2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE
MUNICH, GERMANY
OCTOBER 15-16, 2024

Retention Sufficiency Validation for Optimizing State Retention Cells in Low Power Design

Nitesh Kalwani, Lead Product Engineer, Cadence Design Systems, Noida, India
(*knitesh@cadence.coml*)

Mateus Silva, Principal Software Engineer, Cadence Design Systems, Belo Horizonte, Brazil
(*msilva@cadence.com*)

***Abstract*— This paper introduces a new verification methodology, termed "Retention Sufficiency", designed to optimize the use of state retention cells within power-aware designs. State retention cells are critical for preserving the state of key elements during power-down modes in low-power circuits and allow the system to resume operation seamlessly upon power recovery. A significant challenge in power-aware design is determining the essential state elements that require retention. Many designers, facing such difficulty, adopt a conservative strategy of over-retaining registers within collapsible power domains beyond the functional necessities of the design. This leads to increased circuit area and leakage power, as retention cells must remain powered, adversely affecting the design's Power, Performance, and Area (PPA) metrics. The industry is thus seeking methods to pare down the number of state retention cells without compromising power-aware design integrity. Traditional random verification techniques may overlook critical bugs and cannot assure the complete safety of retention reductions. In contrast, the proposed Retention Sufficiency approach leverages Formal Property Verification (FPV) and Sequential Equivalence Checking (SEC) to assess retention reductions on state elements. This paper details the methodology for implementing the Retention Sufficiency verification process at the block level, offering a more reliable and efficient path to optimizing state retention in low-power designs. A Proof of concept (PoC) of the Retention Sufficiency flow show how a complete retained block can drop its retention use by 22% and customer experiences already show up to 15% of reduction.**

I. INTRODUCTION

CMOS power consumption in the advanced technology nodes is dominated by the leakage power [1]. Given this scenario, power shutoff technique arose to save energy in cutting-edge SoCs. It dynamically disconnects the power supply from the logic gates of the target system. This causes the loss of the values kept into their state elements. State retention cells are used to keep track of the values of essential state elements required for correct operation of the circuit after it wakes up from a power shutoff [2]. These values are saved into the retention cells prior to the power shutoff and restored into the design registers right after their voltage supplies are re-enabled. Identifying the minimal set of candidates (state elements) for state retention is a difficult task, due to which majority of the power intent designers end up with a conservative approach of over retaining the design registers in collapsible domains, even though less state retention cells should be required to guarantee design functionally after a power recovery. This causes an increase of the circuitry area and leakage power as the retention cells must be always powered on, thereby it impacts the overall Power Performance Area (PPA) of the design. Due to this conservative approach of over-retaining the state elements, there is a widespread industry demand to optimize the state retention cells within low-power designs. However, identifying an optimal - or even a suboptimal - set of non-essential state elements is challenging, given the absence of an automated process capable of producing the necessary list of (sub)optimal retention cells for a given power-aware design. Instead, this can be achieved through an approximate analytical method that identifies likely non-essential retention candidates, such as synchronizers and redundant design flip-flops, for retention minimization. Subsequently, the absence of these suspicious non-essential retention candidates must be verified to guarantee the correct functional behavior of the design.

State retention cells are pivotal in preserving and restoring logic states when using power shutoff technique. Overlooking retention for critical state elements can result in silicon defects. Traditional simulation methods, which employ random vector-based approaches, fall short in ensuring the safety of retention optimizations [3]. This limitation has spurred interest in formal verification techniques to tackle the intricate challenges of retention optimization and sufficiency in the Low Power Design Verification field. Historically, formal tools have not been used widely for Low Power Verification, with simulation being the preferred method throughout the industry.

However, over the time this trend has changed, and a shift left is occurring as Design Verification (DV) engineers, driven by the desire to integrate verification earlier in the design process, are beginning to harness formal methodologies. They are crafting custom assertions and coverage properties to validate the functionality of Power Management Unit (PMU), isolation, and retention cells during power down scenarios [6].

