Refinable Macros and Terminal Boundaries in UPF 4.0: Empowering Soft IPs of the Future

Amit Srivastava
Amit.Srivastava@synopsys.com
Synopsys Inc

John Decker
jdecker@cadence.com
Cadence Design Systems

Lakshmanan Balasubramanian
lakshmanan@ieee.org
Texas Instruments (India) Pvt. Ltd

***Abstract*: Systems on Chips (SoCs) increasingly integrate numerous Soft Intellectual Properties (SIPs), each with its own power management architecture. Traditional methods often fail to comprehensively capture the power intent of these SIPs, leading to challenges during integration and risking verification integrity. This paper introduces Refinable Macros in the upcoming UPF 4.0 standard, providing enhanced capabilities for non-intrusive refinement of SIPs' power intent without altering original designs. Refinable Macros allow IP integrators to add implementation-specific details while preserving the extensive validation performed at earlier stages. Additionally, the importance of marking SIPs as terminal boundaries is highlighted to ensure consistent interpretation across design tools and facilitate efficient verification and implementation. The paper discusses these new features, implementation UPF commands, and best practices, showcasing advancements that expedite development and improve efficiency in low-power SoC design.**

I. INTRODUCTION

The rapid advancement of semiconductor technology has led to increasingly complex System on Chips (SoCs) that integrate numerous Soft Intellectual Properties (SIPs) as standalone blocks, each often containing its own power management architecture. Managing the overall power intent of such SoCs presents significant challenges. SIPs are typically provided with their own Unified Power Format (UPF) files, encapsulating power intent in a technology-independent manner, enhancing flexibility and reusability across different technology nodes and design environments.

However, integrating these SIPs into larger designs often requires adding implementation-specific details during stages like synthesis and place-and-route (PnR), which can risk verification integrity if not handled carefully. The existing UPF 3.1 standard [1], which defines the Successive Refinement methodology for bottom-up implementation flows, lacks defined semantics for bottom-up verification flows involving SIPs. Additionally, the soft macros defined in UPF 3.1 are intended for bottom-up implementation flows but have restrictions that hinder necessary refinements from the SoC level, compromising optimization and flexibility.

As a result, designers are often compelled to adopt intrusive methods to alter the original power intent to proceed with implementation, undermining the verification performed at the IP level. This situation forces either re-validation of the IPs, which is time-consuming and counteracts the goal of achieving faster turnaround times, or accepting the risk of introducing silicon bugs into the final product, compromising the reliability and functionality of the SoC.

To address these challenges, UPF 4.0 [2] introduces Refinable Macros, providing enhanced capabilities for non-intrusive refinement of SIPs' power intent without altering the original designs. Refinable Macros enable SIPs to maintain their validation integrity while allowing IP integrators to efficiently refine the power intent with implementation-specific details. By applying UPF's Successive Refinement methodology, integrators can seamlessly enhance the IP power intent without intrusive modifications.

This work builds upon the ideas presented in our earlier paper, where we introduced the concept of Verification Macros and their role in maintaining the integrity of verifiable IP UPF through integration [3]. While the earlier work focused on addressing the challenges within the constraints of UPF 3.1, this paper highlights how the UPF 4.0 standard has evolved to systematically address these limitations, introducing concepts like Refinable Macros and enhanced Terminal Boundaries to better support SIP integration and verification.

This paper provides an in-depth analysis of limitations in UPF 3.1 for SIP integration and introduces key advancements in UPF 4.0. Section II identifies challenges with SIP integration using UPF 3.1. Section III elaborates on Refinable Macros and their significance in preserving verification integrity. Section IV discusses implementation UPF commands with best practices. Section V highlights the role of Terminal Boundaries in ensuring tool consistency, and Section VI provides case studies illustrating these concepts in practical scenarios. The paper concludes with the benefits of UPF 4.0 adoption for efficient and reliable SoC design.

II. LIMITATIONS OF UPF 3.1 IN SOFT IP INTEGRATION

In designing complex System on Chips (SoCs), managing power intent efficiently is crucial due to the integration of numerous Soft Intellectual Properties (SIPs), each with its own power management requirements. The Unified Power Format (UPF) 3.1 provides the Successive Refinement methodology to handle power intent specifications hierarchically and incrementally.

A. Successive Refinement Methodology in UPF 3.1

The Successive Refinement methodology allows designers to develop power intent progressively as the design evolves from individual IP blocks to the complete SoC. This approach facilitates collaboration between IP providers and SoC integrators by enabling the reuse of power intent specifications across different design stages and abstraction levels.

