

# A Power-Centric Timing Optimization Flow for a Quad-Core ARM Cortex-A7 Processor

Bernard Ortiz de Montellano

Product Manager

Processor Division

**ARM**<sup>®</sup>

Dale Lomelino

Staff Applications Consultant

**SYNOPSYS**<sup>®</sup>

March 26, 2013

# Agenda



ARM-Synopsys Project Introduction

Bernard Ortiz de  
Montellano

---

Power-Centric Timing Optimization Flow  
for a Quad-Core ARM® Cortex®-A7  
Processor

Dale Lomelino

# ARM-Synopsys Project Introduction

The ARM Cortex-A7 MPCore™ Processor

Introducing big.LITTLE™ Processing

Implementation Optimization for big.LITTLE

ARM-Synopsys Collaboration

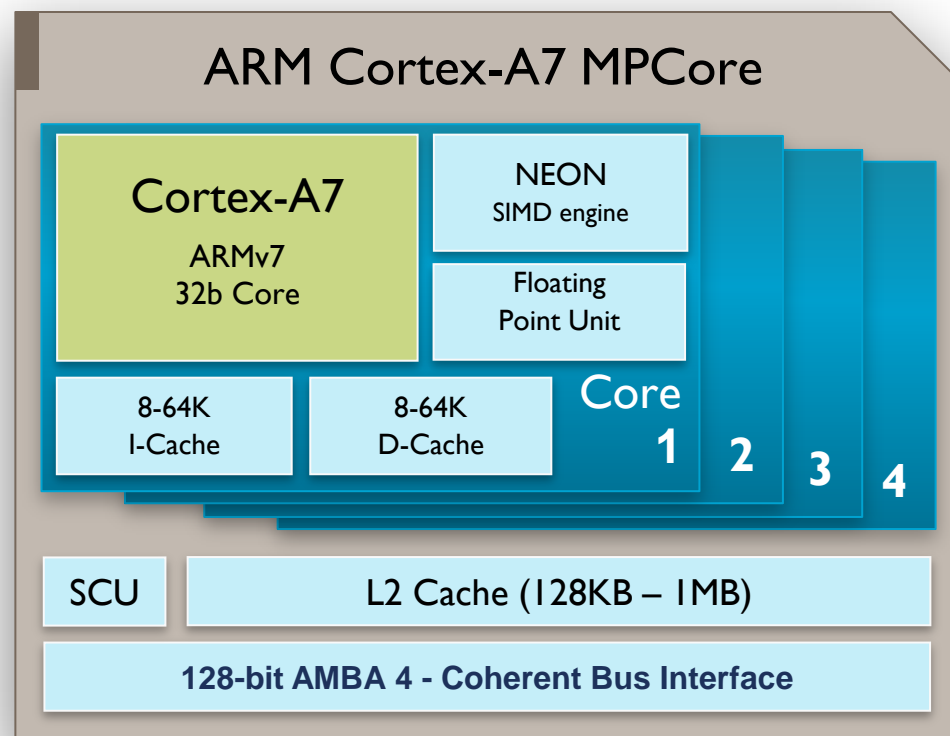
# Introducing the Cortex-A7 MPCore Processor

The most energy-efficient v7A (32-bit) application processor

- Power efficient microarchitecture
  - In-order 8-stage, partial dual-issue
  - Integrated NEON™ and FPU, L2, improved memory system
- Architecture aligned with Cortex-A15 MPCore
  - Hardware enhanced OS virtualization
  - AMBA® 4 ACE system coherency
  - 1 TB physical memory addressable

big.LITTLE processing with Cortex-A15 MPCore and CCI-400

- Processor cluster includes
  - 1-4 processor cores with integrated L2, SCU and bus interface
- IP available now



**Compelling performance at <100mW for big.LITTLE technology and standalone use**

# big.LITTLE Processing: 2013

- Tightly coupled combination of two ARM CPU clusters:
  - Cortex-A15 and Cortex-A7 processors - functionally identical CPUs
  - Same programmers view, looks the same to OS and applications
- big.LITTLE combines high performance and low power
  - Automatically selects the right processor for the right job
  - Redefines the efficiency/performance trade-off

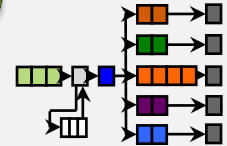


# Right Size Core for the Task

LITTLE

Most energy-efficient applications processor from ARM

- Best efficiency for light tasks
- Simple, in-order, 8-stage pipeline
- Recommended implementation target: highest efficiency

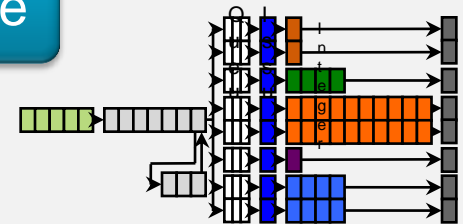


**Cortex-A7**

big

Highest performance in mobile power envelope

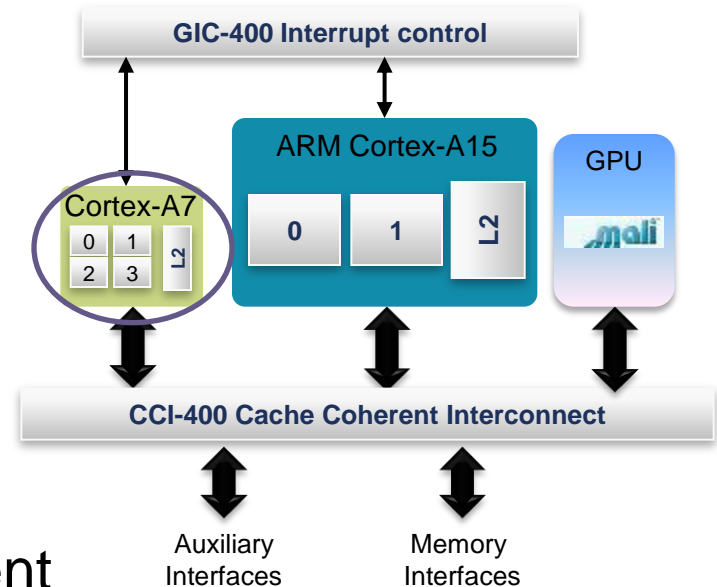
- Performance for heavy tasks
- Complex, out-of-order, multi-issue pipeline
- Up to 2x performance of today's high-end smartphones
- Recommended implementation target: high performance



**Cortex-A15**

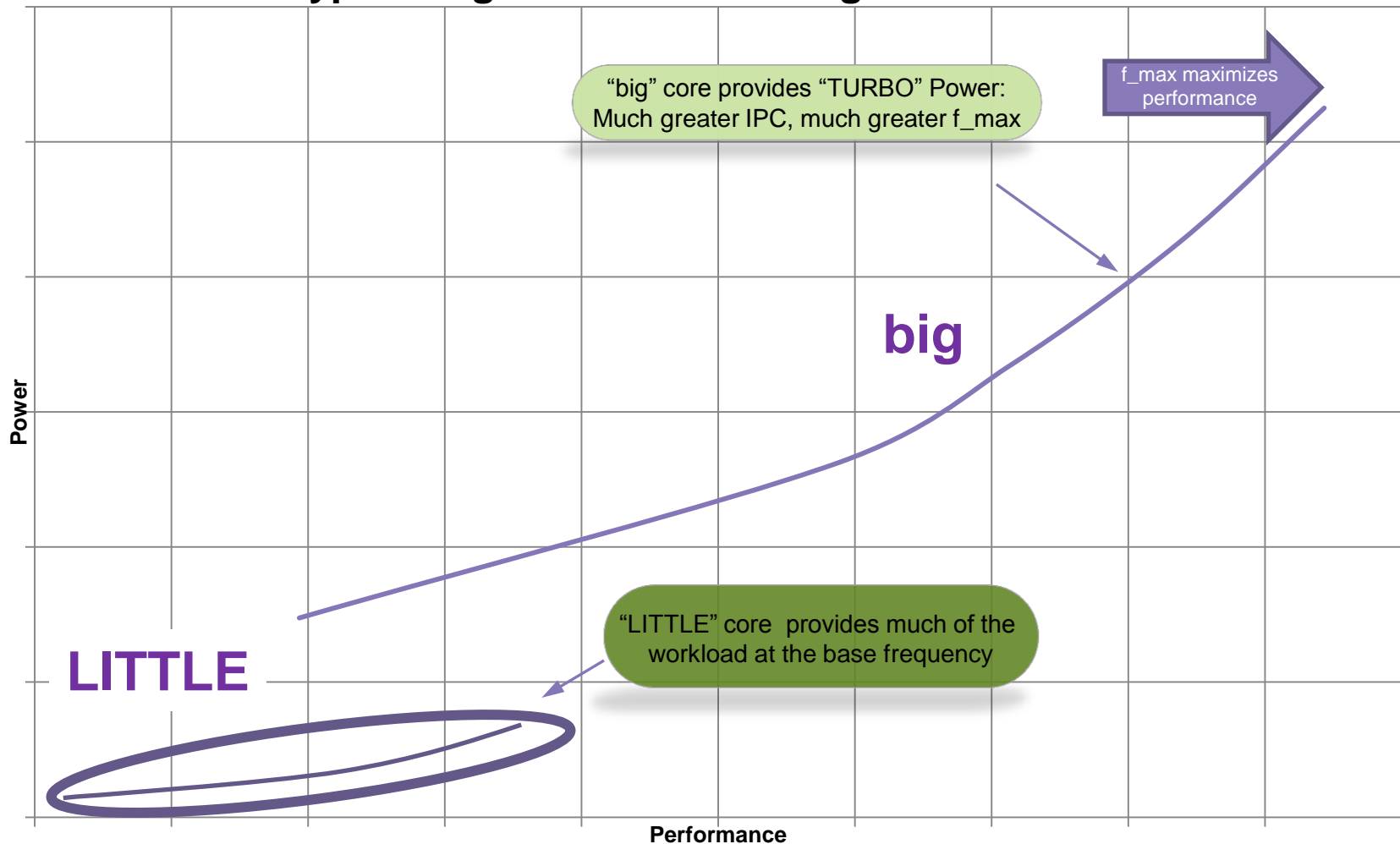
# Implementation Targeting for a big.LITTLE System-on-Chip

- Big cluster: Cortex-A15 processor
  - Choose aggressive frequency target
  - Power is mitigated ~50% with MP software
- LITTLE cluster: Cortex-A7 processor
  - Choose high efficiency target
  - Very small area for quad core!
- CoreLink™ CCI-400 Cache Coherent Interconnect
  - Implement to favor performance
  - Do not starve the big cluster
- GIC-400
  - Provides transparent virtualized interrupt control
  - Implement to favor performance



# Performance and Energy-Efficiency

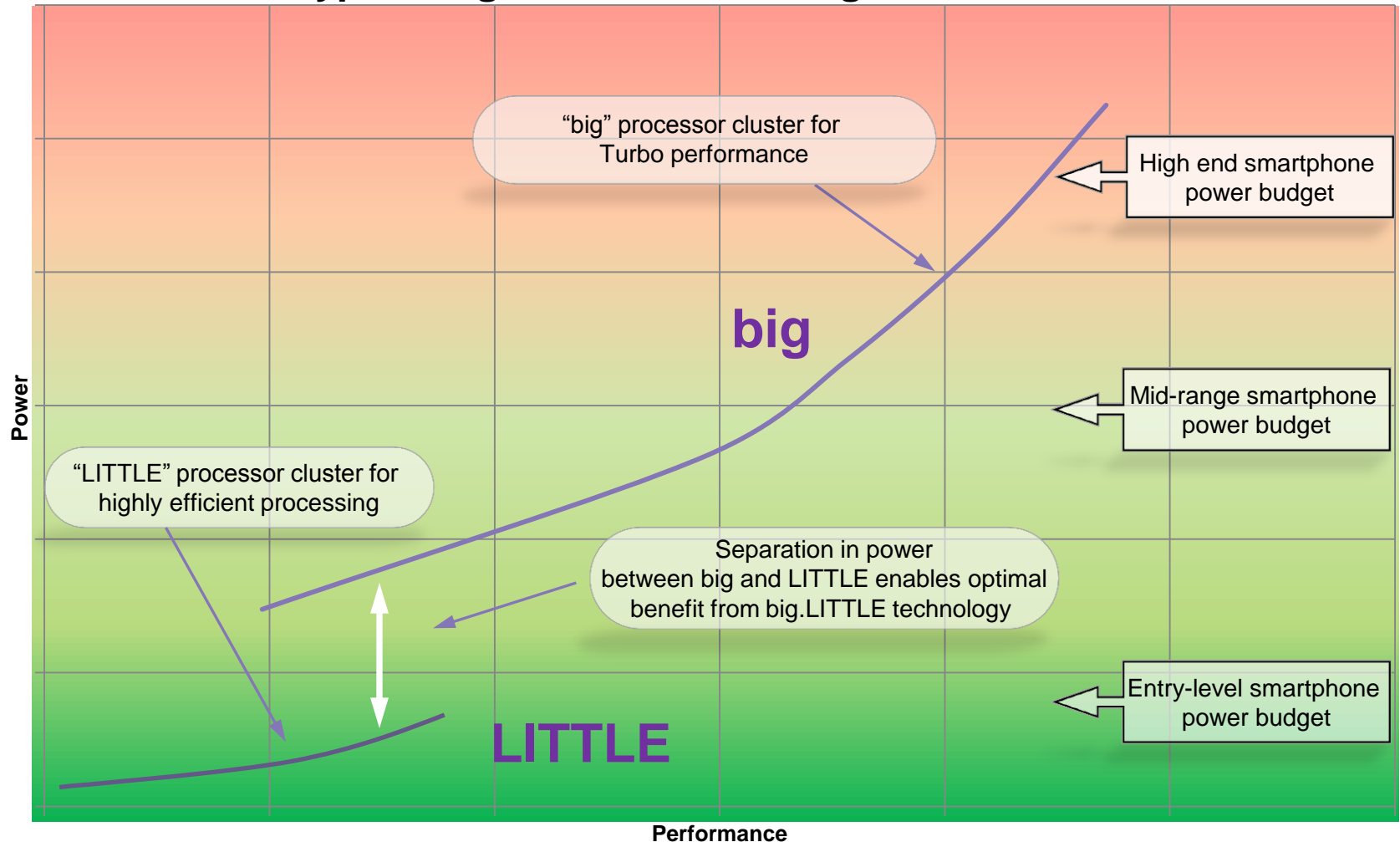
## Typical big.LITTLE DVFS Single-Core Curves



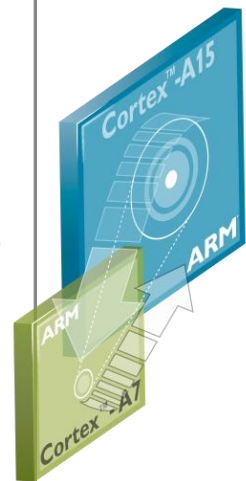
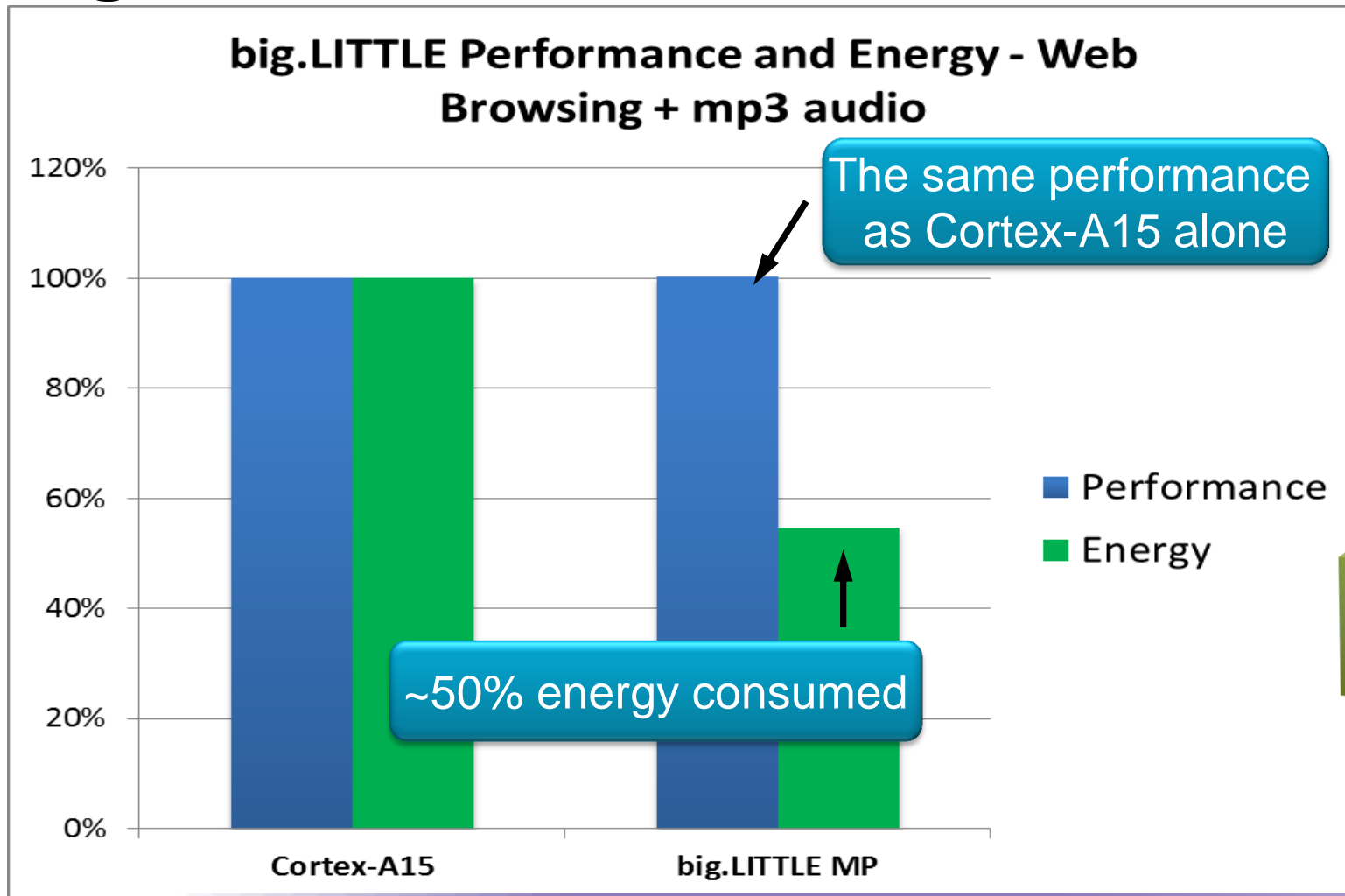