The formal-based flow proposed in this paper- Retention Sufficiency, has been developed to ascertain the safety of retention cell optimizations in low-power designs. This method employs Sequential Equivalence Checking (SEC) to compare two different versions of the same power-aware design in Register Transfer Level (RTL) format. The 'spec model' (SPEC) retains all state elements, incorporating both RTL and the Unified Power Format (UPF), while the 'implementation model' (IMP) omits certain non-essential state retention cells identified by the designer, reflected in an altered UPF (UPF'). The SEC rigorously validates the two models, determining their equivalence. If SEC confirms all formal properties, it validates the equivalence of the two power-aware design versions, indicating that the non-essential elements identified can be safely excluded. Conversely, any failures in the formal properties prompt the user to adjust the UPF' list to achieve full equivalence with the original UPF. The resulting refined UPF' represents the optimal subset of state retention cells necessary to maintain design integrity during power collapse.

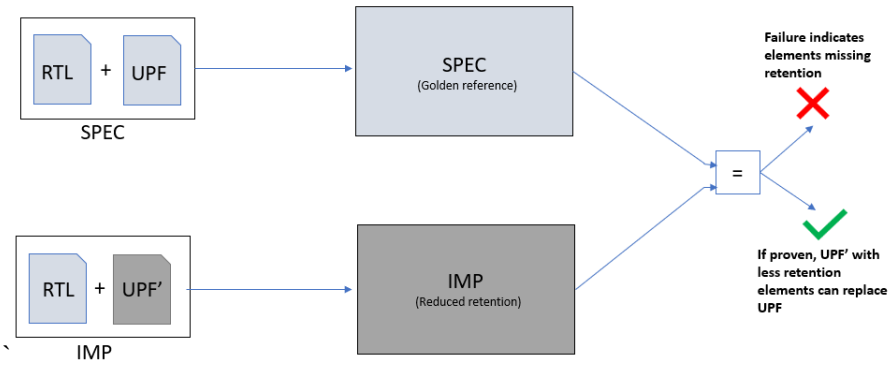

Figure 1: Flow Diagram describing the Retention Sufficiency flow inputs and expected equivalency results. UPF is the golden reference file with all registers retained and UPF' is the power intent file with retention excluded on the identified non-essential registers.

Achieving full convergence—eliminating non-determinism in target properties—is a critical goal in the sufficiency flow to ensure the safety of retention cell reduction. Consequently, selecting an appropriately sized design is a key consideration prior to initiating the sufficiency flow. Block-level designs are particularly well-suited for this process due to their lower setup effort and higher return on investment compared to full System on Chip (SoC) designs, which may not achieve full convergence and are less ideal for formal verification. Moreover, some trivial scenarios of retention exclusion could be validated using a Formal Property Verification (FPV) method to lighten SEC to reached better convergence for the difficult ones – this method will be covered in the upcoming sections.

In addition to design size, establishing the correct power sequencing to ensure proper power collapse and control toggling for low power features such as isolation and retention is another crucial factor in retention sufficiency flow. While power sequencers or the Power Management Unit (PMU) are typically built at the SoC design level, functional low power verification at the block level may require the manual integration of a power sequencer module that replicates the SoC's power sequence behavior. Binding this power sequence to the target Design Under Test (DUT) may not be straight-forward as it may not be clear the conditions or states in which the power collapse may be triggered by the power sequencer. Ensuring that the design conditions for power collapse accurately reflect the intended behavior at SoC level is vital for power-aware verification, as it directly affects the number of essential state retention cells for the DUT. The complexities and trade-offs associated with design conditions for power shutdown will be explored in greater detail in the subsequent sections of this paper.

