Automation of Power On Reset Assertion

Shang-Wei Tu, Penny Yang
Design Technology Division
MediaTek Inc.
HsinChu, Taiwan
Kuma.Tu@mediatek.com
Penny.Yang@mediatek.com

Joydeep Gangopadhyay, Amol Herlekar
Verification Group
Synopsys Inc.
Bangalore, India
Joydeep.Gangopadhyay@synopsys.com
Amol.Herlekar@synopsys.com

***Abstract* – Verifying the reset scheme of a System on Chip (SoC) is crucial to ensure that the device can be reset from any circumstance to a normal state. Nowadays, with many IP blocks integrated in a SoC, the reset scheme could be very complicated. The problem is even more complicated when the power management designs are introduced. This brings a significant challenge for verifying the reset scheme of a SoC. The power-on reset (POR) failure is one of frequently found issues in low power designs. Once this problem is found in real silicon, the only way to workaround this issue is disabling the power gating, but this will cause huge power performance degradation. Another way to solve this issue is respinning. However, either way is painful. Therefore, we propose an automatic solution for extracting POR assertions to identify all non-reset registers in the power-aware simulation. With the proposed solution, we can screen out many real POR failures with the extracted POR assertions in the early RTL simulation.**

Keywords—Lower Power Verification; Power-on Reset; Assertions ; Power-aware Simuation; UPF

I. INTRODUCTION

The power management design has become essential for a mobile device due to strict requirements of extending battery life. Advanced low-power design techniques, such as power gating, achieve significant power savings but also introduce significant design and implementation complexities. Thus, low power designs become highly risky without verifying properly. However, traditional functional verification does not take into account the design's power intent (as defined by UPF), and cannot cover the power management aspects of the design during verification. Therefore, low power verification plays an important role in verifying low power SoC designs [1]. Low power verification covers verifying power-on reset sequence, power mode switching, and so on, where the IPs and power domains are shutdown and turned-on with different scenarios. Besides, low power verification plan should include creating assertions for checking various control signals. The potential targets for adding checkers could be the power control sequence, the clock-gating logic, POR after power up, etc.

The challenges of the reset verification have also been discussed in [2]. POR is a key functional sequence for all SoC designs. POR is supposed to be able to bring the system from any state to a normal state. Any undetected POR bug can lead to disasters in real silicon. Complexities of the reset logic pose increasing challenges for verification engineers to catch such design issues during RTL/GL simulations. The ever increasing size and complexity of the SoCs are accompanied with increasing complexity in its power-on reset logic. Recently, the power management logic further complicates this issue. The factors resulting in increasing complexity of the reset logic are summarized below:

1) *There are usually several sources for the global reset, such as power-on reset, hardware reset, software reset, and watchdog timer reset.*
2) *Number of IPs integrated in a SoC continually increases.*
3) *The reset strategy in different IPs may be different (synchronous reset, asynchronous reset, scan based reset, partial reset, full reset, or non-resettable flops).*
4) *The reset controller logic in a SoC must fulfill requirements of resetting all digital blocks, analog blocks, and different Phase-locked-Loops (PLLs).*
5) *Further complexities are added due to supporting various operation modes which are configured by the values provided from SoC pads.*
6) *Another design complexity will be brought in due to complex power modes and the corresponding transitions defined for different use cases.*

7) *The reset scheme could be even more complex when retention registers are introduced.*

Complexities in SoC reset logics pose various new challenges for the verification with limited resources and within limited SoC execution time. The major verification challenges are listed below:

1) *The SoC reset sequence generates different resets for different blocks in a defined sequence. This sequence should be followed to ensure all blocks can be brought to a normal state. Hence, the verification is required to ensure that various combinations of operating modes of different blocks are covered and checked.*

2) *Non-resettable flops used in POR sampling logics may result in the X-Propagation due to uninitialized flops during RTL and GLS simulations. Hence, to verify such flops, we have to initialize them with forces/deposits with defined/random values at reset time to allow simulations to proceed.*

3) *The various supported SoC functional configurations based on values from pads need to be verified thoroughly. Various functional scenarios covering all possible modes need to be run at SoC level to ensure all configurations working as expected.*