# Performance and Energy-Efficiency

## Typical big.LITTLE DVFS Single-Core Curves



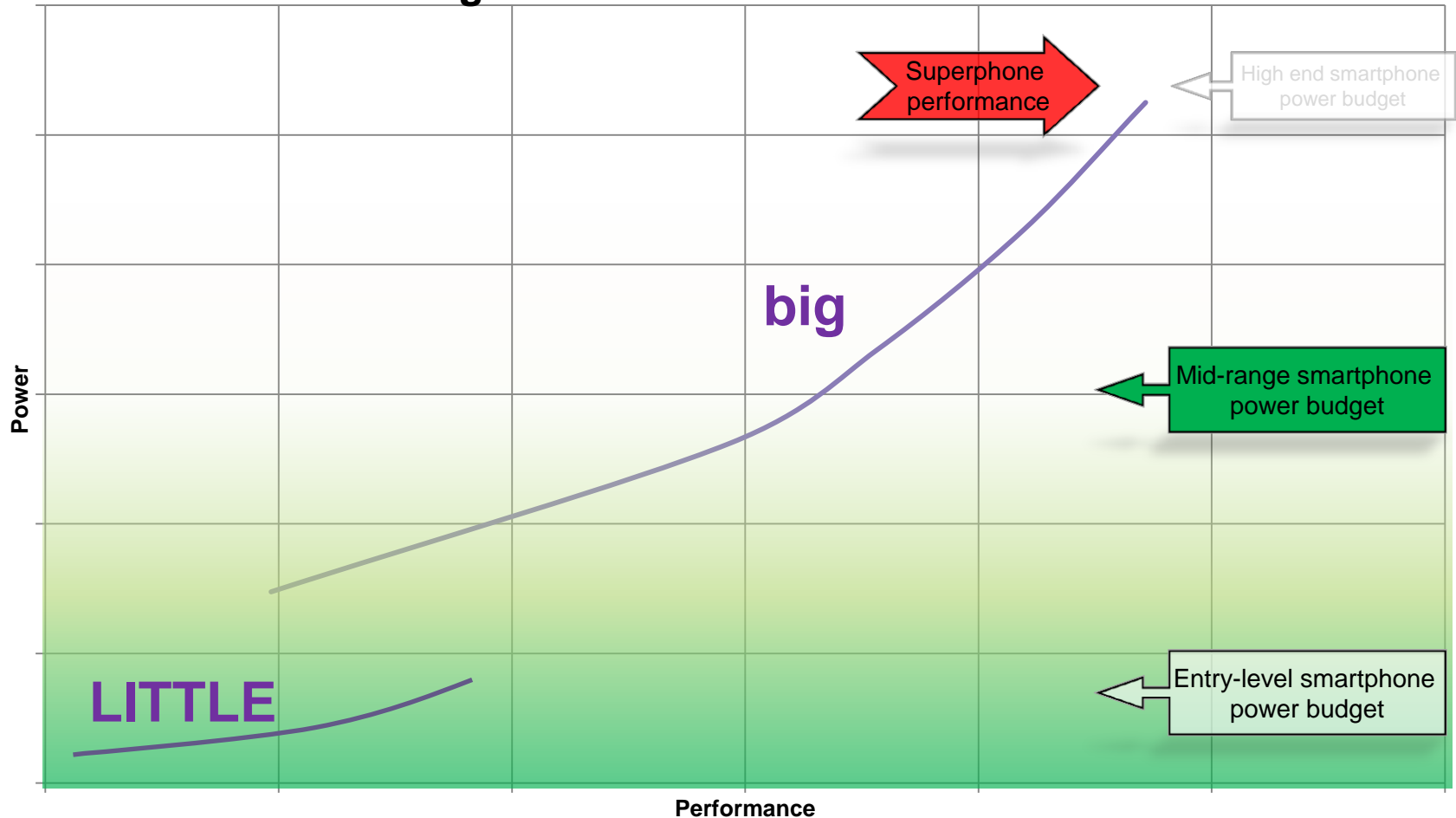
# big.LITTLE Measured Results



Optimized Power and Responsiveness

# Performance and Energy-Efficiency

## big.LITTLE Effective Power Draw



big.LITTLE Technology delivers superphone performance in an effective mid-range power budget

# Collaboration Expanded

*To Deliver Optimized Methodologies For ARM Cortex Processors*



**ARM and Synopsys Expand Collaboration to Optimize Power and Performance, and Accelerate Design and Verification for ARM Technology-based SoCs**

**CAMBRIDGE, United Kingdom and MOUNTAIN VIEW, Calif., Aug. 28, 2012**



**ARM and Synopsys Collaborate to Deliver Optimized Reference Implementations for ARM Processors**

*Optimized Methodologies for ARM's Cortex-A15, Cortex-A7 and CCI-400 Solutions Help Designers Achieve Processor Performance and Power Objectives Faster*

**CAMBRIDGE, UK, and MOUNTAIN VIEW, Calif. Mar. 21, 2013**

# Collaboration Objectives

## *Excellent Implementation For Cortex-A7 Processor*

### QOR

- Meet power target while optimizing for best timing within power budget, best area within power and timing budgets
- Target market requires a power centric implementation

### Schedule

- Develop quad-core Cortex-A7 flow quickly for stand-alone or big.LITTLE
- Enable ARM and Synopsys customers timely access

### Flow

- RTL through Route
- Repeatable, robust, easily modifiable scripts

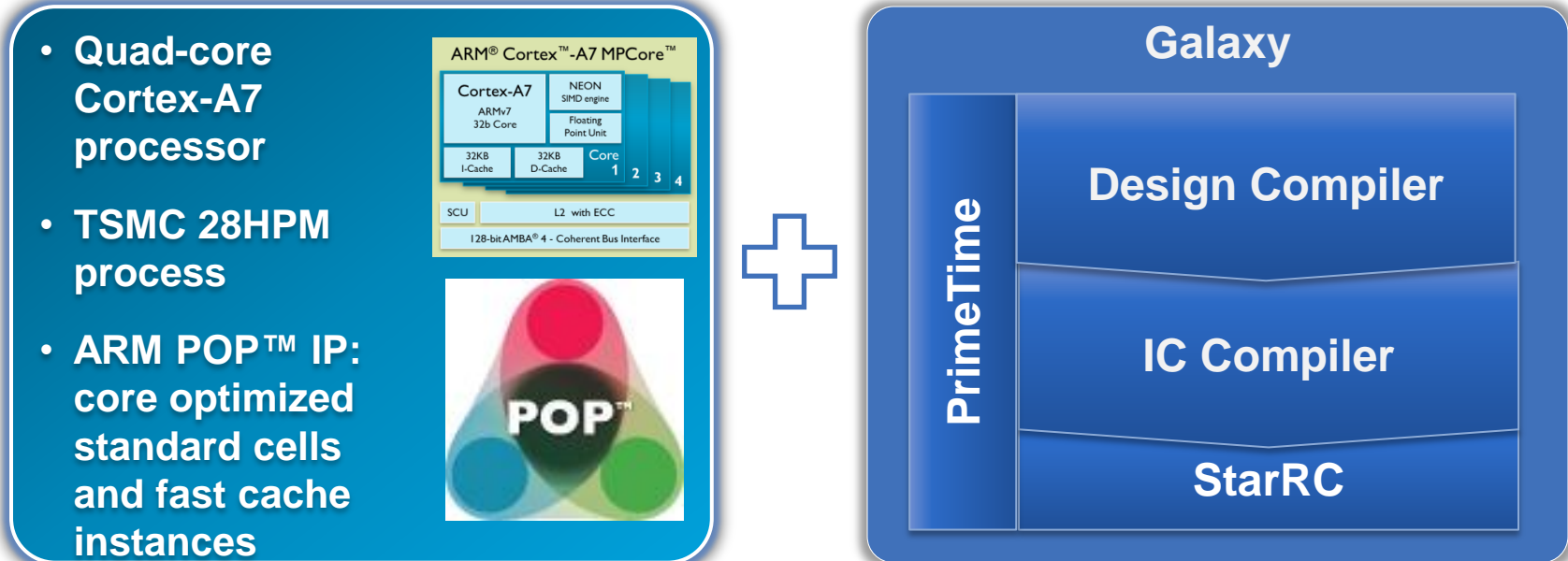
### Documentation

- Guidelines for joint customers to follow when targeting a different configuration
- Best practices and pitfalls



**Primary Deliverables: Reference Implementations (RI)**  
with real, repeatable results

# ARM + Synopsys Collaboration



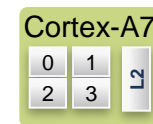
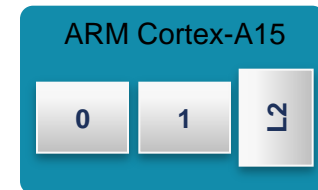
**Synopsys Engineering and Low Power Expertise**



**Reference Implementation** for an ARM Cortex-A7 MPCore processor optimized for excellent efficiency

# Recommended Implementations for your big.LITTLE SoC

- Implementation recommendations developed as a result of the ARM and Synopsys collaboration
- Step 1: Download your ARM Deliverables
  - Processor and fabric IP
- Step 2: Download your Synopsys Deliverables
  - Available now via SolvNet for joint Synopsys and ARM customers
    1. “big” cluster: dual-core Cortex-A15 processor
      - Scripts, design information, documentation
    2. “LITTLE” cluster: Cortex-A7 processor
      - Scripts, design information, documentation
    3. Corelink™ CCI-400 Cache Coherent Interconnect
      - Scripts, design information, documentation



CCI-400 Cache Coherent Interconnect



Reference Implementations are an excellent starting point for implementing an effective big.LITTLE SoC processor

# Agenda

ARM-Synopsys Project Introduction

Bernard Ortiz de  
Montellano



---

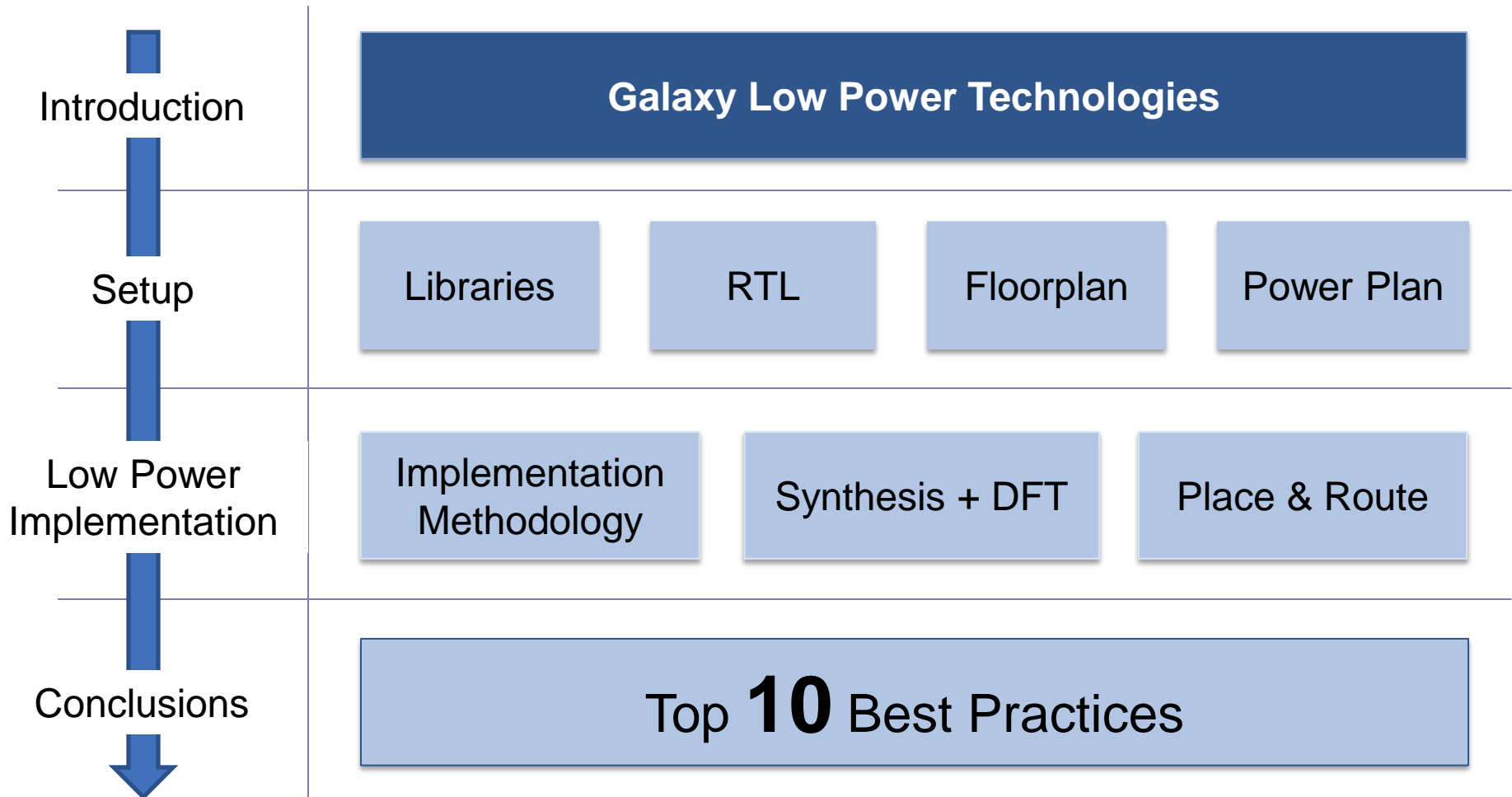
Power-Centric Timing Optimization Flow  
for a Quad-Core Cortex-A7 Processor

Dale Lomelino



# Power-centric Timing Opt. Flow

## For a Quad-Core Cortex-A7 Processor



# Galaxy Low Power Technologies

*Feature Subset Used In Quad-Core Cortex-A7 Implementation*



Synopsys Users Group

## Dynamic Power

- Advanced clock gating
- CTS w/ low power placement (LPP)

## Leakage Power

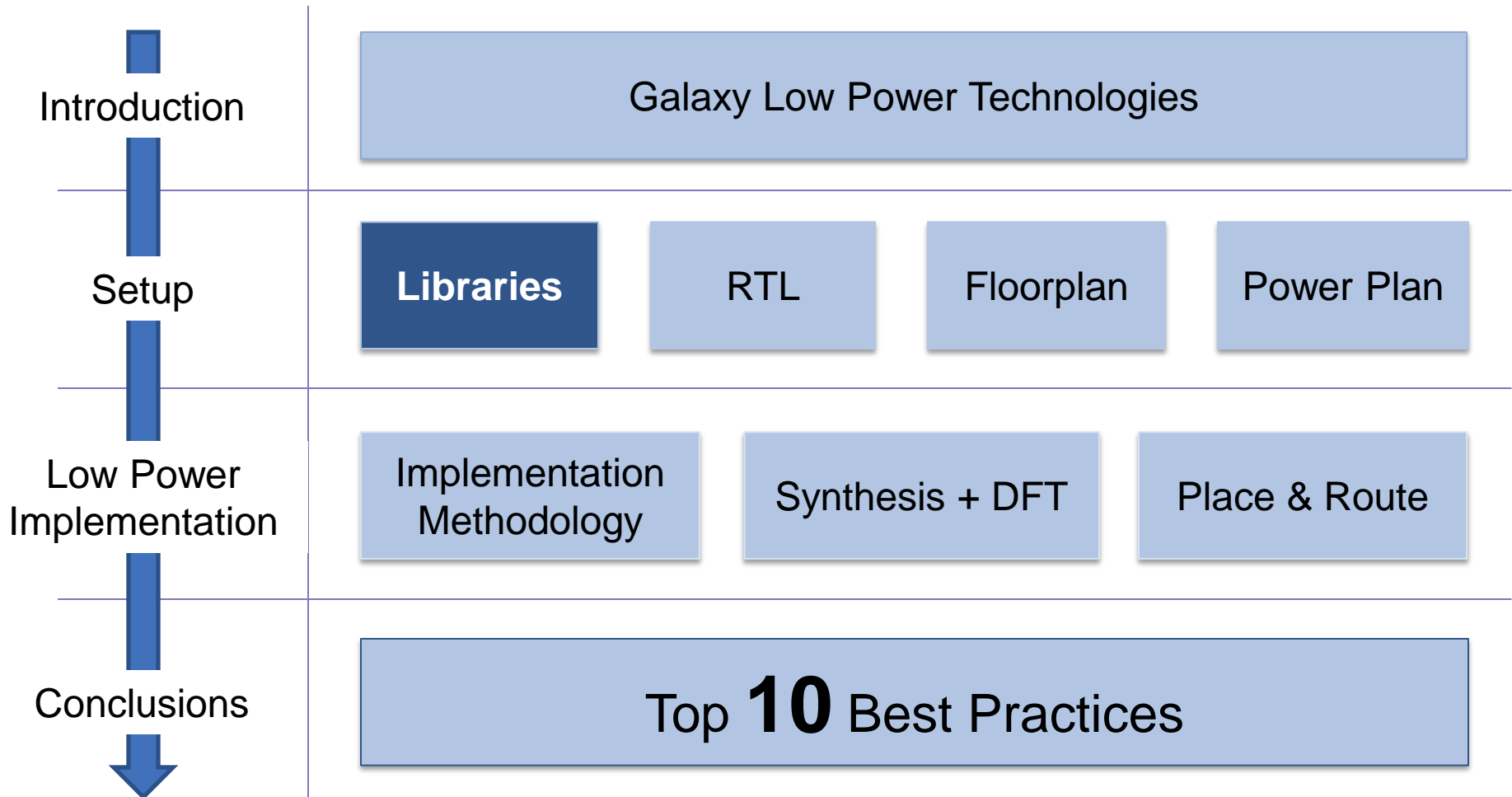
- Multi-threshold (Multi-Vt) libraries
- Limit usage of LVT cells
- Channel length cell variants
- Multicorner / Multimode (MCMM)
- Final-stage leakage recovery (FSLR)

## Multivoltage (UPF)

- Multivoltage power domains
- Shutdown regions

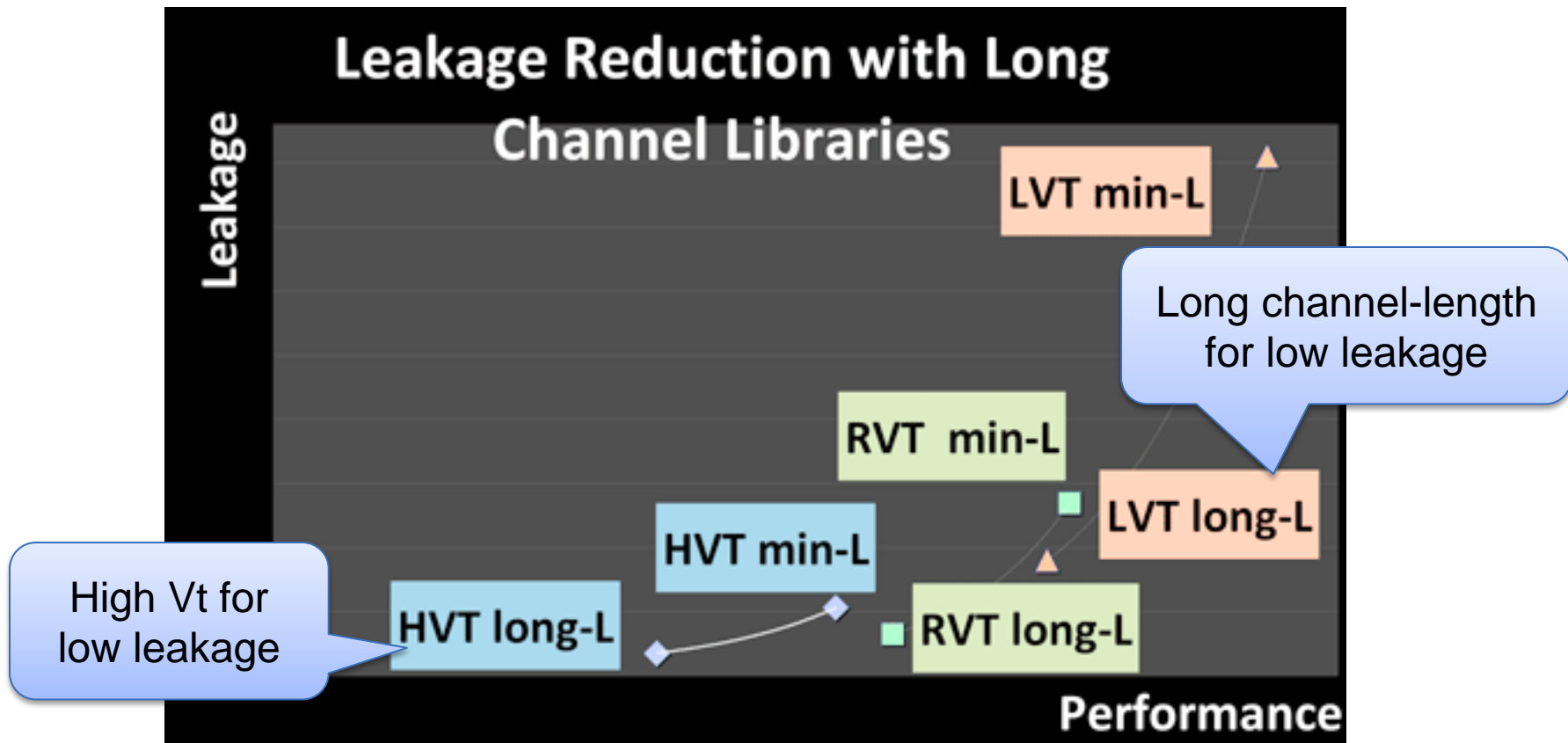
# Power-centric Timing Opt. Flow

## For a Quad-Core Cortex-A7 Processor



# Libs: Vt-classes / Channel-length

## *For Power/Timing Tradeoff*



Source: [www.arm.com/images/PIPD\\_Logic\\_MC\\_animation\\_small.gif](http://www.arm.com/images/PIPD_Logic_MC_animation_small.gif)