In this methodology:

- **Constraint UPF**: IP providers define a Constraint UPF that outlines the fundamental power architecture of the IP block, including definitions of power domains, retention requirements, isolation clamping requirements, and legal power states, all specified in a technology-independent manner.
- **Configuration UPF**: As the design progresses, a Configuration UPF incorporates additional details, adapting the power intent to the specific context of the SoC, such as adjusting power management strategies to align with system-level requirements.
- **Implementation UPF**: Finally, the Implementation UPF adds technology-specific information required for physical implementation, such as power switches, supply nets, and control logic for power management cells.

Figure 1 illustrates the Successive Refinement methodology as defined in UPF.

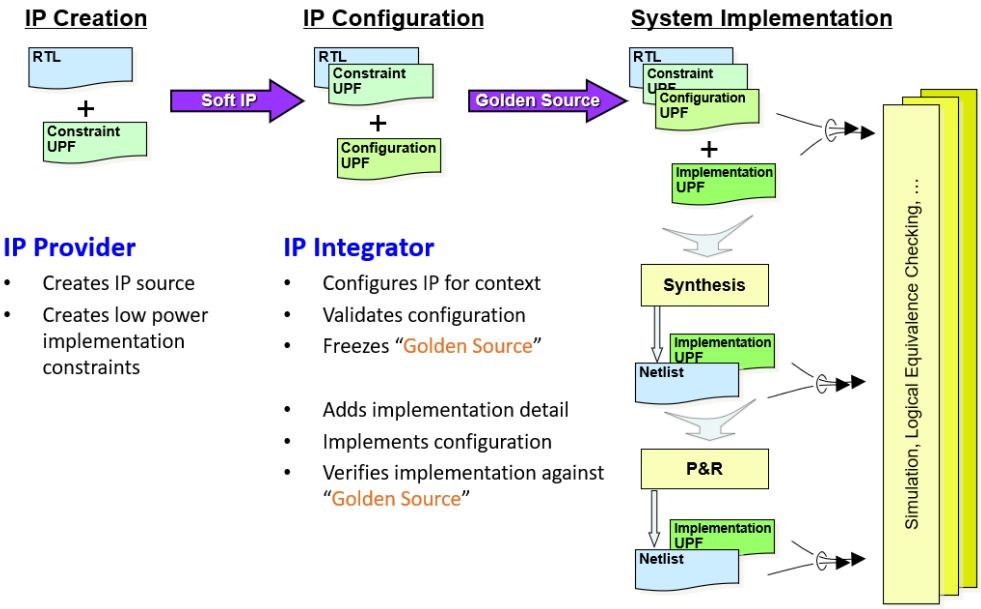

Figure 1 UPF Successive Refinement Methodology

While this methodology promotes modularity and reuse, it primarily supports bottom-up implementation flows and lacks support for bottom-up verification flows involving SIPs.

B. Bottom-Up Implementation Flows and Soft Macros

In a **bottom-up implementation flow**, individual IP blocks are designed and implemented separately before being assembled into a larger subsystem or SoC. This approach allows IP blocks to be developed and optimized independently, leveraging their standalone power intent specifications.

To support this flow, UPF 3.1 introduces **Soft Macros**. A Soft Macro represents an IP block along with its associated UPF, intended to be implemented independently from the rest of the design hierarchy. Soft Macros encapsulate the power intent of the IP block, forming a terminal boundary that isolates the internal power intent from external modifications.

Features of Soft Macros:

- **Self-Contained Power Intent**: The UPF for a Soft Macro fully specifies the power intent within the IP block without relying on power intent definitions from parent or ancestor scopes.
- **Terminal Boundary Behavior**: The ports of a Soft Macro are treated as drivers and receivers, and the power intent outside the macro does not affect the internal power architecture.
- **Consistent Interpretation**: Verification tools interpret the UPF of a Soft Macro in the same way as when the IP was implemented separately, ensuring consistent behavior across different design contexts.

Advantages of Using Soft Macros:

- **Independent Implementation**: Allows IPs to be implemented independently, facilitating modular design and enabling IP vendors to optimize their blocks without dependencies on the SoC environment.

- **Protection of IP Integrity**: Ensures that the IP's internal power management strategies are not altered unintentionally during integration.
- **Consistent Verification Across Tools**: Reduces the risk of discrepancies between verification and implementation by enforcing consistent semantics across different design tools.

Example Scenario:

Figure 2 demonstrates the semantics of Soft Macros in a bottom-up implementation flow.