Therefore, this paper presents the Retention Sufficiency flow, which is a methodology that ensures the safe reduction of state retention cells in a DUT. This flow hinges on three critical user-provided inputs: design intent files (including RTL and Liberty formats), power intent specifications (UPF), and a curated list of retention cells marked for potential reduction. Section II delves into the theoretical underpinnings of the flow, exploring the power sequencer methodology and various scenarios where retention can be securely eliminated from registers

targeted by the verification process. Section III outlines the verification methodology, which includes establishing a SEC setup, executing formal proofs, analyzing results, and refining the UPF. The paper presents empirical results in section IV obtained from a small design comprising approximately 500 flip-flops. Finally, section V concludes this paper highlighting the main outcomes of the proposed flow and its takeaways.

II. Theoretical Aspects

To understand the scenarios in which retention cells are not essential, the activity stages of a power-aware design must be defined. The reset state is the first relevant input for this problem since it is the start point for all the other reachable states. It prevents formal tools to report false assertion failures for scenarios that is not achievable given the reset configuration provided by the user. After that, the design is in the functional stage which covers its normal operation specified in Register Transfer Level (RTL). Then, the block may reach a state where a power collapse is desired; however, some operations may be performed before initializing it (e. g. flush of memories and local clock gating operation). This stage – kwown as "pre-collapse sequence" – will lead the design into a "idle" state, which is detectable by the Power Management Unit (PMU). It is reponsible for orchestrate the power collapse and bring up process by running a well-defined power sequence, described by a Finite State Machine (FSM). Once the design is powered up again and the DUT clock is re-enabled, it may be required some software (SW) configuration for some cycles to reach the functional state and carry out regular transactions. Figure 2 sumarizes these described stages.

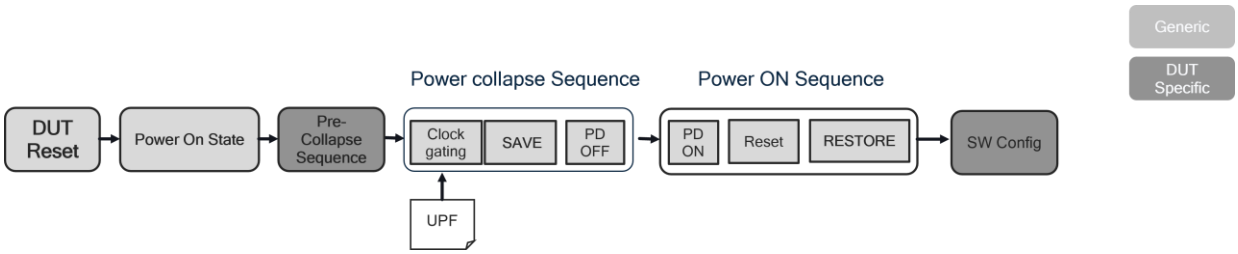

Figure 2 – Design stages present in a power-aware design. PMU is reposible for detecting scenarios for power collapse and bring up according to the designed power intent. Pre-Collapse Sequence and SW config are DUT specific and may not be required in some blocks.

The power sequence implemented by the PMU is composed of well known states needed to be the power shutoff technique [3]. Figure 3 shows a typical power sequence waveform. It begins with the IDLE state, which means that the block is in functional mode. Once the power collapse condition is met, the power collapse trigger (pdown_h) will be active and the power down sequence is run: all remaining DUT clocks are disabled (CLK_DISABLE), the values into the retained flops are saved, and the power domain isolations are enabled. Then, the design is finally ready to have its supplies turned off, which menas that all design states will be corrupted - undefined. The PMU will detect when the block is required to be in function mode again – in this case when pdown_h is deactivate – to run the power bring up states: asyncrounous reset on some design flops (RESET_ASYNC), the value restoration into the retained flops (RESTORATION), deactivation of the isolations (DEISOLATE), the DUT clocks are re-enabled (CLK_ENABLE), and finally the block will be at the functional mode again (IDLE). These are the mandatory states of almost any power sequences implemented by PMU, but there may be some design-specific aspects embedded on it as well. These control signals are connected to the low power cells (e.g isolation, retention and power switch) created by the strategies defined in UPF.