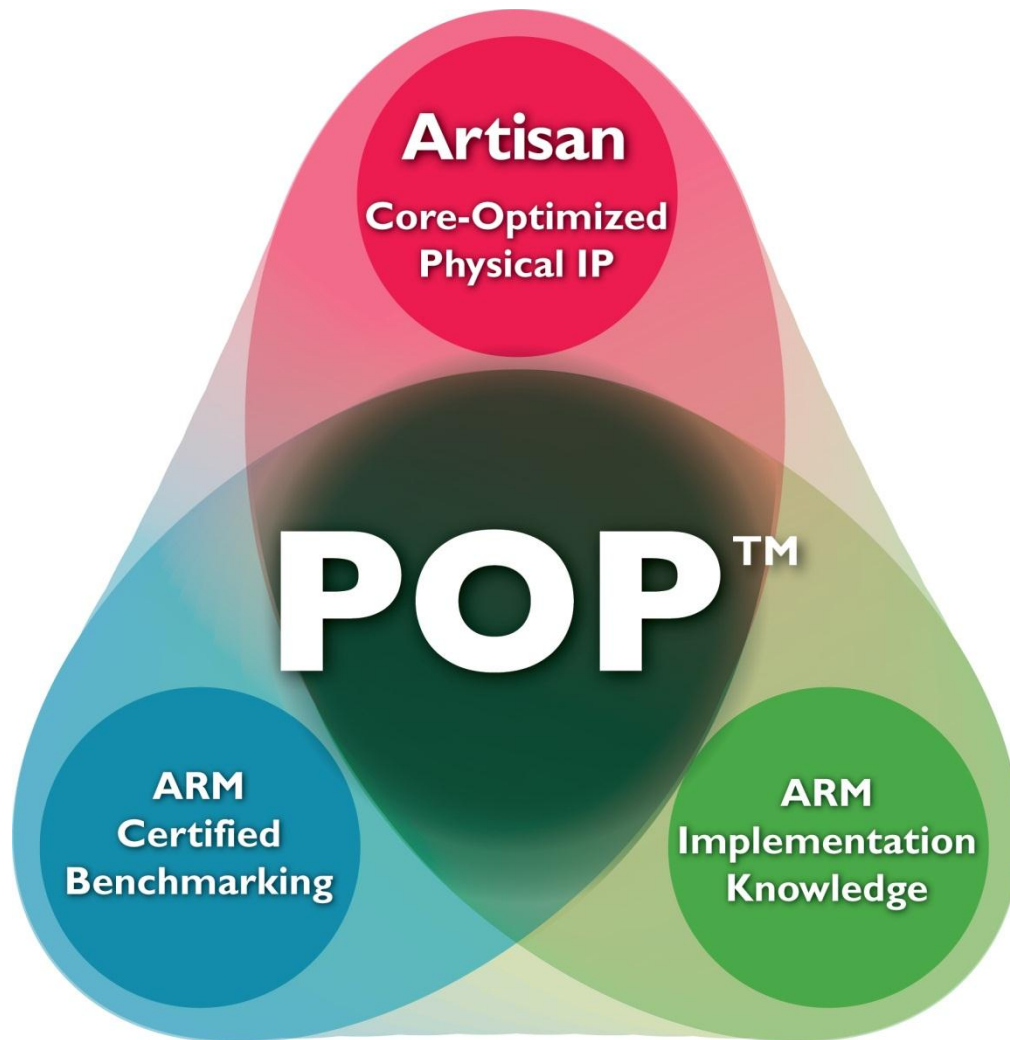
# Choosing Library Subset

## ARM Libraries for Quad-Core Cortex-A7 Processor

- Technology Details
  - TSMC28HPM process
  - 10 layer metal (1p10m\_5x2y2z)
  - ARM POP™ IP libraries
    - Fast Cache Instance (FCI) RAMs
    - Standard cells
  - 9T high-density libraries
    - exclude "CL" for stand-alone config
- PVT Configuration - 4 corners
  - Setup (OC\_WC): SSG / 0.81V / 0C
  - Hold (OC\_BC): FF / 1.05V / 125C
  - Power (OC\_LEAK): TT / 0.90V / 85C
  - IR (OC\_IR): FFG / 1.0V / 125C
- Three transistor channel lengths
  - CS = "short" (faster, more power)
  - CM = "medium" (standard)
  - CL = "long" (slower, less power)
  - same cell\_footprint, swappable

Standard Cell Selection (Multiple Vt / Channel-length variants)		
Vt Class	Channel Length	Cell Family
ULVT	CS	not used
	CM	
LVT	CS	
	CM	
	CL	not used
SVT (RVT)	CS	
	CM	
	CL	not used
HVT	CM	
UHVT	CM	
<ul style="list-style-type: none"><li>• 6 Vt/channel-length classes used</li><li>• ULVT not used due to leakage power</li><li>• <b>CL</b> has add'l monetary cost, not used</li></ul>		

# ARM POP™ IP Core-Hardening Acceleration Technology



*New Processes*

*New Cores*

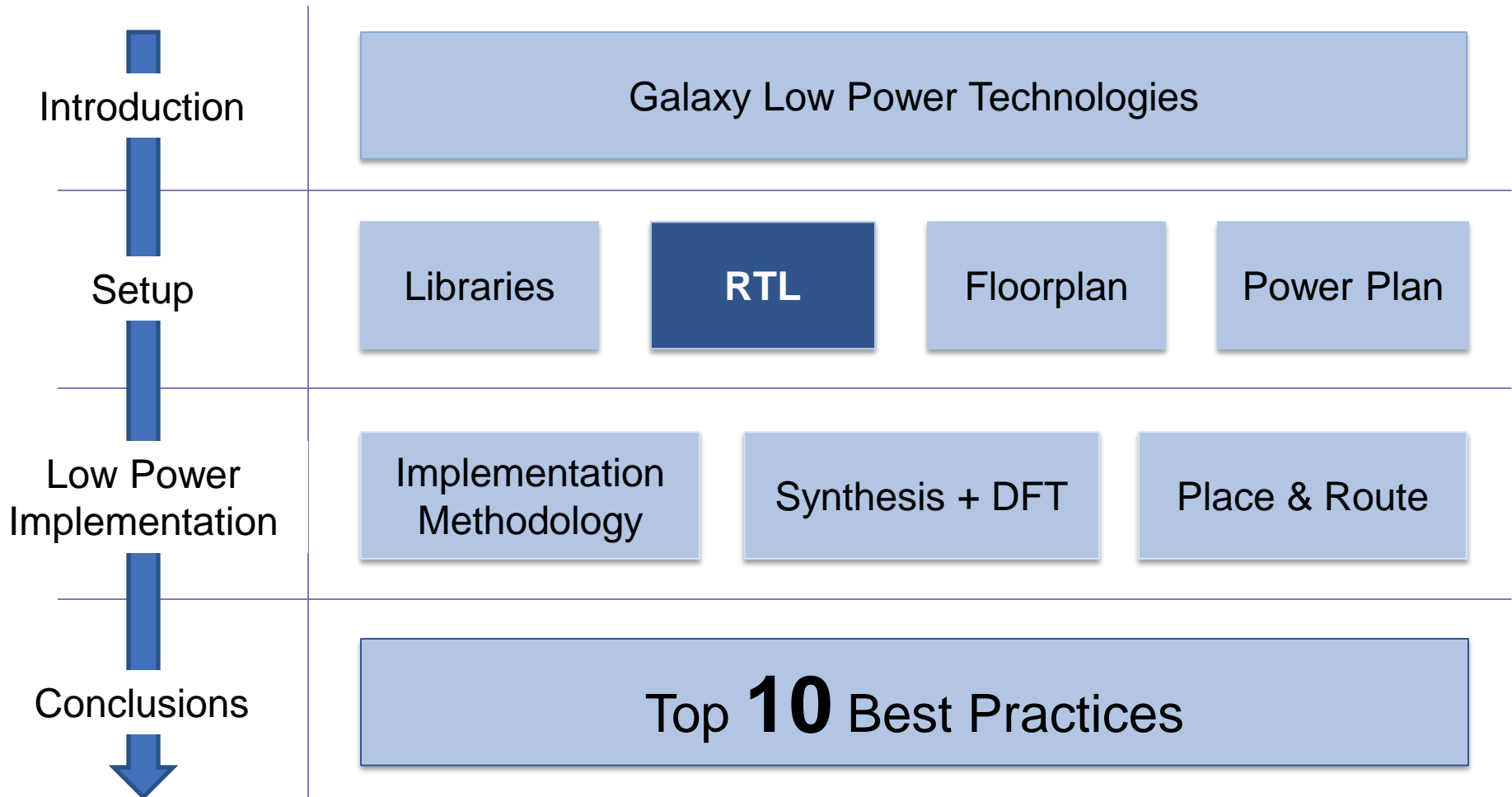
*New Optimizations*

POP IP libraries are used in the quad-core Cortex-A7 MPCore cluster

*Core-Hardening Acceleration by ARM*

# Power-centric Timing Opt. Flow

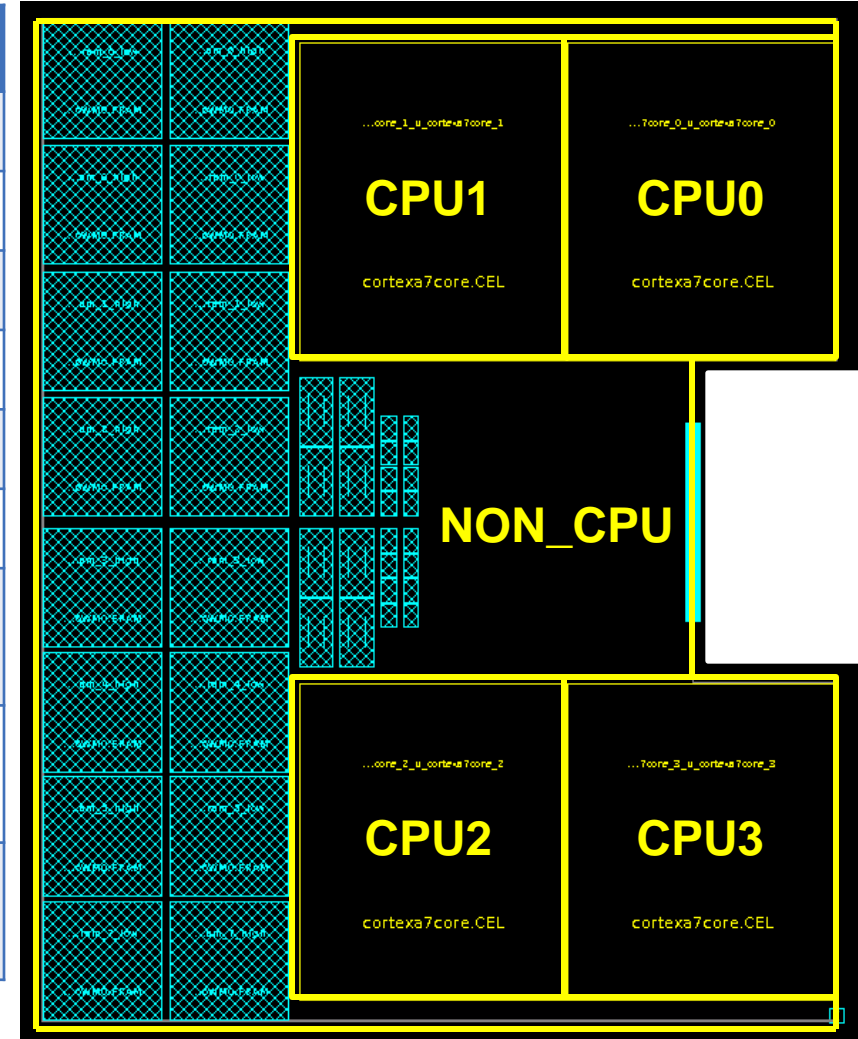
## For a Quad-Core Cortex-A7 Processor



# Quad-Core Cortex-A7 Configuration

## *Representative Across Many Applications*

Configurable Feature	Selected Value
# Cores	4
L2 cache size	1MB
L1 Instruction cache	32KB
L1 Data cache	32KB
NEON™	Included
FPU	Included
Generic Interrupt Controller (GIC)	Included
Embedded Trace Macro Cell (ETM)	Included
Shared Peripheral Interrupts	128



"The Spec": Worked Example Specification for big.LITTLE, Cortex™-A15, Cortex™-A7, Cortex™-A9, and CCI-400



# Edit RTL = Configuration

## *Configure ETM/GIC Modules, Instantiate RAMs*

- Design configuration defined in global configuration file
- Behavioral RAM components changed to library RAM macro instantiations.
- Behavioral clock gating cells changed to library clock gating cell instantiations
- RTL changes during project

CORTEXA7INTEGRATION\_  
CONFIG.v

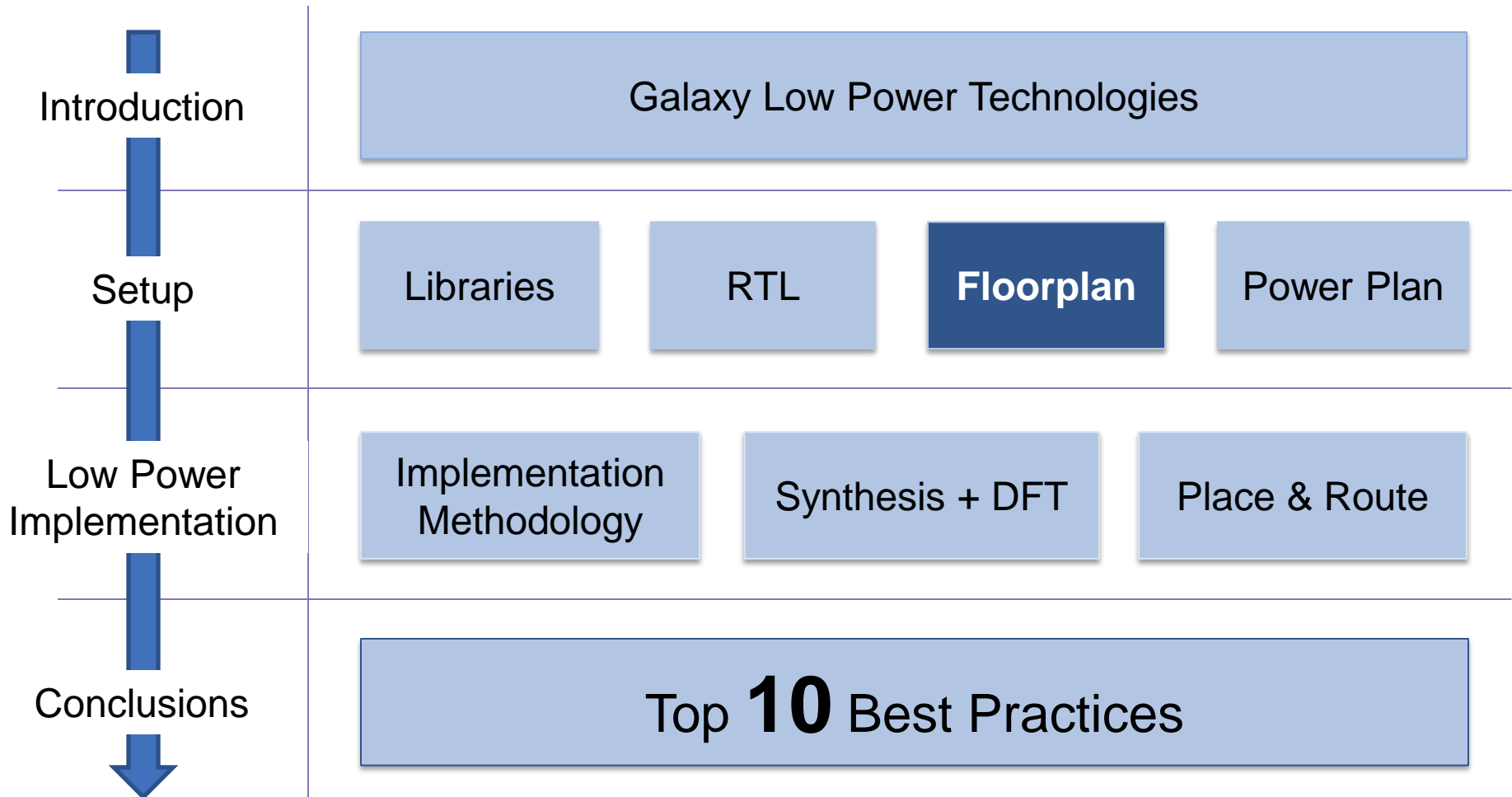
**Fast Cache Instance (FCI)**  
**RAMs**, from POP IP libraries

