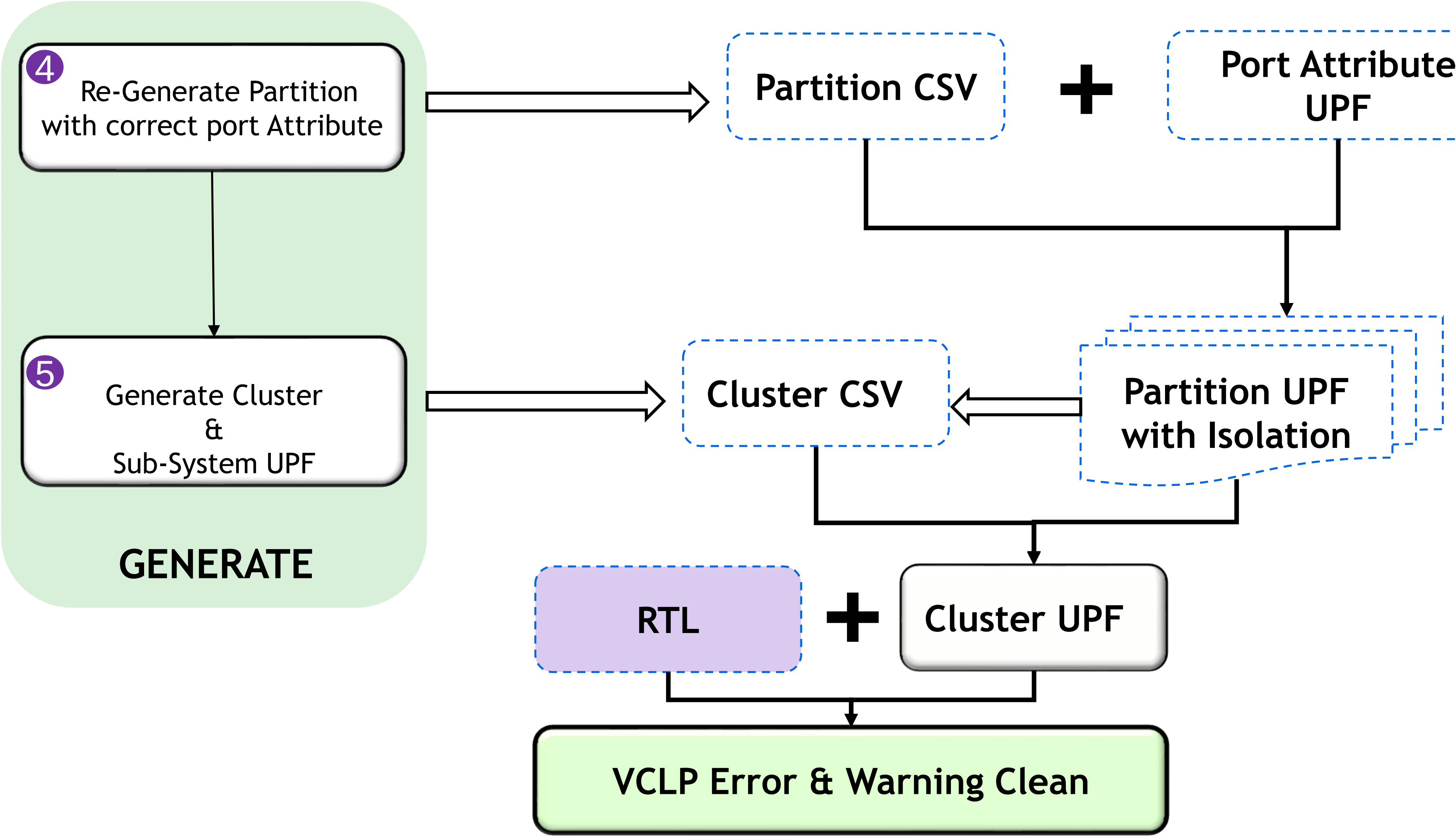
UPF generation without Isolation

- Power Domain
- Supply Net, port
- Supply State
- PST

Isolation Generation

Port Attribute Generation

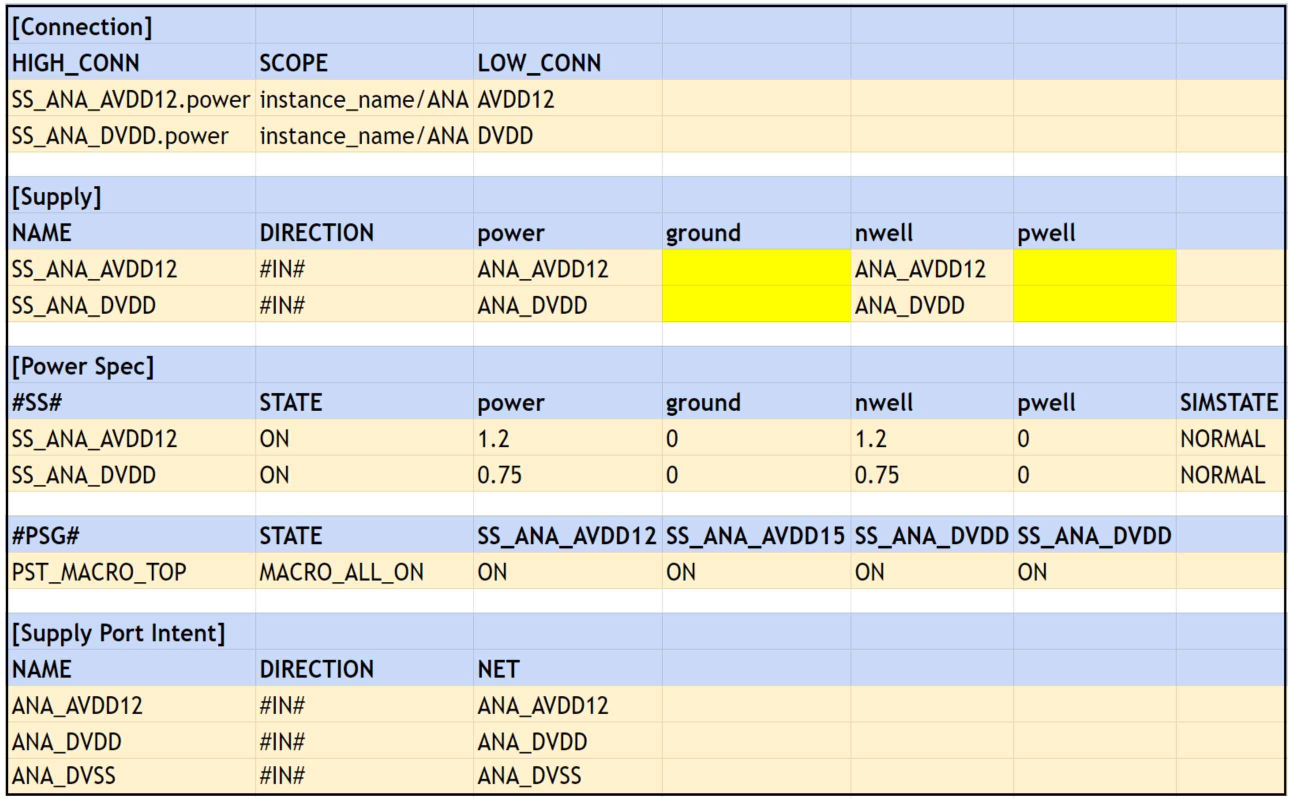
GENERATE: Final UPF generation


4 Re-Generate Partition with correct port Attribute
5 Generate Cluster & Sub-System UPF
GENERATE
Partition CSV
+
Port Attribute UPF
Partition UPF with Isolation
Cluster CSV
RTL
+
Cluster UPF
VCLP Error & Warning Clean

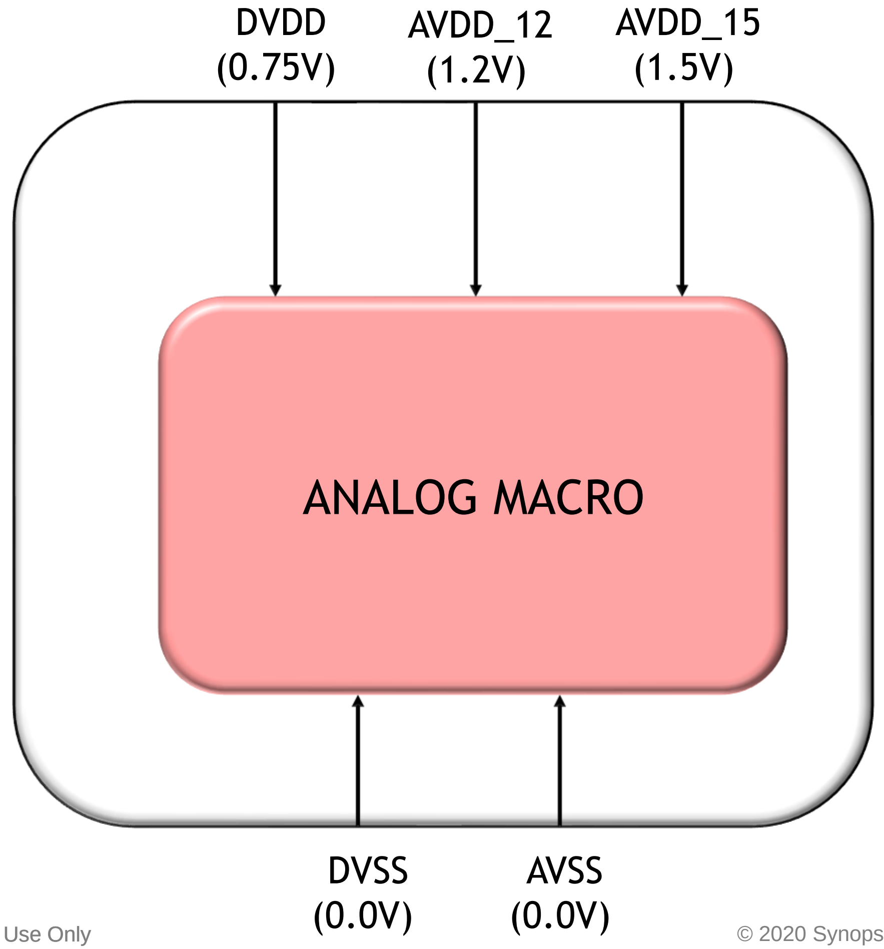
UPFA features: Derive Macro UPF

- Auto generates UPFs for Analog macro instantiated in the design (from .lib)

- Variable to enable Macro CSV generation
  - ugo_dump_macro_pg_pin_report -csv macro.csv -a

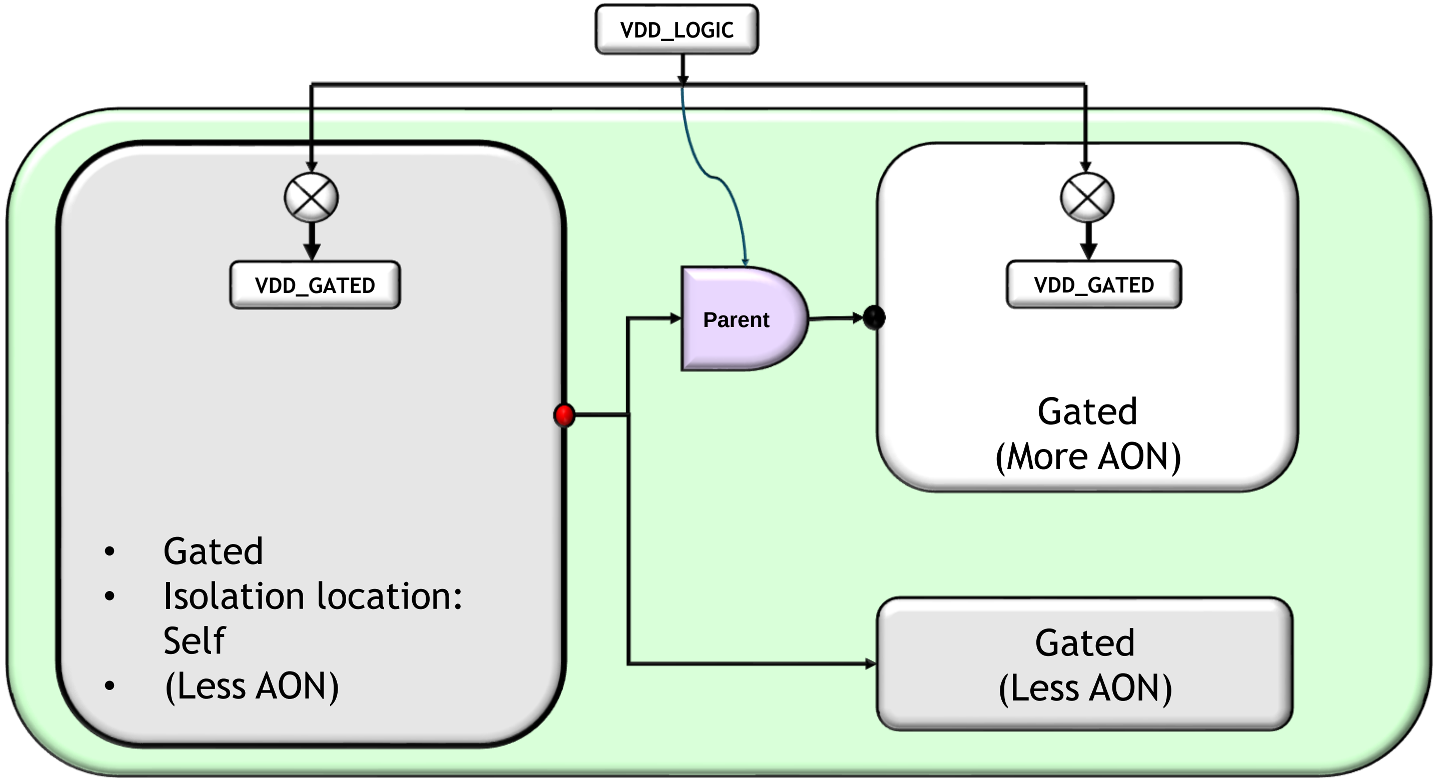


[Connection]						
HIGH_CONN	SCOPE	LOW_CONN				
SS_ANA_AVDD12.power	instance_name/ANA	AVDD12				
SS_ANA_DVDD.power	instance_name/ANA	DVDD				
[Supply]						
NAME	DIRECTION	power	ground	nwell	pwell	
SS_ANA_AVDD12	#IN#	ANA_AVDD12		ANA_AVDD12		
SS_ANA_DVDD	#IN#	ANA_DVDD		ANA_DVDD		
[Power Spec]						
#SS#	STATE	power	ground	nwell	pwell	SIMSTATE
SS_ANA_AVDD12	ON	1.2	0	1.2	0	NORMAL
SS_ANA_DVDD	ON	0.75	0	0.75	0	NORMAL
#PSG#	STATE	SS_ANA_AVDD12	SS_ANA_AVDD15	SS_ANA_DVDD	SS_ANA_DVDD	
PST_MACRO_TOP	MACRO_ALL_ON	ON	ON	ON	ON	
[Supply Port Intent]						
NAME	DIRECTION	NET				
ANA_AVDD12	#IN#	ANA_AVDD12				
ANA_DVDD	#IN#	ANA_DVDD				
ANA_DVSS	#IN#	ANA_DVSS				