A flop may not require having its value retained in the following scenarios [4]:

1. The target register is always written by its fanin logic cone before it is read by fanout registers or observable at the primary outputs.

2. The target register always has the same value as the saved value into the retention cell during restoration phase of the power sequencer.

3. The value saved into the retention cell is always the same. In this case, RTL could be modified to load the constant value into the state element via reset whenever the circuit is powered on again.

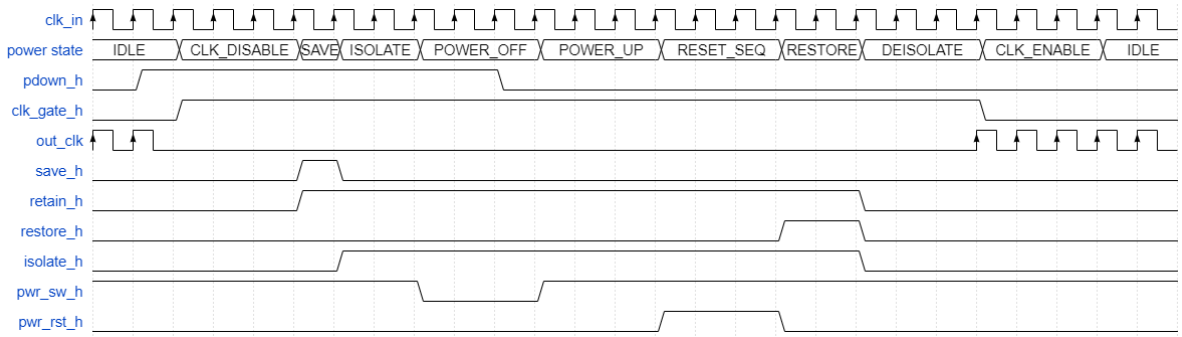

Figure 3 – Typical power sequence waveform implemented by PMUs

There is an observable trade-off between the openness of the scenarios in which a power collapse may happen and the number of retention cells required to guarantee a circuit recovery from power shutoff without any functional flaws. For instance, if the power intent has the requirement of resuming pipelined operations (e.g a float point multiplier) after power shutoff, retention cells will be required on data path registers. However, this may not be a common power intent, as the power collapse is usually triggered when the FSM that controls the pipelined data path is at the idle state. Hence, there will be a design condition which is determined by some registers with concrete - specific - values prior to the power collapse. Figure 4 represents it by the "COLLAPSE CONDITION" set. Also, "POST-COLLAPSE RESET" defines the registers that have asyncronous reset value activated by "RESET_ASYNC" state showed in Figure 3 and "RETENTION ELEMENTS" are the retained flops defined in UPF. These two sets defines what is needed to be concrete at the end of the power bring up sequence to guarantee the correct block fuctionality. However, "RETENTION ELEMENTS" may have redudant state retentions cells according to the scenarios stated before in this section.

The intersection of these three sets (region '5') is the most straight-forward case to remove state retention cells because the power collapse condition leave these registers with the same concrete values as the ones assigned in post-collapse reset stage. In the same way, state retention cells on registers that are part of region '6' are also redundant because they will also have a concrete value prior to power colapse. However, besides removing the retention element from UPF, it is also required to add post-collapse reset for these flops – which involves a RTL change. Region '8' represents a set of retention cells that will not affect the primary outputs before being overwritten no matter other register values are at the power collapse. There are also some tricky cases with multidependency between the flops, meaning that one retention cell on a specific register can only be removed if another set of registers continue to be retained. That will determine few possibilities of retention subsets to be removed, and the biggest one should be chosen – the region '11' in this case. Next section will describe how FPV and SEC techniques can be used together to find out these scenarios of redundant retention cell.

