Future Proofing Power Intent Specification through Unified Power Format (UPF) 4.0 for Evolving Advanced State Retention Strategies

Lakshmanan Balasubramanian[1], Amit Srivastava[2], Raguvaran Easwaran[3], John Decker[4], Rick Koster[5], Progyna Khondkar[4], Paul Bailey[6], Barry Pangrle[7], Shreedhar Ramachandra[2], Phil Giangarra[4], John Biggs[8] and David Cheng[4,9]

[1]Texas Instruments (India) Pvt. Ltd., [2]Synopsys Inc., [3]Intel Technology India Pvt. Ltd., [4]Cadence Design Systems, [5]Siemens Mentor, [6]Nordic Semiconductors, [7]Abacus Semiconductor Corporation, [8]PragmatIC, [9]TBD

Abstract: ***The paper discusses the role of state retention (SR) elements and strategies in optimising power consumption while ensuring quick system recovery. By utilising*** IEEE 1801 standard***, also known as Unified Power Format (UPF), SR can be effectively specified, modelled, verified and implemented coherently to ensure functional and electrical correctness under a wide range of operating conditions.***

Conventional SR techniques mostly involve implementation of underlying SR elements and system-level SR strategies that are potentially sub-optimal for evolving technologies and end applications. Many of these evolving SR strategies for area and power efficient SR elements with lower-level i.e., device-level optimisations utilised in building such elements may impose additional functional and electrical boundary conditions. Comprehensive handling of such boundary conditions is not supported in the current versions of power intent (PI) standards.

In the upcoming version UPF 4.0, we propose to add additional syntactical features supporting scalable semantics of evolving advanced SR requirements and techniques for more complete power intent specification, consistent and coherent interpretation with seamless handling across EDA tools and flows.

A detailed case study is provided demonstrating practical applications of SR strategies in low power designs.

I. INTRODUCTION

As electronic systems evolve to meet the increasing demands for energy efficiency, especially in mobile and embedded devices, low power (LP) design techniques are critical for reducing both dynamic and static power consumption. Standard techniques for realising functionally and electrically consistent LP design with user-controllable power-versus-performance trade-offs include:

- designing with suitable metal-oxide semiconductor (MOS) devices of appropriate gate threshold
- adaptive and static body biasing
- clock gating for sections of designs for periods of no activity
- frequency reduction to sections of designs for periods of low performance
- adaptive voltage and frequency scaling
- power gating.

Additional techniques that are needed to overcome the challenges due to the implementation of any of the aforementioned approaches include:

- isolation of signals from switchable i.e., power gated domains to clearly defined safe states for avoiding functional and power consumption issues in the receiving domains that remain powered-on
- state retention (SR) of memory elements in the switchable power domains
- level shifting for signals crossing between domains of incompatible supply voltage levels
- clock and reset domain crossing (CDC & RDC) techniques like clock and reset synchronisation respectively.

Standardised power intent (PI) specifications like Common Power Format (CPF) and IEEE 1801 standard, also known as Unified Power Format (UPF) support the power gating and scaling related aspects. One of the key challenges in power management is maintaining data integrity during power-down events viz. power shutdown and various gradations of sleep modes. SR strategies, particularly the use of SR elements and methods, provide a solution by preserving critical state information even when the power supply to parts of the system is turned off.

This paper explores the various SR techniques employed in low power, energy-aware systems, including the design and implementation of SR registers under power gated system modes, and state-saving mechanisms. It discusses the role of SR elements and strategies in optimising power consumption while ensuring quick system recovery and minimal power overhead during power state transitions. By utilising UPF, SR can be effectively specified, modelled,

verified and implemented coherently to ensure functional and electrical correctness under a wide range of operating conditions. It delves into verification strategies for design with SR, such as analysis of various power states, checking data integrity across power down cycles, and ensuring seamless power state transitions in complex system-on-chip (SoC) designs.

II. Advanced Retention Challenges

An example LP system, its power-aware (PA) verification setup, and the signal-level behaviour for the system are illustrated in Figure 1, Figure 2, and Figure 3 respectively [1][2][3]. They showcase a typical LP system, demonstrating the interactions between different power domains and the associated control signals that ensure proper state retention across power down cycles. The verification flow integrates a PA simulation tool that supports simulating the more complete and fine-grained behaviour of the system under various power states: exhibiting conservative behaviour by corrupting the states of the logic circuits under such power domains whose supplies are switched-off for minimising power consumption; and ensuring that state information of such switched off power domains can be preserved accurately, if specified appropriately. Figure 1 illustrates an overview of the system architecture, highlighting the role of power gating, clock gating, and adaptive voltage scaling in managing dynamic and static power consumption/dissipation. Figure 2 delves deeper into the verification process, where each power domain is evaluated for functional correctness during mode transitions, ensuring that isolation, level shifting and retention elements behave as expected in the presence of switched power supplies. Figure 3 illustrates the signal-level behaviour, emphasising the interplay between clock, reset, and control signals during transitions between active and power down states.

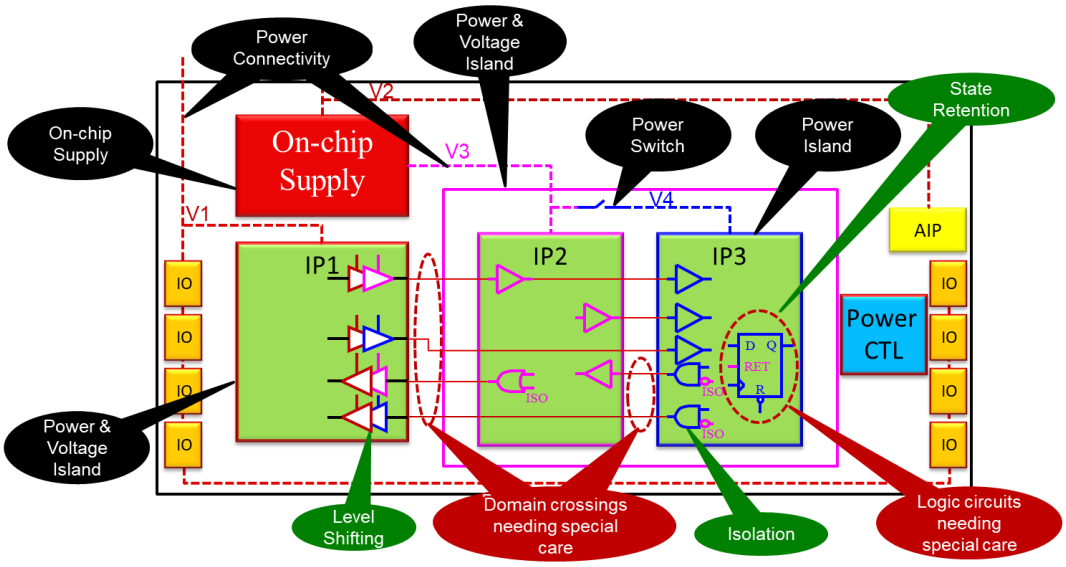


Figure 1. Typical LP System [1]