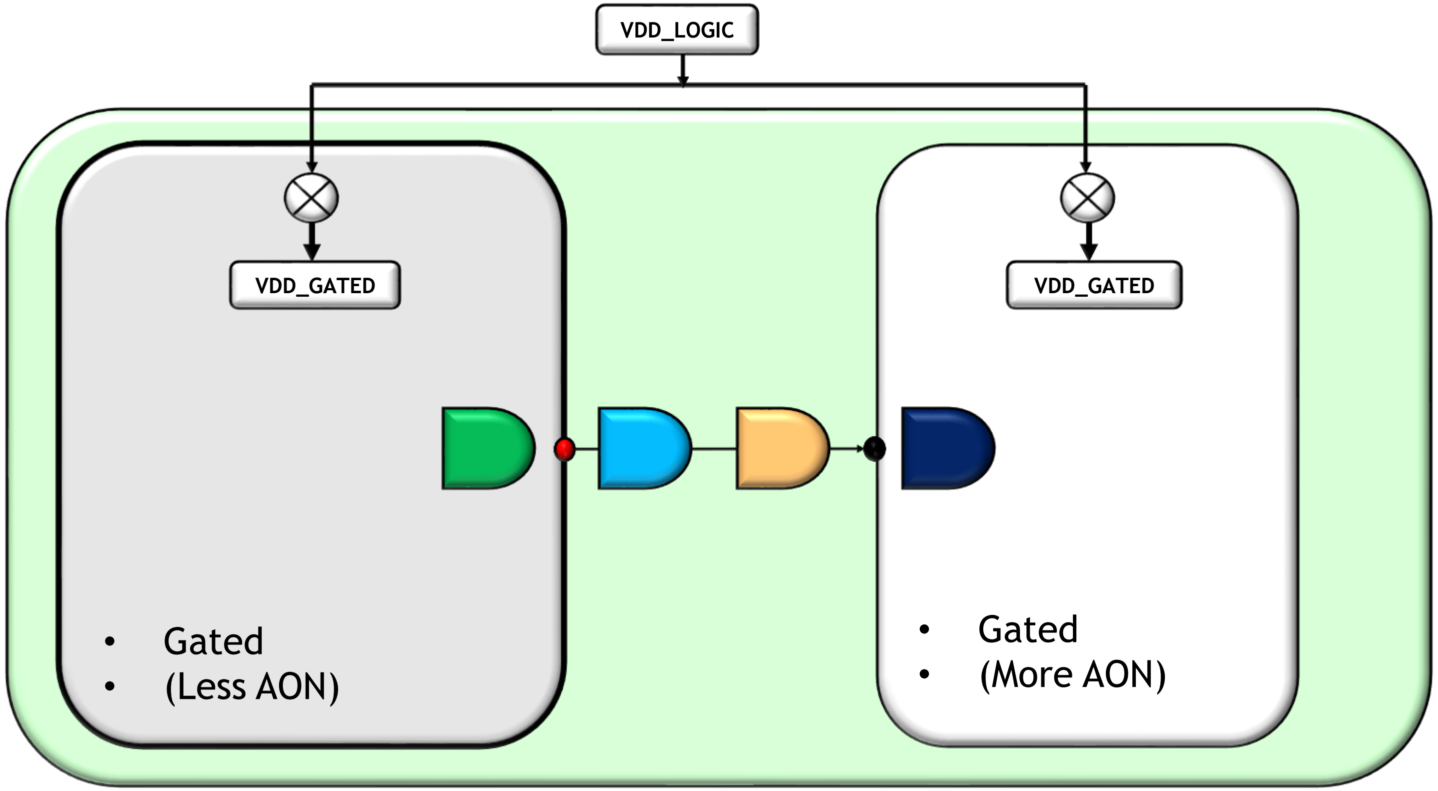
UPFA features: Hetero Fanout

- Hetero fanout: Gated power domain driving multiple sink power domain
- UPFA Optimizer smartly identifies the case of hetero fanout nets, which may lead to additional redundant isolations
- Solves it by adding on the destination domain’s input (Though self-style isolation was specified)



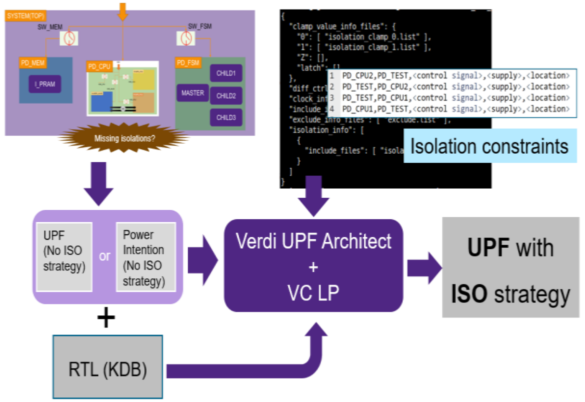
UPFA features: Flexible ISO locations

Isolation	Location	Place
🟩	Self	Source
🟦	Parent	Source
🟧	Parent	Sink
■	Self	Sink



Summary of UPF Architect Features

Optimizer - Generate Isolation Strategies

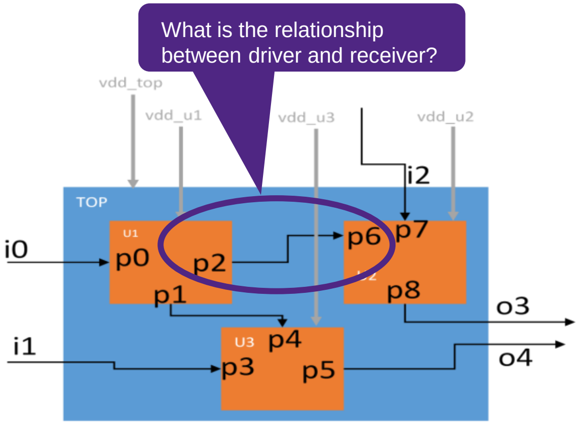


```
{
  "clamp_value_info_files": {
    "0": [ "isolation_clamp_0.list" ],
    "1": [ "isolation_clamp_1.list" ],
    "Z": [],
    "latch": []
  },
  "diff_ctrl
  "clock_inf
  "include_i
  "exclude_info_files": [ "exclude.list" ],
  "isolation_info": [
    {
      "include_files": [ "isola
    }
  ]
}
```

```
1 PD_CPU2,PD_TEST,<control signal>,<supply>,<location>
2 PD_TEST,PD_CPU2,<control signal>,<supply>,<location>
3 PD_TEST,PD_CPU1,<control signal>,<supply>,<location>
4 PD_CPU1,PD_TEST,<control signal>,<supply>,<location>
```

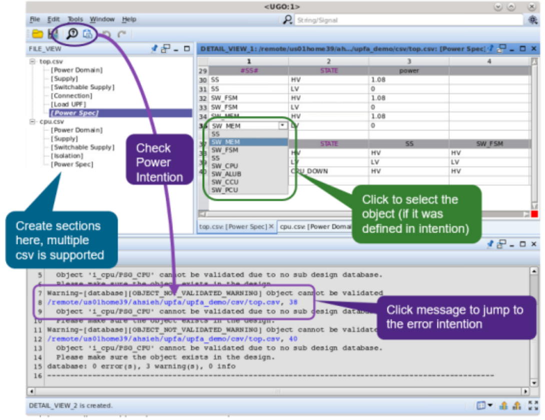
- Leveraging **VC LP** technology
- Analyze RTL design to generate proper Isolation Strategy automatically

Optimizer - UPF Budgeting



- Budgeting sub blocks from top UPF
- Existing UPF for blocks, generate **SPA** (**set_port_attribute**) in top level

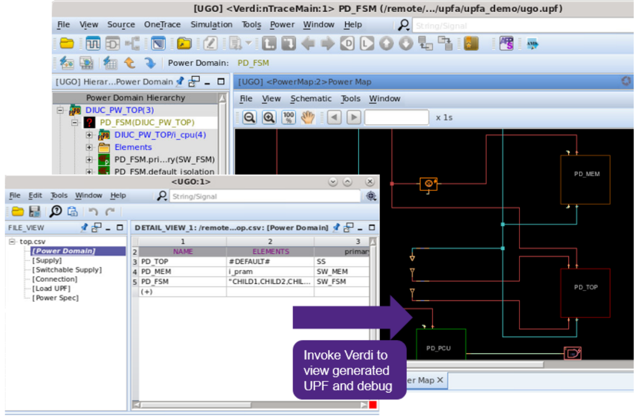
GUI - The Power Intent Editor



```
  Object 'i_cpu/PSG_CPU' cannot be validated due to no sub design database.
  Please make sure the object exists in the design.
Warning-[database][OBJECT_NOT_VALIDATED_WARNING] Object cannot be validated
/remote/us01home39/ahsieh/upfa/upfa_demo/csv/top.csv, 38
  Object 'i_cpu/PSG_CPU' cannot be validated due to no sub design database.
  Please make sure the object exists in the design.
Warning-[database][OBJECT_NOT_VALIDATED_WARNING] Object cannot be validated
/remote/us01home39/ahsieh/upfa/upfa_demo/csv/top.csv, 40
  Object 'i_cpu/PSG_CPU' cannot be validated due to no sub design database.
  Please make sure the object exists in the design.
database: 0 error(s), 3 warning(s), 0 info
--------------------------------------------------------------------------------
DETAIL_VIEW_2 is created.
```

- Input intention in xls-like GUI
  - Intention is syntax/version free
- Check power intention
  - Mismatch objects across the intentions
  - Mismatch objects between intention and design

GUI - Interactive with Verdi


[UGO] <Verdi:nTraceMain:1> PD_FSM (/remote/.../upfa/upfa_demo/ugo.upf)
File View Source OneTrace Simulation Tools Power Window Help
Power Domain: PD_FSM
[UGO] Hierar...Power Domain
Power Domain Hierarchy
DIUC_PW_TOP(3)
PD_FSM(DIUC_PW_TOP)
DIUC_PW_TOP/i_cpu(4)
Elements
[UGO] <PowerMap:2>Power Map
File View Schematic Tools Window
PD_MEM
PD_TOP
PD_PCU
<UGO:1>
FILE_VIEW
top.csv
[Power Domain]
DETAIL_VIEW_1: /remote...op.csv: [Power Domain]
NAME
ELEMENTS
PD_TOP
PD_MEM
PD_FSM
#DEFAULT#
i_pram
SW_MEM
SW_FSM
Invoke Verdi to view generated UPF and debug

- View and debug intentions in **Verdi**
  - Hierarchical Power Domain Tree
  - Power Map: Power domains, ISO, level-shifters…

UPFA Next steps

- Auto generate level shifter strategy commands based on supply connections
- Add support for Full chip integration
  - Black Box models
  - IO PADs connection
- Reduce hierarchical runtime by the recursive reading of CSV’s without the need for intermediate UPFs.
- Enhance Error and Warning reporting.
- Support to read 3rd party UPFs and auto-resolve the strategies at the SS level.
- A more simplistic interface rather than a CSV

Summary

- Generate Analog Macro UPFs.

Impact:

- Reduced manual update for every release of RTL

- 90% reduction in turnaround time for structurally clean UPF generation

Impact:

- Automate hierarchical UPF generation as part of the regression

- Single source of truth

Impact:

- Seamless usage of UPF across all EDA tools.

- Modeling port attributes for dynamically changing interface

Impact:

- Reduced manual update for every release of RTL

Streamlining Low-Power Verification: From UPF to Sign Off

Neeraj Mishra(Google)

Nishant Patel(Synopsys)

Bhaumik Matholia (Synopsys)

Agenda

Predictive VC LP Vefn

- Introduction
- Instrumentation Checks
- Common Issue Resolved
- Benefits & Results
- Future Work

Hierarchical VC LP Vefn

- Why Hier Lp Vefn
- Various Hier VC LP Flows
- SAM Model based Flow
- Qor Analysis
- Benefits & Results

ML Aided VC LP Vefn

- ML-RCA Overview
- Results
- Conclusion Future Work

Agenda

Predictive VC LP Vefn

- Introduction
- Instrumentation Checks
- Common Issue Resolved
- Benefits & Results
- Future Work

Hierarchical VC LP Vefn

Why Hier Lp Vefn

Various Hier VC LP Flows

SAM Model based Flow

Qor Analysis

Benefits & Results

ML Aided VC LP Vefn

ML-RCA Overview

Results

Conclusion Future Work

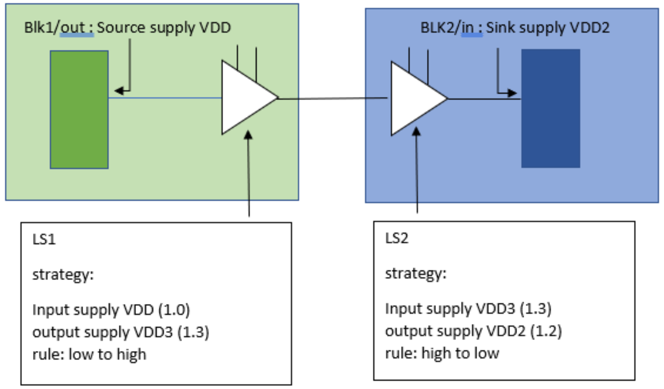
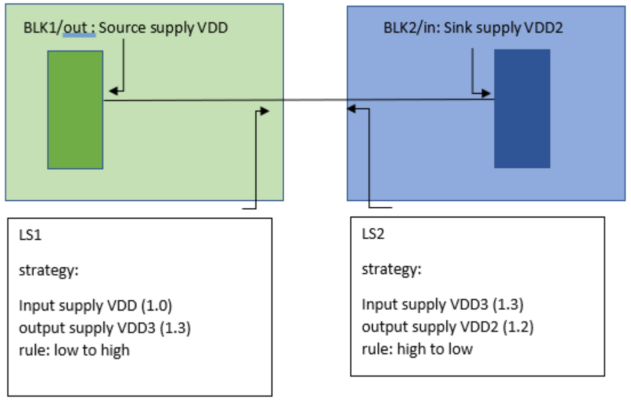
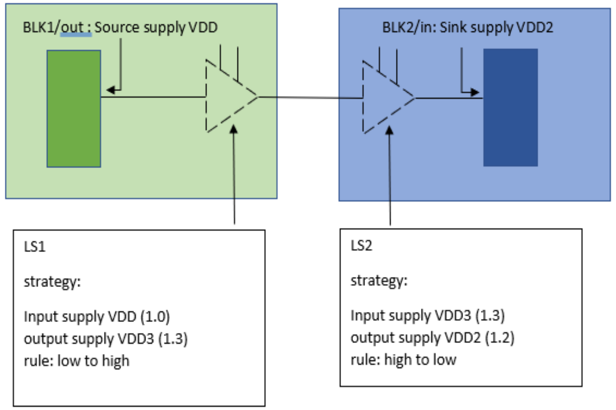
Predictive VC LP Vefn

- Problem Statement -
    - Seeing Inaccurate/Inconsistent violations in Rtl Vclp vs Syn Vclp
        - At RTL stage, VC LP checks do not consider isolation/level-shifter/retention/power-switch cells on the paths where strategies are getting applied
            - post-synthesis electrical or functional issues are introduced due to inserted MV cells
        - Difficulty in finding real violations
    - LP violations identified later in netlist stage has significant cost
        - Very costly to push Rtl fix at late stage
        - Might require upf update and re-synthesis
        - Impacts overall convergence time.
        - Shift Left needed for all Lp checks

- Solution - New **Instrumentation Checks** feature of Vclp
    - Tool inserts and predicts the position of LP cells in netlist based on provided Rtl
    - Runs Vclp check based on these virtual isolation/LS cells
    - Close collaboration with Snps led to almost **80% reduction in noisy violations** for Rtl Level Full flat Soc Run

Common Issues Fixed (The LS checks are considered as noise violations and are ignored at netlist and predictive stage)

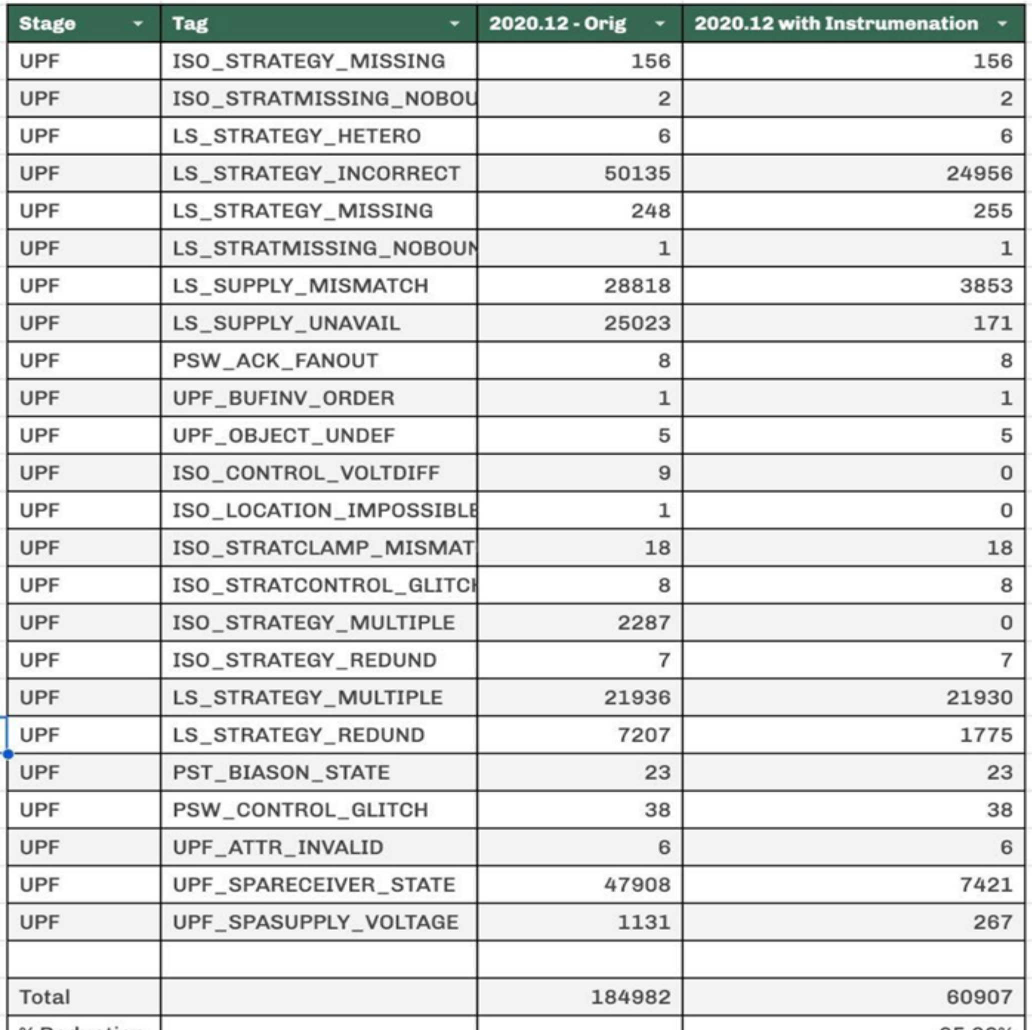
LS_SUPPLY_MISMATCH/LS_STRATEGY_INCORRECT/LS_SUPPLY_UNAVAIL

