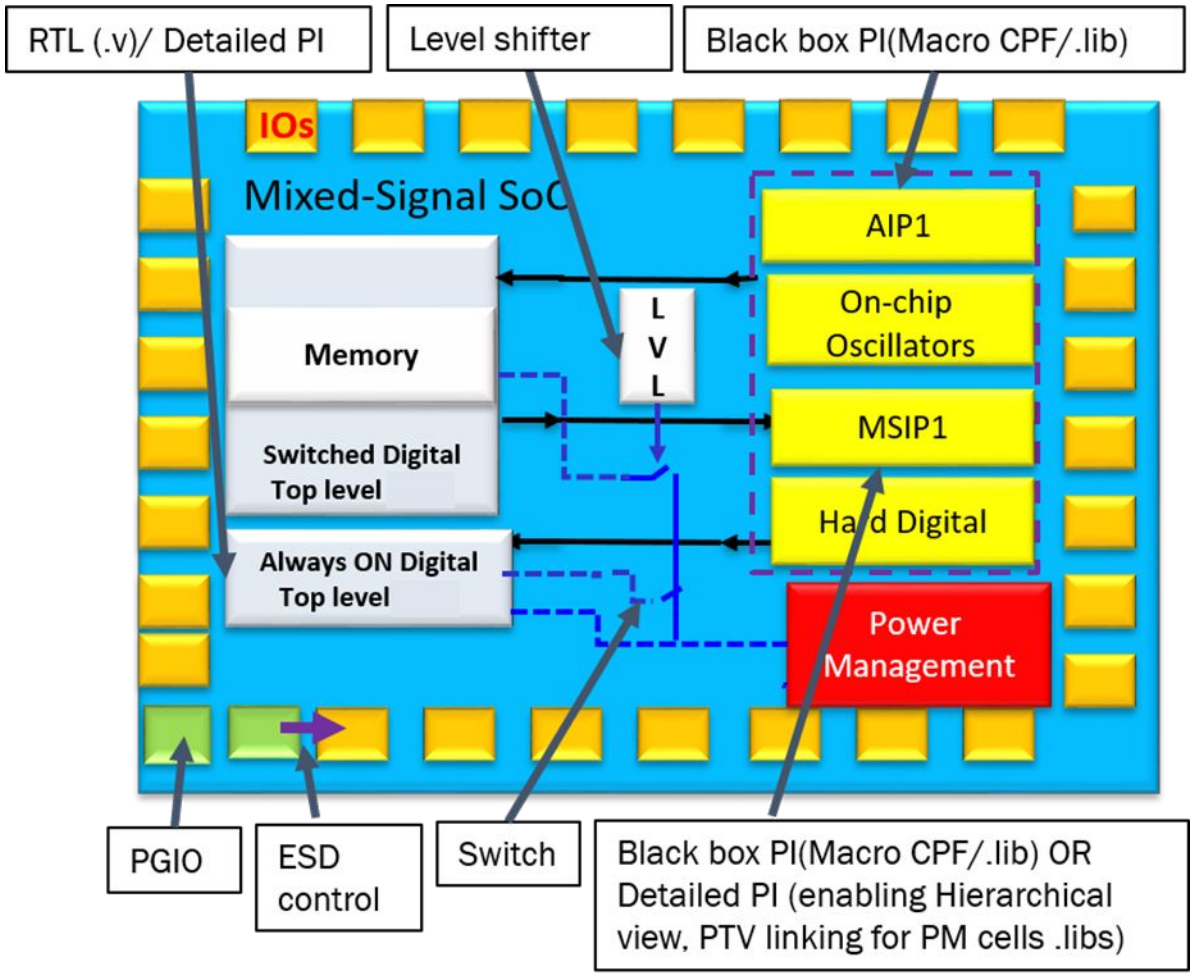
Power Intent Automation for Ultra Low Power Mixed-Signal SoC

Sai Sudarsan Chitrapu, Ruchi Shankar, Ankitha M, Penchal Kumar Gajula, Tejas Salunkhe, Lakshmanan Balasubramanian, Gaurav Kumar Varshney

Texas Instruments

Context

- Newly evolving embedded processing SoCs developed for applications like automotive and IoT are heavily power managed with complex functionality
- Power management (PM) information like power domain (PD) definition, level shifting, isolation, state retention and power switches is captured in a file called power intent (PI)
  - PI can be specified in Common Power Format (CPF) or Unified Power Format (UPF)
  - PI from here on represents PI written in CPF
- Tens of digital IPs, analog and mixed-signal IPs, SoC top-level and verification test bench require PI
- High integration complexity due to multiple voltage, power, reset* and clock* domains several hundred power modes