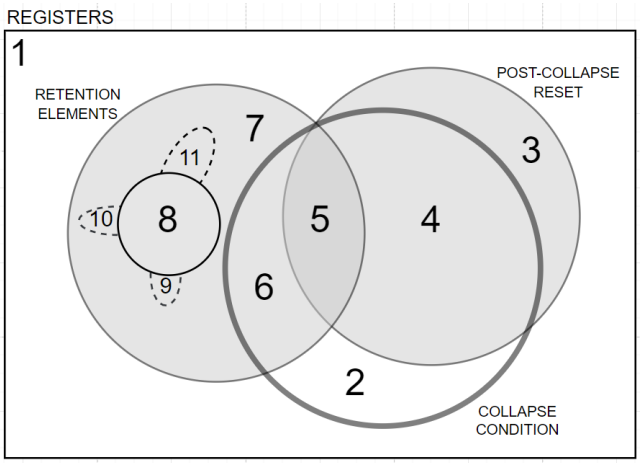

Figure 4 - Venn's diagram to describe the retention sufficiency problem. The union between "POST-COLLAPSE RESET" and "RETENTION ELEMENTS" sets represent the state elements that have concrete values after power recovery (in grey). "COLLAPSE CONDITION" set encompasses all concrete values required to initiate a power collapse. This set will directly affect the optimal number of essential state retention cells required to keep correct functionality after power recovery.

III. Verification Methodology

Two fundamental inputs based on the design knowledge must be provided to enable retention sufficiency flow: the set of retention elements to be removed and the scenarios in which power collapse may happen since PMU is not integrated with block level design. The preciseness of the prior will reduce the number of debugging iterations needed to find out the over retained state retention cells. Also, the accuracy of the latter is crucial because they will be mapped as constraints in formal tools, and wrong assumptions may cause false results if they do not reproduce the real scenario. Figure 5 describes the main steps in the retention sufficiency methodology: the "concreteness check" verifies the number of retention cells (k) that have a concrete value whenever a power collapse is about to happen and if this concrete value matches post-collapse reset values (n) and "SEC analysis" checks for the remaining targets (m-k) whether corruption values stored into the retained registers due to power down are able to leak at the block outputs (i) or not (o). Even though all retention reduction targets could be ultimately handled by "SEC analysis", "Concreteness Check" is a less expensive check that can reduce the number of targets to be dealt by "SEC analysis". At the end, the three categories for retention removal detailed in the previous section are obtained by this verification flow.

The "Concreteness Check" creates a property for each target state retention cell to verify whether it is concrete or not when a power collapse is about to happen. To get some value to be used as reference in the checks, a cover property could be created on the trigger signal of power collapse and the values of the target retained registers can be extracted from this cover waveform. Then, an assertion can be created to check if the outputs of these targets is always equivalent to these extracted values whenever a power collapse is eminent. This approach tends to converge faster than "SEC analysis" since the target is the register output instead of certifying that all block outputs are equivalent. Hence, it catches the trivial "k" cases to alleviate the "SEC analysis", which will require less changes between SPEC and IMP ("m - k").

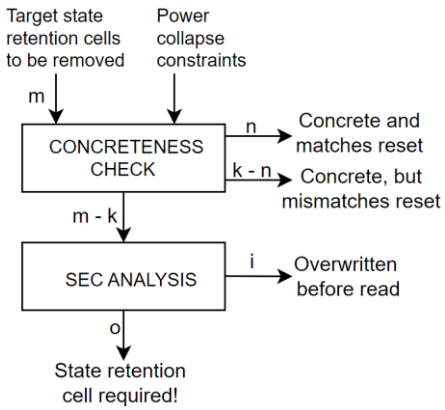

Figure 5 – Verification steps in Retention Sufficiency flow

Some additional steps are required to set up the "SEC Analysis". The validation process begins with initializing the power-aware design within the SEC App. On the Specification (SPEC) side, the original power aware design, including both RTL and UPF, is elaborated. Conversely, on the Implementation (IMP) side, the same RTL design but with a modified UPF (denoted as UPF') that omits retention on the "n - c" candidates are elaborated. UPF' is generated by removing these registers from the retention list. To accomplish this, one can use the -exclude_elements option within the existing set_retention strategy, or alternatively, establish a new set_retention strategy with the -no_retention option to specify the registers that will not retain their states in the synthesized design after power model generation [5].