	NETLIST	RTL	RTL with Level shifter instrumentation
	 LS1 strategy: Input supply VDD (1.0) output supply VDD3 (1.3) rule: low to high LS2 strategy: Input supply VDD3 (1.3) output supply VDD2 (1.2) rule: high to low	 LS1 strategy: Input supply VDD (1.0) output supply VDD3 (1.3) rule: low to high LS2 strategy: Input supply VDD3 (1.3) output supply VDD2 (1.2) rule: high to low	 LS1 strategy: Input supply VDD (1.0) output supply VDD3 (1.3) rule: low to high LS2 strategy: Input supply VDD3 (1.3) output supply VDD2 (1.2) rule: high to low
LS_SUPPLY_MISMATCH	LS Strategies consider the input/output supplies of the adjacent LS cells LS1 : input supply with BLK1/out : No violation LS1 : output supply with LS2/in : No violation LS2 : input supply with LS1/out : No violation LS2 : output supply with BLK2/in : No violation	LS Strategies compare the input/output supplies with actual source/sink supplies LS1 : input supply with BLK1/out : No violation LS1 : output supply with BLK2/in: LS_SUPPLY_MISMATCH □ Noise LS2 : input supply with BLK1/out LS_SUPPLY_MISMATCH □ Noise LS2 : output supply with BLK2/in : No violation	LS Strategies consider the input/output supplies of the adjacent LS strategies LS1 : input supply with BLK1/out : No violation LS1 : output supply with LS2/in : No violation LS2 : input supply with LS1/out : No violation LS2 : output supply with BLK2/in : No violation
LS_STRATEGY_INCORRECT	LS Strategies consider the input/output supplies of the adjacent LS cells LS1 : BLK1/out and LS2/in : No violation LS2 : LS1/out and BLK2/in: No violation	LS Strategies consider the actual source/sink supplies LS1 : BLK1/out and BLK2/in : No violation LS2 : BLK1/out and BLK2/in: LS_STRATEGY_INCORRECT □ Noise	LS Strategies consider the input/output supplies of the adjacent LS strategies LS1 : BLK1/out and LS2/input supply : No violation LS2 : LS1/output supply and BLK2/in: No violation

Benefits & Results

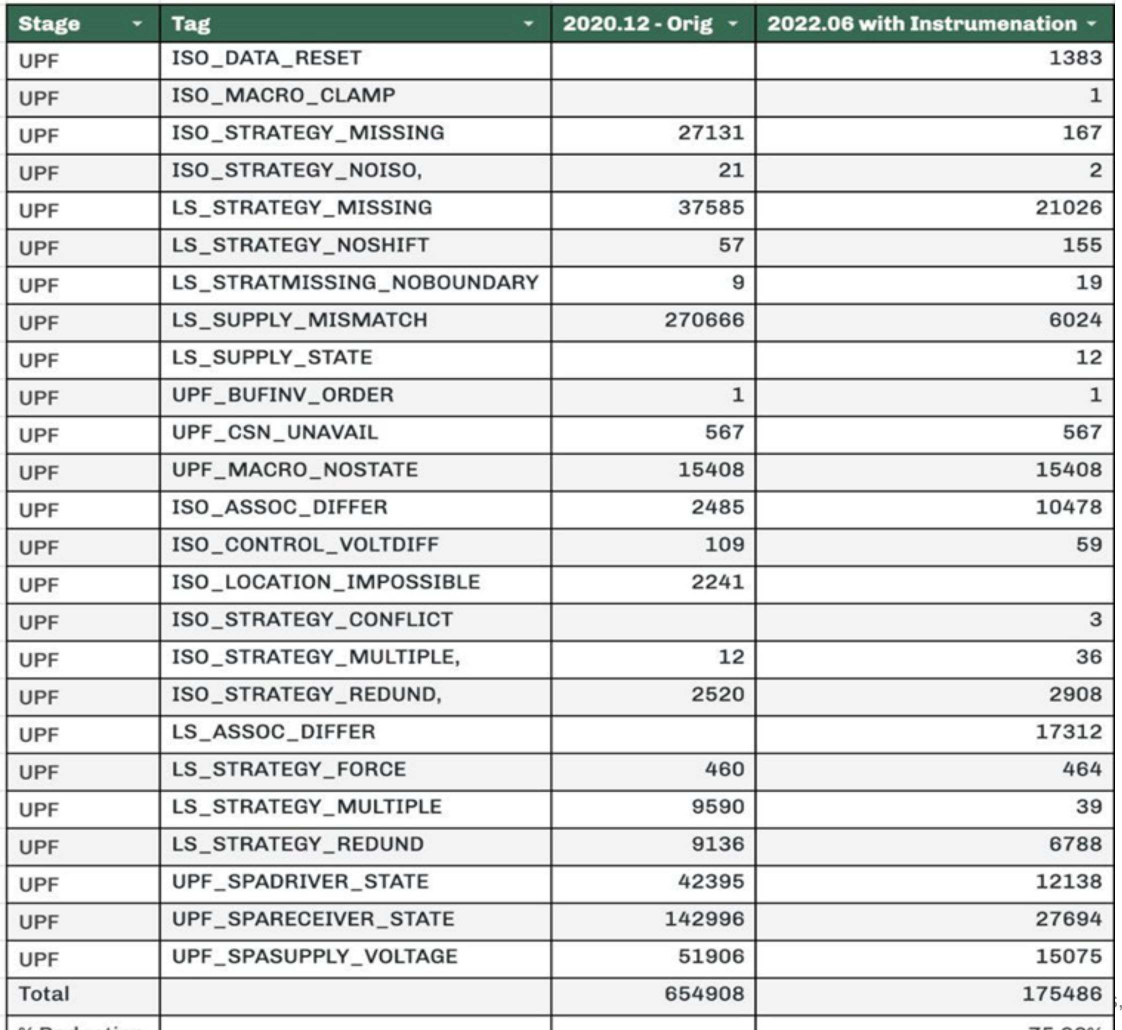
- Collaborative Work with Synopsys to evaluate this feature began 2 years ago
    - Tested over 3 generations of a Mobile Soc with increasing benefit and noise reduction

2020 build was giving 65% reduction in false violations (Soc-1)



Stage	Tag	2020.12 - Orig	2020.12 with Instrumenation
UPF	ISO_STRATEGY_MISSING	156	156
UPF	ISO_STRATMISSING_NOBOU	2	2
UPF	LS_STRATEGY_HETERO	6	6
UPF	LS_STRATEGY_INCORRECT	50135	24956
UPF	LS_STRATEGY_MISSING	248	255
UPF	LS_STRATMISSING_NOBOUN	1	1
UPF	LS_SUPPLY_MISMATCH	28818	3853
UPF	LS_SUPPLY_UNAVAIL	25023	171
UPF	PSW_ACK_FANOUT	8	8
UPF	UPF_BUFINV_ORDER	1	1
UPF	UPF_OBJECT_UNDEF	5	5
UPF	ISO_CONTROL_VOLTDIFF	9	0
UPF	ISO_LOCATION_IMPOSSIBLE	1	0
UPF	ISO_STRATCLAMP_MISMAT	18	18
UPF	ISO_STRATCONTROL_GLITCH	8	8
UPF	ISO_STRATEGY_MULTIPLE	2287	0
UPF	ISO_STRATEGY_REDUND	7	7
UPF	LS_STRATEGY_MULTIPLE	21936	21930
UPF	LS_STRATEGY_REDUND	7207	1775
UPF	PST_BIASON_STATE	23	23
UPF	PSW_CONTROL_GLITCH	38	38
UPF	UPF_ATTR_INVALID	6	6
UPF	UPF_SPARECEIVER_STATE	47908	7421
UPF	UPF_SPASUPPLY_VOLTAGE	1131	267
Total		184982	60907

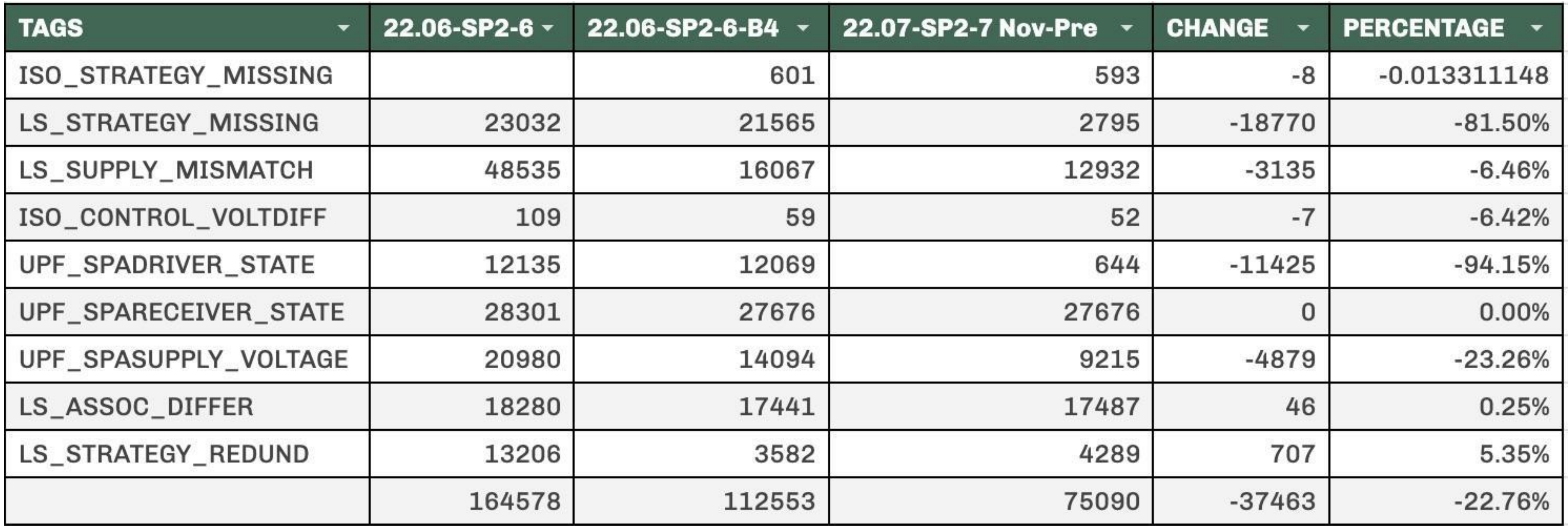
2022 build gave upto 75% reduction in false (Soc-1.1)



Stage	Tag	2020.12 - Orig	2022.06 with Instrumenation
UPF	ISO_DATA_RESET		1383
UPF	ISO_MACRO_CLAMP		1
UPF	ISO_STRATEGY_MISSING	27131	167
UPF	ISO_STRATEGY_NOISO,	21	2
UPF	LS_STRATEGY_MISSING	37585	21026
UPF	LS_STRATEGY_NOSHIFT	57	155
UPF	LS_STRATMISSING_NOBOUNDARY	9	19
UPF	LS_SUPPLY_MISMATCH	270666	6024
UPF	LS_SUPPLY_STATE		12
UPF	UPF_BUFINV_ORDER	1	1
UPF	UPF_CSN_UNAVAIL	567	567
UPF	UPF_MACRO_NOSTATE	15408	15408
UPF	ISO_ASSOC_DIFFER	2485	10478
UPF	ISO_CONTROL_VOLTDIFF	109	59
UPF	ISO_LOCATION_IMPOSSIBLE	2241	
UPF	ISO_STRATEGY_CONFLICT		3
UPF	ISO_STRATEGY_MULTIPLE,	12	36
UPF	ISO_STRATEGY_REDUND,	2520	2908
UPF	LS_ASSOC_DIFFER		17312
UPF	LS_STRATEGY_FORCE	460	464
UPF	LS_STRATEGY_MULTIPLE	9590	39
UPF	LS_STRATEGY_REDUND	9136	6788
UPF	UPF_SPADRIVER_STATE	42395	12138
UPF	UPF_SPARECEIVER_STATE	142996	27694
UPF	UPF_SPASUPPLY_VOLTAGE	51906	15075
Total		654908	175486

Future Work

- Continuous improvement for better correlation w.r.t Synthesis runs (Soc-1.2)
  - Dec‘23 rel-Vclp 22.07 build giving ~80% reduction compared to 2020 build



TAGS	22.06-SP2-6	22.06-SP2-6-B4	22.07-SP2-7 Nov-Pre	CHANGE	PERCENTAGE
ISO_STRATEGY_MISSING		601	593	-8	-0.013311148
LS_STRATEGY_MISSING	23032	21565	2795	-18770	-81.50%
LS_SUPPLY_MISMATCH	48535	16067	12932	-3135	-6.46%
ISO_CONTROL_VOLTDIFF	109	59	52	-7	-6.42%
UPF_SPADRIVER_STATE	12135	12069	644	-11425	-94.15%
UPF_SPARECEIVER_STATE	28301	27676	27676	0	0.00%
UPF_SPASUPPLY_VOLTAGE	20980	14094	9215	-4879	-23.26%
LS_ASSOC_DIFFER	18280	17441	17487	46	0.25%
LS_STRATEGY_REDUND	13206	3582	4289	707	5.35%
	164578	112553	75090	-37463	-22.76%

- Major update in VC LP 2023.12 - Design based instrumentation
  - No need to modify vclp internal db limiting scope miscorelation

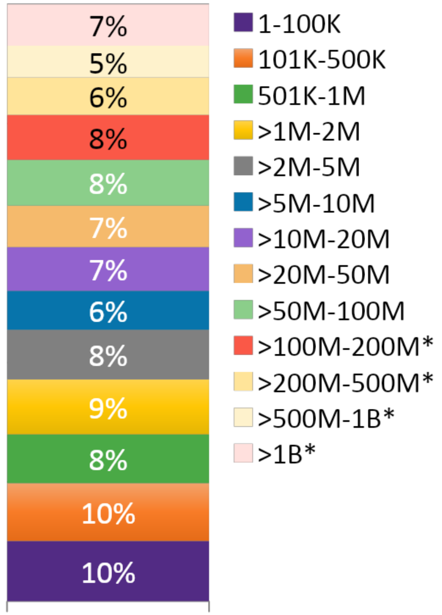
Agenda

Hierarchical VC LP Vefn

- Why Hier LP Vefn
- Various Hier VC LP Flows
- SAM Model based Flow
- Qor Analysis
- Benefits & Results

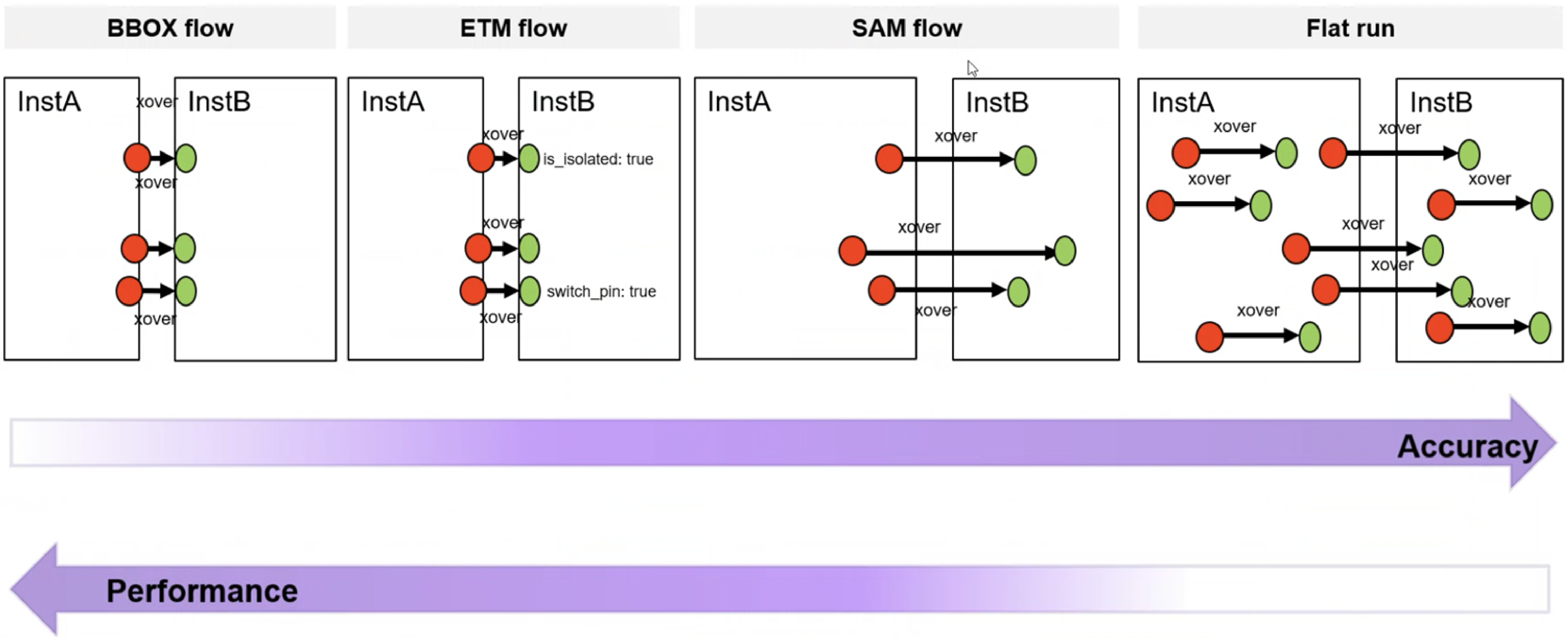
Why Hierarchical VC LP Flow?