4) *Top-level reset signals should be connected to all resettable flip flops in the design, i.e., all resettable flip flops in the design should be reset when the global reset is asserted. In fact, a missing reset connection or reset signal blocked during propagation between different power domains can easily go undetected.*

Currently, most of proposed works for the reset verification are focus on the connectivity check. However, the root causes of POR failures in a low power design could be diverse. Some are "static" and can be caught by inspecting the design structure, but some are "dynamic" and can only be caught during the power-aware simulation. Therefore, we propose an automatic solution for extracting POR assertions to identify all non-reset registers in the power-aware simulation. With the proposed solution, we can screen out many real POR failures with the extracted POR assertions in the early RTL simulation.

The rest of this paper is organized as follows. In section II, we will compare our proposed solution with Mediatek's previous work. Section III explains detailed tool flow and results. Finally, conclusion and future work will be covered in section IV.

II. Comparison With Previous Reset Verification Method

In Mediatek's previous work [3], we have presented how to use formal techniques to verify global reset schemes. Global reset failures could be caused by missing/wrong reset connections, design bugs which reside in the logics between global reset source and the reset pin of the intended storage element, etc. Instead of crafting simulation based tests for all possible scenarios to verify if global reset can really be propagated to the intended storage elements and get its job done to do the reset work, it is much more efficient to add SVA on the reset pins of the intended storage elements then let Formal engine to prove that these reset pins will be asserted when the designed reset condition is met. An example SVA is shown as the following.

global_reset_check: assert property

(@(global_clock) `GLOBAL_RESET |-> ##`DELAY _rst);

`GLOBAL_RESET*** is defined by the global reset condition which is composed of any legal SVA expression to make up the antecedent part of this global reset verification SVA. ***`DELAY is decided by the actual cycle delay between the global reset condition is met and the reset signal is asserted. To eliminate unnecessary false firings, the maximum delay could be used. ***_rst*** is the signal which is directly connected to the reset pin of the intended storage element. The aim of this methodology is targeted to verify the connectivity and logics between the global reset and the reset pin of the intended storage element. The same philosophy is applied on the `GLOBAL_RESET part. Except for the required logics which make up the global reset condition, we should leave as many logics as possible for formal engine to verify.

However, there are still some disadvantages of using formal techniques with global reset assertions. Thus, we develop power-aware simulation with POR assertions to overcome the disadvantages. The comparisons of these two methods are summarized in TABLE I.

TABLE I - Comparison of Mediatek's Reset Verification Methods

Feature/Method	Global Reset Assertions + Formal Techniques	Power-on Reset Assertions + Power-aware Simulation
Checking mechanism	Static, structural	With dynamic power control sequences
Power aware	No	Yes, with UPF
Clocking event	Must	Not necessary, use user specified time interval
Coverage	Exhaustive	Pattern dependant
Tool capacity	Limited by design complexity, abstraction techniques are needed for whole chip	Able to run whole chip simulation (limited by machine memory)
Software included	No	Yes

Formal verification techniques have the strength that can exhaustively explore all possible conditions with respect to the SoC reset scheme, while the coverage of simulation-based techniques is limited by provided patterns. However, formal techniques for the reset verification mainly focus on static and structural checks. Formal techniques have limited ability to describe dynamic power control sequences and cannot include the software or firmware in the verification. Besides, currently, most formal tools are not "power-aware", i.e., UPF are not considered. Another limitation of formal techniques is that they require clocking event for the verification. However, "asynchronous" resets are usually non-clock-based. Among all the disadvantages of the formal techniques, the major one is the limitation of the design complexity. This complexity limitation results in requirements of various reduction techniques for fitting a design into a formal engine. This introduces barriers for engineers to adopt the formal techniques in the execution. Therefore, in order to overcome all the disadvantages of the formal techniques, our proposed solution is the automatic POR assertions in UPF-based power simulation.