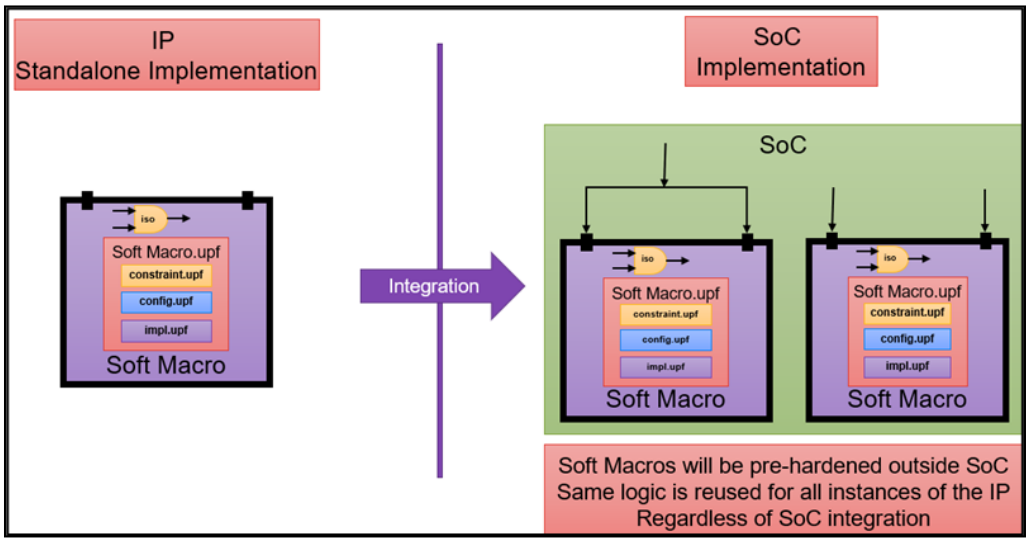

Figure 2. Soft Macro in Bottom-up Implementation Flow

In the figure:

- A Soft Macro IP has two supply pins on its interface. The UPF is written to insert isolation cells within the IP from a source powered by one supply to a sink powered by another supply.
- Since the IP is implemented separately, the isolation cells are present in the netlist.
- When the IP is integrated into a larger block, there may be multiple instances of the IP. In one instance, the two supplies at the interface are connected, making the isolation cells redundant.
- By marking the IPs as Soft Macros, verification tools can identify the IP boundaries and treat them as terminal boundaries, preserving the redundant isolation cells even if the supplies are connected.

Note: The example of redundant isolation cells is just one scenario. Many UPF semantics can be affected by the environment, so it is important to mark these blocks as Soft Macros to maintain consistent behavior across different design contexts.

C. Limitations in Bottom-up Verification Flows

In a **bottom-up verification flow**, SIPs are verified independently but implemented within a larger system context. The entire SoC, including the SIPs, undergoes implementation and verification as a whole, allowing for optimization across the design hierarchy and leading to better overall performance, power efficiency, and area utilization.

However, applying Soft Macros to SIPs in bottom-up verification flows presents significant limitations:

- **Inability to Refine Power Intent**: Soft Macros prevent modifications to the IP's power intent from the parent scope, inhibiting necessary implementation-specific refinements.
- **Constraints on Optimization**: They require standalone implementation, restricting system-level optimizations and potentially limiting performance and area efficiency.
- **Conflict with System-Level Requirements**: The rigid isolation may not align with system-level power management strategies, leading to suboptimal integration.
- **Risk to Verification Integrity**: Without non-intrusive refinement capabilities, designers may resort to modifying the original UPF, compromising the IP's verified power intent.

Thus, Soft Macros are unsuitable for bottom-up verification flows where flexibility in refining power intent during integration is essential.

D. Challenges in Capturing Power Intent of SIPs

UPF 3.1 lacks suitable constructs for capturing the power intent of SIPs in bottom-up verification flows, leaving designers with inadequate options:

1. **No Marking for SIPs**:
    - **Approach**: Treating the SIP's UPF like any other block, without special protection.

- **Consequences**: Lacks mechanisms to ensure the IP's power intent integrity, requiring integrators to modify the UPF and potentially necessitating re-validation.

2. **Using Soft Macros for SIPs**:
   - **Approach**: Marking SIPs as Soft Macros imposes strict restrictions.
   - **Challenges**: Prevents necessary refinements and optimizations at the SoC level due to the inability to modify the IP's power intent.

Neither approach adequately balances maintaining verification integrity with the flexibility needed for system-level optimizations.

