2023 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE 10 YEAR ANNIVERSARY

An Improved Methodology for Debugging UPF Issues at SoC level Power Aware Simulations

Ruchi Misra (ruchi.misra@samsung.com)

S Shrinidhi Rao (shrinidhi.r@samsung.com)

Alok Kumar (kumar.alok@samsung.com)

Garima Srivastava (s.garima@samsung.com)

Sarang Kalbande (sarang.mk@samsung.com)

Samsung Semiconductors India Research, Bangalore, India

***Abstract*— With ever-increasing SoC complexity and advanced power-aware architectures, the challenges involved with Low Power verification at SoC level have grown in substantial proportions. While Unified Power format (UPF) standard enables designers to add power intent for the design, one of the toughest challenges faced in semiconductor chip design and verification is debugging issues related to UPF syntaxes or strategies. Issues in the power format file, the test bench, design or the tool can cause significant turnaround time owing to large run times and iterations involved. For low-power design and verification, these debug challenges are further complicated as a result of the sophisticated power management architectures and techniques that are used. Moreover, the traditional debug technology and methods focus on issues found in a design working in always-on mode and fails to address the new and complex power-related issues thereby consuming more engineering time. In this paper, we will provide an in-depth analysis of various debug challenges and problems faced in low-power design and verification. By using relevant EDA tools and their targeted features, we will demonstrate how these issues can be either avoided or solved. Hardware engineers often do not leverage the debugging capabilities of EDA tools that allow user to debug the simulations in an interactive manner, leveraging breakpoints, live driver and signal analysis, power state networks and schematics. In this paper, we explore the methodologies involving interactive debugging for an efficient power debug and faster verification closure.**

Keywords—Low Power Verification; Live Simulation, Interactive Debug; United Power Format; System-On-Chip

I. INTRODUCTION

Today's most system-on-chip (SoC) designs are composed of different blocks running multiple applications with varying power requirements. This demands SoCs to partition design into multiple power domains as well as various operating modes with different and dynamically variable voltage levels. Interactions between these power domains are done through various power control logics like retention logic, isolation logic, level shifters, etc. The low-power designs demand aggressive power management, which demands complex design. As the complexity of the SoCs increase, the explosion in scope and intricacy caused by low-power design techniques cause significant difficulty of low power verification, as designer captures the power intent such as power switch rules, retention, isolation, and level-shifter rules in UPF (Unified Power Format) by separating from the RTL code.

The UPF is an extension of the TCL (Tool Command Language) and is defined independently of the HDL (Hardware description language). This makes the low power intent creation independent of the design in place. This drives early development of the power distribution architecture, enabling parallel yet independent design and verification of low power architecture alongside the RTL design and verification. Addition of power format files along with RTL increases the low power verification effort hence as a part of complete verification process, all the added power modes, power sequences, and power-mode transitions need to be covered such that the design meets both functional and power intent.

During the low power verification of a SoC with a real CPU RTL, test cases are written in C, and compiled using the standard compiler like ARMGCC. The hex code generated is then loaded on to the chip memory. When

the SoC reset is released, the power controller releases CPU reset, which reads the code from memory and executes instructions. The design along with UPF file is compiled using the EDA tool compiler. Compiler tools that support interactive debugging can be invoked by the use of TCL commands. Exploring signal probing, live signal forcing, enabling break points, step run feature along with the power state network, schematics and ability to dump voltage and power rail information enabled us with a very powerful tool to debug issues in Low power simulations

II. RELATED WORK

Series of programming languages like Specman-e, Cocotb/Python have been accustomed to the flexibility of interactive debug during run time. According to results in [1], employing an interactive debug library dramatically reduced debug turnaround time and regression run time. Engineers can re-run alternative scenarios in shorter time rather than waiting for hours for the test to complete. Multiple researchers have created libraries to support interactive debugging [1][2][3]). With interactive debug, user can run the simulation to a certain point in time, pause the simulation, perform debug actions and resume the simulation. Interactive debug in System Verilog is not a natively supported feature. Due to this, the user has limited control over the test scenario and program flow once the TB is compiled and simulation starts. If a user has to force or change stimulus to debug a long running test, precious time is lost which affects the verification closure time. With interactive simulation and debug support presented in this paper, one can pause the simulation at point of hang, debug, change stimulus if needed and resume to simulation, giving fine grained and advanced control over simulation flow. The work mentioned in this paper is uniquely fine-tuned to find multi driver issues, isolation issues and UPF syntactical issues across different compilers and hence is very useful in debugging.