As illustrated in Figure 1 & Figure 3, the switchable domain *PD_V4* associated with the supply net *V4* supplies IP3 requiring state retention. State retention is enabled in the system by asserting the signal *RET* before the switch powering *V4* is turned-off (by de-asserting *Power Domain* Enable) and eventually *V4* becomes invalid. The state *D5* encountered at the rising edge of *RET* signal is saved and retained during invalid *V4* i.e., power-down state of IP3. Upon exit from power-down by asserting the *Power Domain Enable*, once *V4* becomes valid and subsequently upon de-assertion of *RET* the saved/retained state *D5* is restored. However, the strategy allows a successful SR operation independent of the clock (*CLK*) and reset/preset (*RSTZ_V4*) control signal states during the period between the save and restore events, requiring underlying SR elements having the clock signal *CLK* gated internally with the retention control signal *RET*. This allows clock tree activity when *RET* remains asserted without causing any malfunction including loss of retained state. Such an implementation doesn't pose any constraints on the system-level clock tree allowing its drivers to be powered-off to reduce power consumption or to share with other sections of the design that may require clock activity, though at the cost of retention element-level incremental power consumption and area overhead. Such a system behaviour can be realised using a retention strategy representing an SR element of type single-control balloon-latch, both illustrated by the UPF specification in *Code 1*. Note that while ***set_retention***

specifies the system-level SR strategy at the current scope, the ***define_retention_cell*** specifies the underlying gate-level abstraction of the individual SR element (not detailed herein).

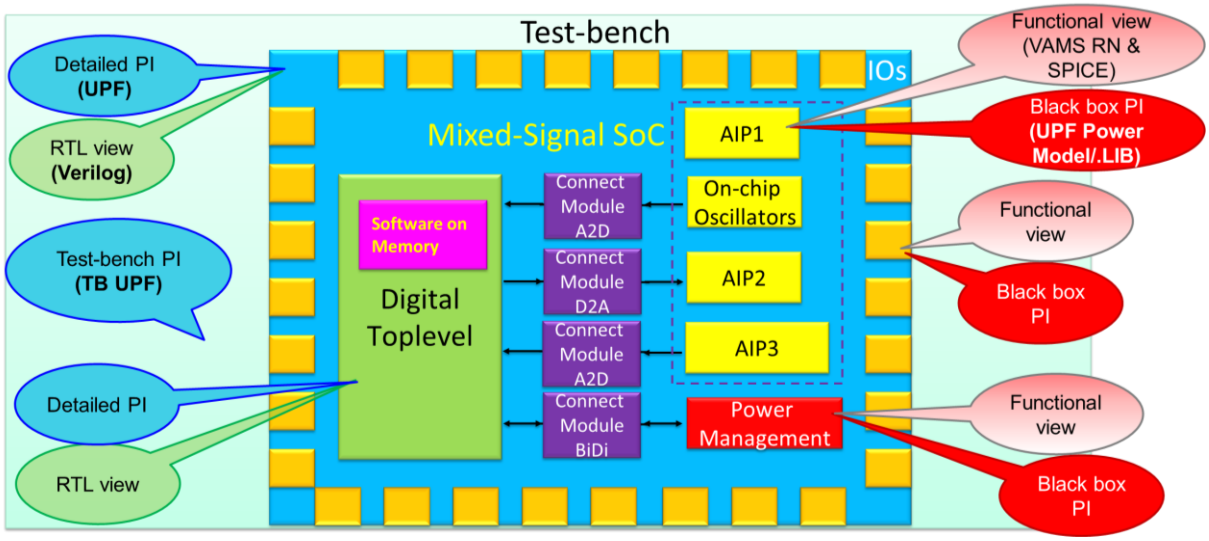
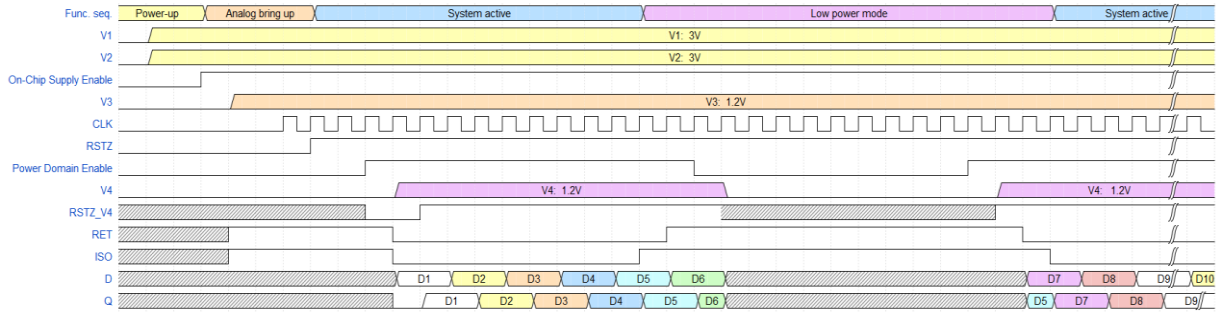


Figure 2. Typical Power-Aware Verification Setup [1]



Legend:

V1, V2: Primary power supplies
V3, V4: On-chip switched power supplies
CLK: Clock signal
RSTZ: Active low reset signal
RET: Active high state retention control signal
ISO: Active high isolation control signal
D: Input data to a state retention cell
Q: Output of a state retention cell
RSTZ_V4: Reset for the power domain PD_V4 corresponding to V4

Figure 3. Signal-Level Behaviour for an LP System [1]