E. Need for Enhanced Methodologies

These challenges highlight the need for methodologies that:

- **Allow Non-Intrusive Refinement**: Enable integrators to add implementation-specific details without altering the original IP's UPF.
- **Preserve Verification Integrity**: Ensure refinements do not compromise the IP's standalone verification.
- **Facilitate System-Level Optimization:** Align the IP's power intent with system-level strategies.
- **Support Bottom-Up Verification Flows**: Provide tool support and defined semantics for effective handling.

UPF 4.0 addresses these needs by introducing **Refinable Macros**, which will be discussed in the next section.

III. REFINABLE MACROS IN UPF 4.0

To address the limitations of soft macros in bottom-up verification flows, UPF 4.0 introduces **Refinable Macros,** providing the necessary flexibility for integrating SIPs into larger systems while preserving their original power intent and verification integrity.

A. Definition and Purpose of Refinable Macros

Refinable Macros are a new construct in UPF 4.0 that form a **refinable terminal boundary**. This boundary allows SoC integrators to add implementation-specific refinements to the IP's power intent during integration without altering the original UPF provided by the IP vendor.

Key Attributes of Refinable Macros:

- **Non-Intrusive Refinement**: Implementation details can be added without modifying the original power intent defined in the IP's UPF, avoiding the need for re-verification of the IP.
- **Preservation of Verification Integrity**: The IP's standalone verification remains valid, reducing the risk of functional discrepancies.
- **Flexibility for System-Level Optimization**: Enables alignment of the IP's power intent with system-level power management strategies, allowing optimizations such as removing redundant logic when it's safe to do so, improving performance and area utilization.

B. Marking IPs as Refinable Macros

IPs can be marked as Refinable macros using **`set_design_attributes`** command.

```
set_design_attributes -models SIP \
  -attribute {UPF_is_refinable_macro TRUE}
                                    OR
#Marks the Model of current scope as Refinable Macro
set_design_attributes -models {.} \
  -is_refinable_macro
```

These commands mark the IP (referred to as SIP or the current scope .) as a Refinable Macro by setting the **`UPF_is_refinable_macro`** attribute to **`TRUE`**.

C. Practical Application

Figure 3 illustrates the use of Refinable Macros in a bottom-up verification flow.

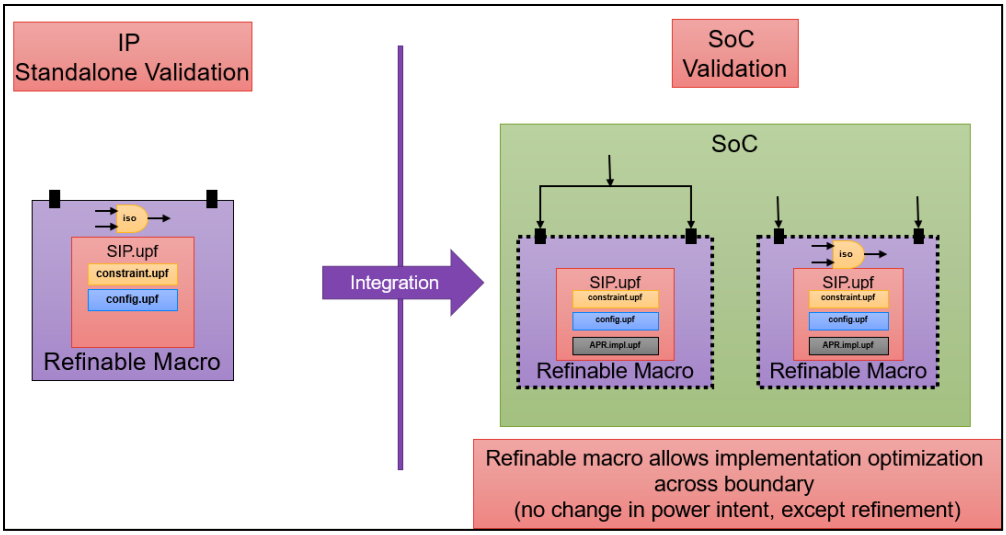

Figure 3. Refinable Macro in a Bottom-up Verification Flow

In Figure 3:

- The IP is marked as a Refinable Macro, enabling the implementation tool to make necessary refinements during integration.
- For instance, if the two supplies of the IP are connected in the larger system, redundant isolation cells can be safely removed to optimize area without violating the IP's original power intent.
- The original power intent remains intact, ensuring that the IP's standalone verification is still valid.

D. Advantages over Soft Macros

Refinable Macros overcome the limitations of Soft Macros by allowing:

- **Implementation-Specific Refinements**: Integrators can add details like power switches, supply nets, and control logic necessary for physical implementation without altering the original power intent.
- **Optimized System Integration:** Enables system-level optimizations and refinements, improving performance and area utilization**.**
- **Preservation of Verification Integrity:** Ensures that the IP's verified power intent remains unchanged, maintaining confidence in the IP's correctness.