The subsequent phase involves configuring the SEC setup. This setup is critical for managing undriven signals, uninitialized registers, and potential sources of low power corruption, which helps prevent false counterexamples (CEXs) during the verification phase. It also includes mapping all primary inputs, black-box outputs and any other internal undriven signal on both the SPEC and IMP sides. Also, X values of uninitialized registers are mapped between SPEC and IMP to be equivalent. Within the SPEC and IMP models, these mapped undriven sources are driven by formal engines, while primary outputs serve as observation points to detect any discrepancies that could lead to functional differences between the SPEC and IMP caused by retention reduction, which is the only difference between the two design versions.

For either “Concreteness Check” and “SEC Analysis”, power sequencer – implemented by the PMU – must be tied to the target DUT, which is vital for the functional verification of the low-power design. While the PMU is usually integrated within system-on-chip (SOC) designs, standalone block or IP level designs needs an external power sequencer state machine to replicate the SOC's PMU functions. Due to the inherent scalability limitations of the formal flow, Retention Sufficiency flow is best applied to block or IP level designs that are amenable to formal analysis and exhibit a high rate of property convergence. Finally, once the PMU setup is finalized, a formal proof is conducted on the SEC properties, which are automatically generated by SEC for the primary outputs. A successful proof across all targeted SEC properties confirms the equivalence between the two versions of the power-aware design. This validation ensures that the non-essential retention cells identified earlier can be safely excluded, thereby optimizing the retention modeling within the design. Figure 6 highlights the “SEC Analysis” process.

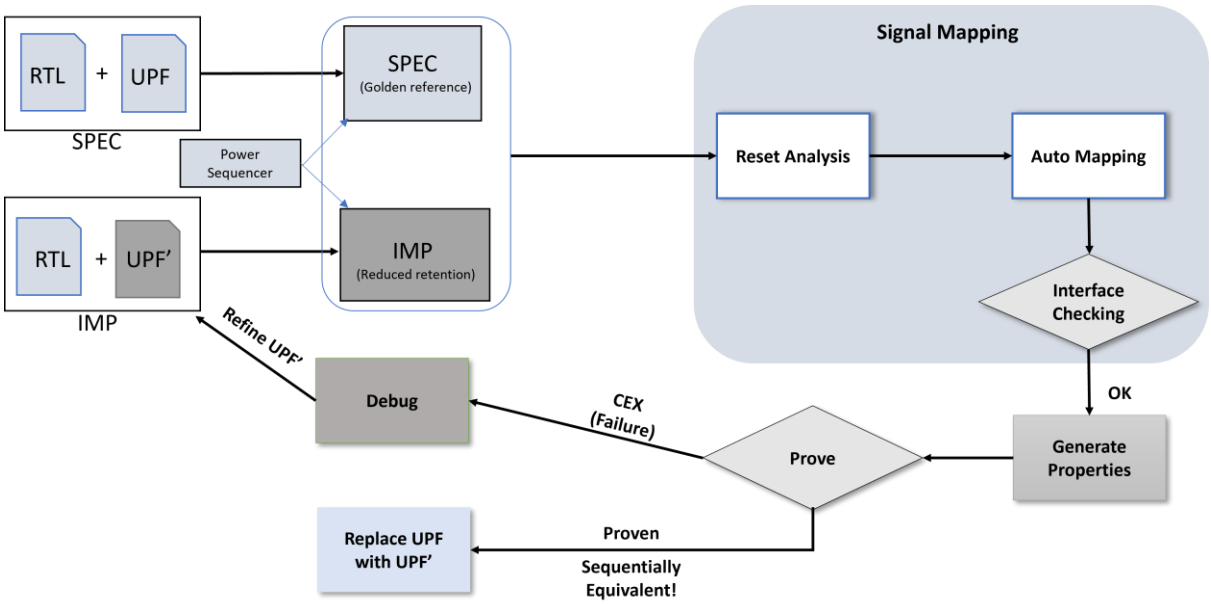

Figure 6 – The “SEC Analysis” Flow. It comprehends power-aware design bring-up (RTL+UPF vs RTL+UPF’), formal setup (clock, reset and DUT constraints), power sequence bind to the low power control signals and the power collapse conditions, SEC mapping and property generation, formal proof analysis and waveform debugging.