- **Increasing design sizes**
  - Considerable number of designs are > 100M
- **LP Signoff in Flat SoC takes a long time**
  - For our designs more than 140 hours
- **Next generation SoCs are expected to be even larger and cannot rely on full-flat run**
- **Integration of large number of IPs with standalone UPF files validated at block level**
- **Existing hierarchical flows do not cover all aspects of Signoff**

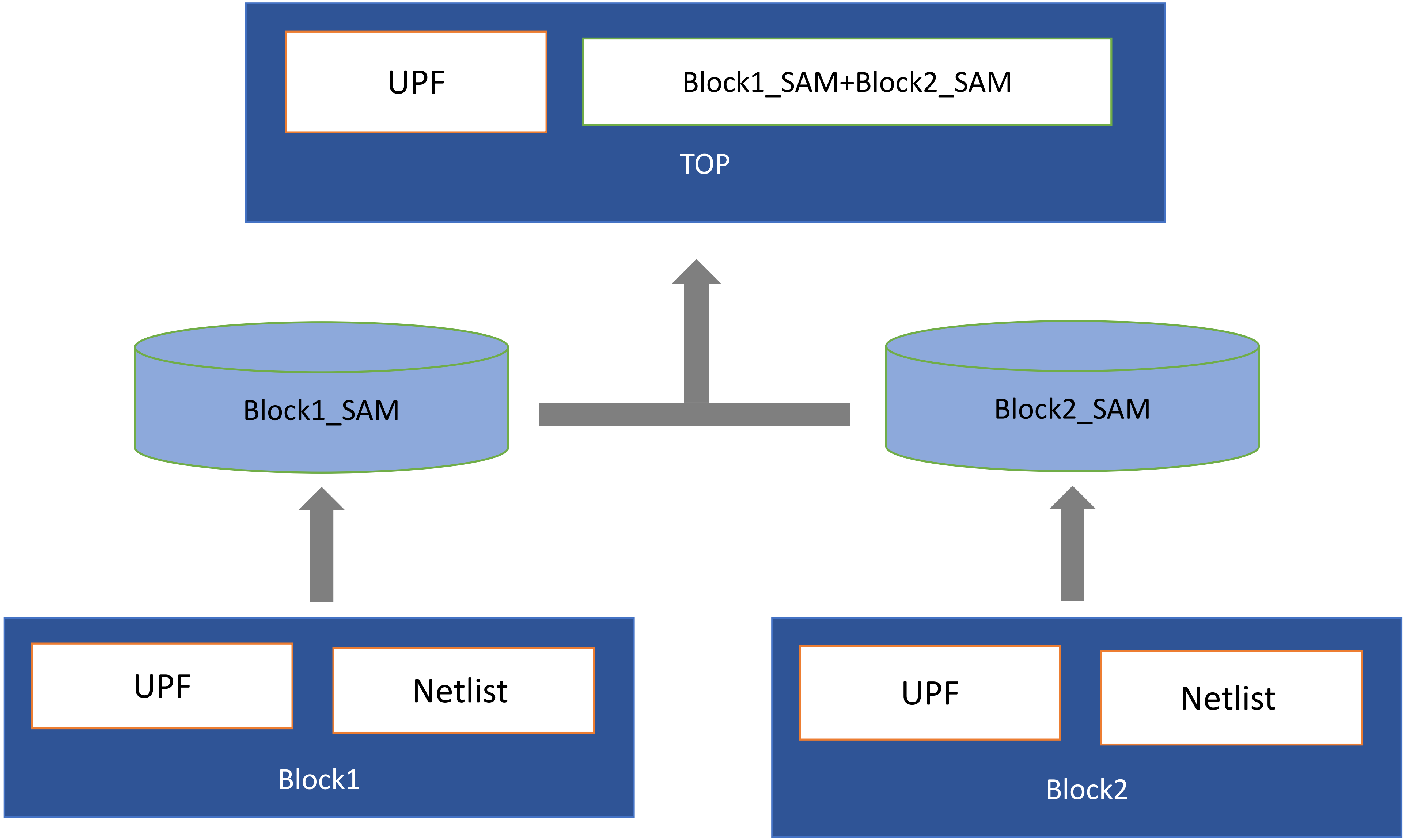


Engineers need a methodology to handle next generation chip signoff with considerably better performance and capacity, and **NO** loss in QoR or signoff confidence.

BBOX/ETM/SAM/FLAT VC LP Flows

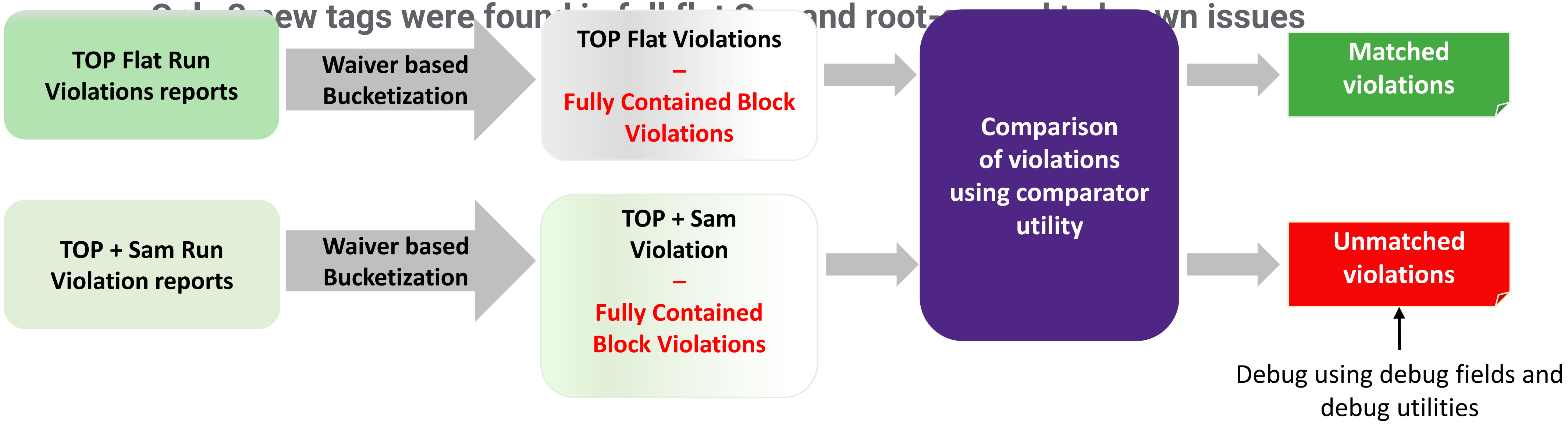


SAM Model based Flow



Qor Analysis - Subtractive QoR Flow

- Ran Full Flat Vclp run for Entire Soc - Compared against SAM model based VCLP
  - Only 3 new tags were found in full flat SoC and root-caused to known issues


TOP Flat Run Violations reports
Waiver based Bucketization
TOP Flat Violations – Fully Contained Block Violations
TOP + Sam Run Violation reports
Waiver based Bucketization
TOP + Sam Violation – Fully Contained Block Violations
Comparison of violations using comparator utility
Matched violations
Unmatched violations
Debug using debug fields and debug utilities

```
Unmatched Violations
UPF     UPF_OBJECT_UNDEF           275          0  ← ok. UPF reference IP internal ports, not present in SAM run
Design  ANALOG_NET_INCORRECT       18         106  ← ok. Not real violation and flagged by other waived checks.

UPF     ISO_STRATCONTROL_GLITCH    37         153  ← warning on iso ctrl signal thru dft_mux-wouldn't find in SAM
```

Benefits & Results

- **Large Improvement in Runtime and memory making daily batch run feasible**
  - **No Impact on QOR**

Design	Runtime Flat	Runtime Top+SAM		Peak Memory Flat (MB)	Peak Memory Top+SAM (MB)	Mem Gain
SOC 1	148 hours	13 hours	**11x**	840540	152624	**5.5x**

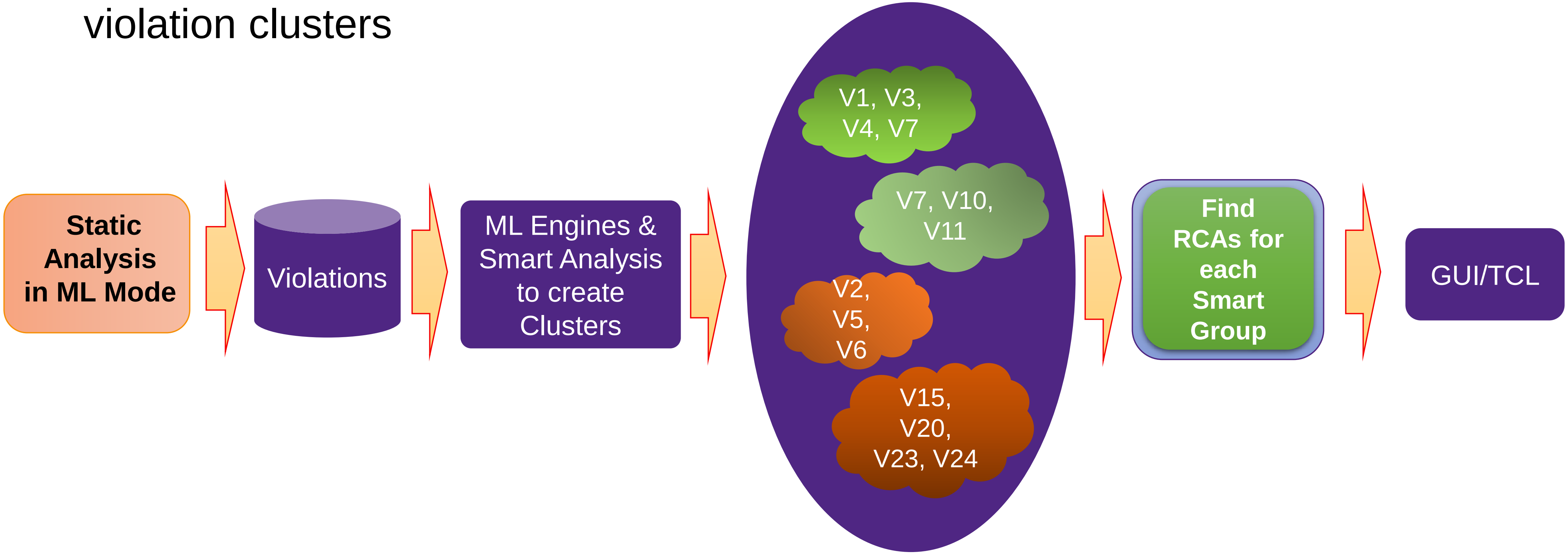
Agenda

ML Aided VC LP Vefn

- ML-RCA Overview
- Evaluation Results
- Conclusion & Future Work

ML RCA Overview

- Uses ML to cluster violations in different categories
- Performs Root Cause analysis to identify reasons for violation clusters


Static Analysis in ML Mode
Violations
ML Engines & Smart Analysis to create Clusters
V1, V3, V4, V7
V7, V10, V11
V2, V5, V6
V15, V20, V23, V24
Find RCAs for each Smart Group
GUI/TCL

Results - IP TOP, Pre-Waiver

- Stage: Post Synthesis
- Initial violation count: **264139**
  - Errors: **7667**
    - Pre RCA categories: **14**
  - Warnings: **255681**
    - Pre RCA categories**:** **32**
- Post RCA Clustering**:**
  - Clusters**: 21**
  - Coverage: **4.23%**

Analysis of Clusters

- Was able to identify the most obvious violations:

🟥 Waived

🟧 Not helpful

🟩 Helpful

Rootcause	Clusters	Total Coverage (%)
🟥 DEBUG_SETUP_CONST	7	1.45
🟥 UPF_SPASUPPLY_CONN	7	0.32
🟧 DEBUG_PST_STATE	4	2.1
🟩 DEBUG_ISO_OPTION	3	0.4

Results - IP Top, Post-Waiver

- Stage: Post Synthesis
- Initial violation count: **1156**
    - Errors: **227**
        - Pre RCA categories: **6**
    - Warnings: **314**
        - Pre RCA categories**: 6**
- Post RCA Clustering**:**
    - Clusters**: 6**
    - Coverage: **43.48%**

Analysis of Clusters

🟥 Waived

🟧 Not helpful

🟩 Helpful

- Helpful in identifying rootcauses of some categories:
- Generic problems identified for majority → no specific cause

Root Cause	Clusters	Total Coverage (%)
DEBUG_ISO_MISSING	2	4.55
DEBUG_ISO_REDUND	2	5.93
DEBUG_ISO_SETUP	1	31.62
DEBUG_PST_STATE	1	1.38

Conclusion

- ML cluster friendly design → Many violations with fewer root causes instead of Few violations with multiple root causes instead of
- Coverage improvement with Waivers using ML RCA = 43.48%
- Not particularly helpful for dirty designs. Best used after initial analysis and waiver of Noisy Violations.