III. FUNDAMENTAL STRATEGIES OF LOW POWER DESIGN

The low power designs leverage multiple techniques to achieve reduced power consumption during operation of the SoC [4]. These strategies are applied during the design and resource planning between multiple power domains and enabling low power yet efficient operation of the logic contained. Some of the key strategies utilized are:

i. Isolation: Isolation is a technique that prevents flow of crowbar current and spurious signal propagation and prevents a dead logic form driving an active logic. Isolation cells are placed in between two power domains, such that the second power domain is isolated from the first when the first power domain is off.

ii. Retention: Retention is a technique of saving certain logic state of memory state when the rest of the logic around it is powered down. This enables faster booting or logical operation once the surrounding logic is powered on. Retention cells ensure that retained logic is saved and restored without any possibility of corruption via leakage current.

iii. Level Shifters: Level shifters ensure that sufficient and optimal voltage is available for logic that is driven between two power domains. These can be High-to-Low or Low-to-High voltage shifters. This ensures that logic value sent by one domain is correctly processed in the next irrespective of individual operating voltage levels.

iv. Power Switches: Power switches are used to switch on or switch off the power supply to the power domains. The design of these switches varies according to the placement and speed of operation. These switches have many attributes such as internal power and threshold voltage that need careful attention during design.

v. Always-On Cells: These cells are always on, even when the surrounding logic is turned off. These are designed to ensure that critical logic is powered on to allow data or logic path for signal crossing between power domains.

IV. UNIFIED POWER FORMAT STANDARD

Unified Power Format (UPF) is the widespread name of the Institute of Electrical and Electronics Engineers (IEEE) standard for insist on power intent in power optimization of electronic design automation. Power domains, Supply networks, Power state table and Isolation are applied to the design which is of interest in this paper.

A. Power domains: Power Domain is intellectual way to include the instance/block instances/block that shares the same supply should be included in one power domain.
B. Supply networks: Supply network comprises of supply ports, supply nets. Supply ports consists of the VDD, VSS and supply nets which is used to provide the connection between the power domains and supply ports.
C. Power State Table (PST): A Power State Table (PST) defines the legal combinations of all the states that can present at the same time during process of a design.
D. Isolation: Isolation is a method which provides a definite performance of a logic signal when its driving logic is not active. An Isolation cell is used to pass normal logic value during normal operation and clamps its output to some definite logic value when a control signal is declared

The design flow will get changed with the addition of UPF into the flow. Firstly, power intent should be designed followed by its verification to ensure the designed power intent is intact with the design. Next phase is simulation to capture the potential bugs in the design along with UPF. Conventional design flow is a straightforward mechanism as architects will finalize the design that will be implemented by RTL engineers followed by simulation, synthesis and physical design. The power aware flow adds the power intent into the design by means of a set of low power constraints. Along with RTL, UPF will be given as input file for the synthesis process.
There are lot of processes that need to undergo to deliver final power intent of the design.

At Architecture level:

- Defining when the blocks are to be powered on or powered off.
- Identifying and partitioning of the blocks which have common characteristics under single power domain
- Specifying legal power states and their sequencing
- Identifying the domain crossings that paves the way to invoke low power techniques or strategies such as retention, Isolation and level shifter.

At RTL level:

The constraints obtained from the architecture level are configured along with RTL which helps to,

- Create power domains and merge power domains which are showing common characteristics.
- Define Isolation strategies at the OFF to ON domain crossings to avoid floating of the signals.
- Define Level shifter strategies at the domain crossings of different voltage levels to avoid voltage mismatches and power losses.
- Define retention strategies to satisfy the retention needs. Apart these, there are certain steps need to be performed to complete the power intent design,
- Create power switches.
- Create supply nets and ports

The power intent once brought up need to be verified to ensure the generated UPF is matching with the expected power architecture and that can be termed as power aware verification. It involves verification of power management schemes to ensure the design is operating as expected in all power states successfully. Different bugs can be deducted that can be classified into,