*Not in scope of this paper

Prior Art

- SoC PI uses black box PI for MSIP[1], adequate for driving backend implementation
- Detailed / Hierarchical PI view is required for enabling early DV[1], Flat STA, faster debug of PI violation(s)
- Absence of PGIOs and PGIO driven ESD control connections in SoC PI
  - Eventually defined downstream (Physical Design)
  - Results in late findings of incorrect domain mappings causing unprotected IOs under ESD events[2,3]
- SoC development involves few formal compile milestones and tens of integration iterations that require continuous PI modification
  - Takes about 2 to 3 person months
  - Significant manual effort to build SoC PI
  - Multiple iterations as IPs and SoC RTL mature
  - Error prone manual interventions impact design execution cycle for any late findings

SoC PI Complexity		
SL.NO	CPF Construct	Count
1	create_power_domain	100
2	update_power_domain	100
3	-shutoff_condition (for all create_power_domain)	70
4	create_power_nets	50
5	create_nominal_condition	50
6	set_instance	20
7	create_power_modes	20
8	-domain_conditons (for all power_mode)	2000
9	create_level_shifter_rule	100
10	update_level_shifter_rule	100
11	create_global_connection	350
12	Manual coded lines for SoC PI	3000

Source:

[1] Aswani Kumar Golla, et al, "Hierarchical Power Intent Driven Efficient Power Integration Methodology for Ultra Low Power Mixed-Signal SoC,", DAC 2019

[2] Lakshmanan Balasubramanian, et al, "Advances to CPF Based Low Power Mixed Signal Integration and Verification," Cadence Live India 2021

[3] Vijay Kumar Sankaran,et al, "Modern Recipes for Brewing the Inevitable Methodology for Today's ICs: Low-Power Mixed-Signal Design Verification", DAC 2019

Proposal

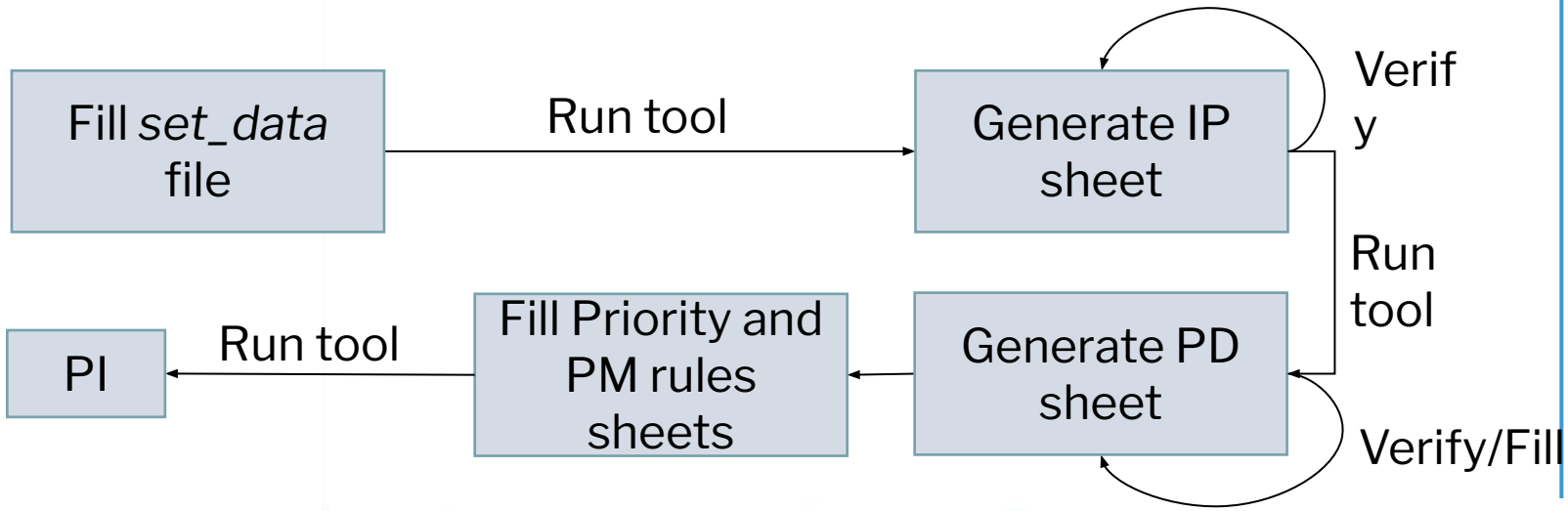
- **Automation:** Time consuming and manually intensive process of PI development warrants an automation to generate correct by construct PI with RTL as key input
- **Flavors of PI**: Supports multiple flavors – detailed and macro PI views generated automatically with same interface
- **Hierarchical generation of PI**: PI generated is hierarchical in nature – generates PI for intermediate hierarchies which are used to generate SoC toplevel PI
- **Simple and efficient user interface**: Easy and intuitive interface for specifying PI with minimum user inputs
- **Adaptation:** PI automatically adapts to RTL incremental updates without manual intervention, reducing manual effort and time consumption