Used **LVT ICGs**, for smallest  
insertion delay

Parameters included:  
ETM\_PRESENT/GIC\_PRESENT

# Power-centric Timing Opt. Flow

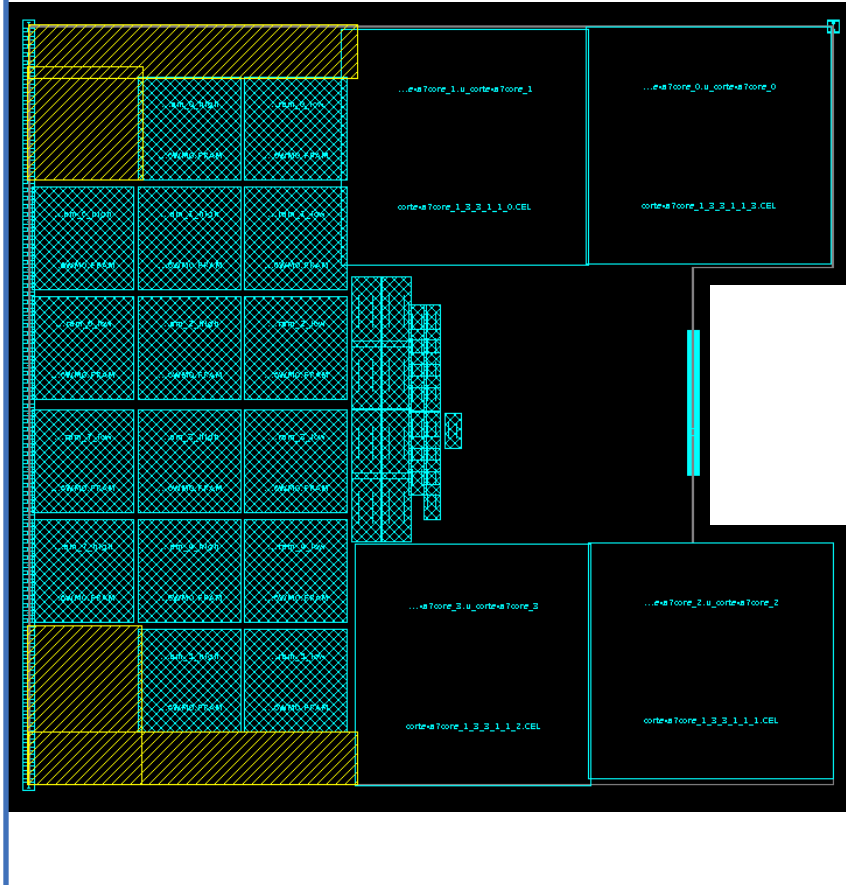
## For a Quad-Core Cortex-A7 Processor



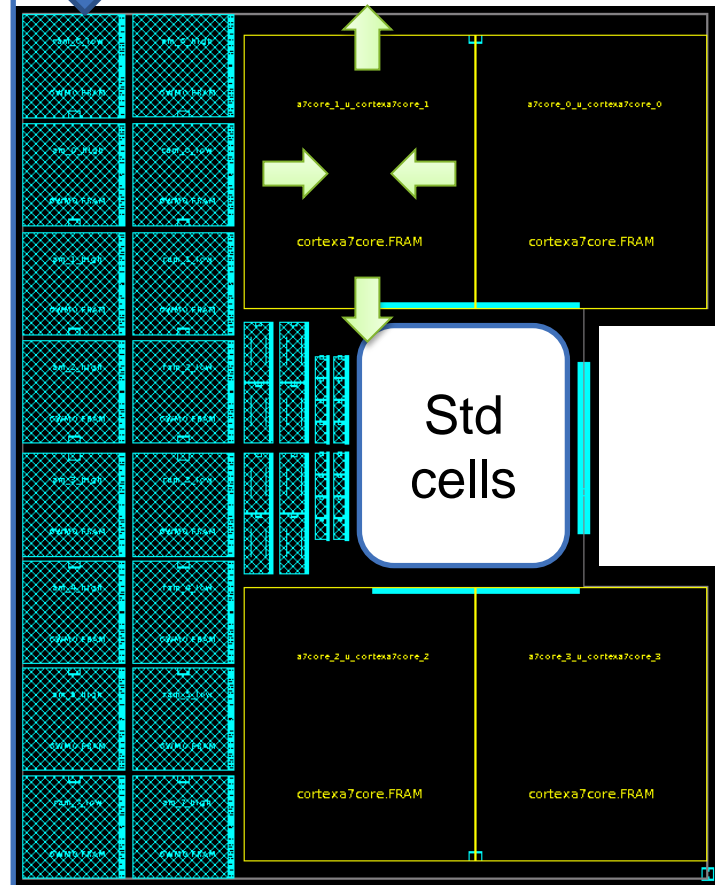
# Floorplan Refinement

*Leveraged ARM's Experience, Adjust For This Cfg*

## Early Floorplan



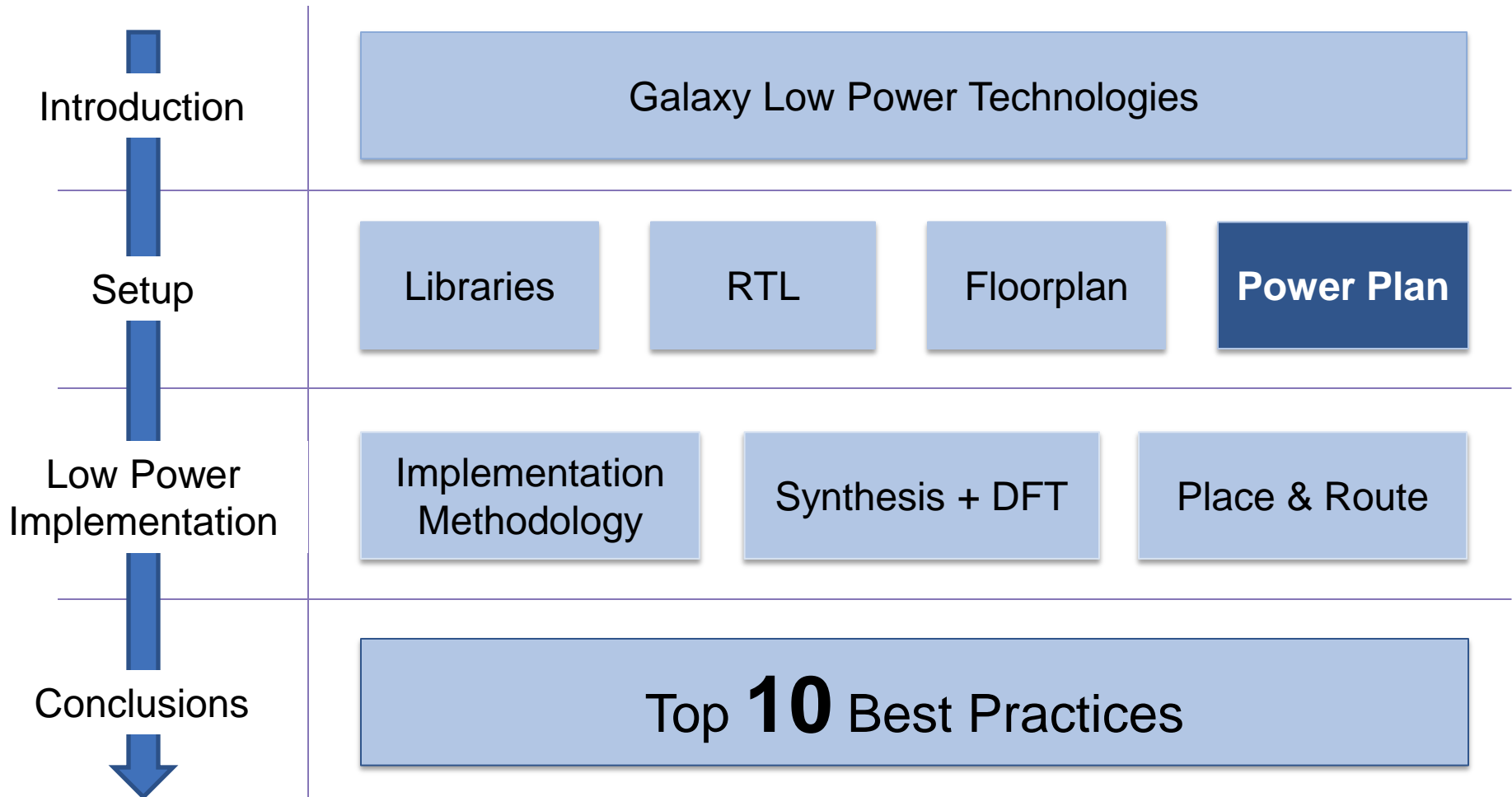
## Refined Floorplan



**Saved Area - Made Use Of Corners, Reshaped CPU To Match**

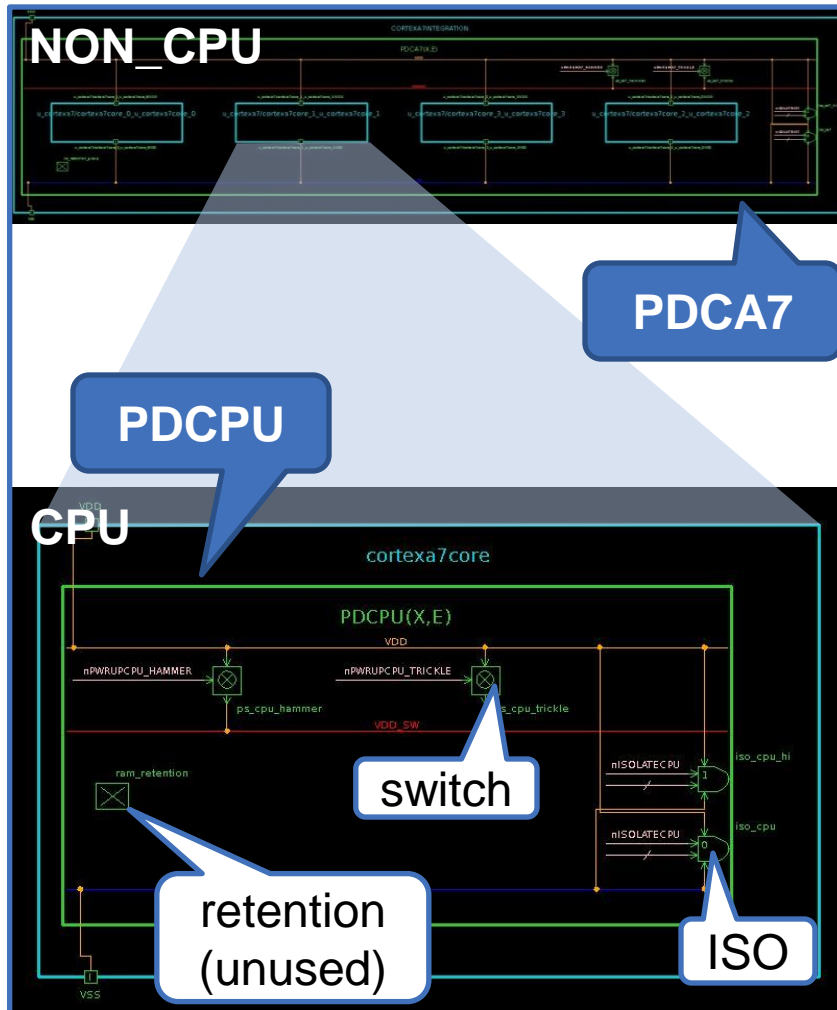
# Power-centric Timing Opt. Flow

## For a Quad-Core Cortex-A7 Processor



# UPF & Power Plan - ARM iRM Based

## Use Design Compiler Design Vision GUI To Visualize



Name	Type
<b>H</b> CORTEXA7INTEGRATION	Scope
<b>H</b> u_cortexa7/cortexa7core_3_u_cortexa7core_3	Scope
VSS	Supply Port
VDD	Supply Port
<b>H</b> u_cortexa7/cortexa7core_2_u_cortexa7core_2	Scope
VSS	Supply Port
VDD	Supply Port
<b>H</b> u_cortexa7/cortexa7core_1_u_cortexa7core_1	Scope
VSS	Supply Port
VDD	Supply Port
<b>H</b> u_cortexa7/cortexa7core_0_u_cortexa7core_0	Scope
VSS	Supply Port
VDD	Supply Port
q7_pst	Power State Table
VSS	Supply Port
VDD	Supply Port
<b>PD</b> PDCA7	Power Domain
ps_ca7_trickle	Power Switch
ps_ca7_hammer	Power Switch
no_retention_group	Retention Strategy
iso_ca7_hi	Isolation Strategy
iso_ca7	Isolation Strategy
VSS	Supply Net
VDDCA7	Supply Net
VDD	Supply Net

Design Vision > Power > Visual UPF...

Design Hierarchy | Diagram | Power Hierarchy | UPF Script | Error/Warning

# UPF Modifications

## *Changes From ARM's Cortex-A7 iRM*

- Started with `supply_net` UPF



Added commands for I/Os

`set_related_supply_net`

- For top-level (NON\_CPU)



Added supply net connections for CPU domains

Added logic ports, logic nets for

- Switch cell controls
- Isolation cell controls

Supports verification in Formality

- Updated to use new UPF constructs supported in 2011.09 and later versions



Created additional bias.upf file

Added variable:

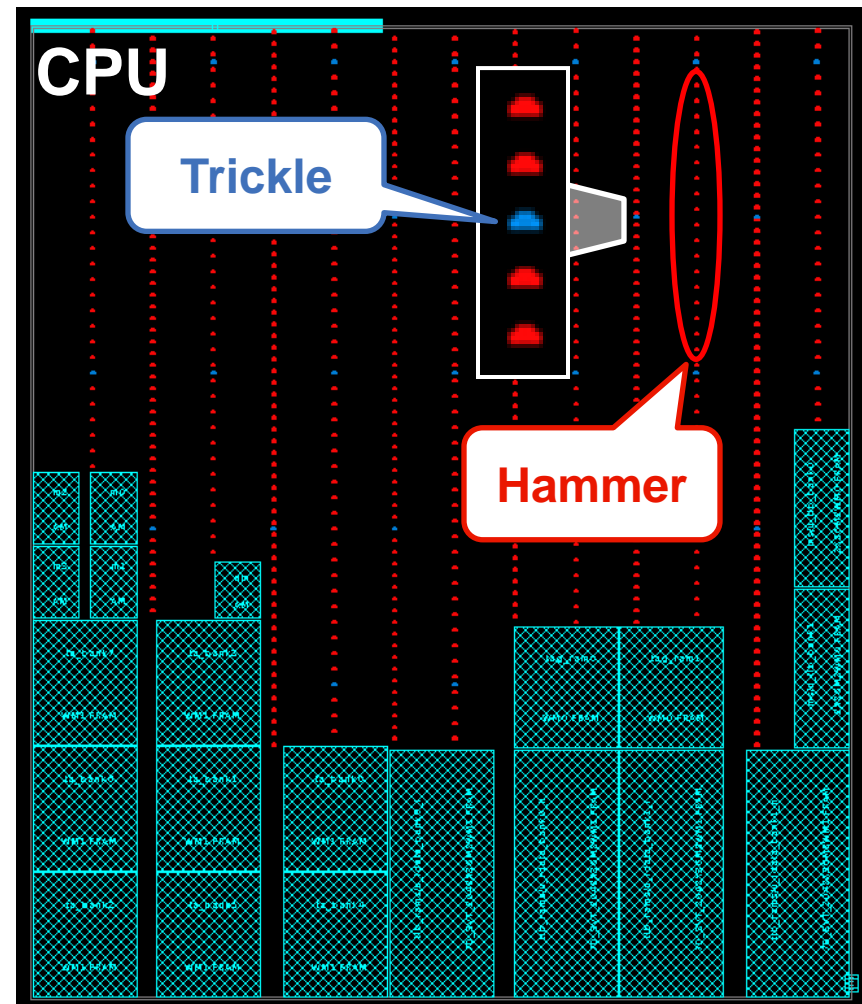
```
set
upf_create_implicit_supply_
sets false
```

- Updated port\_state(s) and PST setup

# Power Switches - ARM iRM Based

## *Leveraged ARM's Experience To Balance IR-drop And In-rush Current*

- **"Trickle"** switches (blue)
  - Few = limits in-rush current
  - HEADBUFTIE26\_X3M\_A9TS\_CM
- **"Hammer"** switches (red)
  - Many = low on-resistance (~20:1 hammer:trickle)
  - HEADTIE22\_A9TS\_CM
- SVT\_CM switches for low leakage
- RAMs have built-in switches

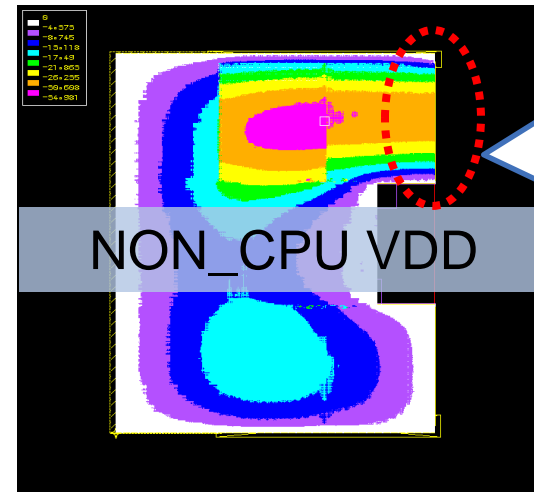
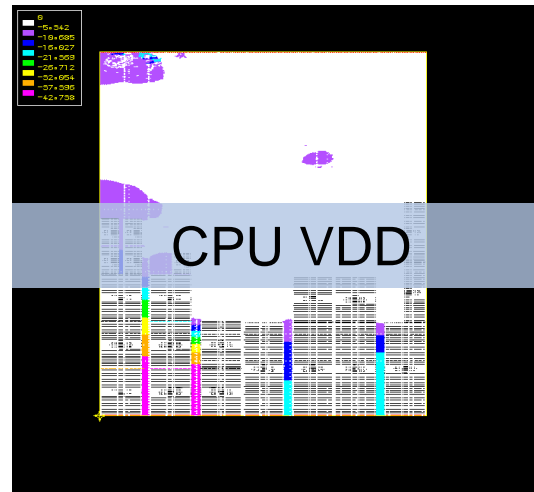