Thanks & Questions

VC SpyGlass CDC – UPF Aware CDC

Power-aware CDC

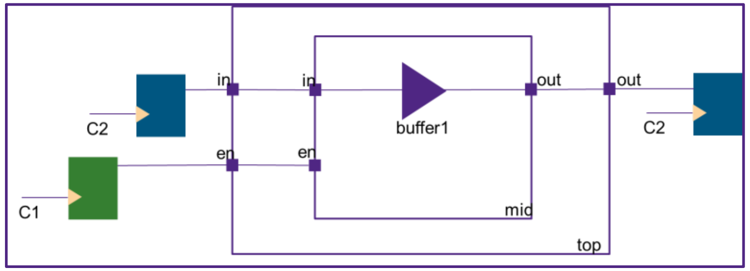
Synopsys Product Team

Sept 2022

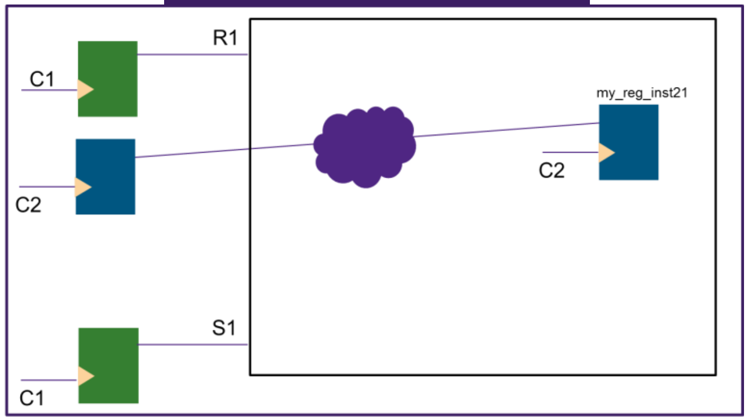
UPF Aware Instrumentation

No Instrumentation

Pre-Isolation Insertion



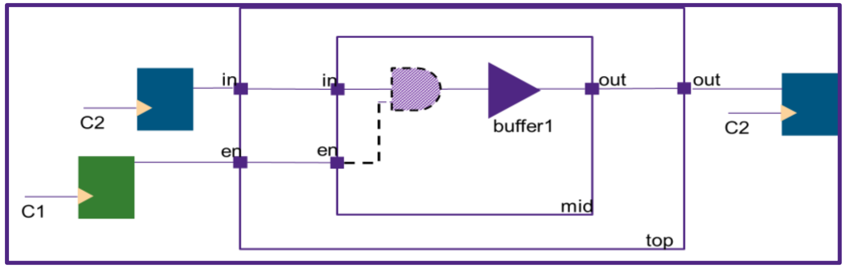
Pre-Retention Insertion



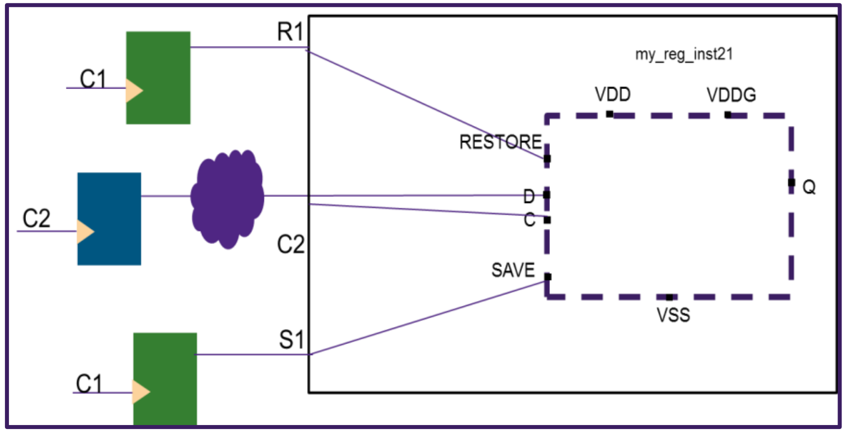
Signoff ✗

Virtual Instrumentation

Virtual-Isolation Insertion



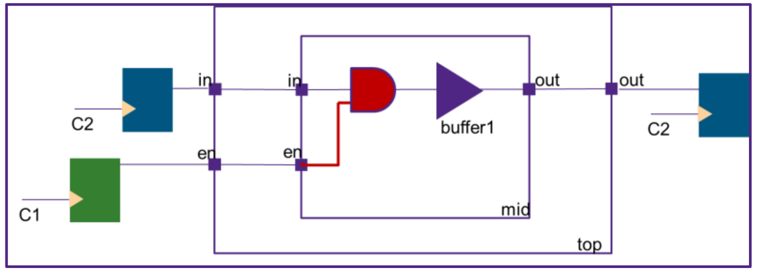
Virtual-Retention Insertion



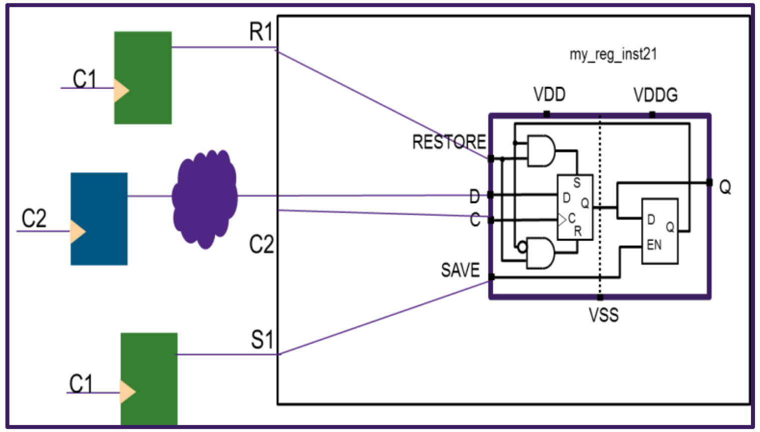
Signoff ?

Real Instrumentation

Real-Isolation Insertion

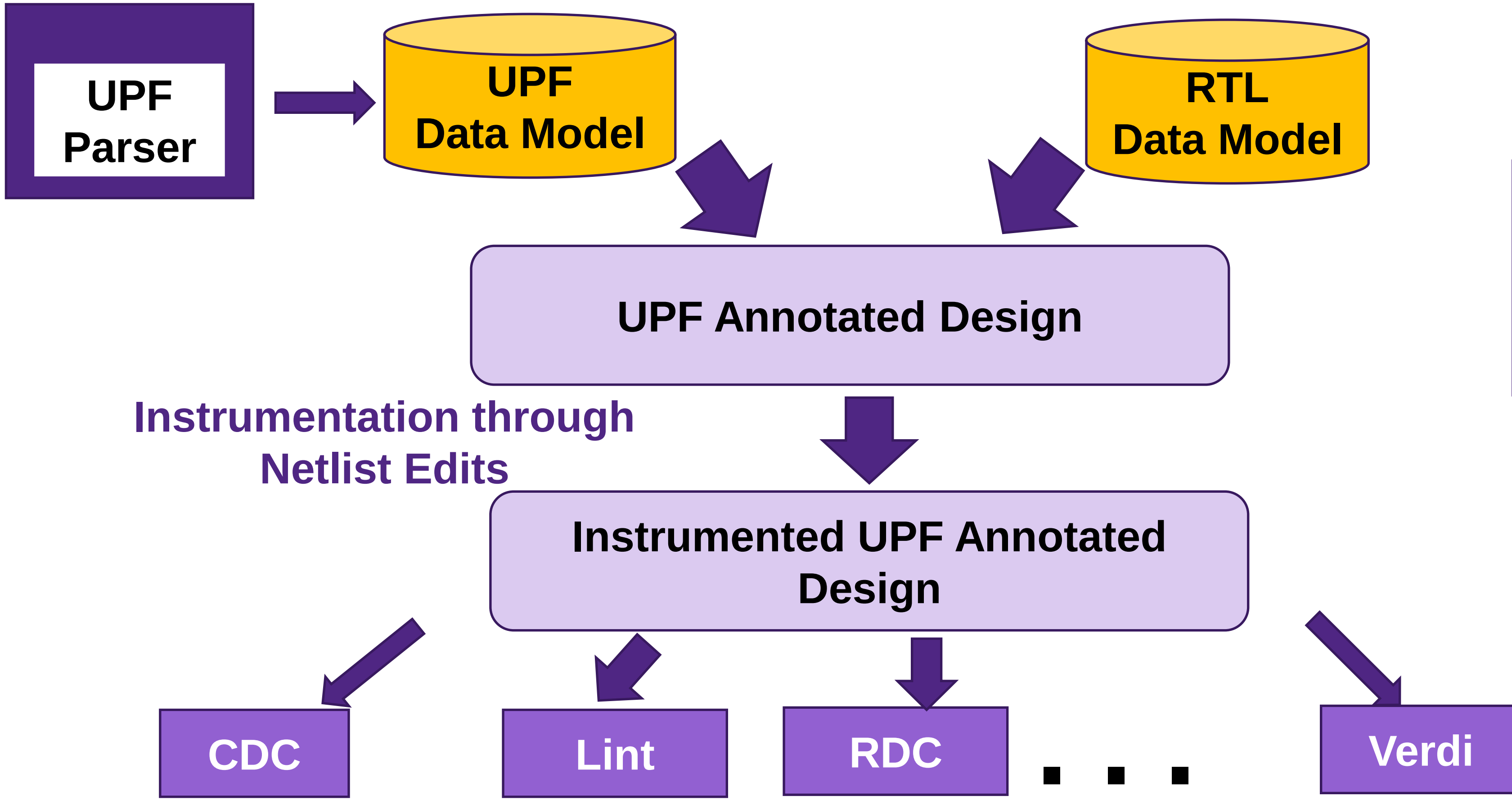


Real-Retention Insertion

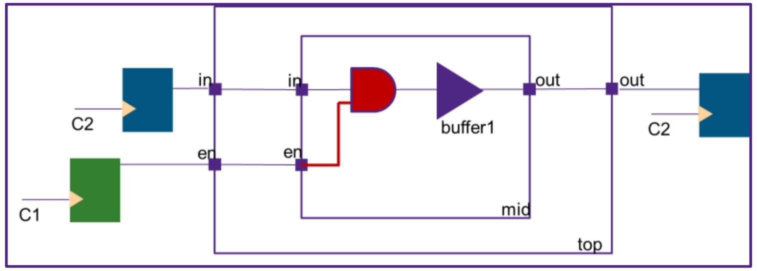


Signoff ✓

Power-aware Static Verification



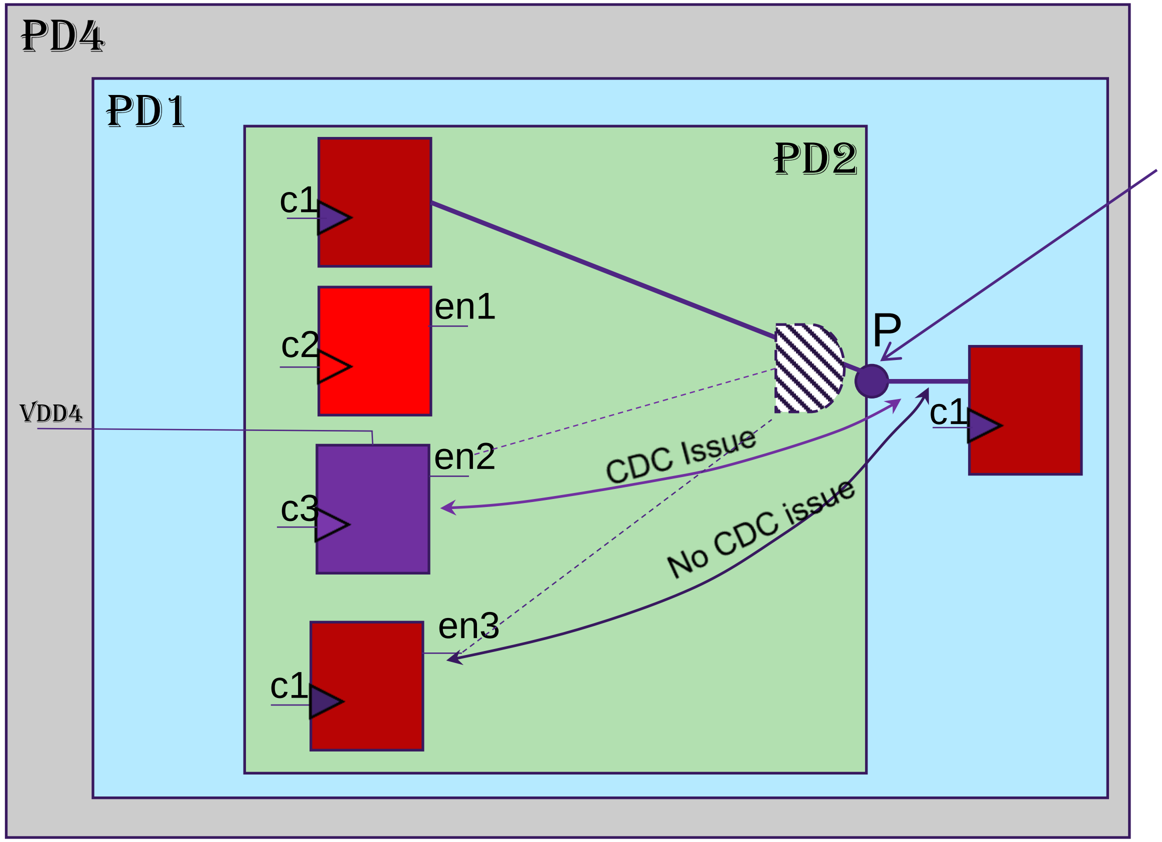
Real-Isolation Insertion



```
set_app_var enable_lp_instrument true
….
read_upf top.upf
```

Leverage industry-proven VC LP engines to detect Static bugs due to UPF instrumentation

Consistent UPF Analysis is Crucial



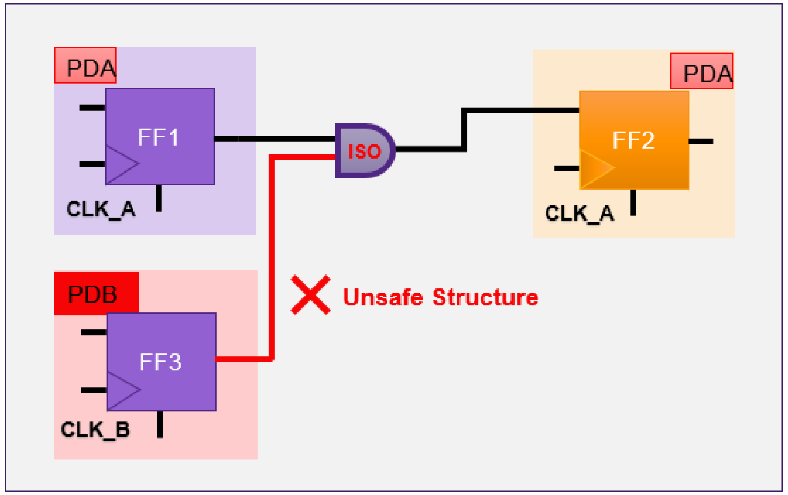
Multiple ISO Policies on P

set_isolation iso3 –enable **en3** –applies_to output
set_isolation iso2 –enable **en2** –source PD4 –diff_supply
set_isolation iso1 –enable **en1** –source PD3 –sink PD1

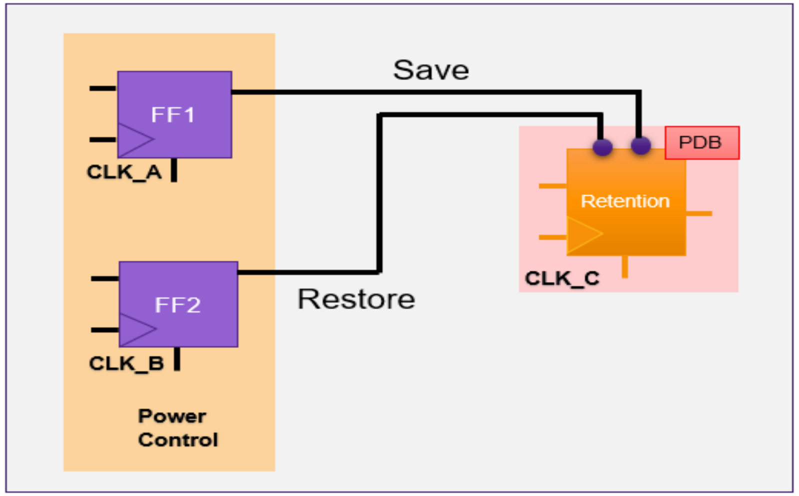
- ISO policy precedence analysis must be done exactly same as what DC/ICC will do
- **Otherwise potential bugs will be missed in RTL**, or false negative violation

CDC Verification Scenarios on UPF Instrumented Logic

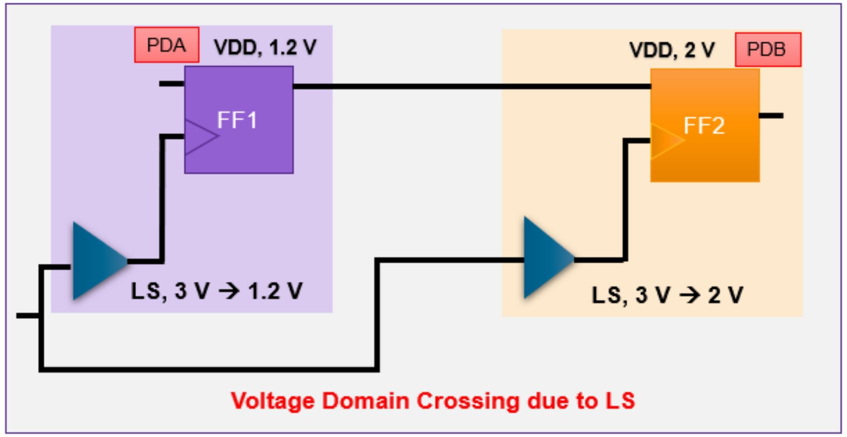
Isolation Cell Insertion Causing Metastability



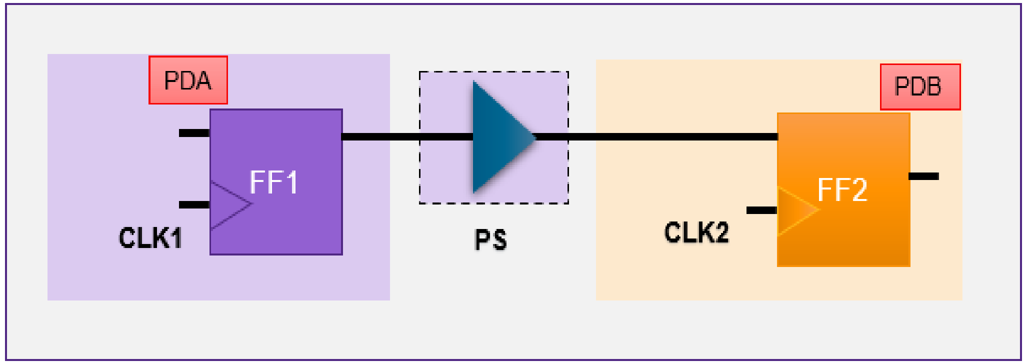
CDC on Save-restore pins of Retention Flops