Tool Inputs and Setup

set_data file

Specifies:

- Folder and file paths – input excel, output folder, RTL files
- Library cells used in the design
- Virtual ports and port mapping information



Excel file contents

IP sheet
IPs, logic cells, PM cells from RTL and their PI

PD sheet
PDs in design and their information

Priority sheet
PD priority - used in generating power modes

PM rules sheet
PM cell rules (isolation, retention, level-shifter and switch rules)

Tool Features

- Additional user commands:
  - *'-run_clp'* – runs static checks with Cadence® Conformal Low Power (CLP) on the generated hierarchical PI
  - *'-reset_manual_changes'* – restores Excel file by removing additional manual changes by user apart from auto populated data
- Handling legacy IP PI:
  - Tool automatically set back the top level PI version to the required version even if the IP PI were written in an older version
- Automated file extraction:
  - Automatically searches and extracts PI files, both from workspace

Handling PM Cells

Switches

Generate PD names

Short PDs created by Functional and DFT switches (as per architectural requirement)

Extract switch instances and their shutoff conditions from RTL

Get PM cell name from .v, pin names from .lib file

User input – base PD (source PD)

Create Switch rule, global connection and PDs

Source PD

DFT

Functional

Derived PD

Tool generates two PDs for DFT and Functional switch, these will be shorted – by user

Level shifters

Extract level shifter names from RTL

Get pin names from .v file, and .lib file

User input – power net ,ground net association, from and to PD specification

Level shifter rule and global connection creation

Isolation and retention cells

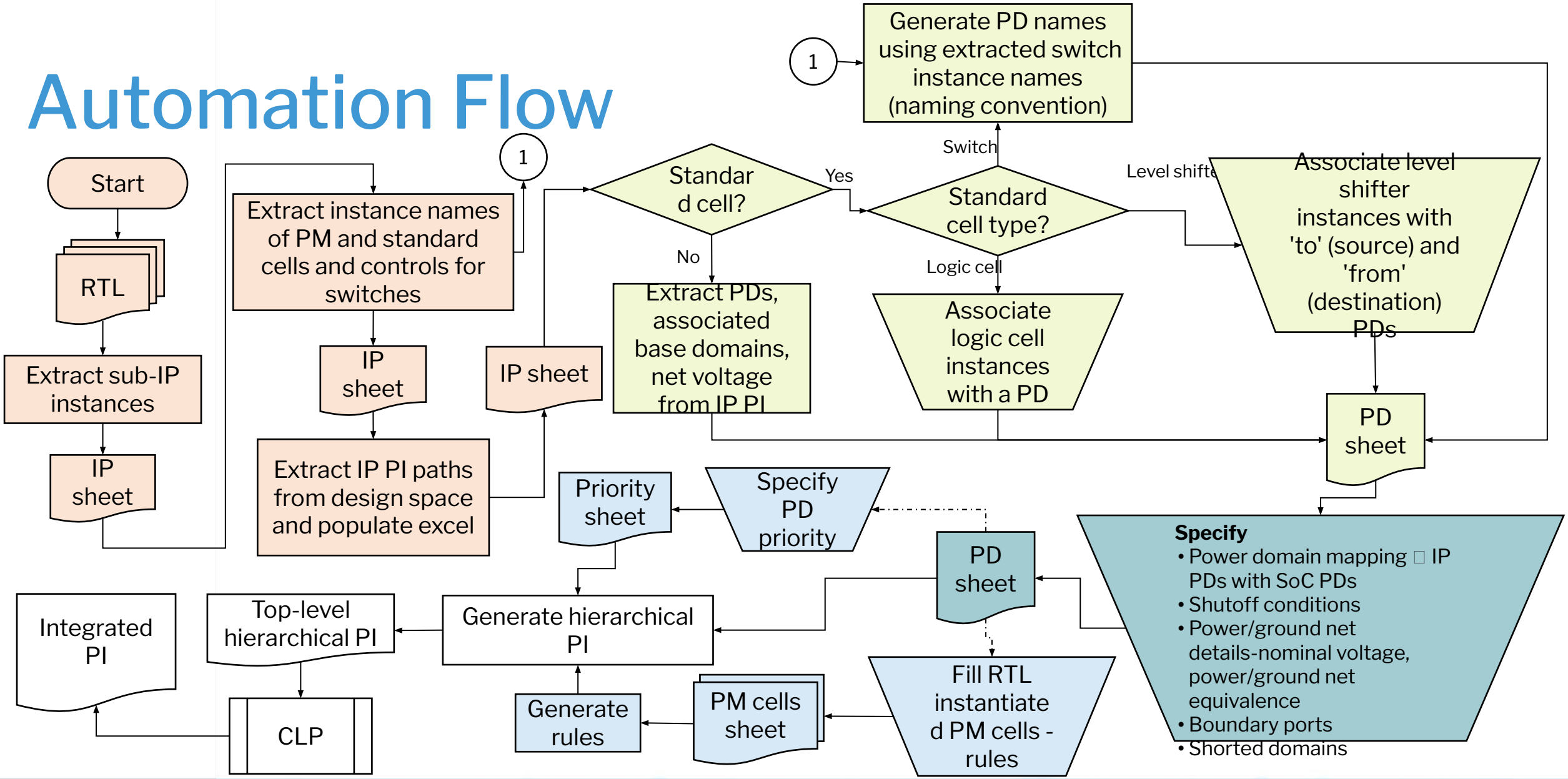
User input – define isolation/retention rule in PM rule sheet