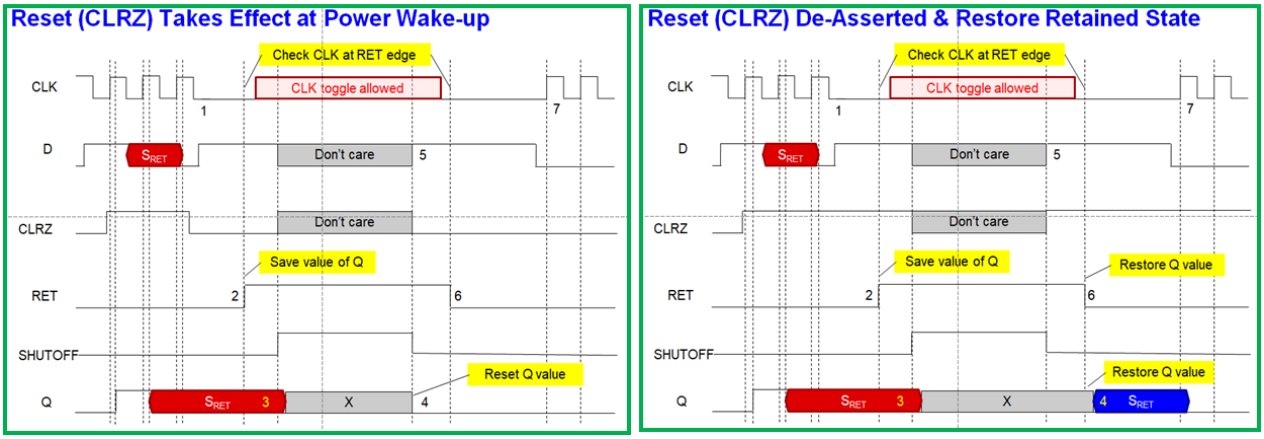
```
upf_version 3.1

set_retention ret -domain PD_V4 -save_signal {RET posedge} \
-restore_signal {RET negedge} ...

define_retention_cell -cells SR1 -clock_pin CLK -save_function {RET posedge} \
-restore_function {RET negedge} -power_switchable VDD -power VDD_RET \
-ground VSS
```

Code 1. UPF 3.1 code specifying single-control balloon-latch type SR element

Conventional SR [1][2] techniques mostly involve straightforward simpler implementation of underlying SR elements, such as SR registers or flip-flops and system-level SR strategies that are effective in conventional design paradigms but are increasingly getting sub-optimal for evolving technologies and end applications. An example of the signal-level behaviour of one such conventional SR technique is illustrated in Figure 4, involving a strategy (as shown in Code 2) using underlying retention elements having the clock signal *CLK* gated internally with the retention

control signal *RET*, allowing clock tree activity when *RET* remains asserted without causing any malfunction including loss of retained state. Such an implementation doesn't pose any constraints on the system-level clock tree allowing its drivers to be powered-off to reduce power consumption, though at the cost of retention element-level incremental power consumption and area overhead. However, as shown in Figure 4(ii), it requires the clock to be frozen at logic '0'- level at the save & restore control events for the save and restore functions to operate successfully, the condition for the same is specified using ***save_condition*** and ***restore_condition*** options of ***set_retention*** as shown in Code 2.

As semiconductor process nodes shrink and device architectures become more complex, conventional SR techniques struggle to meet the new demands for higher power efficiency, faster recovery times, and lower area overheads. In this figure, the state of a SR register is maintained during power down, but the recovery time is relatively slow, and power consumption remains a concern due to the static nature of the control signals and activity on clock path. Additionally, conventional techniques may not fully account for the intricate power domain crossings, leading to potential data integrity issues during power mode transitions.



(i) Reset Functionality *(ii) Retention Restore Functionality*

Figure 4. Conventional SR Behaviour [1]

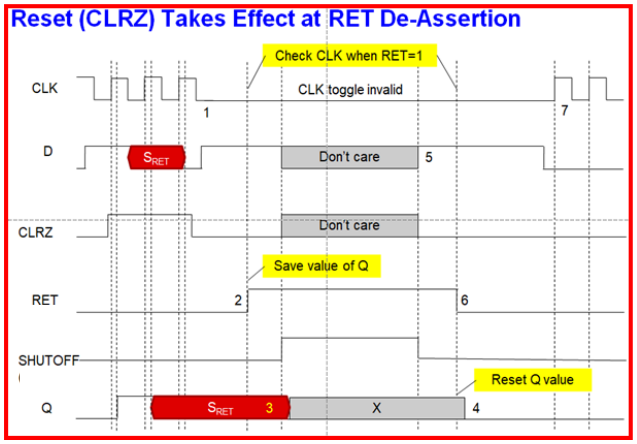
```
upf_version 3.1

set_retention ret -domain PD_V4 -save_signal {RET posedge} \
-restore_signal {RET negedge} -save_condition {!CLK} \
-restore_condition {!CLK} …
```

Code 2. UPF 3.1 code specifying a conventional SR behaviour in Figure 4

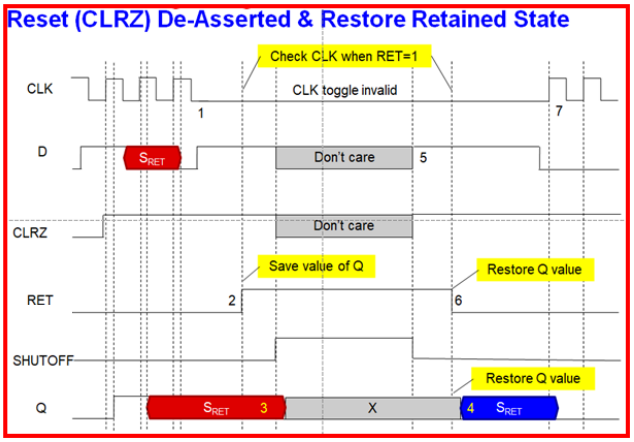
Many of the evolving advanced SR strategies [1][2] aim to address these shortcomings by incorporating more sophisticated, area and power efficient SR elements with lower-level i.e., device-level optimisations utilised in building such elements. These optimisations include the use of lower leakage transistors, improved retention mechanisms, and more dynamic control of retention elements based on real time power and performance needs. However, these improvements often may impose additional functional and electrical boundary conditions [1][2], such as stricter timing requirements for signal transitions, more complex interactions between power domains, and the need for tighter control over clock and reset signals during state and mode transitions. An example of the signal-level behaviour of samples of such advanced SR techniques are illustrated in Figure 5 which for example provides a more fine-grained micro architectural-level trade-off. Such advanced SR strategies can significantly reduce power consumption while maintaining fast recovery times by dynamically adjusting the retention control signals based on the system's operational state.