E. Using Implementation UPF for Refinements

Refinements to the IP's power intent are specified in an **Implementation UPF**. This UPF contains only allowed commands that add implementation details without altering the fundamental power architecture. The details are present in section IV.

F. Ensuring Compliance and Verification Integrity

By combining Refinable Macros with the controlled use of Implementation UPF, designers achieve:

- **Controlled Refinement**: Only safe and permissible changes are made to the IP's power intent.
- **Verification Integrity Preservation**: The IP's original power intent and standalone verification remain unaffected.
- **Tool-Assisted Compliance**: Tools enforce constraints, ensuring refinements adhere to UPF 4.0 standards.

IV. IMPLEMENTATION UPF AND THE LOAD_UPF -IMPLEMENTATION COMMAND

In UPF 4.0, the **Implementation UPF** allows designers to add implementation-specific details to the power intent of SIPs without altering the original power intent defined by the IP provider. This enables safe refinement during integration, maintaining verification integrity while accommodating necessary implementation adjustments.

A. Allowed Commands in Implementation UPF

To ensure refinements do not violate the fundamental power architecture, the Implementation UPF is restricted to specific commands and options. Table 1 lists the UPF commands and options permitted within the Implementation UPF.

Table 1 UPF Commands and options that can be used in Implementation UPF

UPF Commands and required options	Implementation UPF
add_power_state -update	Only for options **-supply_expr**, **-legal**, **-illegal**
create_logic_port	All options
create_logic_net	All options

connect_logic_net	All options except **-reconnect**
create_supply_port	All options
create_supply_net	All options
connect_supply_net	All options
create_abstract_power_source -update	Only for option **-power_switch**
create_power_domain -update	Only for options **-elements**, **-available_supplies**, **-boundary_supplies**, **-define_func_type**
create_power_switch	All options
create_supply_set -update	Only for option **-function**
find_objects	All options
map_power_switch	All options
map_repeater_cell	All options
map_retention_cell	All options
map_retention_clamp_cell	All options
use_interface_cell	All options
set_isolation	All options
set_isolation -update	Only for **-elements**, **-location**, **-isolation_supply**, **-instance**, **-force_isolation**
set_level_shifter	All options
set_port_attributes	All options
set_design_attributes	All options
set_repeater	All options
set_retention -update	Only **-instance**, **-retention_supply**
set_variation	All options
set_correlation	All options
load_upf	All options
set_scope	All options
apply_power_model -update	Only **-port_map**, **-elements**

These commands focus on:

- **Adding Implementation Details:**
  - **Creating and Connecting Ports and Nets**: Commands like `create_logic_port`, `create_supply_net`, and `connect_supply_net` add necessary logic and supply connections for implementation.
  - **Defining Power Switches**: Using `create_power_switch` to specify power switches needed for power gating strategies.
  - **Mapping Constructs to Physical Cells**: Commands such as `map_power_switch` and `map_retention_cell` associate logical power intent elements with physical library cells.
- **Updating Existing Power Intent Objects:**
  - **Refining Power Domains**: With `create_power_domain -update`, designers can update power domains with implementation-specific details like available supplies and elements (e.g., adding Design for Test [DFT] logic).

    - **Adjusting Power States and Supplies**: Using `add_power_state -update` and `create_supply_set -update` to modify power states and supply functions without altering the core power intent.
- **Specifying Cell Insertion Strategies:**
    - **Isolation Strategies**: `set_isolation` and `set_isolation -update` enable the definition or refinement of isolation strategies for signals crossing power domain boundaries.
    - **Level Shifting and Retention**: Define level shifting requirements using `set_level_shifter` and update retention strategies with `set_retention -update`.
- **Control and Configuration:**
    - **Setting Attributes**: `set_design_attributes` and `set_port_attributes` specify attributes affecting implementation without changing the original power intent.
    - **Managing Scope and Loading**: Commands like `set_scope` and `load_upf` assist in managing context and loading additional UPF files as needed.

B. Using the load_upf -implementation Command

To apply the Implementation UPF, the `load_upf` command with the `-implementation` option is used:

```
load_upf <implementation_upf_file> -scope <IP_instance> -implementation
```

This command loads the Implementation UPF for the specified IP instance, enforcing that only permissible refinements are applied.