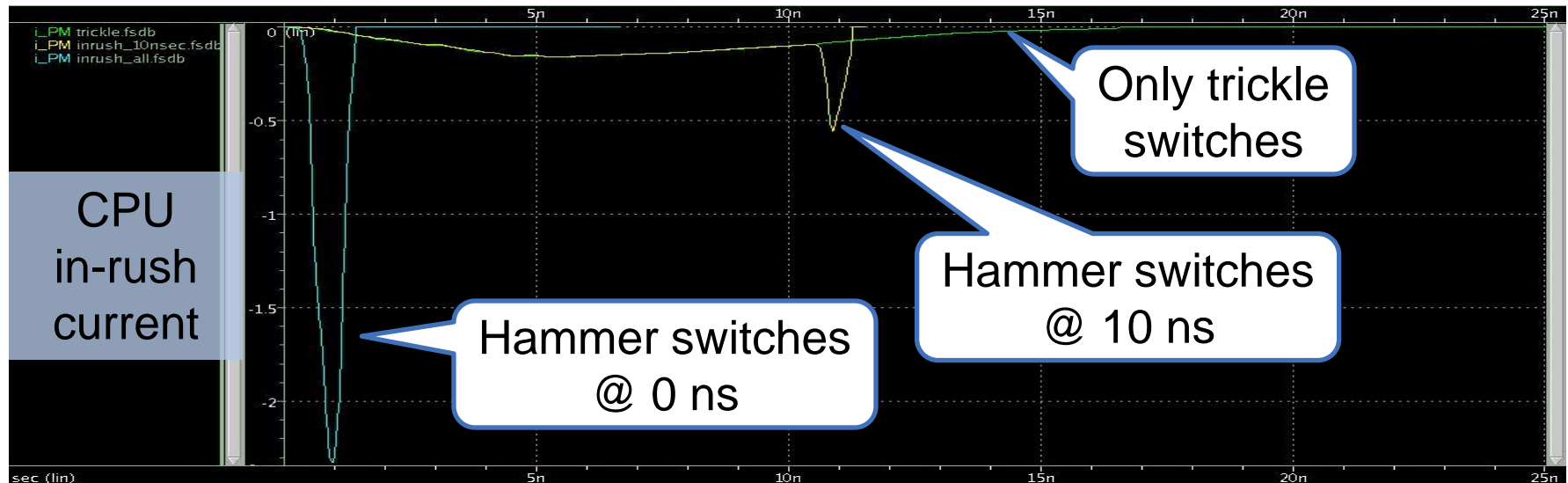
# Power Analysis w/ PrimeRail

*Used To Debug IR-drop & In-rush Current Trade-offs*

IR-drop



Missing power connections, fixed by aligning pins



Only trickle switches

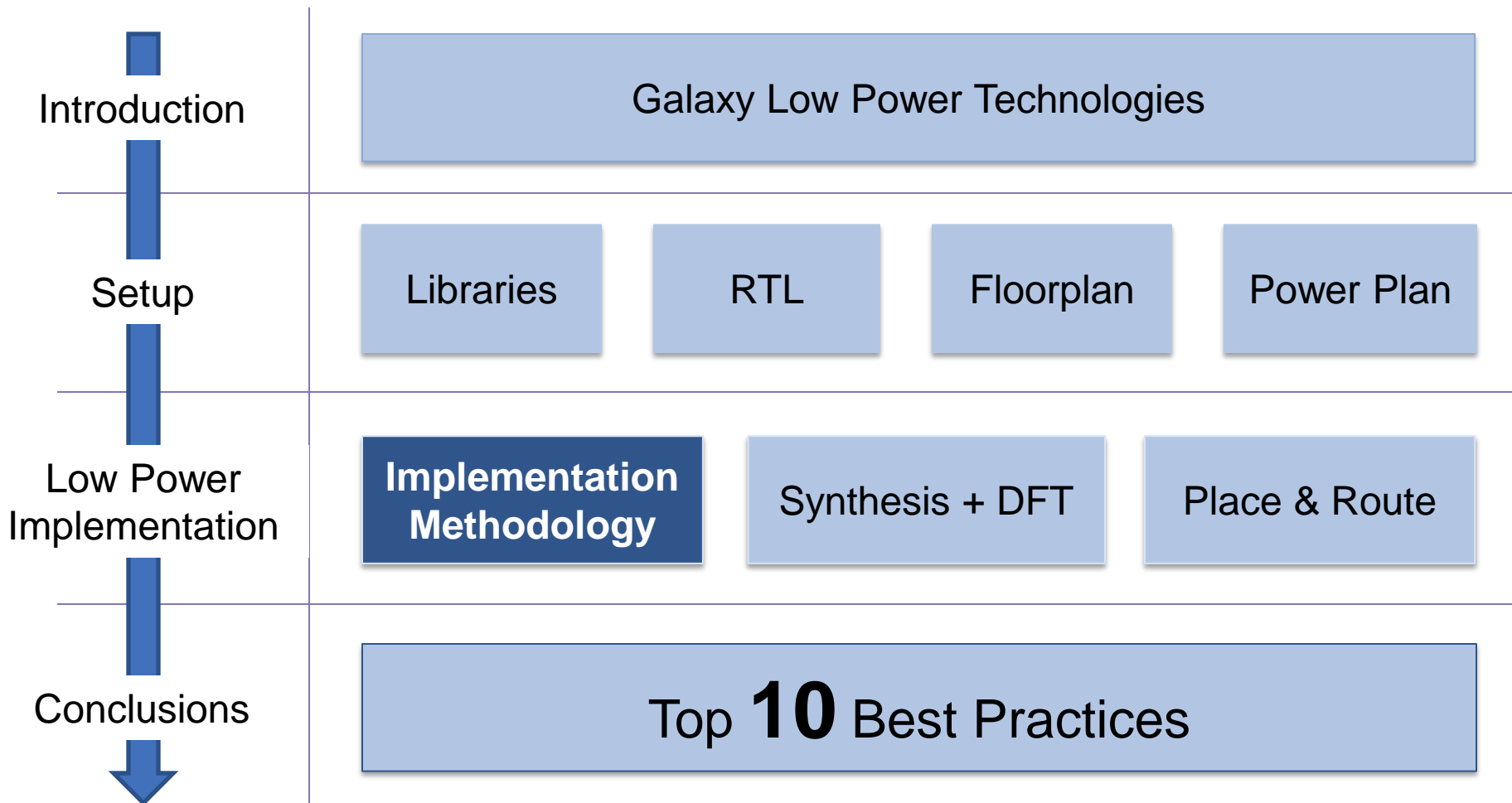
Hammer switches @ 10 ns

Hammer switches @ 0 ns



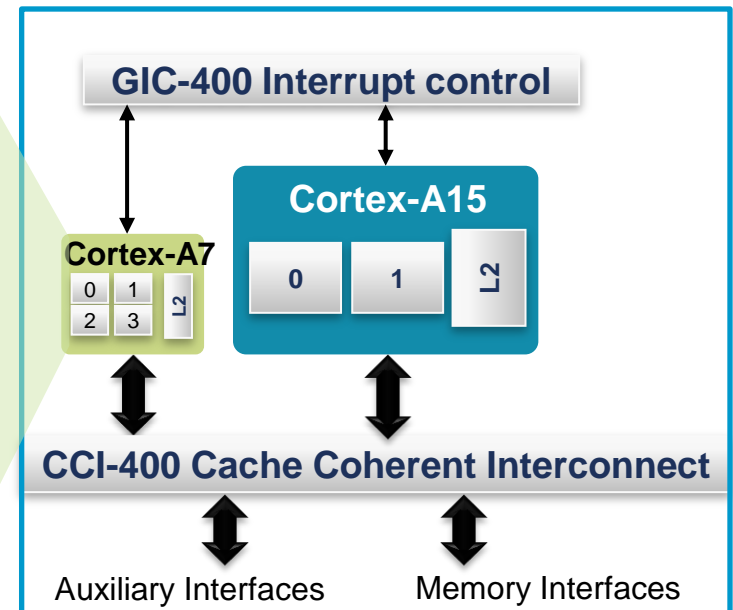
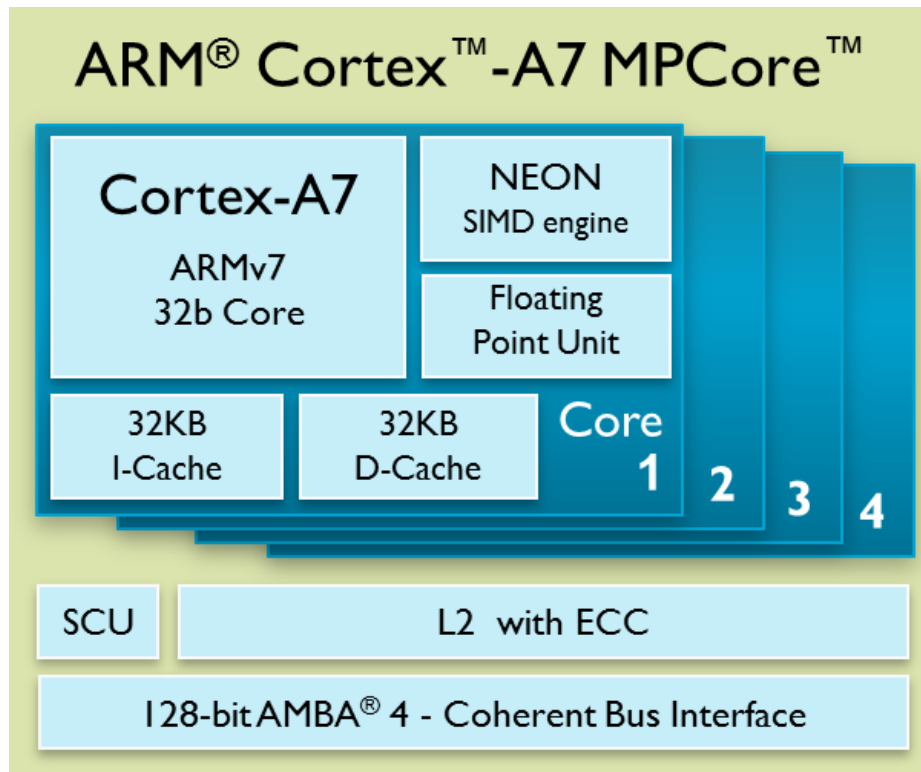
# Power-centric Timing Opt. Flow

## For a Quad-Core Cortex-A7 Processor



# Quad-Core Cortex-A7 Configuration

A7 = "LITTLE" in big.LITTLE



# Quad-Core Cortex-A7 Priorities

1. Meet the power target
2. Optimize for best timing  
(within power budget)
3. Optimize for area  
(within power & timing budgets)

*Goal: Deliver Cortex-A7 "Reference Implementation" flow*

- *scripts for cpu (cortexa7core)*
- *scripts for non\_cpu (CORTEXA7INTEGRATION)*

**Power is #1 in the "LITTLE" processor**

# Synopsys' Core Optimization Collateral

*Built on Galaxy Tool RMs*

- Leverages HPC
- Core and technology library specific
- Includes scripts, floorplan, constraints

Reference  
Implementations  
(RIs)

Lynx  
Plug-Ins

- RI scripts instrumented for Lynx environment

- Leverages RMs, tuned for high perf cores
- Core and technology library independent

Hi-Performance  
Core (HPC)  
Methodology

Reference  
Methodologies  
(RMs)

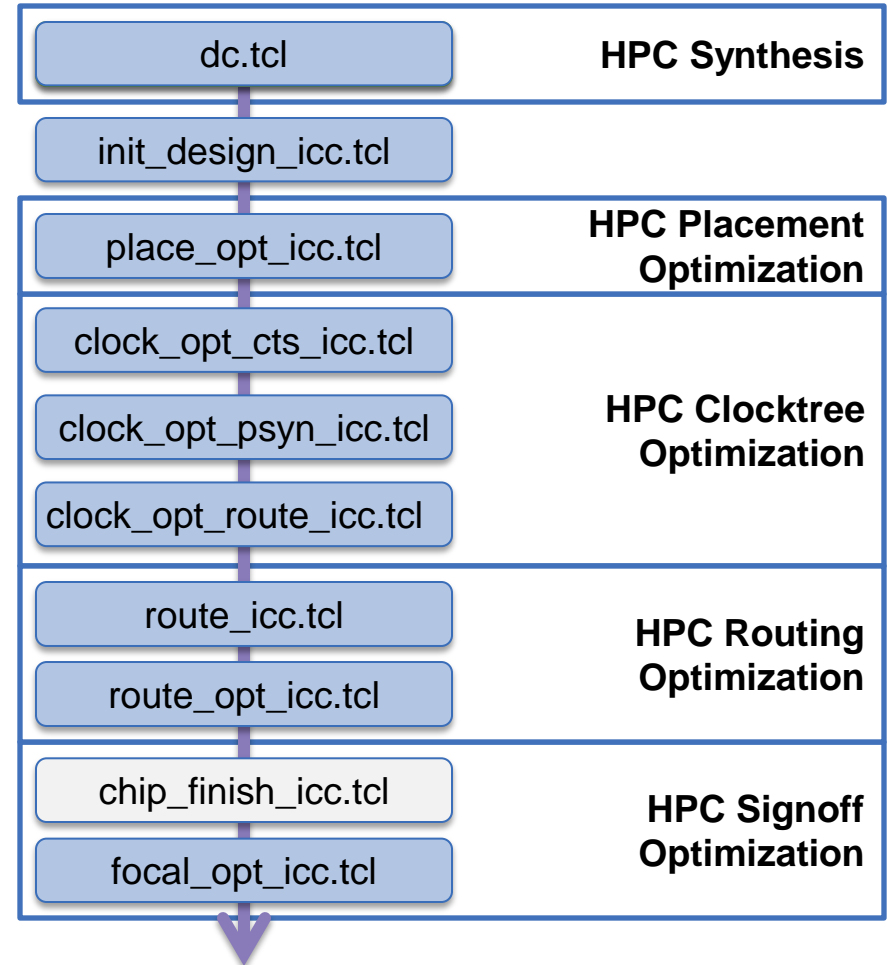
- Tool- and release-specific scripts
- Core and technology library independent

More design/technology specific

# Reference Implementation Flow

## *Captures Best Practices in the Scripts*

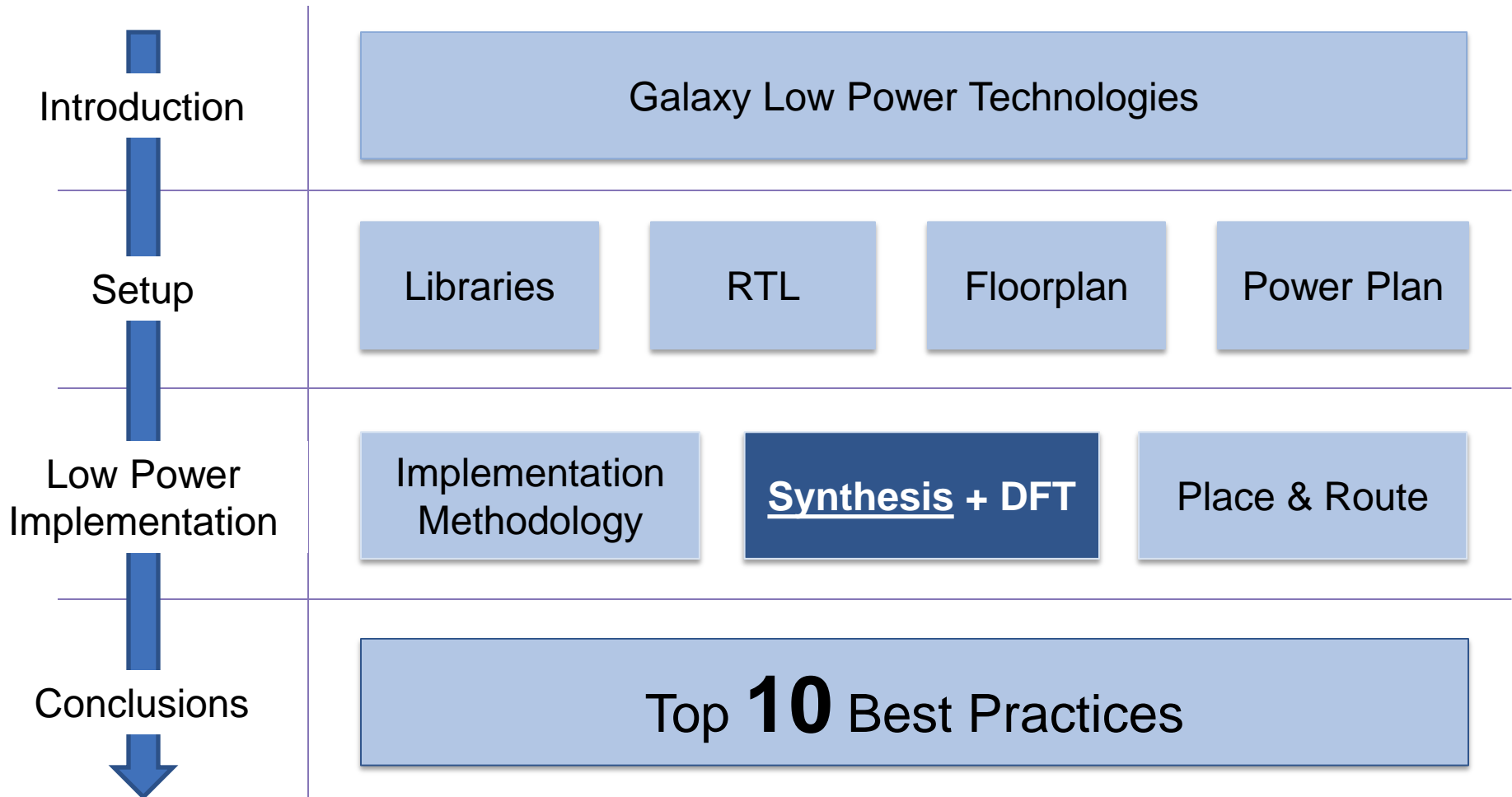
- Start w/ HPC with **-power** enabled throughout
- Floorplan is HPC input
- DCG with SPG
  - with **-power**
- ICC w/ SPG-based placement
  - with **-power**
- ICC CTS/CTO
- ICC optimized routing
  - with **-power**
- ICC focal\_opt timing-closure
  - with **-power**



**Reference Implementation is tuned for Quad-Core Cortex-A7 Processor**

# Power-centric Timing Opt. Flow

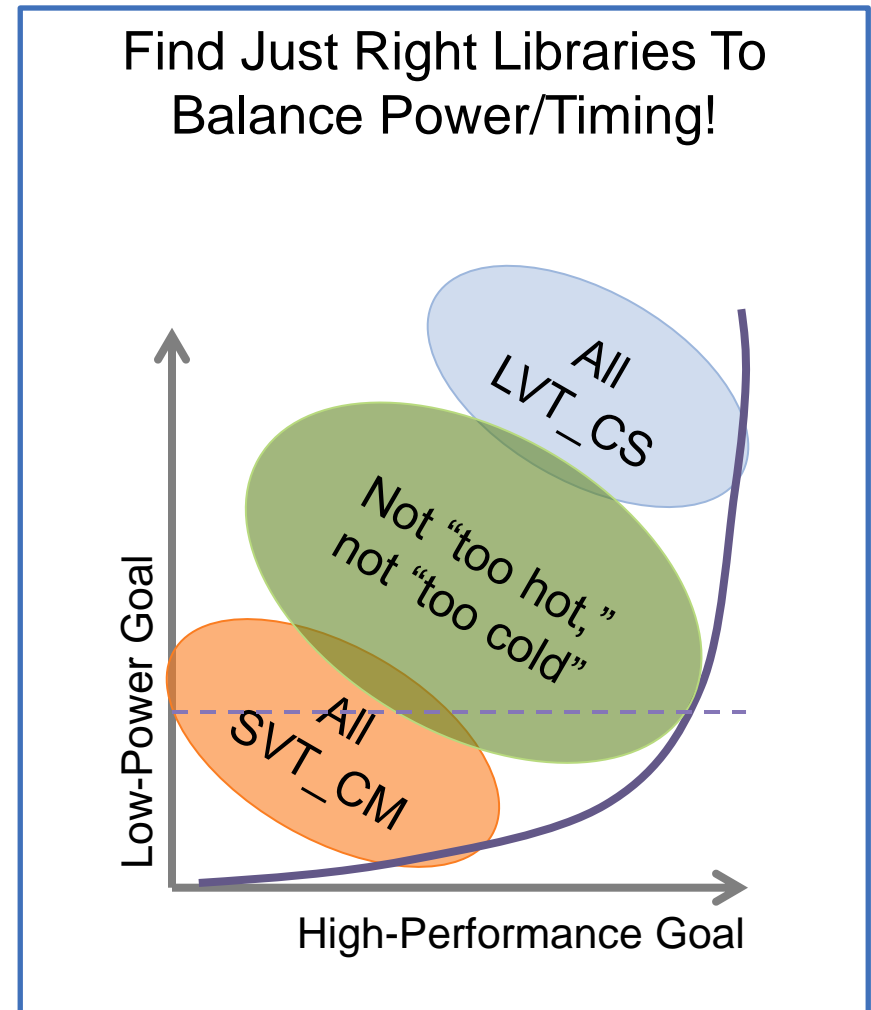
## For a Quad-Core Cortex-A7 Processor



# First, Check the Library Limits

*Then Adjust The Flow To Meet The Power Target*

- Explore the libraries
- Use Synopsys Physical Guidance (SPG) for best DC-G/ICC correlation:
  - `compile_ultra -spg`
- Manage cell density  
`placer_max_cell_density_threshold`
- Reserve 20-30% of power budget for:
  - DC-G/ICC correlation
  - Clock tree insertion
  - Pre/post-route correlation



# Manage Power: target\_library

SVT\_CM = "**Base**" Library, Balances Power/Timing

target_library	UHVT CM	SVT CL	HVT CM	SVT CM	LVT CL	SVT CS	LVT CM	LVT CS
compile_ultra				✓				
place_opt				✓		✓*		
clock_opt_cts	✓		✓	✓		✓	✓	✓**
clock_opt_psyn				✓		✓		
clock_opt_rt				✓		✓		
route	✓		✓	✓				
route_opt	✓		✓	✓				
focal_opt			✓	✓		✓	✓	

\* `set_multi_vth_constraint -lvth_percentage 15 -cost area \`  
`-type hard -lvth_groups {stdcell_9t_rvt_CS}`

\*\* Use lvt\_CS for minimum insertion delay (clock OCV: 8% setup, 14% hold)