During the SEC Analysis, the prove may either confirm the validity of the identified retention cells (with full proven results) or reveal failures (CEXs). A full proven result indicates that the state retention cells excluded in UPF’ can be reliably excluded, while any failures suggest that UPF’ removes essential state retention cells. To address these failures, we must scrutinize the CEXs generated by the formal analysis to pinpoint the specific retention cells responsible for the functional discrepancies. This involves identifying which cells, when retention is removed, lead to mismatches. Failure analysis is often a repetitive process because each failure trace is associated with a particular functional discrepancy linked to a specific retention cell – or a small set of it. It is possible that multiple retention cells excluded in UPF’ may cause a CEXs. The debugging process continues with the elimination of all problematic retention cells from the exclusion list. The refined list, which allows all properties to be proven together, will then represent the final list of non-essential retention cells that can be safely eliminated. Given that this analysis is iterative, it is crucial to strategically select the retention cells for removal based on an in-depth analysis of the design's behavior. Random reductions in retention cells could inadvertently lead to functional failures, underscoring the importance of a rational approach for the selection of the target exclusion list.

IV. Results

An internal block - the Packet Processor (PP) - has been used to demonstrate the “Retention Sufficient” flow as a PoC. This block parses, stores, and routes packets through two interfaces: ingress (ig) and egress (eg) interfaces. The design serves as a buffer for packets that are waiting to be consumed by a multi-threaded CPU which will perform further analysis on each packet. Packets enter the design and are marked as pending until a third interface (route) issues a decision on how to route them. It also has an interface for interruptions (intr) and another one for configuration registers using an Advanced Peripheral Bus (APB) interface. Figure 7 shows the IP interfaces. This block has two top level clocks: a fast clock for APB and a slow clock for the remaining subblocks. Then, synchronizers are required these APB registers to transmit data to the other subblocks. Also, every subblock has a clock gating cell that automatically disables its internal clock when idleness is detected. The total number of registers of this IP is 470 registers (2389 bits).

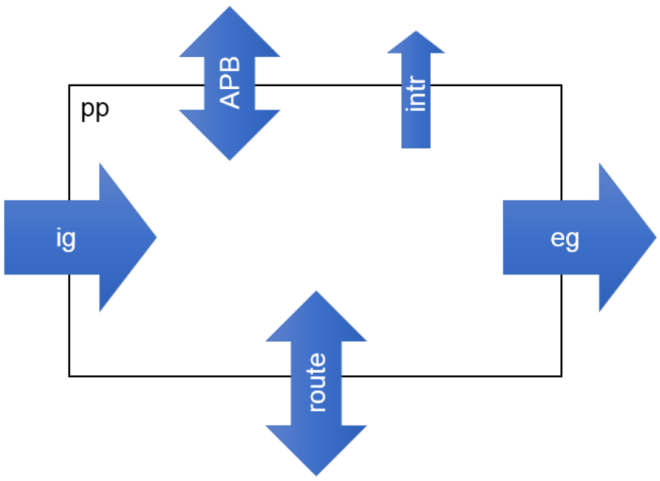

Figure 7 – Top-level interfaces of Packet Processor.

This block has a single power domain, and all registers are retained initially. For this PoC, a scenario condition for power collapse was established to when all clock gating cells of the internal subblocks are deactivated. That means complete idleness of the block. Another power collapse condition is that the APB synchronizers should not be processing anything prior to the power collapse. Like any formal verification flow, the protocol rules of the block interfaces should be incorporated as assumptions to avoid any false failures. Also, some block outputs must also have “disabled iff” conditions - data outputs can only have their equivalence checked if their related valid signal is asserted. After this setup is done, then “Concreteness Check” and “SEC Analysis” can be performed.