Tool generates rule

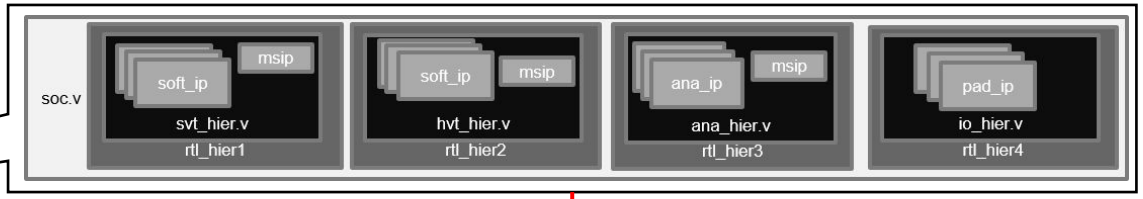
User specifies instances to which isolation/retention rule is applied

Isolation/retention rule creation

Automation Flow



Excel View



IP Sheet:

rtl_wrapper	IP group hierarchy	IP instance name	IP module	P	(CPF) file path	PI type
rtl_hier1	ana_hier	ip_1	IP_1	ip	_1_macro_cpf	macro
rtl_hier3	svt_hier	ip_2	IP_2	ip	_2_design_cpf	design
rtl_hier3	svt_hier	ip_3	IP_3	ip	_3_design_cpf	design
rtl_hier3	svt_hier	ip_4	IP_4	ip	_4_macro_cpf	macro
rtl_hier3	svt_hier	ip_5	IP_5	ip	_5_macro_cpf	macro
rtl_hier3	svt_hier	ip_6	IP_6	ip	_6_macro_cpf	macro

Tool generates PI of each of these sub hierarchies which are instantiated in top-level CPF □ hierarchical CPF

Start

Set_data file

RTL

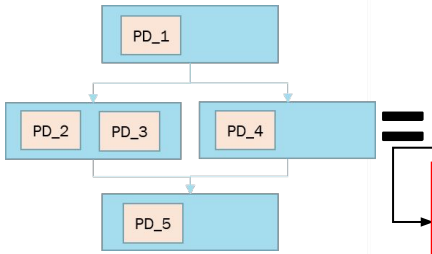
Automatically picks IP PI from design area