In Figure 5, the advanced SR technique leverages single-control baloon-latch retention strategy, where both the clock and reset signals are managed separately to ensure optimal power savings during periods of inactivity. This approach allows the system to selectively power down specific regions while retaining critical data in key areas of the design. For example, Figure 5(a) illustrates a strategy using underlying retention elements exposing the clock signal *CLK*, without any internal gating, thus imposing a constraint on the system-level clock tree to maintain a known state (logic level 0 in this illustration) when *RET* remains asserted to avoid loss of retained state. However, such an

implementation requires additional system-level constraints with the benefit of low power and low area retention elements. Figure 5(b) illustrates another more advanced strategy using an underlying retention element having no internal gating on the clock (*CLK*) path but with a specific circuit-level optimisation of the SR element that is resilient to operate with an undriven clock signal *CLK* by allowing the clock tree drivers to be powered-off during the low power mode, with more fine-grained power saving entitlement. Unlike conventional methods, the advanced strategy adapts to varying power and performance demands, making it more suitable for modern applications with diverse operating conditions, such as mobile and embedded systems. These strategies not only minimize power consumption during standby modes but also enable rapid recovery from power down states, thus ensuring that the system can resume full functionality quickly when needed.

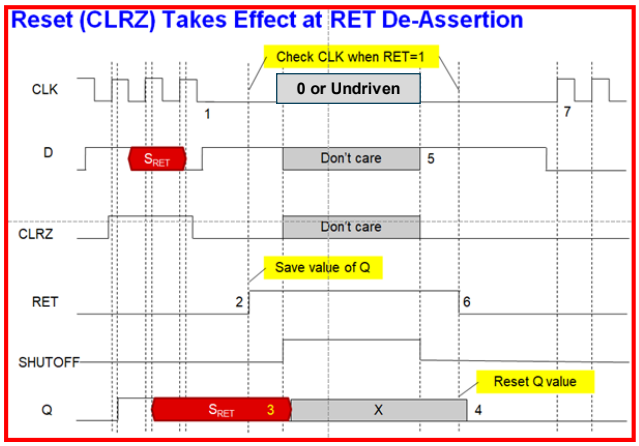


(i) Reset Functionality

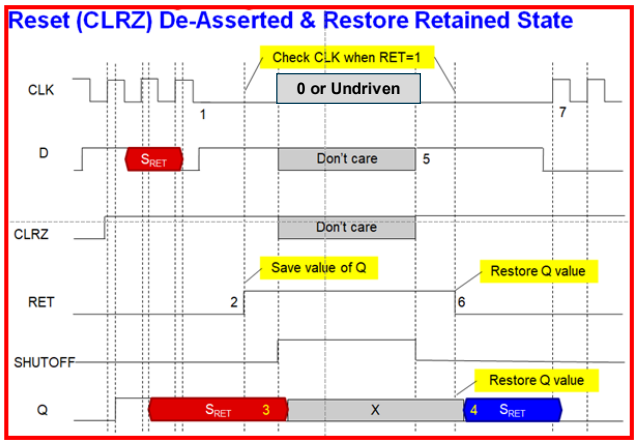


(ii) Retention Restore Functionality

(a) Clock to stay low when RET is asserted



(i) Reset Functionality



(ii) Retention Restore Functionality

(b) Clock to stay low when RET is asserted; Clock to stay low or undriven, due to driver whose power is off when SHUTOFF is asserted

Figure 5. Advanced SR Behaviour [1]

As these advanced SR techniques evolve, they impose new challenges in terms of verification and implementation. The additional boundary conditions introduced by device-level optimisations require more granular control over SR elements, leading to increased complexity in specifying and verifying power intent, including fine grained microarchitectural optimisation and trade-off to keep the entire clock tree either powered-on or powered-off during the power down modes. The upcoming UPF 4.0 aims to address these challenges by introducing new features and commands that allow for more detailed specification of retention behaviours, power domains, and the interactions

between them. These enhancements enable designers to model advanced SR strategies more effectively and earlier in the design cycle, ensuring that all functional and electrical constraints are met, even as power management techniques continue to evolve.

III. Limitations of Existing Power Intent Standards

Comprehensive handling of the boundary conditions imposed by the evolving advanced SR strategies discussed in the earlier section is not supported in the current versions of PI standards. For example, the strategies shown in Figure 4(i), Figure 5(a)-(i) & Figure 5(b)-(i) require reset (*CLRZ*) to take effect if asserted at the power-up event and the saved/retained state information to be lost. Additionally, there are more nuanced conditions for successful save, restore and retention operations as illustrated:

a) in Figure 5(a) requiring *CLK* to remain low through the entire period between save and restore events.
b) in Figure 5(b) requiring CLK to remain low through the period between save event and power-down event, between power-up event and restore event; and CLK to either remain low or driven by a powered-off driver during the period between power-down and power-up events.

Thus, most of the electrical and behavioural constraints on *reset* (*CLRZ*), set/*preset*, *clock* (*CLK*) signals of such SR elements and those related to the SR strategies, across different states of the related power domains (both primary and backup power domains) and states of the retention control signal (*RET*) are not specifiable in the current PI specification standards (UPF 3.1 and earlier). They allow for checks only at save event, restore event and the period between power-down and power-up events. This state-of-the-art causes either late finding of basic architectural and infrastructural specification violations or inappropriate (both inaccurate and incorrect) implementation details to evolve until very late in the design cycle as they cannot be comprehended in RTL stage due to PI limitations with significant specification and verification gap. Existing synthesis and implementation tools too cannot comprehend and hence handle such requirements comprehensively due the aforementioned PI limitations.