III. Tool Flow and Results

Figure 1 shows the flow diagram for our proposed solution. The required inputs of the flow are the software/firmware which is optional, the testbench, RTL codes, and UPF codes. The testbench together with the software/firmware is for stimulating the POR with different sources and conditions, and the RTL codes are for extracting POR assertions. The UPF codes are for turning on the power-aware simulation and provide the power on trigger events for POR assertions to check. The first step of the flow is parsing and compiling the input source codes. During this step, POR assertions are automatically extracted and bounded to registers with asynchronous set, reset or even both by identifying the predefined coding style in **Figure 2**. One thing we have to mention is that only asynchronous set/reset registers are considered in this work, since most of our designs are implemented with asynchronous set/reset registers. The second step of the flow is the power-aware simulation with the extracted POR assertions. During this step, the software/firmware together with the testbench will stimulate the design with different low power and reset scenarios, and, hence, simulator will generate the power on trigger event for POR assertions to check. Finally, when the simulation is done, the failures of the assertions will be shown in the simulation logs for review.

We collaborated with Synopsys to implement this solution in VCS NLP [5] from version 1209SP1-7 onward to capture the POR issues in power-aware simulations. By parsing and identifying the coding style in **Figure 2**, POR assertions are automatically generated and bound to registers to check if they can be reset correctly when they are re-powered. These assertions have been implemented only for flops with asynchronous reset present in Verilog modules and not for flops which are presented in VHDL part of a design or synchronous flops currently. In short, these assertions are implemented in Verilog modules, and these Verilog modules are instantiated per qualified flop in the design with information like the power domain state, clock and reset signal values, reset_sense (polarity of reset) as port connects. The clock and reset signal names presented as part of the assertion message are instrumented as string constants which were populated through an initial block. The hierarchical path of the UPF power domain name and the flop are instrumented using simulator DPI calls.

After discussing with design and verification teams, we conclude 3 reset scenarios as shown in **Figure 3**. These 3 reset scenarios can cover all the reset behaviors in our designs. Following are the definitions of *Tmax* and *Twidth* in **Figure 3**:

1) *Tmax :* reset should be asserted within *Tmax* time unit after power on.
2) *Twidth :* reset should be lasted at least *Twidth* time unit after power on.

where the time unit is defined with the *timescale* specified in the Verilog code or the simulator option.

For case 1 in **Figure 3**, only *Twidth* will be checked with re-powered registers, since the reset signal keeps active during whole power off period. For case 2 and 3 in **Figure 3**, both *Tmax* and *Twidth* will be checked with re-powered registers, since the reset signal asserted after power on. We should note that the checking is based on the simulation time interval specified by users, not clock cycles because there may be no clock event when the reset is asserted after power-up.

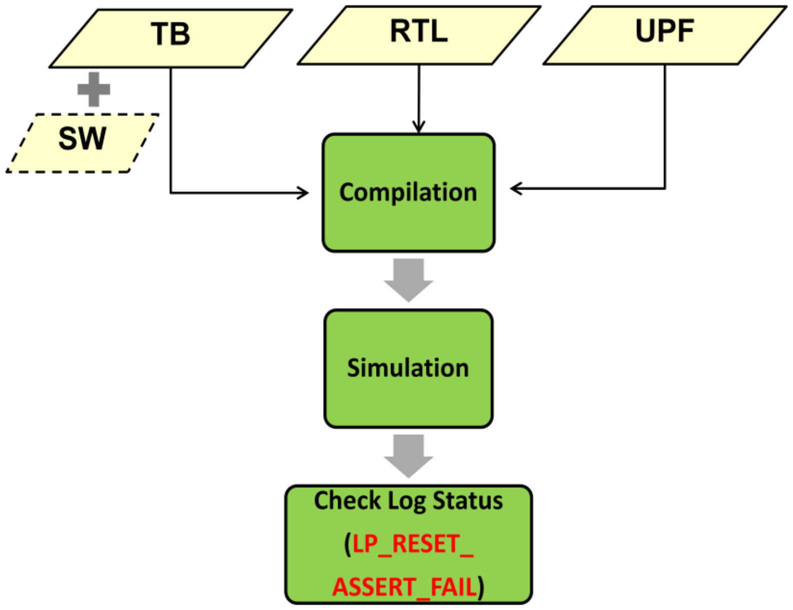

Figure 1 - The verification Flow of Power-on Reset Assertions with power-aware simulation

```
//coding style 1
always @(posedge clk or negedge rstb)
    if (rstb == 0)
        q <= 1'b0;
    else
        q <= dint;

//coding style 2
always @(posedge clk or posedge rst)
    if (rst == 1)
        q <= 1'b0;
    else
        q <= dint;

//coding style 3
always @(posedge clk or negedge rstb or
posedge rst)
    if (rst == 1)
        q <= 1'b0;
    else if (rstb == 0)
        q <= 1'b0;
    else
        q <= dint;

```

Figure 2 - The coding styles for binding power on reset assertion.