Level-shifters causing New Clock-domain

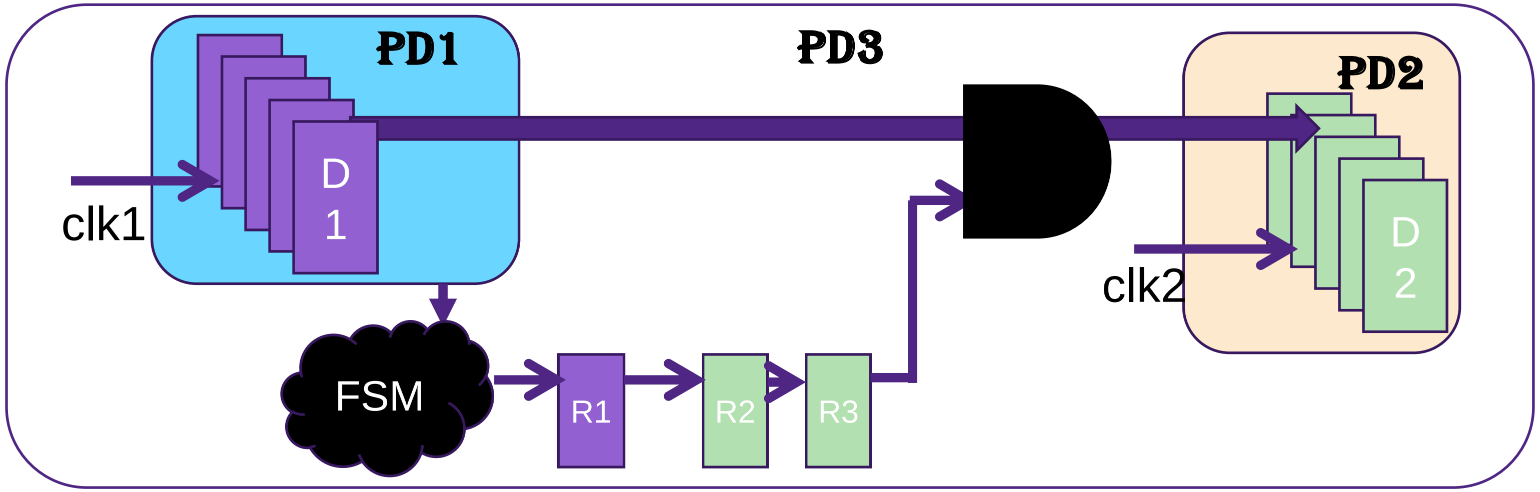


Logic Connect Net introducing New CDC Path



Case Study: Graphics Sub System

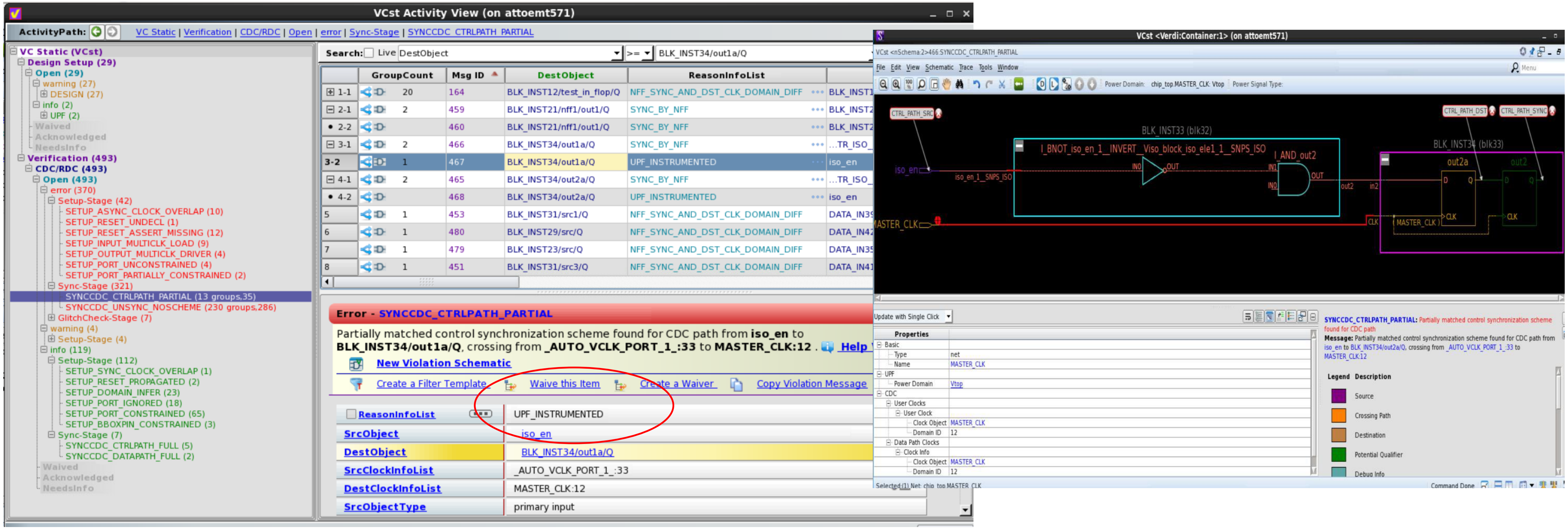
Detecting LP aware CDC paths at Netlist



- Requirements : Identify all CDC paths which cross power domains
- Solution:
  - VC Static Platform contains both VC SpyGlass CDC and VC LP (LP Signoff tool)
  - CDC algorithm identifies all clock domain crossing paths (get_cdc_paths –from clk1* -to clk2*)
  - VC LP Algorithm identifies power domain crossing (get_crossover –source net1 –sink net2)
  - CDC Path with power domain crossing can be identified with small enhancement
    - get_cdc_paths –from clk1* -to clk2* **-filter "power_domain_crossing==true"**

PA CDC Violations Debug in VC SpyGlass

- VC SpyGlass supports in all the checks – No separate rules
  - Reports reason code for the violations on UPF instrumented logic
  - Verdi highlights the instrumented logic in different color



VCst Activity View (on attoemt571)

ActivityPath: VC Static | Verification | CDC/RDC | Open | error | Sync-Stage | SYNCCDC_CTRLPATH_PARTIAL

- VC Static (VCst)
  - Design Setup (29)
    - Open (29)
      - warning (27)
        - DESIGN (27)
      - info (2)
        - UPF (2)
    - Waived
    - Acknowledged
    - NeedsInfo
  - Verification (493)
    - CDC/RDC (493)
      - Open (493)
        - error (370)
          - Setup-Stage (42)
            - SETUP_ASYNC_CLOCK_OVERLAP (10)
            - SETUP_RESET_UNDECL (1)
            - SETUP_RESET_ASSERT_MISSING (12)
            - SETUP_INPUT_MULTICLK_LOAD (9)
            - SETUP_OUTPUT_MULTICLK_DRIVER (4)
            - SETUP_PORT_UNCONSTRAINED (4)
            - SETUP_PORT_PARTIALLY_CONSTRAINED (2)
          - Sync-Stage (321)
            - SYNCCDC_CTRLPATH_PARTIAL (13 groups,35)
            - SYNCCDC_UNSYNC_NOSCHEME (230 groups,286)
          - GlitchCheck-Stage (7)
        - warning (4)
          - Setup-Stage (4)
        - info (119)
          - Setup-Stage (112)
            - SETUP_SYNC_CLOCK_OVERLAP (1)
            - SETUP_RESET_PROPAGATED (2)
            - SETUP_DOMAIN_INFER (23)
            - SETUP_PORT_IGNORED (18)
            - SETUP_PORT_CONSTRAINED (65)
            - SETUP_BBOXPIN_CONSTRAINED (3)
          - Sync-Stage (7)
            - SYNCCDC_CTRLPATH_FULL (5)
            - SYNCCDC_DATAPATH_FULL (2)
      - Waived
      - Acknowledged
      - NeedsInfo

Search: □ Live DestObject >= BLK_INST34/out1a/Q

	GroupCount	Msg ID	DestObject	ReasonInfoList	
1-1	20	164	BLK_INST12/test_in_flop/Q	NFF_SYNC_AND_DST_CLK_DOMAIN_DIFF •••	BLK_INST1
2-1	2	459	BLK_INST21/nff1/out1/Q	SYNC_BY_NFF •••	BLK_INST2
• 2-2		460	BLK_INST21/nff1/out1/Q	SYNC_BY_NFF •••	BLK_INST2
3-1	2	466	BLK_INST34/out1a/Q	SYNC_BY_NFF •••	...TR_ISO_
3-2	1	467	BLK_INST34/out1a/Q	UPF_INSTRUMENTED •••	iso_en
4-1	2	465	BLK_INST34/out2a/Q	SYNC_BY_NFF •••	...TR_ISO_
• 4-2		468	BLK_INST34/out2a/Q	UPF_INSTRUMENTED •••	iso_en
5	1	453	BLK_INST31/src1/Q	NFF_SYNC_AND_DST_CLK_DOMAIN_DIFF	DATA_IN39
6	1	480	BLK_INST29/src/Q	NFF_SYNC_AND_DST_CLK_DOMAIN_DIFF	DATA_IN42
7	1	479	BLK_INST23/src/Q	NFF_SYNC_AND_DST_CLK_DOMAIN_DIFF	DATA_IN35
8	1	451	BLK_INST31/src3/Q	NFF_SYNC_AND_DST_CLK_DOMAIN_DIFF	DATA_IN41

Error - SYNCCDC_CTRLPATH_PARTIAL

Partially matched control synchronization scheme found for CDC path from **iso_en** to **BLK_INST34/out1a/Q**, crossing from **_AUTO_VCLK_PORT_1_:33** to **MASTER_CLK:12** . Help

New Violation Schematic

Create a Filter Template | Waive this Item | Create a Waiver | Copy Violation Message

□ ReasonInfoList	UPF_INSTRUMENTED
SrcObject	iso_en
DestObject	BLK_INST34/out1a/Q
SrcClockInfoList	_AUTO_VCLK_PORT_1_:33
DestClockInfoList	MASTER_CLK:12
SrcObjectType	primary input

VCst <Verdi:Container:1> (on attoemt571)

VCst <nSchema:2>466:SYNCCDC_CTRLPATH_PARTIAL

Power Domain: chip_top.MASTER_CLK: Vtop   Power Signal Type:

Update with Single Click

Properties	
Basic	
Type	net
Name	MASTER_CLK
UPF	
Power Domain	Vtop
CDC	
User Clocks	
User Clock	
Clock Object	MASTER_CLK
Domain ID	12
Data Path Clocks	
Clock Info	
Clock Object	MASTER_CLK
Domain ID	12

Selected:(1) Net: chip_top.MASTER_CLK

SYNCCDC_CTRLPATH_PARTIAL: Partially matched control synchronization scheme found for CDC path

Message: Partially matched control synchronization scheme found for CDC path from iso_en to BLK_INST34/out2a/Q, crossing from _AUTO_VCLK_PORT_1_:33 to MASTER_CLK:12

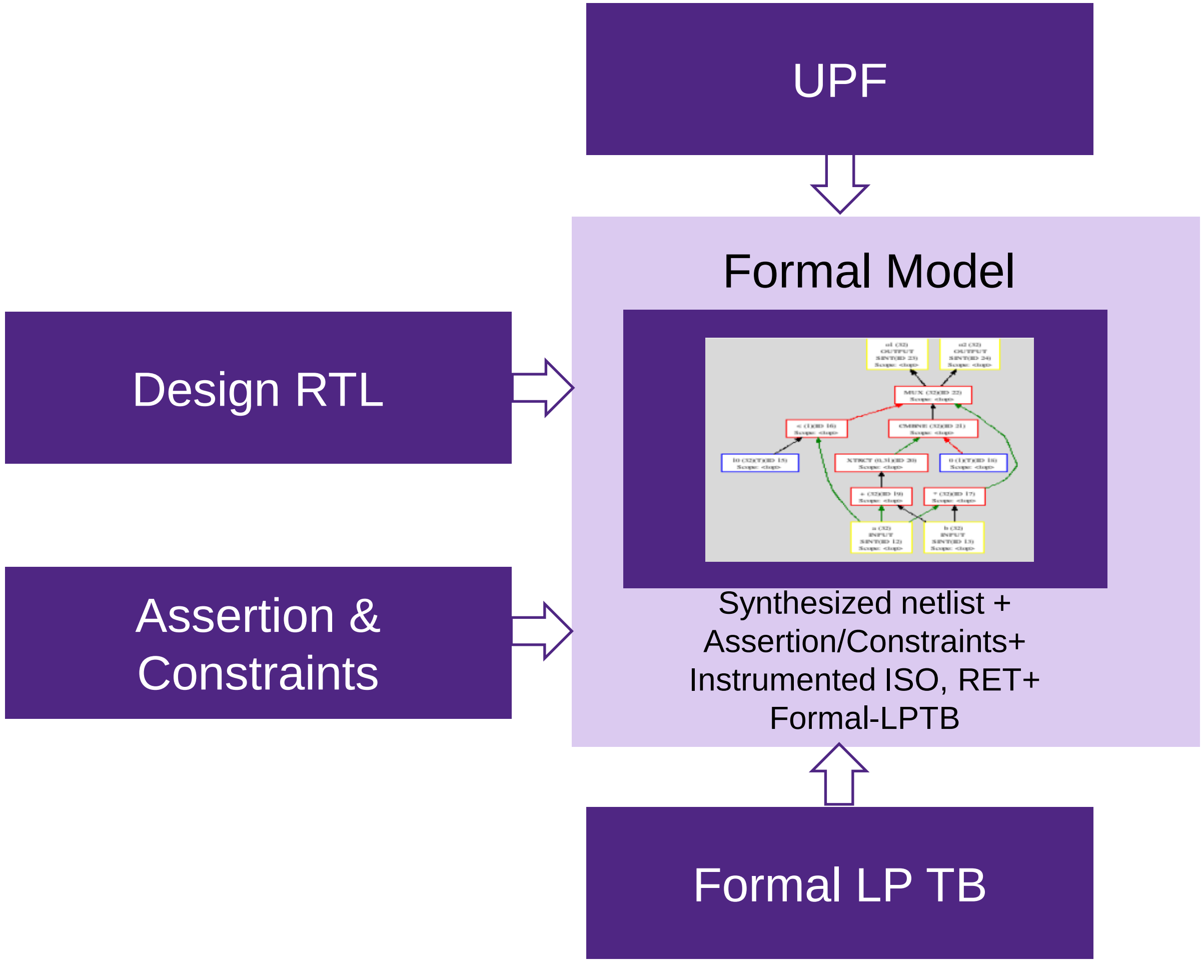
Legend	Description
🟪	Source
🟧	Crossing Path
🟫	Destination
🟩	Potential Qualifier
▨	Debug Info

Command Done

VC Formal FLP

Formal Low Power Connectivity and Property Verificaiton

VC Formal LP: Formal Analysis of Power Aware Model



UPF

Design RTL

Assertion & Constraints

Formal Model

Synthesized netlist + Assertion/Constraints+ Instrumented ISO, RET+ Formal-LPTB

Formal LP TB

Additional Specification for PMC AIP, constraining Power Network Model (PNM)

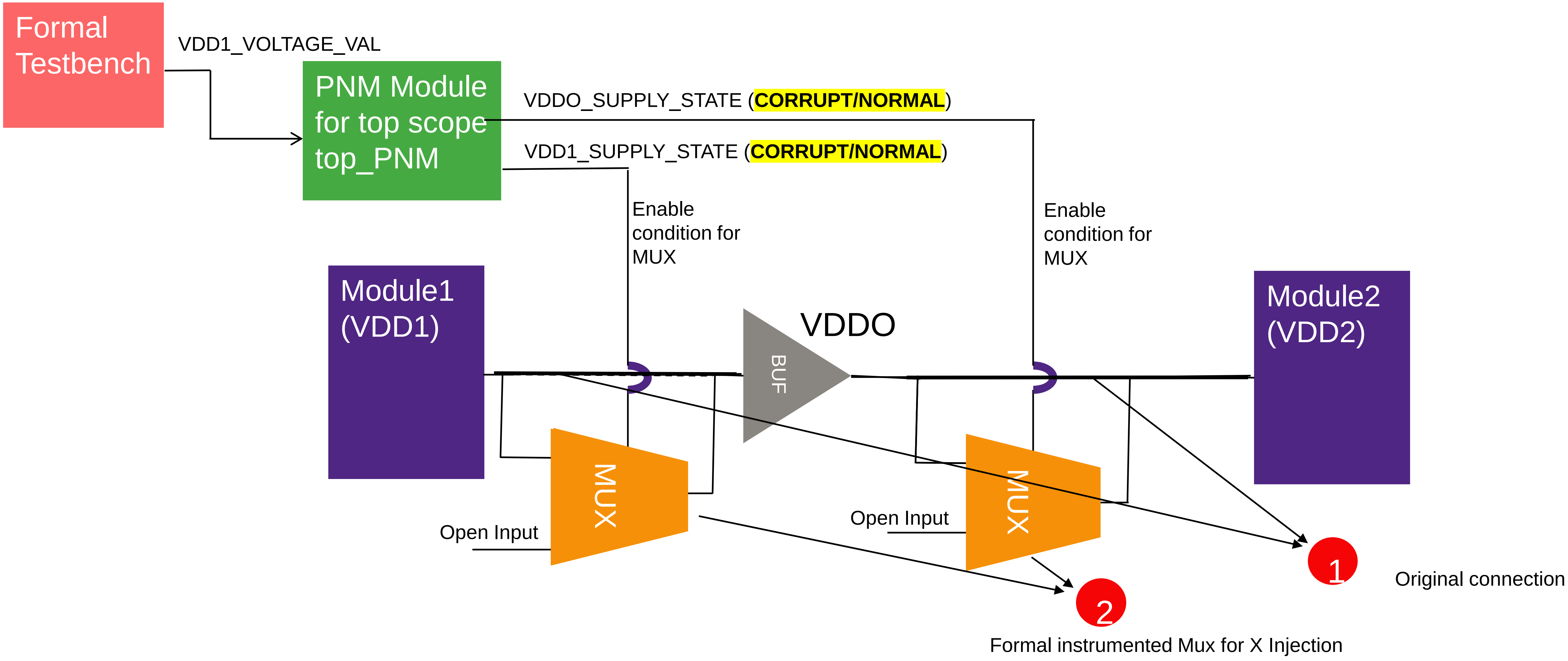
1. Power Aware Connectivity Checking (CC)
   - PG Pin : Power Network connected correctly
   - Functional: Is RTL connection bug free with UPF