Typically, such functional behaviours can only be comprehended at a mature design stage post PA Gate-Level abstraction, mostly with PA simulations and some electrical behaviours require even transistor-level simulation like Analog and Mixed-Signal (AMS) co-simulation which can be prohibitively costly or even impractical to be performed due to the complexity and computational costs involved in performing such simulation especially starting at pretty late stages of design potentially resulting in escaped design issues and silicon bugs.

IV. Advanced Retention Handling in Proposed UPF 4.0

In the upcoming version UPF 4.0 [4], we propose to add additional syntactical features supporting scalable semantics of evolving advanced SR requirements and techniques for more complete power intent specification, consistent and coherent interpretation with seamless handling across EDA tools and flows. These features include semantic and syntactical extensions to existing retention related commands viz. ***define_retention_cell, map_retention_cell, & set_retention*** supported by introducing predefined command options e.g., new predefined conditions to identify system states viz. ***save_event_condition, restore_event_condition, restore_period_condition, & power_down_period_condition***. The options ***save_condition, restore_condition, & retention_condition*** are made legacy and cannot be specified consistently and coherently with newer options. A new option ***async_set_reset_effect*** is introduced to specify how the reset/set signals affect the output and retained value during the restore period, while having no effect in normal operation. Note that currently all these enhanced and new options are only for the purposes of verification and have no explicit or unique effect on the synthesis and implementation flows. The correctness of implementation depends on the right ***map_retention_cell & define_retention_cell*** **specified** for SR strategies as always.

Basic retention register operation and modelling that are useful in describing the simulation semantics for ***set_retention*** command is shown below:

```
upf_version 4.0

set_retention retention_name   -domain domain_...  [-save_signal {logic_net
<high | low | posedge | negedge>} \
-restore_signal {logic_net <high | low | posedge | negedge >}]  [-
save_event_condition {boolean_expression}]  \
[-restore_event_condition {boolean_expression}] [-powerdown_period_condition
{boolean expression}] \
[-restore_period_condition {boolean expression}]
[-async_set_reset_effect <ignored | retained_value | output_value>]
  … [-save_condition {boolean_expression}] [-restore_condition
{boolean_expression}]
[-retention condition {boolean expression}]
```

The ***save_event_condition*** gates the save event, defining the save behaviour of the register. The register contents are saved when the save event occurs and the ***save_event_condition*** is *True*. The retained value shall be the register's value at the time of the save event when ***save_event_condition*** evaluates to *True*. For edge-sensitive save, the save event is the rising or falling edge of the ***save_signal*** at the specified edge; for level-sensitive save, the save event is the trailing edge of the ***save_signal***.

The ***restore_event_condition*** gates the restore event from triggering the restore operation of the register. The register is restored when the restore event occurs and the ***restore_event_condition*** is *True*. The retained value is transferred to the register on the restore event when ***restore_event_condition*** evaluates to *True*. The restore event is the rising or falling edge of an edge-triggered ***restore_signal*** or the leading and trailing edges of a level-sensitive ***restore_signal***. In addition, in the case of a level-sensitive restore, the restore operation is continuous throughout the restore period, taking the value of the ***async_set_reset_effect*** option into account.

The ***restore_period_condition*** also gates the restore operation. The restore operation occurs continuously during the entire restore period. The ***restore_period_condition*** shall be *TRUE* throughout the entire restore period; otherwise, the restore operation will fail, and the retention element will be corrupted. If the ***restore_period_condition*** is not specified, then the restore operation is not gated.

The ***powerdown_period_condition*** defines the conditions under which the retained value is maintained when the domain power is *OFF*. If the ***powerdown_period_condition*** is specified, it shall evaluate to *TRUE* the entire time that the domain's primary power is *OFF* for the value of the state element to be retained. If the ***powerdown_period_condition*** evaluates to *FALSE* and the primary supply is not *NORMAL*, the retained value of the state element is corrupted, else the retained value is not affected by this option. The driver supply of any pin listed in the ***powerdown_period_condition*** shall be at least as on as the retention supply of the retention strategy.

The ***save_event_condition***, ***restore_event_condition***, ***restore_period_condition*** and ***powerdown_period_condition*** shall only reference the predefined name ***UPF_GENERIC_CLOCK, UPF_GENERIC_ASYNC_SET_RESET***, or logic nets or ports rooted in the current scope.

For edge-triggered flip-flops, ***UPF_GENERIC_CLOCK*** represents a signal whose rising edge triggers the register to load data. For level-sensitive latches, this is a signal whose high value enables the latch. ***UPF_GENERIC_ASYNC_SET_RESET*** is an expression derived from the set and reset inputs of the flipflop as following:

- (set signal normalized to active high) || (reset signal normalized to active high)
- This expression is TRUE when either the set or reset signal is active

For improving the ease of strategy definition, a new option ***applies_to*** is introduced as a filter to restrict the strategy to apply only to state elements of a given type viz. *[<flop | latch | any>]*.

V. Case Studies and Discussion

This case study demonstrates some practical applications of SR strategies in LP designs highlighting design trade-offs that enable the optimisation techniques to minimise area and power overhead while maintaining rapid recovery of system state. The clock and reset/set behaviour dependencies on the retention functionality for single-control and dual-control SR elements are discussed.

A. Single-Control Retention

A single-control SR element has just one control signal to indicate whether the register is operating in normal mode or retention mode. In addition to the standard RTL defined clock (*CLK*), data (*D*), output (*Q*), set (*SET*)/reset (*RESET*) ports, single-control SR registers have an additional retention control port that is specified in the ***set_retention*** command. In this case study *RETN* is defined as the active low retention control port. A typical retention sequence is as follows:

- Optionally stop the clock at a specified state (*CLK*)
- Assert the retention control signal (*RETN*)
- Save the value in the flipflop
- Power-down
- Power-up
- Restore the saved value
- De-assert the retention control signal

In addition, it is recommended to have the clock remain in its inactive state and that the circuit isolate the outputs of the domain before powering down, to prevent unknown values from propagating from the powered-down domain

to the inputs of powered-up domains. A level-sensitive control signal should be used to specify the restoration of Q at power-up (*@pwrup*); while an edge-triggered control signal is specified if Q value requires to be restored at the end of the retention cycle (*@retn*). Some retention cells require the clock or reset to be held at a certain value during specific intervals of the retention period. These requirements can be specified using various condition options in the ***set_retention*** command. If any of these conditions are false at any time during the entire interval (level sensitive), or at the edge (edge sensitive), then the save or restore operation will fail, and the retained value will be set to x. Figure 6 illustrates the behaviour of the retention flipflop under various save/restore conditions, in which ***UPF_GENERIC_CLOCK*** is present, to allow the user to model the retention behaviour with respect to the state of the clock. The UPF specification for each of these variants are listed below.