By restricting the Implementation UPF to these commands and options, UPF 4.0 ensures that refinements are made in a controlled and non-intrusive manner. This approach allows integrators to:

- **Enhance the Power Intent for Implementation**: Add necessary details for physical realization without altering the fundamental power architecture verified by the IP provider.
- **Maintain Verification Integrity**: Ensure that the original power intent remains intact, preserving the validity of the IP's standalone verification.
- **Leverage Tool Support**: Enable tools to enforce constraints and check for compliance, reducing the risk of errors during integration.

V. Importance of Marking IPs as Terminal Boundaries

In complex System on Chip (SoC) designs, accurately managing power intent requires not only specifying power domains and strategies but also ensuring that design tools interpret these specifications consistently across different stages of the design flow. A critical aspect of achieving this consistency is marking Intellectual Property (IP) blocks as **Terminal Boundaries**.

A. Isolation Strategies in UPF

UPF provides two primary methods for defining isolation strategies between power domains:

1. **Port-Based Strategies**: Isolation is explicitly defined on specific ports or signals. Designers manually specify the ports requiring isolation, resulting in consistent interpretation across tools. This method has been available since UPF 1.0.

   Example:

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

2. **Path-Based Strategies**: Isolation is automatically inserted when certain source and sink requirements are met, based on the analysis of cross-domain paths. This method scales better with design changes and has been available since UPF 2.0. However, it requires visibility of the global design and appropriate macro markings to function correctly.

   Example:

   ```
   set_isolation iso_out \
     -domain pd_gated \
     -source ss_gated \
     -sink ss_always_on \
     -diff_supply_only true
   ```

B. Challenges with Path-Based Strategies

While path-based strategies simplify UPF specifications and adapt well to design changes, they introduce challenges:

- **Global Design Visibility**: Tools must analyze the entire design to identify cross-domain paths, which can be computationally intensive in large designs.
- **Consistency Between Simulation and Synthesis**: Without proper boundaries, different tools may interpret the isolation requirements differently, leading to inconsistencies.
- **Functional Correctness**: Inaccurate path analysis can result in missing or incorrectly inserted isolation cells, potentially causing functional errors.

These challenges necessitate a mechanism to control the scope of path-based analysis.

C. Terminal Boundary Semantics

UPF addresses these challenges by defining **Terminal Boundary Semantics**, which stop cross-domain tracing at specified boundaries and rely on boundary constraints:

- **Driver/Receiver Supplies on Boundary Ports**: Tools consider the supply conditions at the ports of the boundary, without analyzing the internal details of the IP.
- **Macro Markings**: By marking IP blocks as macros, designers inform tools where to apply terminal boundary semantics.

D. Types of Macros:

UPF defines three kinds of macros that form **Terminal Boundaries**:

- Soft Macros (UPF 3.1): Represent IPs intended for separate implementation, encapsulating their power intent.
- Hard Macros (UPF 3.1): Pre-implemented IPs with fixed functionality and power intent.
- Refinable Macros (UPF 4.0): Introduced for SIPs, allowing non-intrusive refinements while preserving the original power intent.

All three macro types require tracing to stop at their boundaries, relying on boundary constraints to ensure correct power management.

E. Benefits of Marking IPs as Terminal Boundaries

Marking IPs as Terminal Boundaries provides several advantages:

- **Controlled Path Analysis**: Tools limit cross-domain path analysis to within terminal boundaries, reducing computational complexity and improving scalability.
- **Consistent Tool Behavior**: Ensures that verification and synthesis tools interpret the power intent consistently, reducing the risk of discrepancies.
- **Parallel Processing**: Enables tools to partition the design at macro boundaries, facilitating parallel processing and faster analysis.
- **Functional Correctness**: By relying on boundary constraints, tools accurately insert necessary isolation and level-shifting cells, maintaining functional integrity.

F. Practical Implications

In designs employing path-based strategies, not marking IPs as Terminal Boundaries can lead to inconsistent isolation cell insertion across tools and stages:

- **Simulation vs. Synthesis**: Without terminal boundaries, simulation tools might insert isolation cells differently from synthesis tools, causing mismatches.
- **Complex Designs**: Large designs with numerous power domains and IP blocks exacerbate these issues, making terminal boundaries essential for manageability.