2. Formal LP Property Checks
   - Checking effect of PoR (Power On Reset) sequence
   - Checking effect of isolation on output of DUT

3. Functional Verification of Power Management Controller (PMC)

4. Formal LP Query & LP Assertion Generation (bind_checker)

Power Network Model (PNM) – internally modelled



New commands for driving PNM

```
fvlp_get_free_supply_nets
```

```
{"V1", "V1_ret", "V2", "VSS", "VSS2", "Vdd", "v_u2"}
```

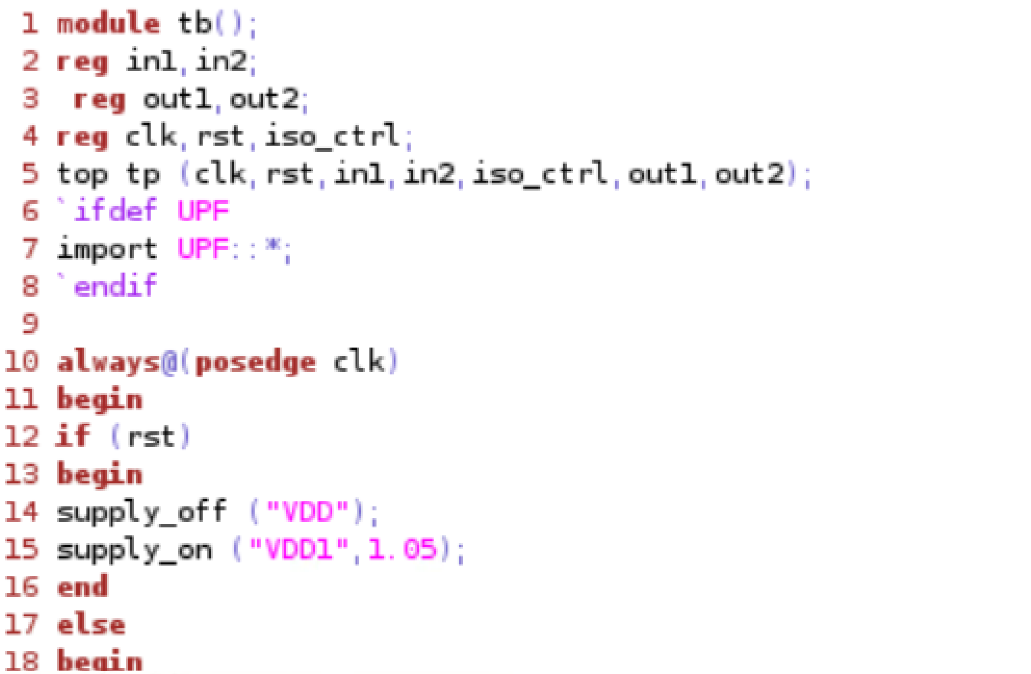
```
fvlp_set_supply_aon {"V1" "V1_ret"  "V2" "VSS" "VSS2"    "v_u2"}
```

```
[Info] FVLP_SET_INPUT_SUPPLY_VOLTAGE_I: Input supply 'V1' set to voltage 1v.
[Info] FVLP_SET_INPUT_SUPPLY_VOLTAGE_I: Input supply 'V1_ret' set to voltage 1v.
[Info] FVLP_SET_INPUT_SUPPLY_VOLTAGE_I: Input supply 'V2' set to voltage 1v.
[Info] FVLP_SET_INPUT_SUPPLY_VOLTAGE_I: Input supply 'VSS' set to voltage 1v.
[Info] FVLP_SET_INPUT_SUPPLY_VOLTAGE_I: Input supply 'VSS2' set to voltage 1v.
[Info] FVLP_SET_INPUT_SUPPLY_VOLTAGE_I: Input supply 'v_u2' set to voltage 1v.
```

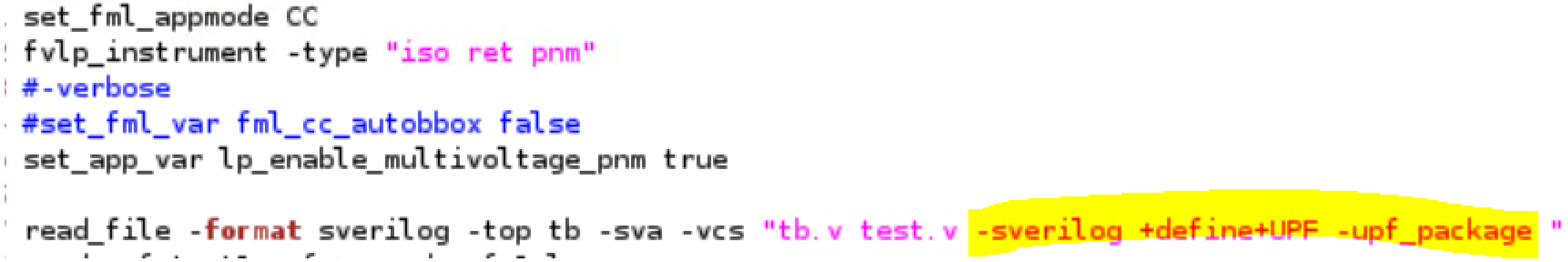
```
fvlp_set_supply_onoff_net -netname ctrl  { "Vdd"  }
```

```
[Info] FVLP_SET_INPUT_ONOFF_I: Input supply 'Vdd' connected to be driven on/off by RTL
signal 'ctrl'.
```

UPF supply_on and supply_off commands

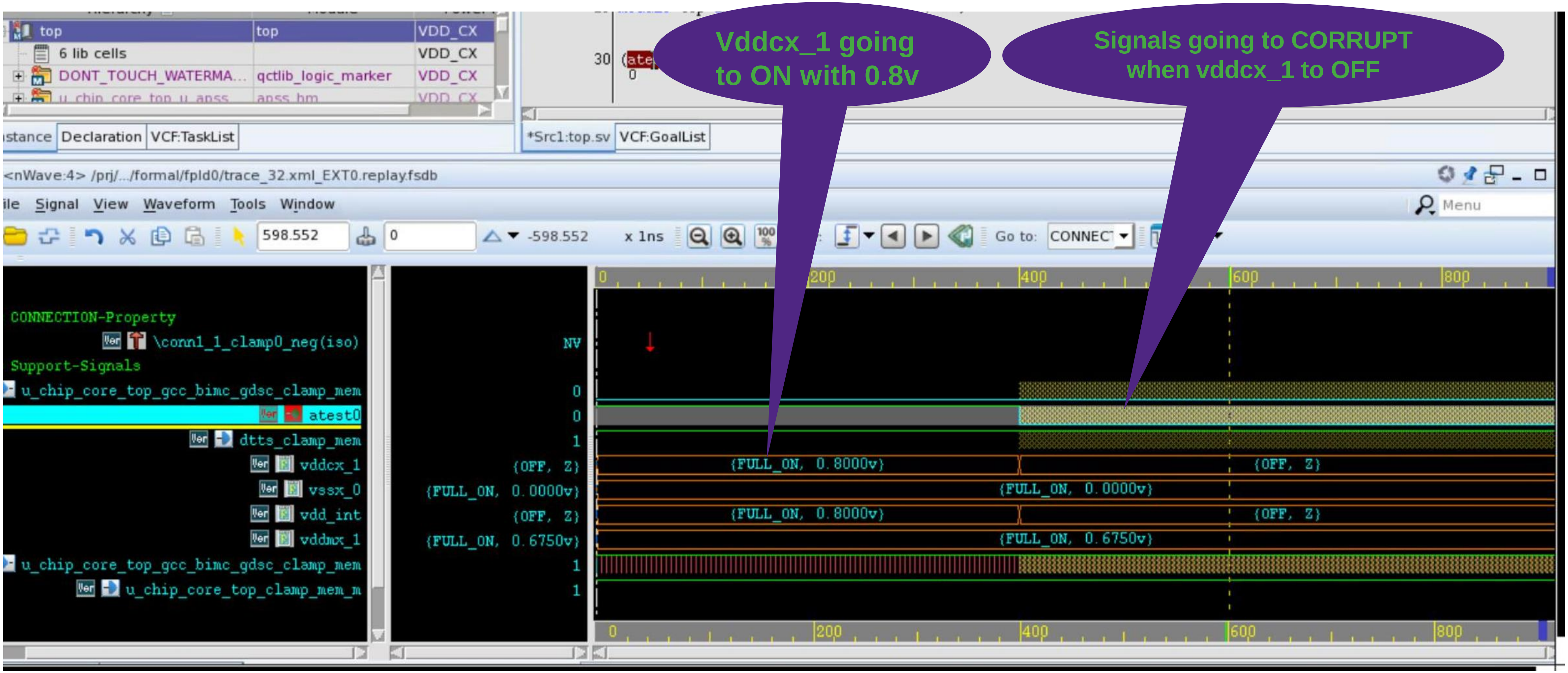


```
module tb();
reg in1,in2;
 reg out1,out2;
reg clk,rst,iso_ctrl;
top tp (clk,rst,in1,in2,iso_ctrl,out1,out2);
`ifdef UPF
import UPF::*;
`endif

always@(posedge clk)
begin
if (rst)
begin
supply_off ("VDD");
supply_on ("VDD1",1.05);
end
else
begin
```



```
set_fml_appmode CC
fvlp_instrument -type "iso ret pnm"
#-verbose
#set_fml_var fml_cc_autobbox false
set_app_var lp_enable_multivoltage_pnm true

read_file -format sverilog -top tb -sva -vcs "tb.v test.v -sverilog +define+UPF -upf_package "
```

Multi voltage Waveforms For Debug….

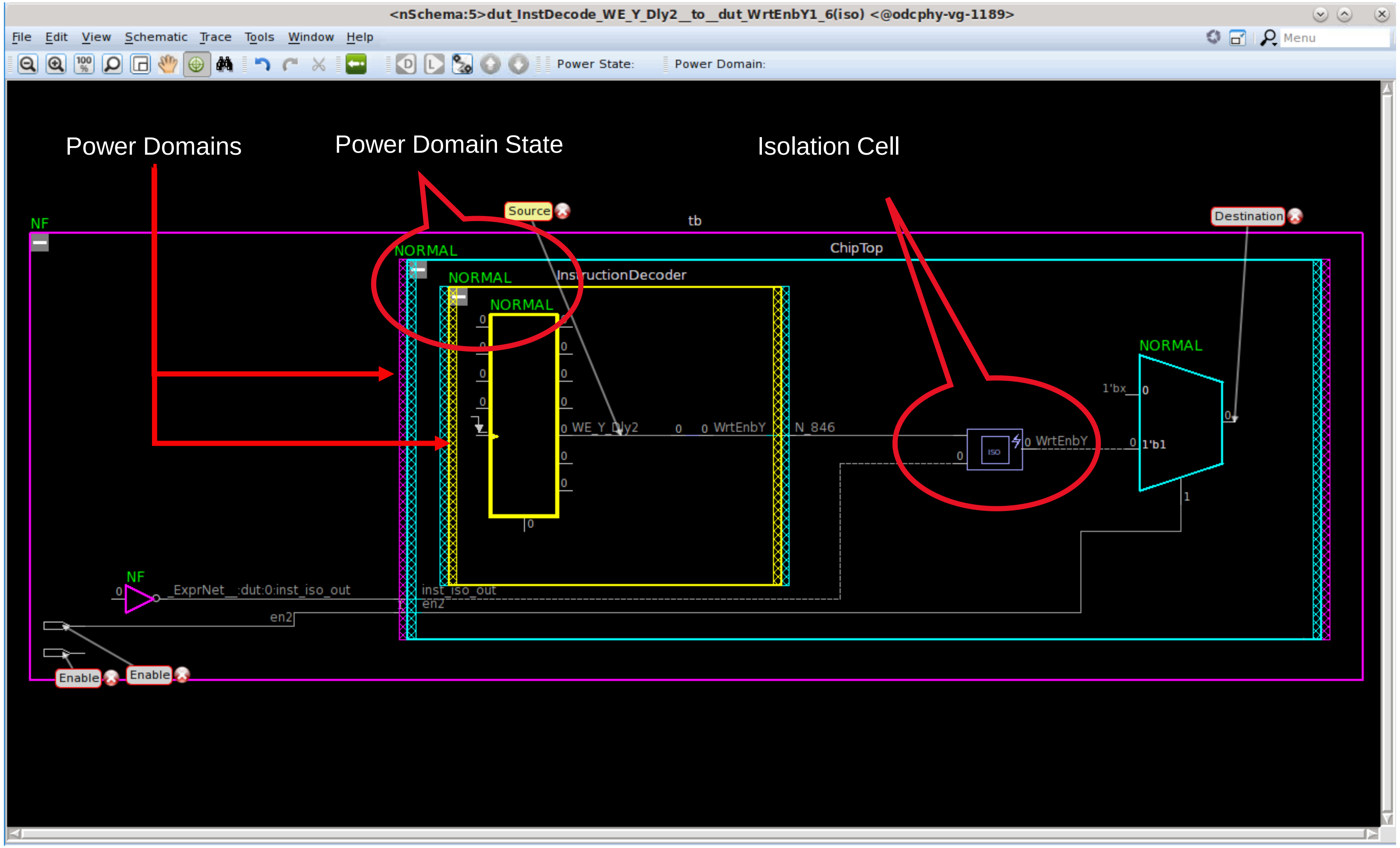


```
CONNECTION-Property
    \conn1_1_clamp0_neg(iso)                     NV
Support-Signals
  u_chip_core_top_gcc_bimc_gdsc_clamp_mem         0
                                   atest0         0
                            dtts_clamp_mem        1
                                  vddcx_1   {OFF, Z}        {FULL_ON, 0.8000v}                {OFF, Z}
                                   vssx_0   {FULL_ON, 0.0000v}      {FULL_ON, 0.0000v}
                                  vdd_int   {OFF, Z}        {FULL_ON, 0.8000v}                {OFF, Z}
                                  vddmx_1   {FULL_ON, 0.6750v}      {FULL_ON, 0.6750v}
  u_chip_core_top_gcc_bimc_gdsc_clamp_mem         1
                 u_chip_core_top_clamp_mem_m      1
```

Vddcx_1 going to ON with 0.8v

Signals going to CORRUPT when vddcx_1 to OFF

Verdi: Schematic



Power Domains

Power Domain State

Isolation Cell

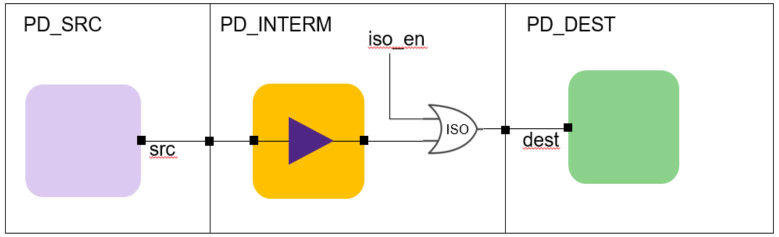
Agenda:

- Formal Verification of Low Power Designs
- **Low Power Connectivity Checking**
- Low Power Property Verification

New Categories of Properties of LP Connectivity Checking -lpa_type

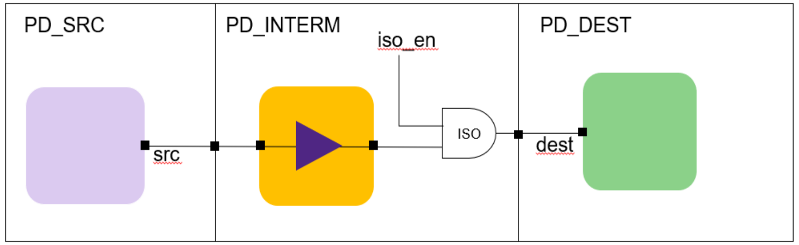
- **LPA BASIC**
  - **SD:** Source power domain (PD_SRC) is NORMAL and **DD:** Destination power domain (PD_DEST) is NORMAL

```
En && SD && DD |-> (dest == src)
```

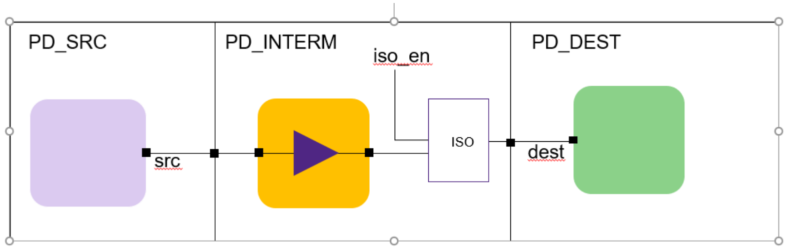


- **LPA_CLAMP1/CLAMP0**
  - Connectivity check for paths having OR-type ISO cell (CLAMP1) or AND-type ISO cell (CLAMP0)

```
En && SD && DD && (iso_en != ISO_SENSE) |-> (src == dest) … (connectivity component)
En && SD && DD && (iso_en == ISO_SENSE) |-> (dest == 'b1) … (clamping component)
```



- **LPA_LATCH**
  - Same as LPA_CLAMP1/0 but having LATCH-type iso cell.
  - Consequent of the clamping component becomes (not $fell(dest) and not $rose(dest)).