# Manage power: clock\_uncertainty

*Timing target impacts power*

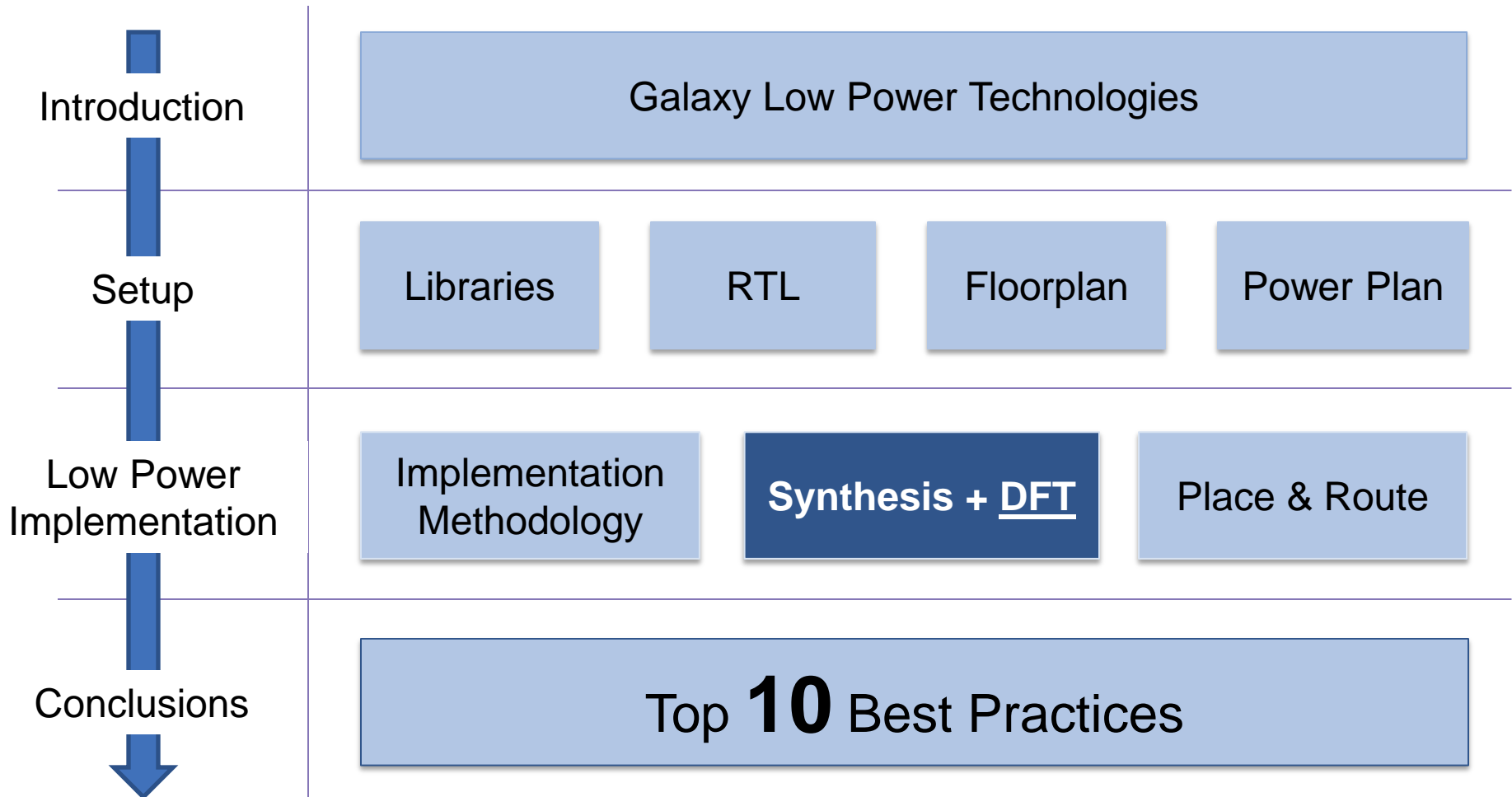
- Do NOT over-constrain in synthesis
  - impacts power
- Relax hold uncertainty for pre-route fix\_hold
  - saves power prior to route
- Use signoff constraints throughout
  - avoids pessimism
- Signoff uncertainty used for both cpu & non\_cpu

Cortex-A7 CPU & NON_CPU Flow Step	Clock Setup Uncertainty (ps)	
	setup	hold
compile_ultra	50	-
place_opt	50	-
clock_opt	50	-50
route_opt	50	50
focal_opt	50	50
signoff	50	50



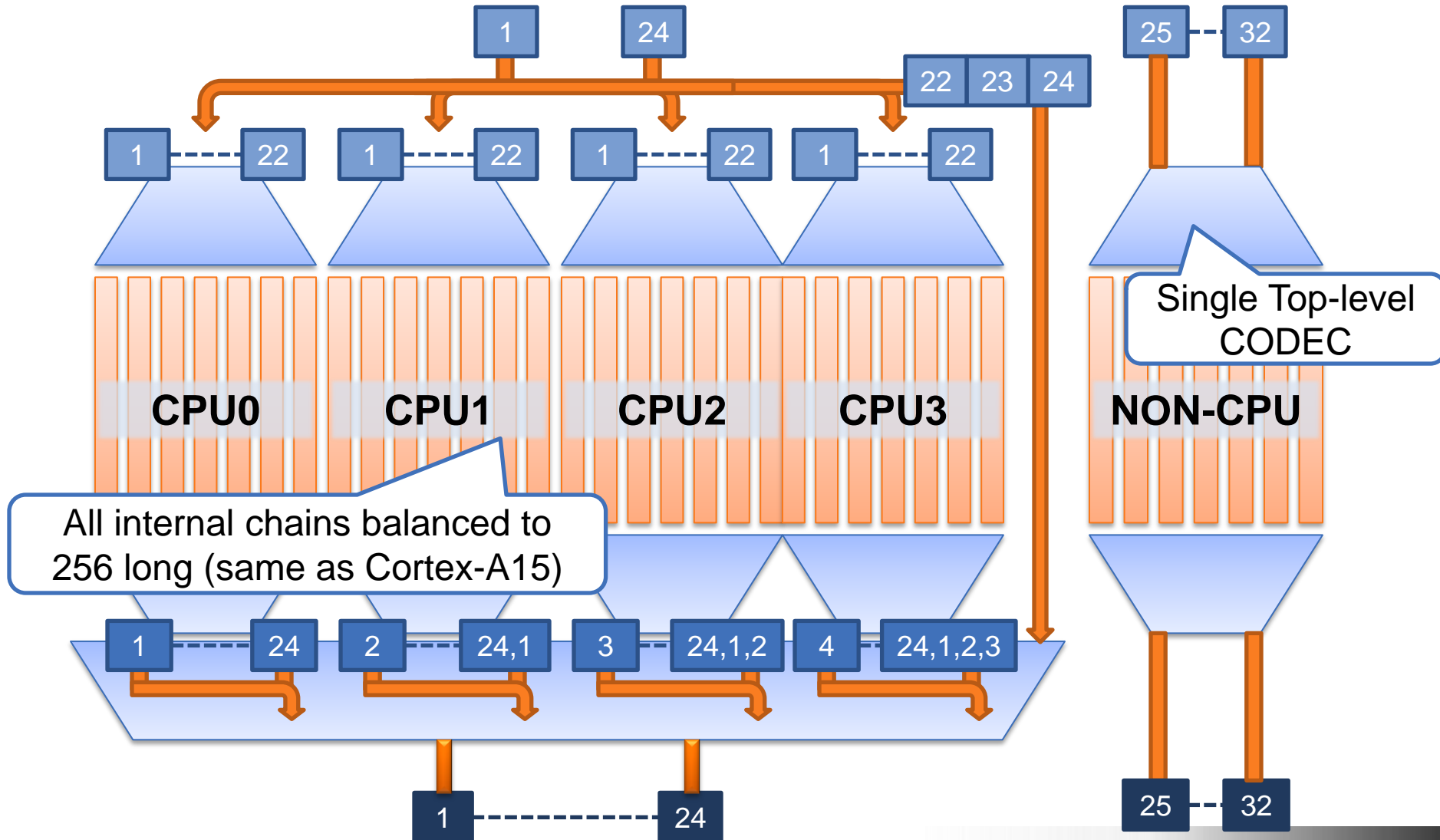
# Power-centric Timing Opt. Flow

## For a Quad-Core Cortex-A7 Processor



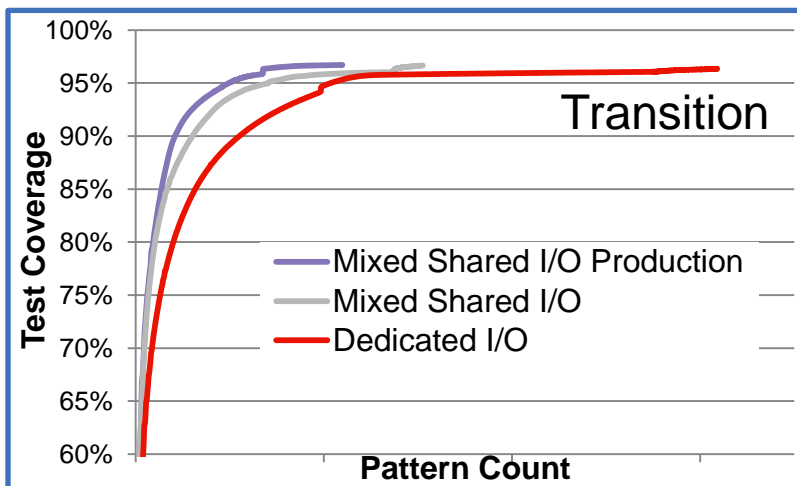
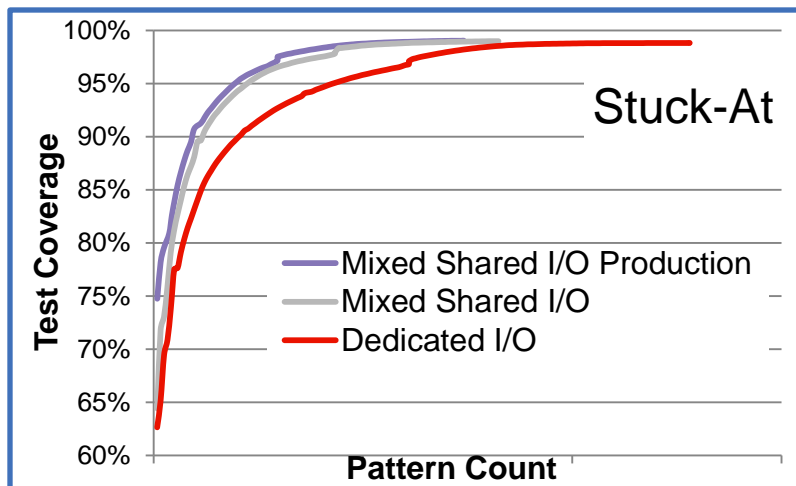
# DFT Strategy - Scan Compression

## Mixed Shared/Non-Shared CODEC I/O Architecture



# TetraMAX<sup>®</sup> ATPG Results

*Improved Coverage, Reduced Tester Time*



Pattern Count Reduction Compared to Dedicated I/O	Mixed Shared I/O
Transition	55%
Stuck-At	52%

**Reduces pattern  
count by >50%**

Test Time Reduction Compared to Legacy Scan	Dedicated I/O	Mixed Shared I/O	Mixed Production
Transition	16X	36X	73X
Stuck-At	18X	37X	43X

**Improves TATR  
up to 4.5X**

**Note:**

Baseline ATPG patterns: `run_atpg -auto`, production ATPG patterns: `run_atpg -optimize_patterns`

# DFT-aware Synthesis

## *Power and Congestion-aware DFT/ATPG*

### Timing constraints sensitive to DFT architecture

- RTL: Add DFT ports for ease of verification in Formality
- SDC: Constrain CODEC feedthroughs to avoid false timing violations
- MCMM: Test Mode STA scenarios needed as a minimum
  - Slow-speed shift
  - At-speed capture
    - provides timing data for ATPG within TetraMAX

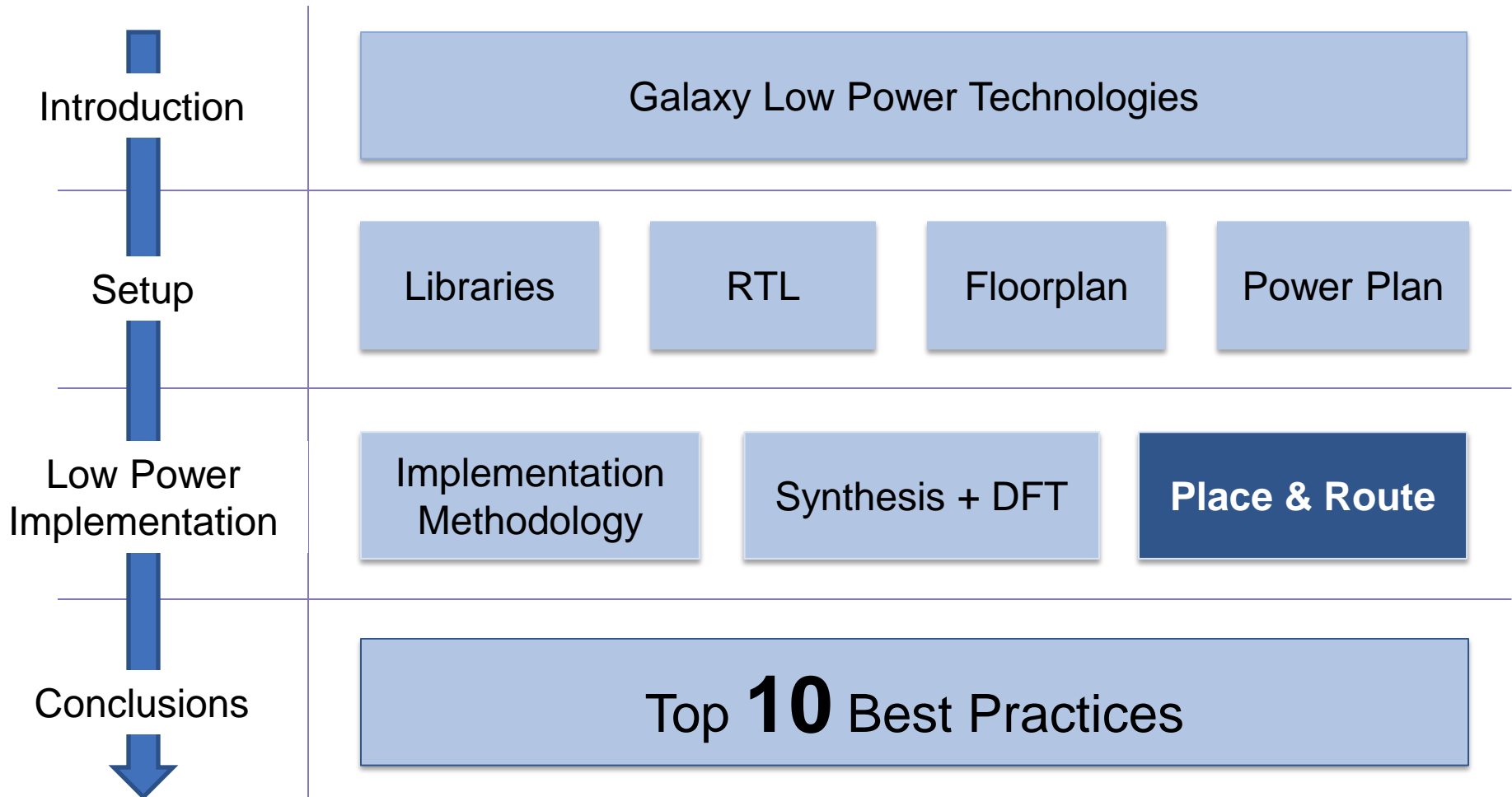
### Benefits of Synthesis-Based Test

- Concurrent handling of timing/power/area & physical effects
- Compression logic optimized to remove **congestion** and chains re-ordered during incremental compile to improve ICC correlation
- **UPF power-intent** encompasses test signals to ensure DFT insertion 'correct by construction'

**Power-Aware ATPG** defines switching activity budgets to test within functional conditions, avoiding false failures on tester due to ground bounce

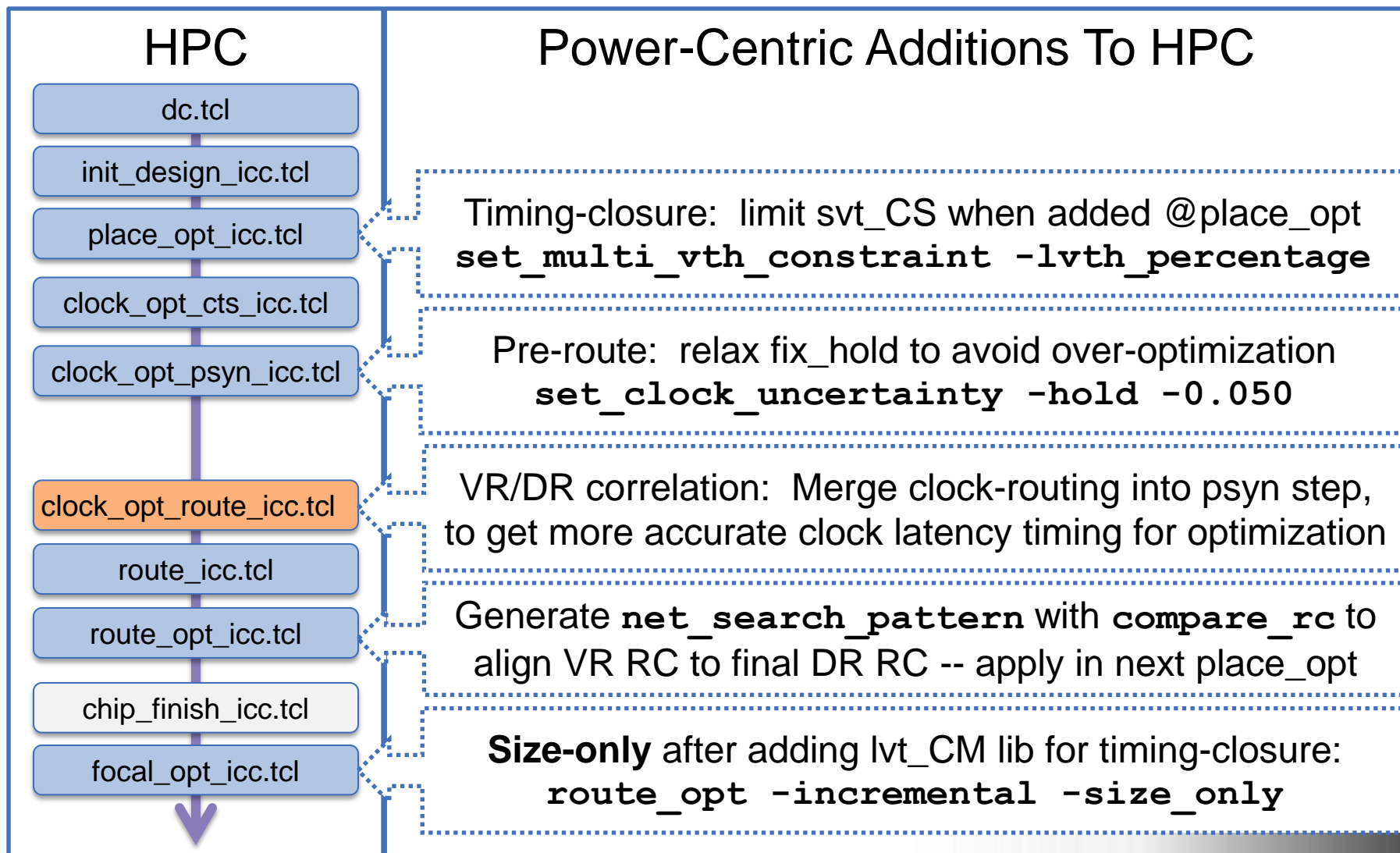
# Power-centric Timing Opt. Flow

## For a Quad-Core Cortex-A7 Processor



# Summary of Power-centric P&R

## *Micromanage Power (By Limiting Timing Opt.)*



# Upgrade To Latest Tool Release For Best Power Results

- Power optimization enhanced in IC Compiler 2013.03
  - Exact same starting init\_design\_icc.CEL
  - Exact same script, no changes

cortexa7core	Frequency	TNS	Area	Leakage
2012.06-SP5	1.00	1.00	1.00	1.00
2013.03-Beta2	1.02	0.6	0.996	0.93