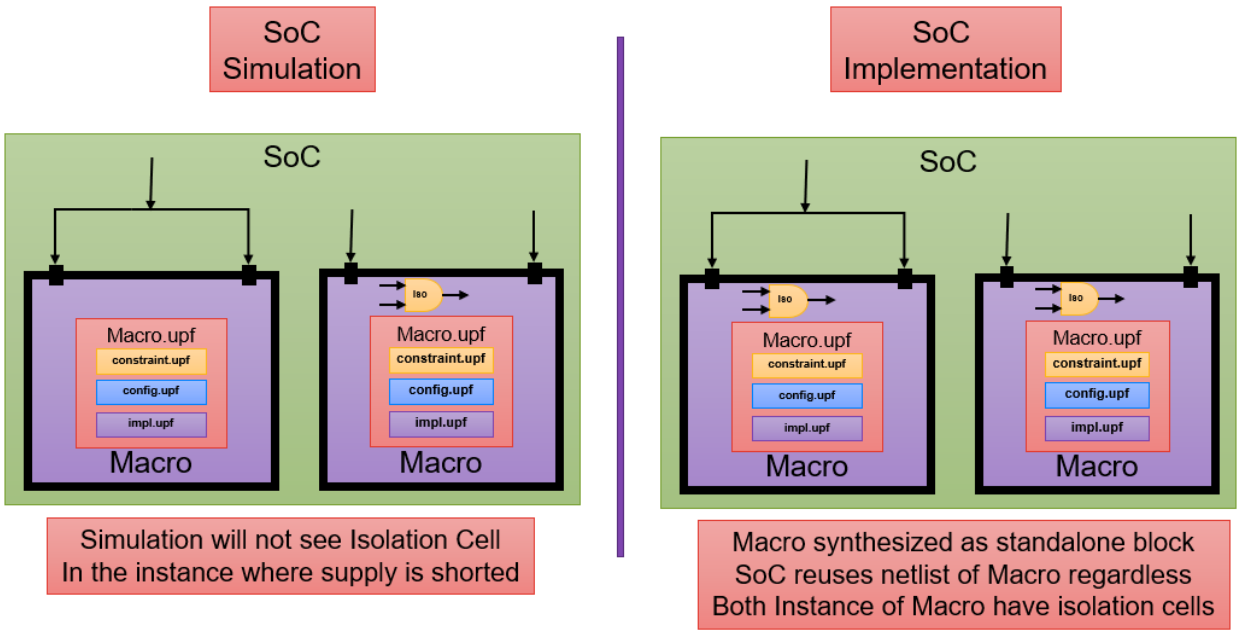

Figure 4 Simulation vs. Synthesis mismatch without terminal boundaries

Figure 4 illustrates a common inconsistency between simulation and synthesis when terminal boundaries are not properly defined. An IP block implemented as a standalone macro has two input supplies powering its internals. When instantiated multiple times in the SoC, there are scenarios where these supplies may be shorted by connecting them to the same supply in the SoC. Within the IP, crossings between logic powered by these supplies are handled by isolation strategies that insert isolation cells if the source and sink supplies differ.

During standalone synthesis, the isolation cells are inferred at these crossings based solely on internal conditions. These isolation cells are then included in the netlist for all instances of the IP in the SoC as it reuses the macro netlist for all its instances. In contrast, RTL simulation of the SoC considers the shorting of supplies and omits the isolation cells. This discrepancy between simulation and synthesis can lead to missed issues, such as incorrect clamping of isolation cells during normal operation in instances where supplies are shorted.

Marking the IP as a Soft Macro enforces terminal boundary semantics, ensuring that simulation respects the presence of isolation cells as inferred during synthesis, even in instances where supplies are shorted. Figure 2 highlights these semantics and demonstrates how simulation tools account for soft macro behaviors. This consistency catches potential issues during RTL simulation, ensuring reliable verification of the final hardware. This highlights that macro markings are crucial for functional correctness and tool consistency in UPF-based designs.

Table 2 summarizes the key features of UPF 4.0 discussed in this paper and their benefits.

Table 2

Feature	Description	Benefit
Refinable Macros	Enables non-intrusive refinements	Preserves IP verification integrity
Terminal Boundaries	Ensures consistent tool interpretations	Reduces simulation-synthesis mismatch
Implementation UPF	Adds implementation-specific details	Improves system-level optimizations

VI. Practical Application and Case Studies

To illustrate the practical benefits of Refinable Macros and terminal boundary markings, we present an example demonstrating their application.

Example: Integrating a Pre-Verified IP with Refinable Macros

A pre-verified IP has GATED and Always ON (AON) domains. The GATED domain is switched through a SoC-implemented power switch. The IP implements isolation between the AON and GATED domains and validates its power intent. The SoC integrator needs flexibility in choosing the isolation location based on place-and-route (PnR) requirements without altering the IP's original power intent.