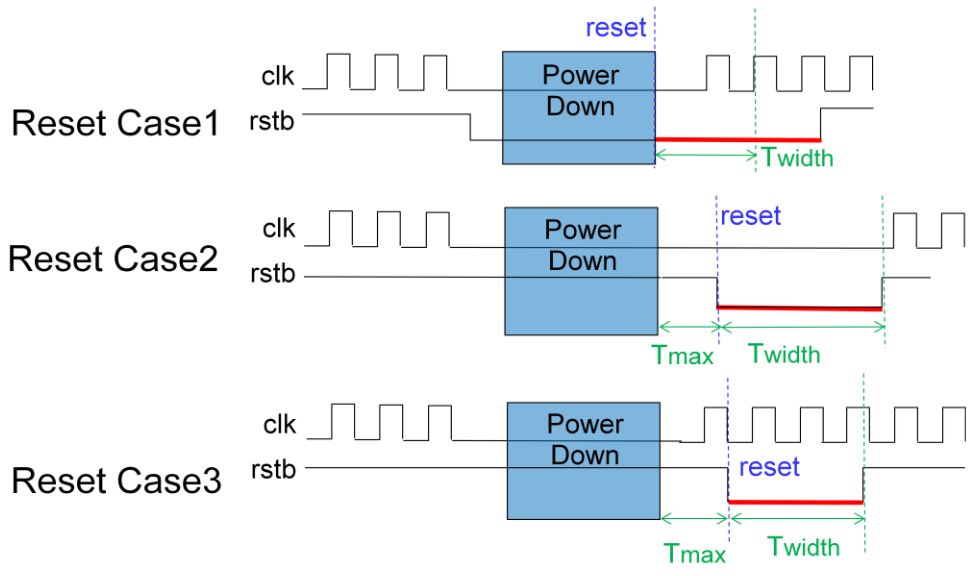

Figure 3 - The 3 reset scenarios that checked by power on reset assertion.

To turn on the POR assertion, users need to add a compile-time option to VCS as shown in **Figure 4** and set *Tmax* (**SNPS_active_reset_duration**) and *Twidth* (**SNPS_inactive_reset_duration**) in UPF using **set_design_attributes** as shown in **Figure 5**. The delay values of *Tmax* and *Twidth* are propagated as package parameters (global) for all the assertions of the whole design flops. Then you can get the warning messages as displayed in **Figure 6** and **Figure 7** when *Tmax* or *Twidth* is violated during the power-aware simulation. The value of **SNPS_active_reset_duration** and **SNPS_inactive_reset_duration** are parts of the assertion messages in absolute values. It should be interpreted as resolved time units of VCS for that particular simulation run. The LP_RESET_ASSERT_FAIL assertion has been postponed to reactive region of the simulation queue to avoid firing of this assertion if the reset changes at the same time of power-up time + **SNPS_inactive_reset_duration** delay value.

With the proposed solution, we found real bugs in our SoC designs which have been verified by our previous formal solution. This proves that the proposed solution can overcome the disadvantages of the formal techniques.

```
vcs -power=assert_reset_sequence …
```

Figure 4 – The compile-time option to turn on power on reset assertion

```
set_design_attributes \
-attribute SNPS_active_reset_duration 20
set_design_attributes \
-attribute SNPS_inactive_reset_duration 15
```

Figure 5 – The UPF command to set active and inactive reset duration

```
[555000 ps][WARNING][LP_RESET_ASSERT
_FAIL] Reset has not been asserted
within ‘15’ timeunit after power up.
Instance: ‘tb.u_top.FF_sub’, Domain:
‘tb/u_top/PD_FF’, clock: ‘clk’, reset:
‘rst’.
```

Figure 6 - Warning message when *Tmax* is violated.

```
[575000 ps] [WARNING] [LP_RESET_
DEASSERT_FAIL] Reset is deasserted
within ‘20’ timeunit after power up.
Instance: ‘tb.u_top.FF_sub’, Domain:
‘tb/u_top/PD_FF’, clock: ‘clk’, reset:
‘rst’.
```

Figure 7 - Warning message when *Twidth* is violated.