7% CPU Leakage Power Savings In 2013.03



# Add skew\_opt

*skew\_opt helps to/from\_hard\_macro path\_groups*

- Limit skew\_opt to a "few" endpoints for best results

```
set paths [get_timing_paths -slack_lesser_than -0.010 -max_paths 10000]
append_to -unique skew_opt_pins [get_attr $paths startpoint]
append_to -unique skew_opt_pins [get_attr $paths endpoint_clock_pin]
echo Applying skew_opt on [sizeof $skew_opt_pins] pins ...
skew_opt -no_auto_source -resolution 0.005 -pins $skew_opt_pins -output
```

log: Applying skew\_opt on **8353 pins** ...

cortexa7core	Frequency	TNS	Area	Leakage
w/o skew_opt	1.00	1.00	1.00	1.00
w/ skew_opt	<b>1.02</b>	<b>0.17</b>	<b>0.99</b>	<b>0.98</b>

**Better timing, area, AND power!**

# Manage Pre/Post-Route Correlation

*Use compare\_rc to Generate net\_search\_pattern*

```
set enable_net_pattern_rc_scaling TRUE
create_net_search_pattern -net_length_upper_limit 2
set_net_search_pattern_delay_estimation_options \
  -max_horizontal_capacitance_scaling_factor 1.12564 \
  -max_vertical_capacitance_scaling_factor 1.12564 \
  -max_horizontal_resistance_scaling_factor 1.00355 \
  -max_vertical_resistance_scaling_factor 1.00355 \
  -via_count_scale 0.801555 \
  -via_resistance 0.00201186\
  -pattern 1

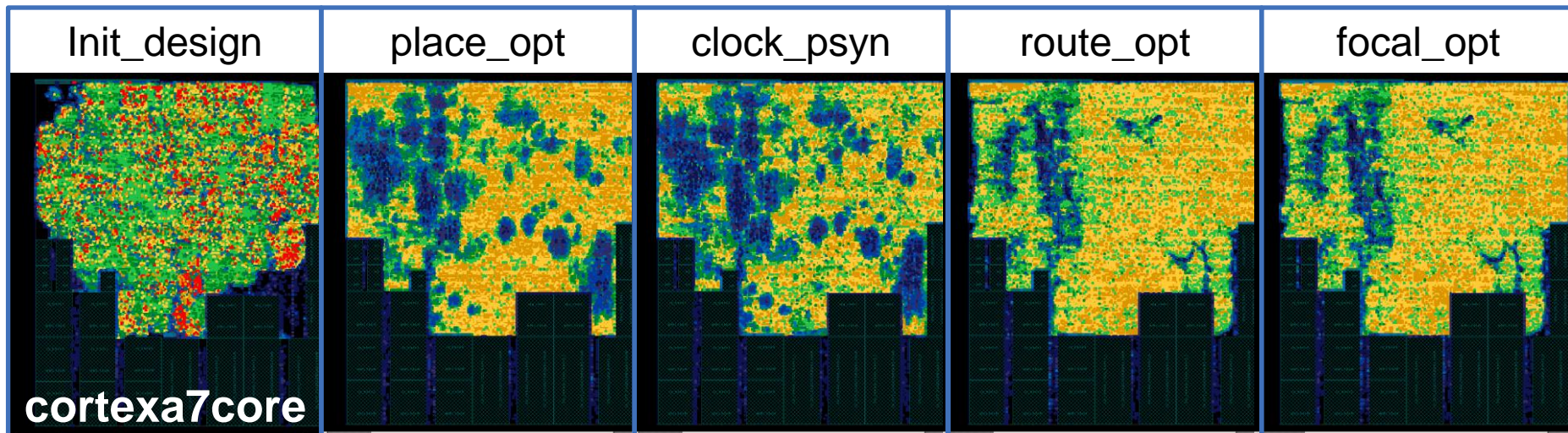
...net_length_upper_limit: 5, 10, 20, 50, 100, 200, 500
```

Net patterns avoid over-constrained timing, for low-power

# Manage Cell Density for Low Power

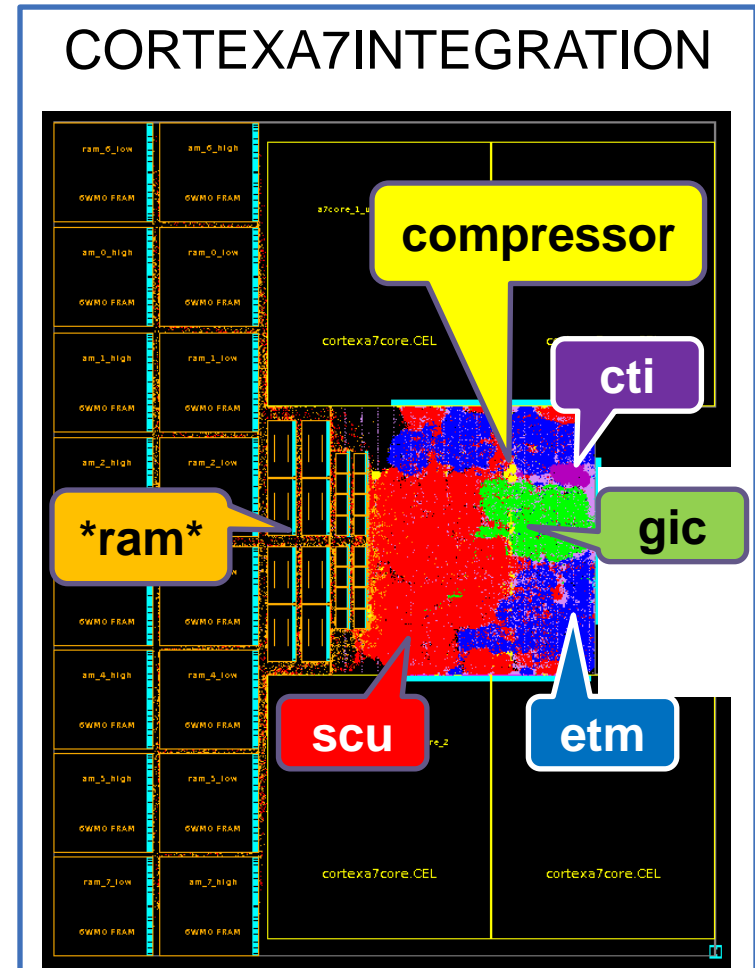
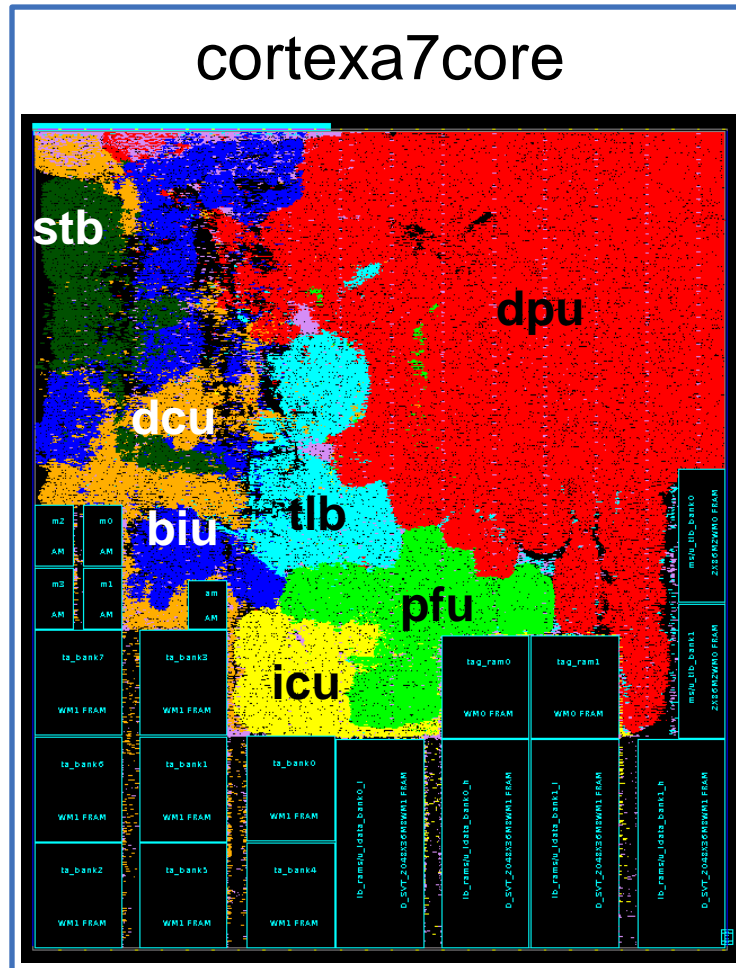
## *Target Consistency Across Flow*

- Check cell density as early as DC-G
  - `restore_spg_placement` shows placement at `init_design` step
  - `"-congestion"` is default in `compile_ultra`
- Manage cell density, because it also impacts power/area
  - `set_placer_max_cell_density_threshold 0.7`
  - `set_congestion_options -max_util 0.8 -coord $score`

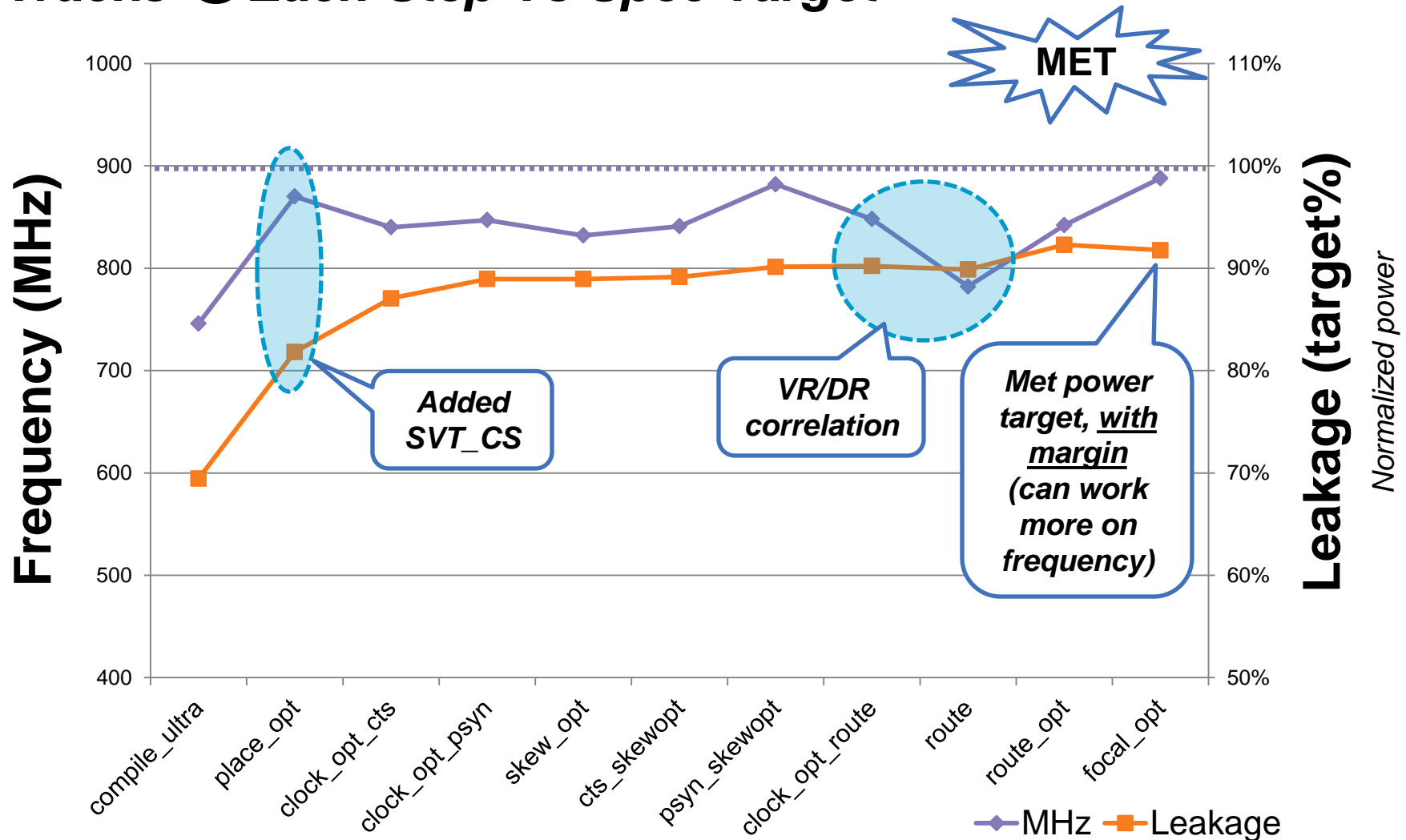


# Quad-Core Cortex-A7 Processor

## Low-Power Results Address Target Market



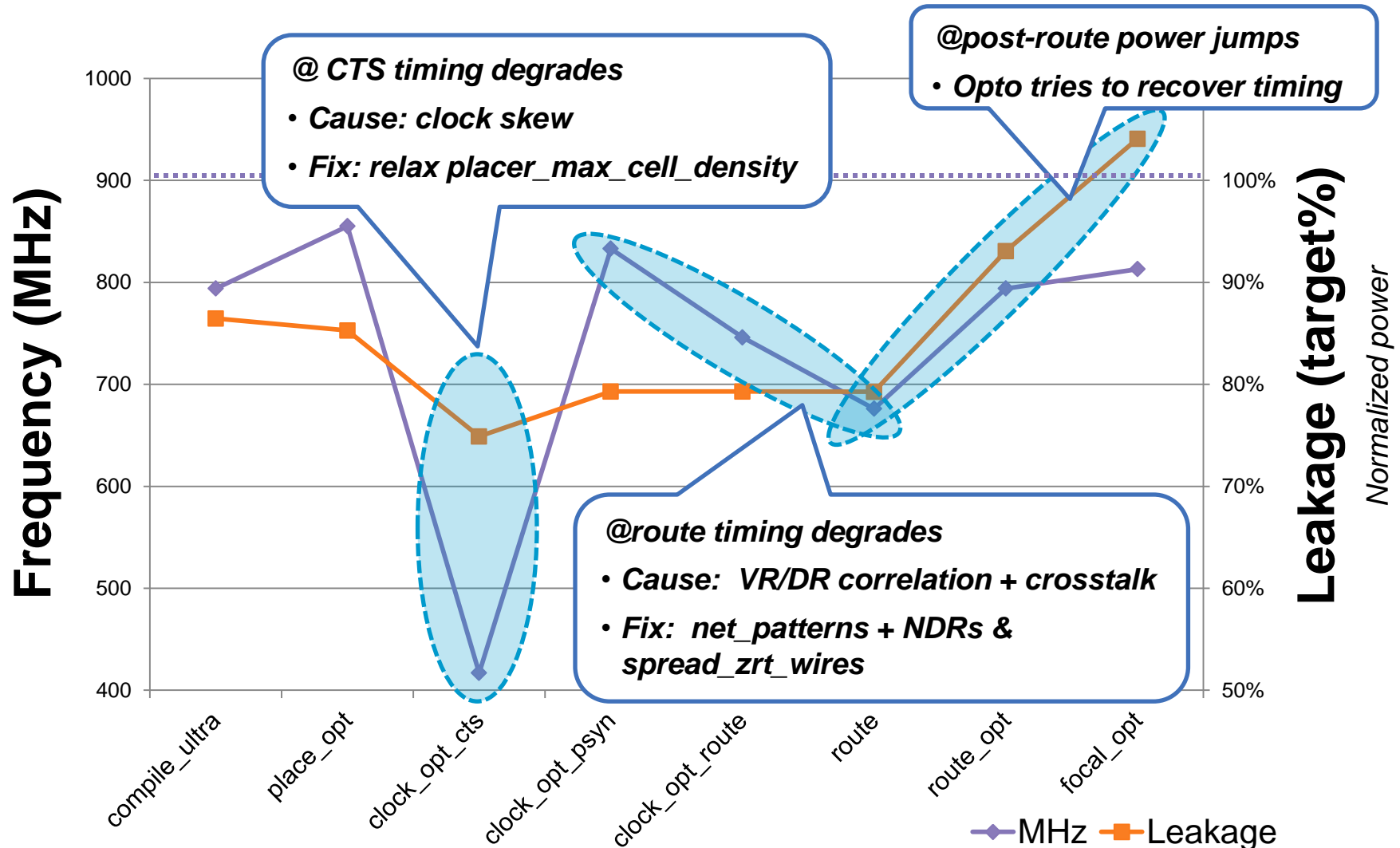
# CPU Has Consistent Power/Timing Tracks @ Each Step Vs Spec Target