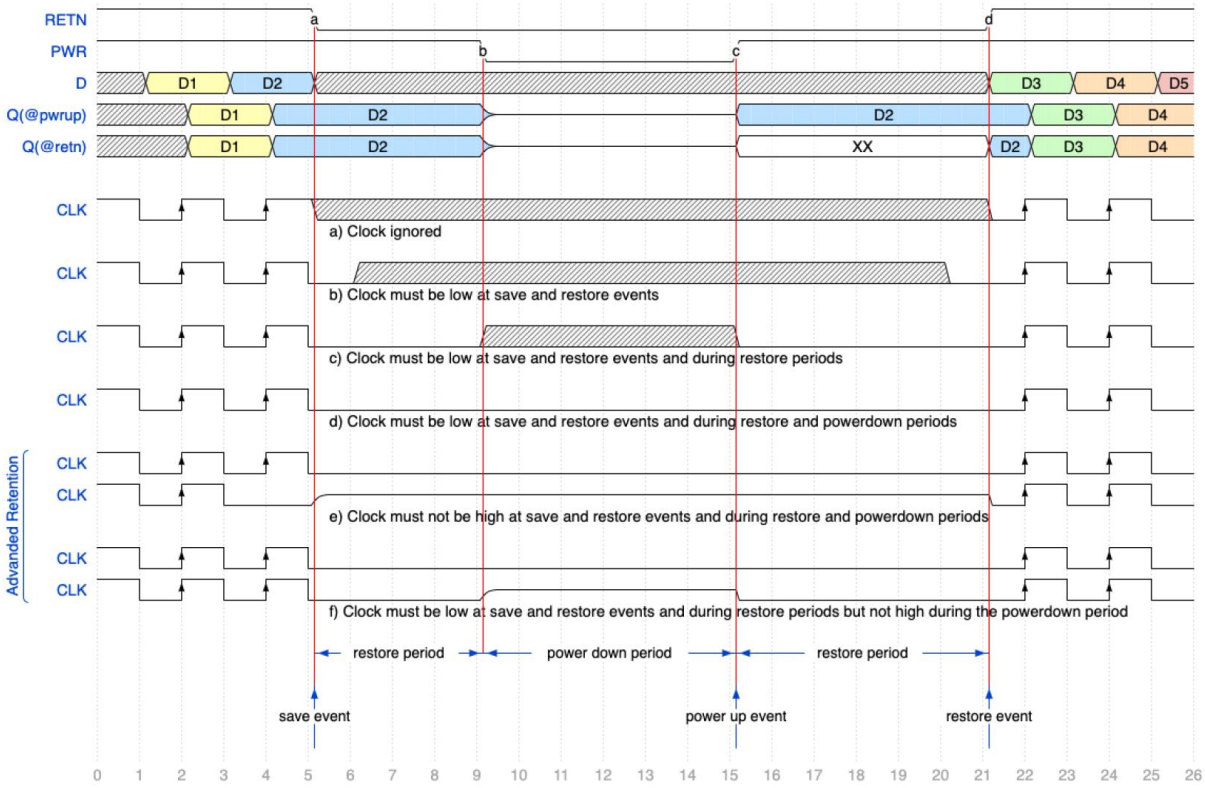


Figure 6. Single-Control Retention: Clock Behaviour [4]

a) Clock ignored during RETN active:
No ***save_event_condition***, ***restore_event_condition*** or ***restore_period_condition*** present in the command.

b) Clock must be low at the save and restore events:
-save_event_condition {!***UPF_GENERIC_CLOCK***} ***-restore_event_condition*** {!***UPF_GENERIC_CLOCK***}

c) Clock must be low at save event and restore events, and during the restore periods:
The ***restore_period_condition*** applies to both the restore event and restore period
-save_event_condition {!***UPF_GENERIC_CLOCK***} ***-restore_period_condition*** {!***UPF_GENERIC_CLOCK***}

d) Clock must be low during save and restore events, and during restore and power-down periods:
-save_event_condition {!***UPF_GENERIC_CLOCK***} ***-restore_period_condition*** {!***UPF_GENERIC_CLOCK***} ***-powerdown_period_condition*** {!***UPF_GENERIC_CLOCK***}

e) Clock must not be high during save and restore events, and during restore and power-down periods:
-save_event_condition {***UPF_GENERIC_CLOCK***==0 || ***UPF_GENERIC_CLOCK***==Z}
-restore_period_condition {***UPF_GENERIC_CLOCK***==0 || ***UPF_GENERIC_CLOCK***==Z}
-powerdown_period_condition {***UPF_GENERIC_CLOCK***==0 || ***UPF_GENERIC_CLOCK***==Z}

f) Clock must be low at save and restore events and during restore periods, but not high during power-down period:

-save_event_condition {!***UPF_GENERIC_CLOCK***} ***-restore_period_condition*** {!***UPF_GENERIC_CLOCK***} ***-powerdown_period_condition*** {***UPF_GENERIC_CLOCK***==0 || ***UPF_GENERIC_CLOCK***==Z}

Figure 7 shows the retention behaviour for various values of ***async_set_reset_effect***. In this figure, reset is active during the restore period after power-up. The figure shows the value of Q during the restore period, which changes depending on the value of ***async_set_reset_effect***. The UPF specification for each of these variants are listed below.