IP's UPF (ip.upf):

```
# IP hierarchy
# ip1_top in AON domain
# |_ ip1_pgd_wrapper in PGD domain
```

```
set_design_attributes -models {.} -is_refinable_macro true

create_supply_set ss_AON
create_supply_set ss_PGD

create_power_domain AON -elements {.}

create_power_domain PGD -elements {ip1_pgd_wrapper}

.....

## isolation strategy for ports crossing from PGD to AON.
## -location not specified by IP team
##  -location,  if  specified  and  hardcoded,  cannot  be  overridden  through
Refinable macro refinement
set_isolation "o_PGD_to_AON" \
 -domain “PGD” \
 -isolation_supply_set "ss_AON" \
 -clamp_value "0" \
 -elements {
    ip1_pgd_wrapper/portA
    ip1_pgd_wrapper/portB
 } \
 -isolation_signal pwr_manager/iso_en_b \
 -isolation_sense low

....

create_pst ip_PST   -supplies "ss_AON.power ss_PGD.power ss_AON.ground"
add_pst_state ON    -state {ON  ON  ON} -pst ip_pst
add_pst_state GATED -state {ON  OFF ON} -pst ip_pst
add_pst_state OFF   -state {OFF OFF ON} -pst ip_pst
```

The IP defines AON and PGD power domains and specifies an isolation strategy without the `-location` option, allowing the SoC integrator to define the isolation location based on implementation needs.

SoC Implementation UPF (ip.socimpl.upf):

```
set_isolation o_PGD_to_AON \
 -domain PGD \
 -location self \
 -update
```

The SoC integrator refines the IP's isolation strategy by specifying the `-location` option. Using the Refinable Macro semantics, the integrator can make this refinement without altering the IP's original UPF.

SoC-Level UPF (parIP1.upf):

```
# soc hierarchy
# parIP1
# |_ ip1 ip1_top

create_power_domain par_AON -elements {.}

# Load the IP's UPF and the SoC implementation UPF
load_upf ip.upf -scope ip1
load_upf ip.socimpl.upf -scope ip1 -implementation
```

By using the **`load_upf`** command with the **`-implementation`** option, the SoC integrator ensures that the refinements comply with UPF 4.0 standards and that the IP's original power intent remains intact.

Benefits Demonstrated:

- **Flexibility in Implementation:** The SoC integrator can specify implementation details like isolation location without modifying the IP's original UPF.

- **Preservation of Verification Integrity:** The IP's standalone verification remains valid, avoiding the need for re-validation.
- **Tool-Assisted Compliance:** Tools enforce constraints, ensuring that refinements adhere to UPF 4.0 standards.

VII. CONCLUSION

The introduction of Refinable Macros and enhanced terminal boundary definitions in UPF 4.0 represents significant advancements in low-power design for complex System on Chips (SoCs). By addressing the limitations of UPF 3.1, these new features enable designers to integrate pre-verified SIPs more efficiently and safely, without compromising verification integrity.

Key contributions of UPF 4.0 include:

- **Non-Intrusive Refinement Capabilities**: Allowing implementation-specific details to be added to the SIP's power intent without altering the original definitions, preserving extensive verification performed by IP providers.
- **Flexible Terminal Boundary Definitions:** Providing mechanisms to mark IPs as refinable terminal boundaries, facilitating efficient partitioning of designs and enabling tools to manage complexity effectively.
- **Enhanced Tool Support:** Empowering design tools to enforce constraints, verify compliance, and assist in implementing power management strategies, reducing the likelihood of errors.

By adopting these methodologies, designers can achieve:

- **Improved Design Quality:** Enhanced optimization opportunities lead to better performance, lower power consumption, and efficient use of silicon area.
- **Reduced Development Time:** Streamlined integration and verification processes accelerate development cycles, helping companies meet aggressive time-to-market goals**.**
- **Risk Mitigation:** Preservation of verification integrity minimizes the risk of functional errors, contributing to more reliable and robust products.

The evolution of UPF standards reflects the industry's response to the growing complexity of SoC designs and the need for efficient, scalable methodologies. Refinable Macros in UPF 4.0 offer a balanced approach, providing the necessary flexibility for implementation refinements while safeguarding the original power intent and verification efforts.

VIII. REFERENCES

[1] IEEE Standard 1801-2018 (UPF 3.1), “IEEE Standard for Design and Verification of Low-Power, Energy-Aware Electronic Systems,” 2018.

[2] IEEE Standard 1801-2024 (UPF 4.0), “IEEE Standard for Design and Verification of Low-Power, Energy-Aware Electronic Systems,” 2024.

[3] Amit Srivastava and Shreedhar Ramachandra, "Verification Macros: Maintain the Integrity of Verifiable IP UPF Through Integration," Design and Verification Conference (DVCon) US, 2023.