For the “Concreteness Check”, all retention cells were evaluated. 45 state elements have concrete values right before power collapse, and 31 matches and 14 mismatch the post-collapse reset. For the matched ones, most of them are related to control registers that assume reset values as there is not any packet transmission or reception. For the 14 mismatches, they are related to status registers related to procedures done after reset to allow functional operation of the block (i.e memory initialization).

For the remaining retention registers, synchronizers of the configuration registers were checked with “SEC Analysis” by removing them in UPF’. To avoid leakage of the corruption values of these synchronizers at the DUT outputs after power recovery, only retention cell is applied in the first element of the register chain and its propagation to the output of the synchronizer should be done before entering to the functional mode. With these design changes “SEC Analysis” could prove all the 74 synchronizers – each one has 6 registers. Even though the power collapse condition happens when the block is idle (no active transmission or reception), that does not mean that the memory (132 registers) is not filled with relevant info. In this case, the power collapse could only occur when memory is empty, which may not be the design intent. The remaining registers were not analyzed, but they are likely to require state retention cells as most of them are control signals. The “SEC Analysis” could be performed on each of them to understand if they are required or not. Table I summarizes the results.

Table I. – Retention Sufficiency Results for PP

Category	# of registers (bits)	Action
Concrete with post-collapse reset match	31 (66)	Remove elements from UPF rule
Concrete with post-collapse reset mismatch	14 (23)	Change post-collapse reset
Synchronizers	74 (444)	Remove elements from UPF rule
Memory registers	132 (1460)	Restrict power-collapse condition or keep retention on them
Other registers	219 (396)	Perform detailed analysis

Retention Sufficiency flow has been successfully validated on <CUSTOMER> design with 11881 flops out of which 1804 synchronizer flops were successfully proven as non-essential for retention. Accounting to ~15% retention optimization in the target block.

V. CONCLUSIONS

This paper introduced the Retention Sufficient flow, which is a formal-based approach that enables finding non-essential state retention cells. As functional aspects are not negotiable, this method provides a safe solution to increase PPA, which is not necessarily guaranteed by random vector verification methods. However, the applicability of this formal-based flow is limited by design size and a single power domain. Moreover, the selection of retention cells to exclude is crucial; it must align with the design intent and functional modes to minimize redundant debugging and proof cycles when refining the non-essential retention list. The manual binding of the power sequencer to the DUT poses a challenge since it requires a manual constraint of the power collapse trigger, which is equivalent to the state the DUT should be to allow power collapse to happen. This must be correct to obtain accurate results from this flow. Even with that, section IV demonstrates that effective results can be achieved using just high-level knowledge of the DUT.

REFERENCES

[1] Bikki, Pavankumar, Pitchai Karuppanan, et al.: SRAM cell leakage control techniques for ultra low power application: A Survey. Circuits and systems, 8(02):23, 2017.

[2] "Low Power Methodology Manual: For System-on-Chip Design" by Michael Keating, David Flynn, Rob Aitken, Alan Gibbons, and Kaijian Shi

[3] Shlomo Greenberg, et al. “Selective State Retention Power Gating Based on Formal Verification” IEEE Transactions on Circuits and Systems I: Regular Papers (Volume: 62, Issue: 3, March 2015)

[4] Darbari, Ashish, et al. "Selective state retention design using symbolic simulation." *2009 Design, Automation & Test in Europe Conference & Exhibition*. IEEE, 2009.

[5] IEEE 1801 - Unified Power Format (UPF) for specifying power-saving techniques in the design process.

[6] Reza Sharafinejad, Bijan Alizadeh “UPF-based Formal Verification of Low Power Techniques in Modern Processors” IEEE VLSI Test Symposium, 2015