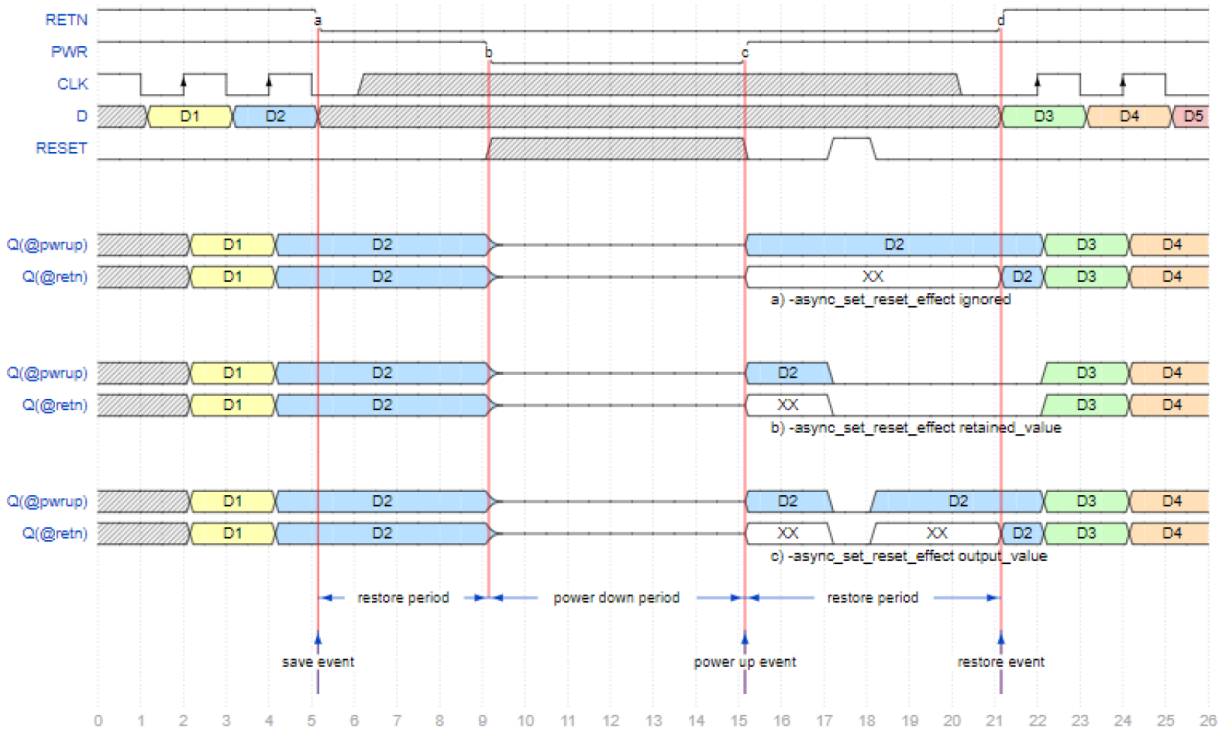


Figure 7. Single-Control Retention: Reset Behaviour [4]

a) Reset is ignored during the restore period:
-async_set_reset_effect *ignored*

b) Reset affects the retained value and therefore the output value during the restore period:
-async_set_reset_effect *retained_value*

c) Reset affects the output value during the restore period:
-async_set_reset_effect *output_value*

B. Dual-Control Retention

A dual-control SR element has one signal to indicate "save", and a different signal for "restore". When neither control signal is active, then the register acts normally. In addition to the standard RTL defined clock (*CLK*), data (*D*), output (*Q*), set (*SET*)/reset (*RESET*) ports, dual-control SR registers have two additional retention control ports viz., *SAVE* & *RESTORE* that are specified in the ***set_retention*** command. A typical retention sequence is as follows:

- Optionally stop the clock at a specified state (*CLK*)
- Save the state by asserting *SAVE* signal
- De-assert *SAVE* signal
- Power-down
- Power-up
- Restore the saved value by asserting *RESTORE* signal
- De-assert the *RESTORE* signal

Figure 8 illustrates the behaviour of the retention flipflop under various save/restore conditions, in which ***UPF_GENERIC_CLOCK*** is present in those conditions. The UPF specification for each of these variants are listed below.

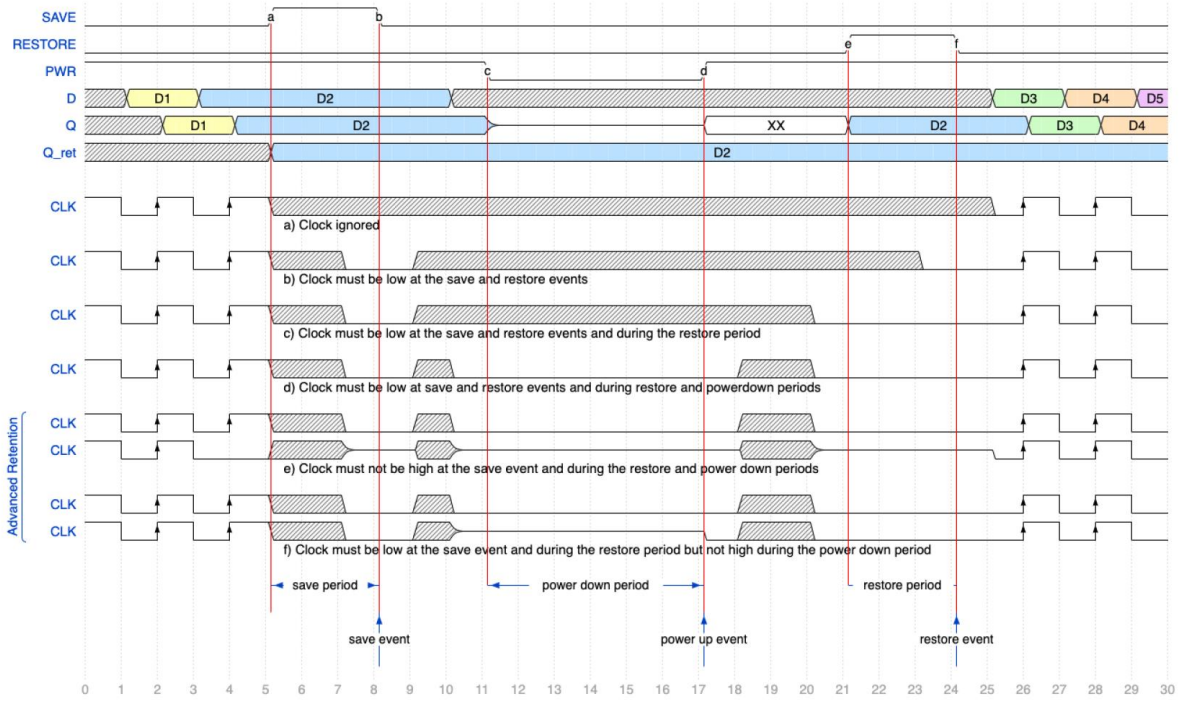


Figure 8. Dual-Control Retention: Clock Behaviour [4]