OpCond: ssg\_typical\_max\_0p81v\_0c + cworst.tluplus

# NON\_CPU - Intermediate Results

## Debugging "Jumps" in Power/Timing

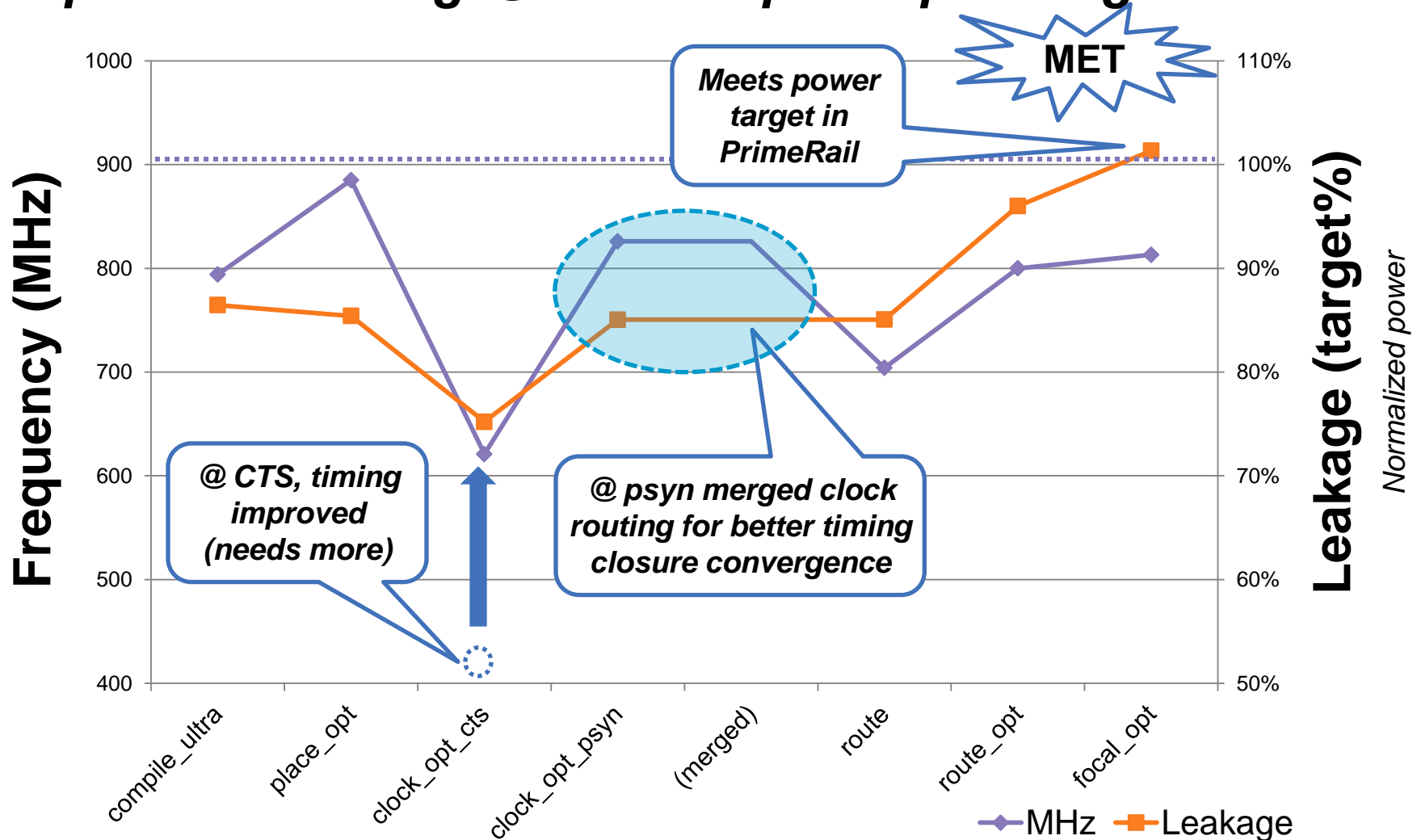


OpCond: ssg\_typical\_max\_0p81v\_0c + cworst.tluplus



# NON\_CPU - Power/Timing Improved

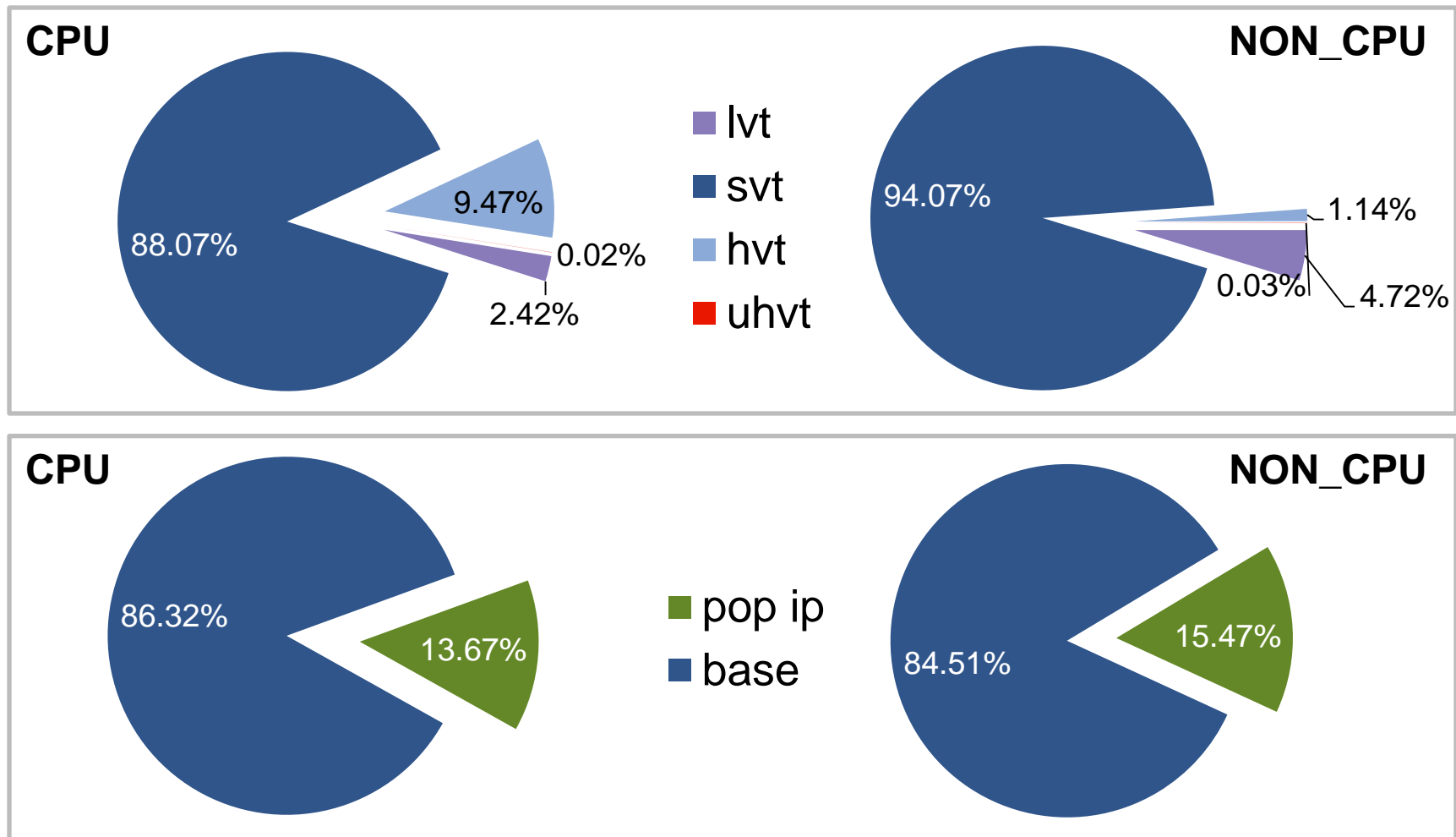
*Improved Tracking @ Each Step Vs Spec Target*



OpCond: ssg\_typical\_max\_0p81v\_0c + cworst.tluplus

# SVT Balances Power/Timing

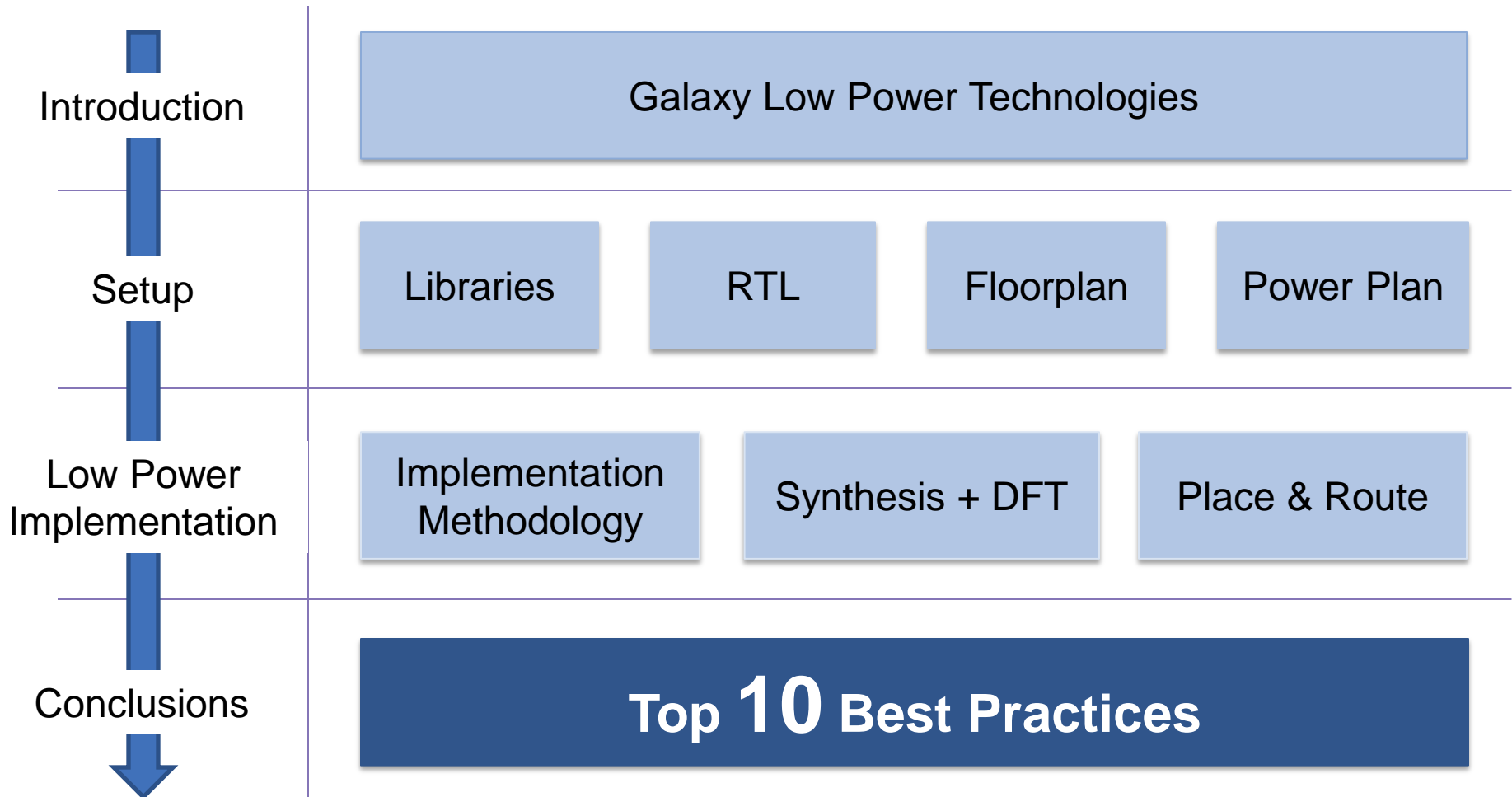
*POP IP Libraries Usage At ~15%*





# Power-centric Timing Opt. Flow

## For a Quad-Core Cortex-A7 Processor



# Top 10 Best Practices

## *For A Power-Centric Implementation*

1 Optimize for power FIRST, starting at synthesis

2 Manage for power at each step, adjust timing target

- `target_library, -lvth_percentage -type hard`

3 Do NOT over-constrain timing

- `set_clock_uncertainty -hold -0.050`

4 Focus on correlation - no power/timing "jumps"

- DCG/ICC: use `compile_ultra -spg; place_opt -spg`
- Pre/post-route (VR/DR) in ICC
  - `compare_rc -net [<net length>]; net_search_pattern`

5 Manage cell density, because it impacts power/area

- `set placer_max_cell_density_threshold 0.7`
- `set_congestion_options -max_util 0.85 -coord $score`
- Use of `-congestion` has power impact, use only if needed

# Top 10 Best Practices

## *For A Power-Centric Implementation*

### 6 Size-only after introducing new Vt libs

- `route_opt -incremental -size_only`

### 7 Use -power & enable power-aware (avoid script errors)

- `set icc_preroute_power_aware_optimization true`
- `set_route_opt_strategy -power_aware_optimization true`

### 8 Use same low power target\_lib technique in PT/ECO

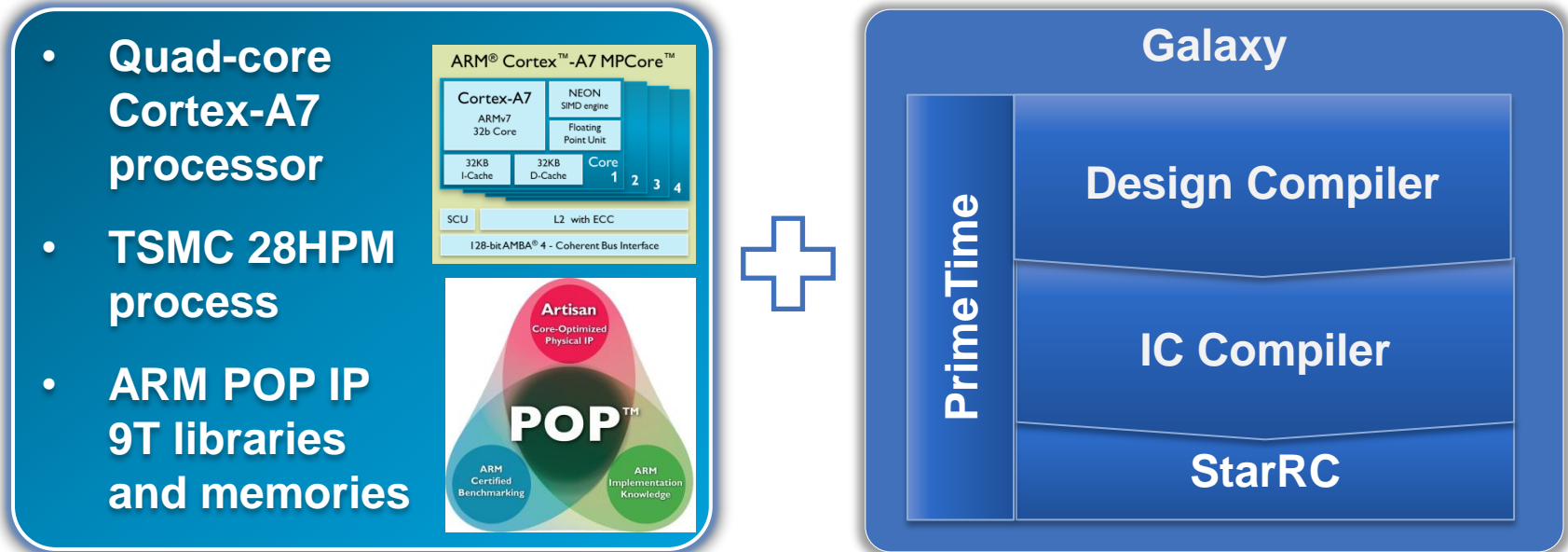
- Otherwise, `fix_eco_timing` may degrade power

### 9 Iterative design-closure process

- Good timing => good power & good power => good timing

### 10 Power-centric flow is possible!

# ARM + Synopsys Collaboration



High Performance Core (HPC) scripts + Low-Power Experience

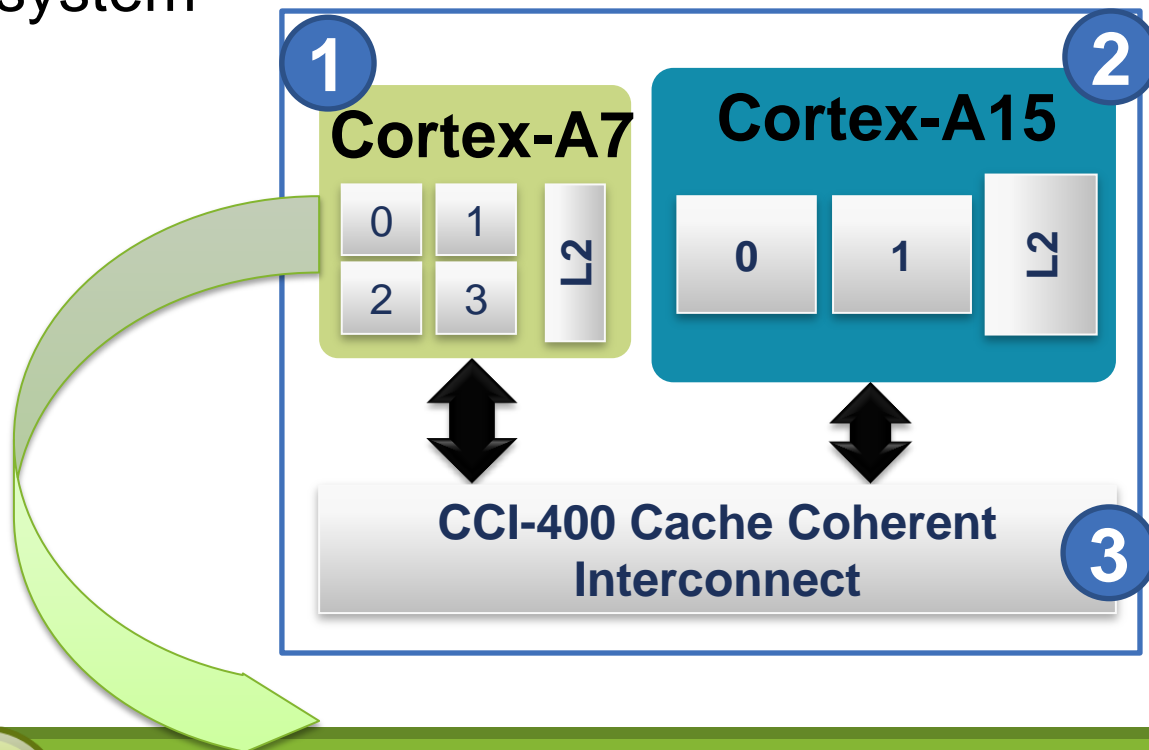


**Reference Implementation** for an ARM Cortex-A7 Processor  
Optimized for low power and performance  
Available Through SolvNet To Joint Customers Today!

# Reference Implementation

## *Collateral & Availability (1/2)*

- Available for key components of the ARM big.LITTLE system



**Reference Implementation** for the ARM Cortex-A7 Processor  
Your best starting point for optimized implementation!

# Reference Implementation

## *Collateral & Availability (2/2)*

- ARM & Synopsys joint customers can download RI scripts & documentation from

[www.synopsys.com/ARM-Opto](http://www.synopsys.com/ARM-Opto)

- For other processor cores, contact Synopsys technical support to help you configure and deploy HPC scripts
- For further optimization and customization support contact Synopsys Professional Services



**Reference Implementation** for the ARM Cortex-A7 Processor  
Your best starting point for optimized implementation!

# High-Perf. Core Implementation

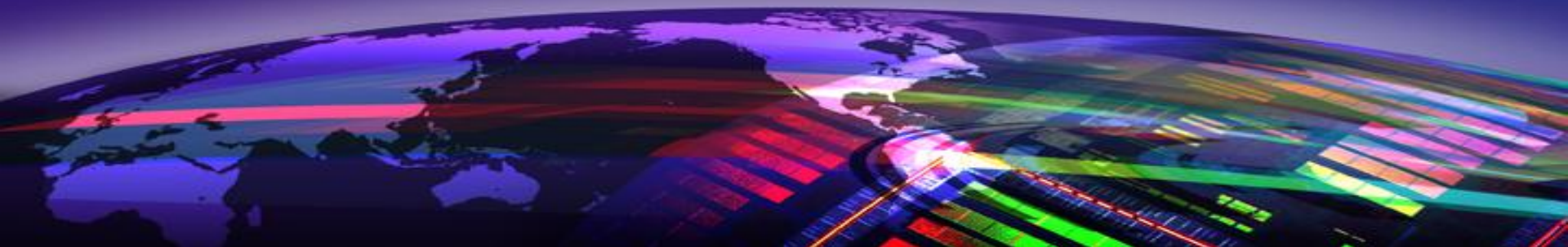
*Sessions of Interest - Tuesday, March 26<sup>th</sup>*



Presenters	Time	Session
Synopsys Lunch & Learn	12:00 PM to 1:30 PM	1. Optimization Exploration of ARM <sup>®</sup> Cortex <sup>™</sup> Processor-Based Designs with the Lynx Design System
ARM & Synopsys Joint Tutorial	1:30 PM to 3:30 PM	2. Power-centric Timing Optimization of an ARM <sup>®</sup> Cortex <sup>™</sup> -A7 Quad Core Processor 3. Engineering Trade-Offs in the Implementation of a High Performance ARM <sup>®</sup> Cortex <sup>™</sup> -A15 Dual Core Processor
Broadcom MediaTek Samsung STMicroelectronics Customer Panel	4:15 PM to 5:15 PM	4. Achieving Optimum Results on High Performance Processor Cores

# Thank You

**SYNOPSYS®**  
Accelerating Innovation





# Q&A

# Acronyms

CTS:	Clock Tree Synthesis	MCM:	Multi-Corner Multi-Mode
DCG:	Synopsys Design Compiler Graphical	MIM:	Multiply Instantiated Module
DR:	Detail Route	OCV:	On-Chip Variation
DRC:	Design Rule Check	POP:	Processor Optimization Pack (ARM POP IP library)
DVFS/AFS:	Dynamic Voltage and Frequency Scaling	PT:	Synopsys Primetime
FCI:	Fast Cache Instance (ARM POP IP RAMs)	PTSI:	Primetime SI (timing analysis with xtalk & noise)
FSLR:	Final Stage Leakage Recovery	PTPX:	Primetime PX (gate-level power analysis)
HPC:	Synopsys High Performance Core (scripts)	RI:	Reference Implementation
HVT:	High Threshold Voltage	RM:	Synopsys Reference Methodology
ICC:	Synopsys IC Compiler	SAIF:	Switching Activity Interchange Format
ILM:	Interface Logic Model	SPG:	Synopsys Physical Guidance (synthesis => place)
iRM:	ARM's Implementation Reference Methodology	SVT/RVT:	Standard Threshold Voltage
LCRM	Synopsys Lynx Compatible Reference Methodology	TATR:	Test Application Time Reduction
LPP:	Low Power Placement	VCD:	Value Change Dump
LVS:	Layout vs Schematic	VR:	Virtual Route
LVT:	Low Threshold Voltage	Vt:	Threshold Voltage