- **LPA_SUPPLY**
  - Power/ground (PG) pin connectivity checking.
  - Source & Destination should be power supply objects from the UPF.
- **LPA_CLAMP1/CLAMP0/LATCH_EN**
  - Connectivity of enable source to enable pin of instrumented isolation cell

```
En && SD && DD && (src == ISO_SENSE) |-> (dest == 'b1) (clamped)
```

FV LP-CC: Formal Verification Low Power Connectivity Check

- Extensions to current Connectivity Check App: use existing load_cc and add_cc

- add_cc -src ”...” -dest ”...”

- add_cc **-lpa_type (NON_LPA | LPA_BASIC | LPA_LATCH | LPA_CLAMP1 | LPA_CLAMP0 | LPA_SUPPLY)** -src ”...” -dest ”...” [*]

[*] After read_upf

Agenda:

- Formal Verification of Low Power Designs
- Low Power Connectivity Checking
- **Low Power Property Verification**

Examples: LP Property Verification

- **Checking effect of isolation on output signal of DUT**

```
property p_isolation_check1;
disable iff(rst )
  @(posedge clk)
    ##1 (gds_enf ==0) && (gds_enr==0) &&(clamp_enable==1) |=> (qnm_rot_Rsp_U_Urgency[0] ==1)  ;
Endproperty

isolation_check1 : assert property (p_isolation_check1);
```

- **Checking effect of reset sequence on power up**

```
property p_qreqn_transition_low_to_high_when_qacceptn_and_qdeny_low2;
     disable iff(rst|| ((!gds_enf)&&(!gds_enr)) ) @(posedge clk)
           ($rose(qreqn) && ($past(rst)===1'b0) &&(count==1)) |->
           ((((qacceptn) === 1'b0) && ((qdeny) === 1'b0)) || (((qacceptn) === 1'b1) && ((qdeny) === 1'b1)));
endproperty

qreqn_transition_low_to_high_when_qacceptn_and_qdeny_low2 : assert property
      (p_qreqn_transition_low_to_high_when_qacceptn_and_qdeny_low2));
```

- **Check if sequential logic is uninitialized on wakeup, propagating through:**

```
property p_tx_isunknown_check_0;
disable iff(rst )
  @(posedge clk)
    ##1 (gds_enf ==0) |=> not ($isunknown(qnm_rot_Rsp_U_Tx[0] ) ) ;
endproperty

tx_isunk_check_0 : assert property (p_tx_isunknown_check_0);
```

Added Power Aware FPV

- Features added for PA FPV
  - Separating “bound” formal testbench from UPF synthesis
  - Support for reset sequence on power-up as a part of formal analysis
  - Support for $isunknown() during shut down and power up
- What is the Purpose of Power Aware FPV (PA FPV)
  - Any formal testbench can be migrated to power aware testbench
  - Properties will pass or fail if there are LP issues in the design
  - Will help catch POR issues in the design.

Formal Testbench

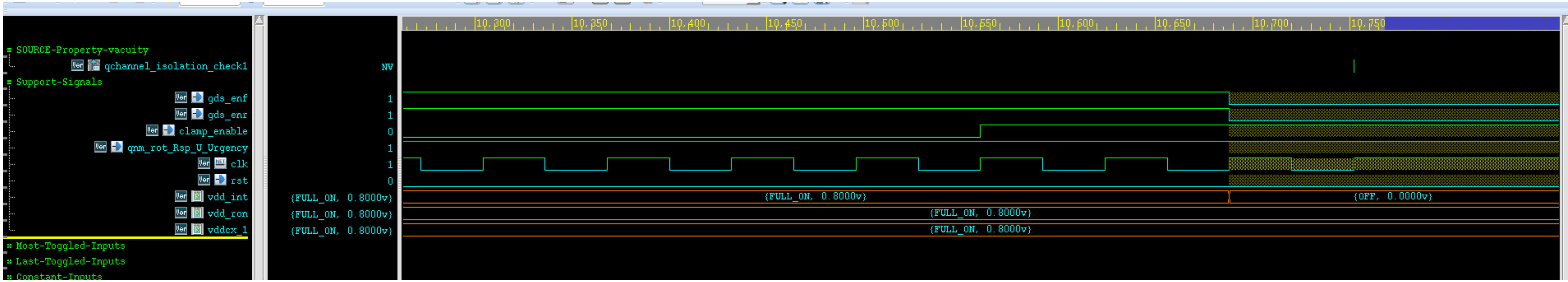
- No changes need to be made to the formal testbench
- VC Formal will automatically separate any “bound” instance from UPF synthesis
  - Testbench will not be a part of the power network synthesis
  - Connection from testbench to logic in the DUT will not influence isolation analysis OR annotation of isolation cells
- No additional input required from user to do this

```
module assert_iso (clk, ctrl, isoen, output_port, rst);
   input clk, ctrl, isoen, output_port, rst;

   property p_isolation_check;
      disable iff(rst )
           @(posedge clk)
       ##1 (ctrl) && (isoen) |-> (output_port==1)  ;
   endproperty

   a_isolation_check: assert property (p_isolation_check);

endmodule // assert_iso

bind top assert_iso  inst_assert_iso  (.rst (reset), .clk (clk), .ctrl(psw_ctrl), .isoen(iso_ctrl_bot),  .output_port(out1[0]) );
```

Example:

Checking effect of isolation on output signal of DUT

```
property p_isolation_check1;
disable iff(rst )
  @(posedge clk)
    ##1 (gds_enf ==0) && (gds_enr==0) &&(clamp_enable==1) |=>
(qnm_rot_Rsp_U_Urgency[0] ==1)  ;
endproperty

isolation_check1 : assert property (p_isolation_check1);
```



Retention Support in LP Property Verification

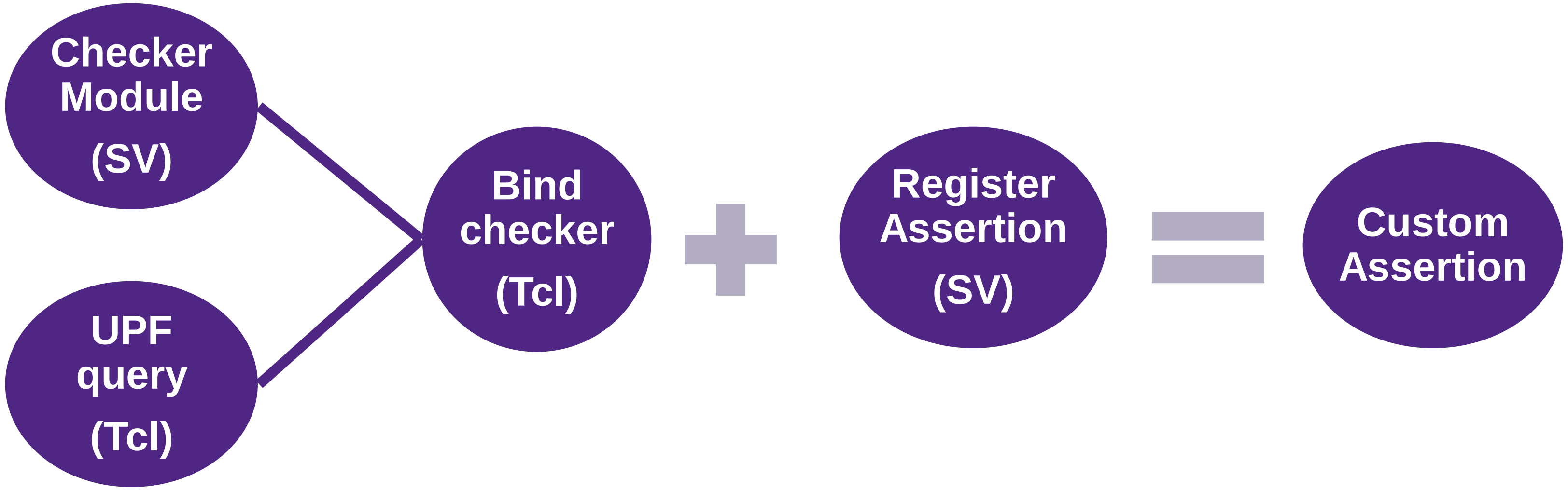
- Enables Retention instrumentation
  - *latches will be ignored by default like DC*
- Supports 2-pin/1-pin/0-pin retention styles

- Supports retention reporting just like NLP
  - instrumentation_summary.rpt
  - retention_policy_assoc.rpt

- Supports CEX nWave and nSchematic debug
  - tracing of retention supply-nets from nSchematic to UPF source code pointing to CSN
  - trace Load / Drivers for retention supply nets in nWave

Support for Writing Properties on Design + UPF

1. **Coding SV checker module**
2. **Populating the necessary ports and parameters (query_*)**
3. **Bind the checker module to the target (bind_checker)**
4. **Register the assertion (APIs)**



Support for Writing Properties on Design + UPF

bind_checker

- Similar to sva bind
- Binds the checker module to the UPF/design objects
- Support for passing parameters

```
bind_checker  state_change_msg \
          -module pd_state_change \
          -elements [ list @$PD ] \
          -parameters $pd_params_list \
          -ports $pd_ports_list
```

Support for Writing Properties on Design + UPF

- Infrastructure (UPF 2.0 extensions)
- Query commands
  - Query the power intent
  - 8 query commands supported

query_power_domain
query_supply_net
query_power_switch
query_isolation
query_retention
query_retention_control
query_pst
query_pst_state

- Example

```
foreach PD [ query_power_domain *] {
                set PD_NAME [list PD_NAME $PD]
                array set pd_detail [query_power_domain $PD -detailed]
                set POWER [ list POWER $pd_detail(primary_power_net) ]
                set GROUND [ list GROUND $pd_detail(primary_ground_net) ]
                set SIMSTATE [ list SIMSTATE $pd_detail(sim_state) ]
```

2024
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
UNITED STATES
SAN JOSE, CA, USA
MARCH 4-7, 2024

Isolation Consistency Checker

Satya Ayyagari & Yoong Yaw Yong

intel®

accellera
SYSTEMS INITIATIVE

The Isolation Strategy

Element based strategy

→ Isolation is inserted according to the element list
→ Available since UPF 1.0
→ Element list needs to be updated as per design change.

```
set_isolation iso_out
-domain pd_gated
-elements {
    unit1/a
    unit1/b
    .
    .
    .
}
```

Path based strategy

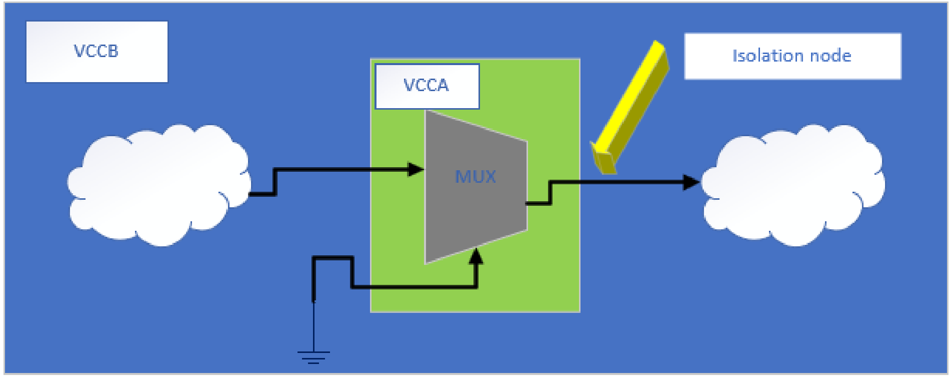
Getting Popular

→ Isolation is inserted if a node fulfills the source/sink requirement
→ Available since UPF 2.0
→ No need to maintain element list as it's generated dynamically.

```
set_isolation iso_out
-domain pd_gated
-source ss_gated
-sink ss_always_on
```

The Problem Statement

- With source/sink isolation strategy, it relies on tools to infer isolation elements.
- An isolation node might meet the requirement in one tool but not in another tool.
- This might be due to following reasons (but not limited to):
  - Logic being optimized differently between tools
  - Different collaterals used (bmod vs timing lib).



If the mux is optimized away

- Source is VCCB, no isolation is inserted

If the mux is not optimized away

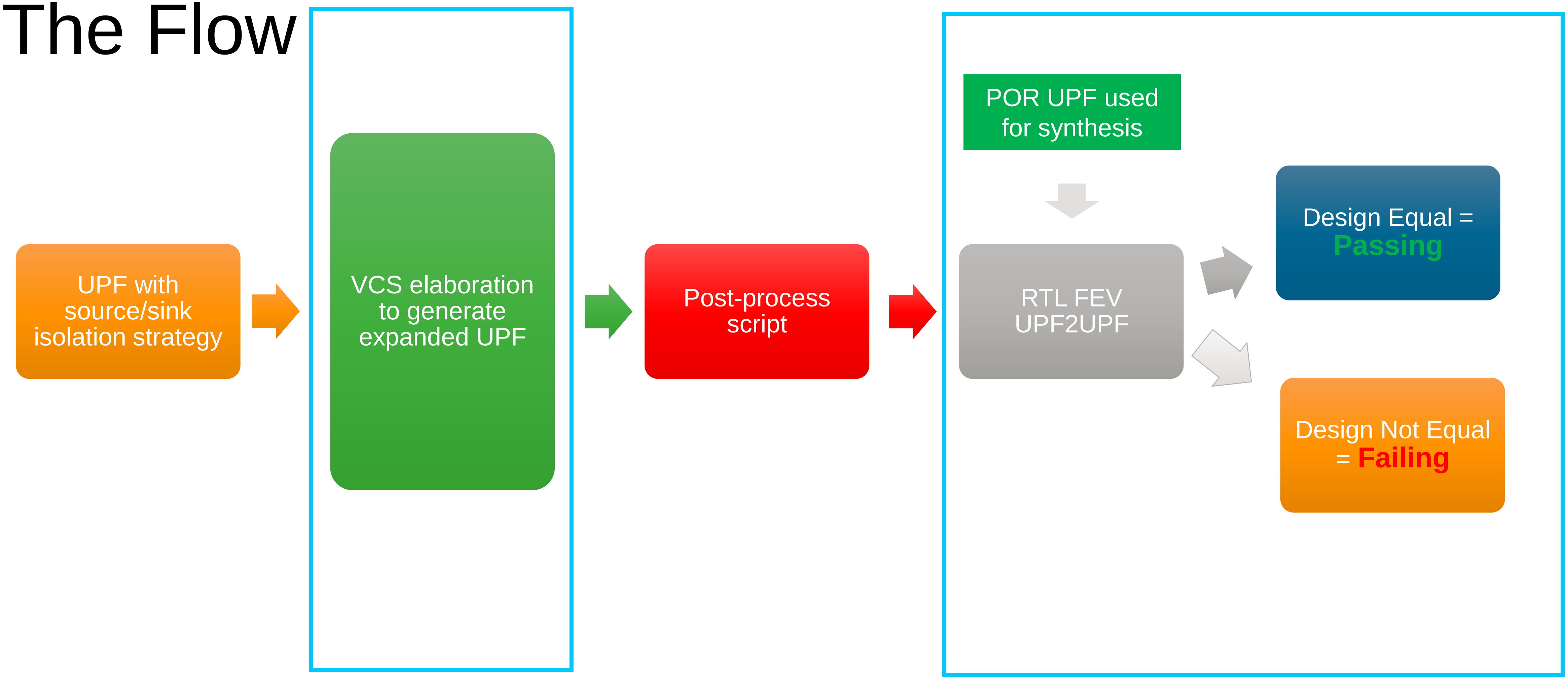
- Source is VCCA, isolation is inserted

- For non-simulation, Low Power FEV between RTL and netlist ensures isolation insertion is consistent.
- For simulation, there's no check to guarantee simulation is consistent with implementation especially when flows different sets of collateral

The Solution (Isolation Consistency Flow)

- Multi tool flow that performs consistency check on isolation insertion between RTL simulation and netlist.
- Ensure simulation and implementation match

The Flow



UPF with source/sink isolation strategy

VCS elaboration to generate expanded UPF

Post-process script

POR UPF used for synthesis

RTL FEV UPF2UPF

Design Equal = **Passing**

Design Not Equal = **Failing**

SYNOPSYS®
Silicon to Software™

Thank You