PD Sheet:

rtl wrapper	IP group hierarchy	IP instance n	ip power dom	oC level power domain mapp	/standard c	fault P	flag	se doma	toff conditi	orted_doma	power net name	internal	er net voltage	nominal cond	on	eq	valent power nets	ground net	equivalent ground nets	boundary ports
rtl_hier1	ana_hier	ip_1	ip_pd_1	D_1			1				V D_1		v1	n1				gn1	gn2	bp1,bp2
rtl_hier1	ana_hier	ip_1	ip_pd_2	D_2				P _1			V D_2		v2	n2		VD	_1	gn1		bp3
rtl_hier2	svt_hier	ip_2	ip_pd_3	D_3			1				V D_3		v3	n3				gn2		
rtl_hier3	svt_hier	ip_3	ip_pd_4	D_4							V D_4		v4	n4				gn1		
rtl_hier3	svt_hier	ip_4		D_5	s tch			P _1	c 1, con2		1 V D_5		1 v5	n5				gn1		
rtl_hier3	svt_hier	ip_5		D_5	s tch			P _2	c 1, con2		1 V D_5		1 v6	n6				gn1		
rtl_hier3	svt_hier	ip_6		D_2	l el_shifter			P _1												

Priority Excel:

Priority		
1	PD_1	
2	PD_2,PD_3	PD_4
3	PD_5	



Hierarchical top-level CPF generation

PM rules

Fill these

PM rule name	rule_type	power_domain	secondary domain	include instances	exclude instances	target	type	conditions	cells
iso_rule1	Isolation	{from domain, to domain}		[pins]		from	high		
iso_rule2	Isolation	{from domain, to domain}		[pins]		to	low		
iso_rule3	Isolation	{from domain, to domain}		[pins]		drop_list	drop_list		
iso_rule4	Isolation	{from domain, to domain}		[pins]		drop_list	hold		
							low		
							high		
							hold		
ret_rule1	Retention	power_domain		[instances]		flop	save_edge		
ret_rule2	Retention	power_domain		[instances]		latch	restore_edge		
ret_rule3	Retention	power_domain		[instances]		both	restore_edge-save_edge		
ret_rule4	Retention	power_domain		[instances]		drop_list	drop_list		
							restore_edge		
							save_edge		
							restore_edge-save_edge		
							restore_level-save_level		

Results

S. No.	Aspect	Conventional flow	Automated flow
1	Time required for **initial** SoC PI bring up	2-3 weeks	2-3 days, (86% reduction in manual effort and cycle time)
2	Iterations	PI adaptation per SoC RTL incremental updates is time consuming	PI automatically adapts to SoC RTL incremental updates
3	SoC PI views	Manual generation of different SoC PI flavors (with Black box PI for MSIP)	Automatic generation of: • SoC PI with black box PI for MSIP • SoC PI with detailed PI of MSIP for early DV, flat STA, faster debug
4	Reliability	Error prone due to manual modifications	More reliable since tool extracts necessary information from RTL and IP PI
5	Prerequisite skills	• Knowledge of power architecture of SoC and IPs • Knowledge of PI language (CPF/UPF)	• User only needs to input data in predefined fields in Excel • Agnostic to PI language & syntax
6	Portability	Manually porting from CPF to UPF is very tedious and error prone	Enhancement to support UPF underway
7	Schedule	• Significant effort to generate PI and manually align PI with design changes • Repetitive iterations and late findings can affect time to market	• Generates reliable and exhaustive PI with minimum user intervention, which easily adapts to changing design intent • Improves time to market

Conclusions

- *Power Intent (PI) is a critical design collateral for power managed SoC development*
- *Significant manual effort in developing, validating and delivering PI*
- *Conceptualised and developed an automated PI generation tool with intuitive user interface*
  - *Freedom from language specific syntactical information*
  - *Design competency disassociated from knowledge of specific PI or HDL language skill*
  - *Design information captured automatically from RTL*
  - *Associated PI aspects captured manually in an Excel utility*
  - *Inbuilt validation or constraint input range to avoid manual errors*
- *Saves at least 86% of effort in PI capture*
- *Significant quality improvement*

References

1. *Aswani Kumar Golla, et al, "Hierarchical Power Intent Driven Efficient Power Integration Methodology for Ultra Low Power Mixed-Signal SoC,", DAC 2019*
2. *Lakshmanan Balasubramanian, et al, "Advances to CPF Based Low Power Mixed Signal Integration and Verification," Cadence Live India 2021*
3. *Vijay Kumar Sankaran, et al, "Modern Recipes for Brewing the Inevitable Methodology for Today's ICs: Low-Power Mixed-Signal Design Verification", DAC 2019*