IV. Implementation Challenges in tool

Since the majority of the reset techniques are employed as “Reset Case 1” shown in the **Figure 3**, the reset assertion on power-on is a challenge in low power verification. If resets are synchronous, then it can be assumed that it will take effect only once the clock gating is stopped on power-up and the reset is asserted. The problem is more complex in the case of asynchronous reset scheme defined in “Reset Case 1” than the synchronous reset case. Suppose the reset is asserted before the shutdown period, then the corresponding logic will get reset at the same time. Now since the power domain goes to shutdown, all the logic will be set to unknown by the simulator. On wake-up, since there is no negedge on the reset (assuming the reset is being driven from an ON domain), it becomes a challenge to reset the logics again on power-up. The problem is similar when the reset is asserted during shutdown. In this case, the negedge of the reset is lost during shutdown as all the design is set to unknown and cannot be reset. Therefore it becomes imperative for the tool to infer resets in the design for each of the sequential block. Once these resets are properly detected, their levels on wake-up are checked and the corresponding logics are reset with an induced edge.

With the need for reset assertion techniques, the tool needs to identify each local clock and reset for each sequential block in designs and the required checks for *Tmax* and *Twidth* need to be triggered on power-on. If the reset is already in an asserted state at power-on, only *Twidth* check should kick-in. If the reset is in a deasserted state at the time of power-on, *Tmax* check should kick in followed by *Twidth* check once the reset is asserted. Therefore, to properly implement POR assertions, special care has been taken to reinitialize the assertion sequence at the time of power down so that these assertions can be checked at every power-up of the domain throughout the simulation.

V. Conclusion

In this paper, we proposed an automatic solution for extracting POR assertions to find out all non-reset registers in the power-aware simulation. The solution has been deployed to multiple projects and has been proven to be able to overcome the disadvantages of the formal techniques, since they have caught design bugs which would be missed from the existing formal solution. Based on the experiences from the project execution, we have found that the POR assertions in power-aware simulation can be more comprehensive than the traditional static verification solutions. In summary, low power simulation with automatically generated assertions to verify the POR logics can benefit design teams with minimum barriers for adopting and much higher confidence on the verification results.

VI. Future Work

POR assertions have been proven to be a good solution in power-aware simulation using VCS-NLP. With the same concept, it is also possible to use VC Apps [5] (formerly Verdi Interoperable Apps) to develop in-house solution even with other simulators to achieve better customization. In addition, these assertions are currently only supported for Verilog constructs. Thus, it is still required enhancement for flops which are present in VHDL part of a design or synchronous flops. Besides, the **SNPS_active_reset_duration** and **SNPS_inactive_reset_duration** delay values are global constants currently. They can be made more granular controllable by making them as per power domain or flop. Moreover, the waiving mechanism is not included in the tool itself, and extra script is still required for post-processing the result currently. Therefore, users also require to enhance the tool to waive the registers which they do not care, e.g., testing logics. We will keep working on how to automate the reset verification. Our goal is to reduce manual efforts and achieve higher verification confidences.

Acknowledgment

We would like to thank Tiger Hsu from Synopsys Taiwan for helping communicate this enhancement request.

References

[1] Boobalan Anantharaman and Arunkumar Narayanamurthy, “Power Aware Verification Strategy for SoCs”, DVCON 2013.

[2] Deepak Jindal and Mayank Digvijay Bindal, “Gaps and Challenges with Reset Logic Verification”, SNUG 2012.

[3] Kaowen Liu, Penny Yang, Jeremy Levitt, Matt Berman, Mark Eslinger, “Using Formal Techniques to Verify SoC Reset Schemes”, DVCON 2013.

[4] Synopsys VCS NLP, http://www.synopsys.com/Tools/Verification/LowPowerVerification/Pages/MVSIM.aspx

[5] VC Apps https://www.vc-apps.org/