a) Clock ignored:
No ***save_event_condition***, ***restore_event_condition*** or ***restore_period_condition*** present in the command
b) Clock must be low at the save and restore events:
-save_event_condition !***UPF_GENERIC_CLOCK*** ***-restore_event_condition*** ***UPF_GENERIC_CLOCK***
c) Clock must be low at the save and restore events, and during the restore period:
The ***restore_period_condition*** applies to both the restore event and restore period:
-save_event_condition !***UPF_GENERIC_CLOCK*** **-restore_period_condition** !***UPF_GENERIC_CLOCK***
d) Clock must be low at the save and restore events, and during the restore period and during the power-down period:
-save_event_condition !***UPF_GENERIC_CLOCK*** ***-restore_period_condition*** !***UPF_GENERIC_CLOCK*** ***-powerdown_period_condition*** !***UPF_GENERIC_CLOCK***
e) Clock must not be high at the save event and during the restore and power down periods:
-save_event_condition {***UPF_GENERIC_CLOCK***==0 || ***UPF_GENERIC_CLOCK***==Z}
-restore_period_condition {***UPF_GENERIC_CLOCK***==0 || ***UPF_GENERIC_CLOCK***==Z}
-powerdown_period_condition {***UPF_GENERIC_CLOCK***==0 || ***UPF_GENERIC_CLOCK***==Z}
f) Clock must be low at the save event and during the restore period, but not high during the power-down period:
-save_event_condition !***UPF_GENERIC_CLOCK*** ***-restore_period_condition*** !***UPF_GENERIC_CLOCK*** ***-powerdown_period_condition*** {***UPF_GENERIC_CLOCK***==0 || ***UPF_GENERIC_CLOCK***==Z}

Figure 9 shows the behaviour when reset is active both after power-up before *RESTORE* is active, and after power-up during the restore signal active. For case (a), when *RESTORE* is inactive, the *RESET* signal is always obeyed, since the flipflop is in normal mode. However, when *RESTORE* is active, then *RESET* is ignored, since ***async_set_reset_effect*** is set to *ignored*. In the second and third cases (b) & (c), when the flipflop is in restore mode, the ***async_set_reset_effect*** option takes effect. The signal *Q_ret* in the figures below represent the actual

saved value in the circuit. For performance reasons, simulators may choose to save the value of *Q* at the ***save_event*** (edge sensitive) or the trailing edge of the ***save_signal*** (level sensitive).

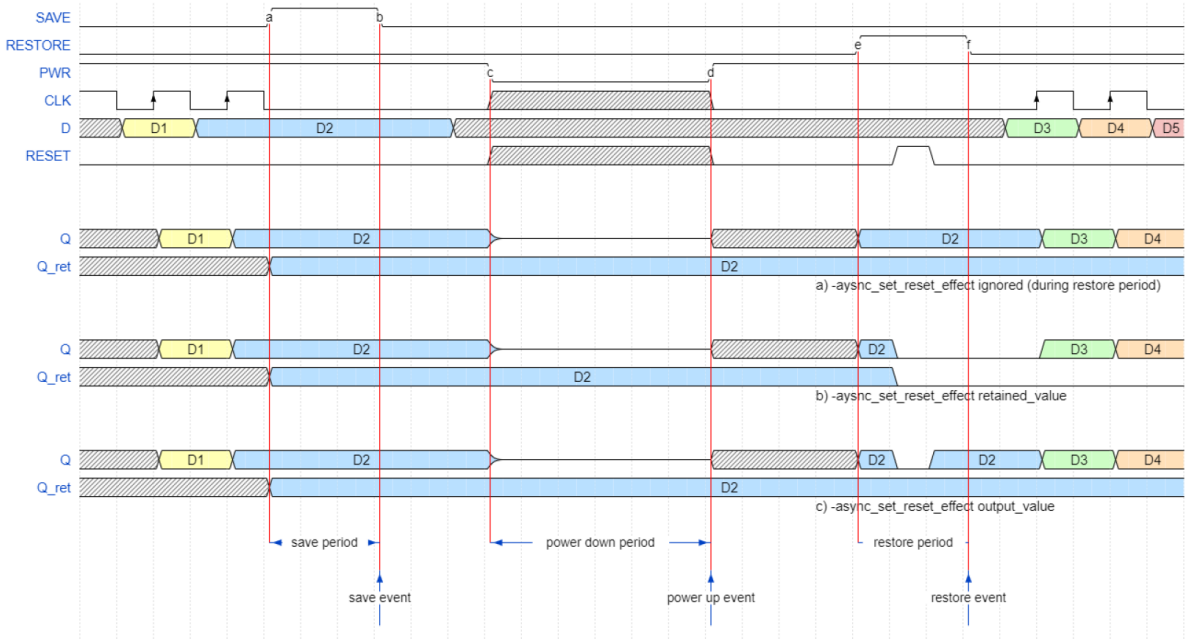


Figure 9. Dual-Control Retention: Reset Behaviour [4]

VI. Conclusion

This paper presents proposed syntactic and semantic enhancements to existing UPF commands to support evolving advanced SR strategies. It throws light on some insights how SR strategies can be effectively implemented to balance energy efficiency, system performance, and data integrity, making it an essential component of modern LP design flows. It presents recommendations on best practices for designing and verifying SR strategies in the current, continuously evolving and future energy-aware electronic systems.

Acknowledgment

The authors thank their colleagues of the IEEE P1801 Working Group for their creativity in devising many of the design examples of this paper.

References

[1] Vijayakumar Sankaran, et al, Modern Recipes for Brewing the Inevitable Methodology for Today's ICs: Low-Power Mixed-Signal Design Verification, Tutorial, DAC 2019.

[2] Lakshmanan Balasubramanian, et al, Advanced Low Power Retention Simulation Framework, Designer Track, DAC 2019.

[3] Lakshmanan Balasubramanian, et al, A holistic approach to low power mixed-signal design verification using power intent: CPF-Based Interface Elements, Methods, and Guidelines, DVCON 2016.

[4] IEEE Standard 1801-2024, "IEEE Standard for Design and Verification of LP, Energy-Aware Electronic Systems," 2024.