- Architectural bugs
- Control sequence bugs
- Structural bugs

The Verification is done at the RTL and netlist level to catch structural bugs in the design. Here are the few instances of the structural bugs:

- Incorrect PG pins (Power Ground pins) connectivity.
- Incorrect supplies to ISO cells, Level shifter cells and Always-On buffers -This indicates the supply set of an instance assigned in UPF is incorrect and conflicts with the supply set of instances in the design.
- No Isolation strategy on a path that connects a powered OFF domain to a ON domain -This indicates the isolation cell is missing in an OFF to ON crossing.
- No Level Shifter strategy on a path that bridges between two Power Domains of different voltages – This indicates the level shifter cell is missing in between two different voltage domains.

V. TYPICAL CHALLENGES IN LOW POWER VERIFICATION

Power Aware Simulation helps to capture architectural and control sequence bugs in the design. Control sequence bugs are resulted due to the timing mismatches between the events and unwanted transitions. Here are the few instances of control sequence bugs:

- Input switching when the power domain is in corrupt state.
- Transitions of the power states, which is illegal.
- Early Isolation control disable, or Delayed Isolation control enable. Apart from the simulation of supply network based on UPF, power aware simulation will also perform the following,
- Invoking virtual isolations cells and there by simulated in RTL for the strategies as mentioned in the UPF.
- Simulating retention flops at RTL level as per the retention strategies mentioned in the UPF.
- Propagating the ‘X’ value at the gate outputs and register contents when a corresponding power domain is OFF.

The process of debugging in power aware simulation is much more complex as an error found in power-aware designs may well be caused by RTL functions or by the behavior defined in the power-specification. Verification of “Power intents” in aggregation with RTL remains a challenge with sprouting power format. One of the Major challenges in Power aware verification is to validate all valid states /Transitions and report invalid state/ transitions early in design cycle (Figure 1).

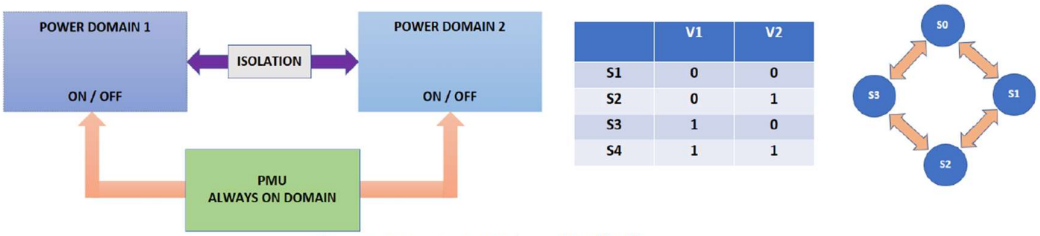

Figure 1 : Power Aware Design and Verification

Figure 2 is a representative version of a design with low power features implemented. Power management unit (PMU) represented in green box implements the power state machines to control the complete power sequencing. This block is in 'always ON' domain i.e. there is no case when the chip will be powered but this logic will be shut down. There are 2 blocks A and B and their respective power domains (PD1 & PD2) defined wherein the PD1 can be shutdown independently of PD2 (Sleep mode) or both PD1 and PD2 can be shut down together (Deep Sleep mode). In absence of qualified signals from second block for a defined period, the power manager initiates Sleep mode. During Sleep mode, if there is a wake-up interrupt (from different sources with one represented here), the power manager initiates the wake-up cycle. To avoid re-initialization of the first unit, it is expected that the last functional state of the PD1 be retained. State retention cells help in saving the state of the block and restoring it when required.

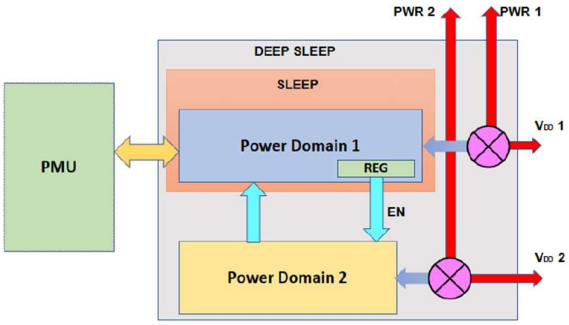

Figure 2 : DUT representation and the source of bugs

Let's discuss some typical issues which we have encountered during the course of the work conducted as a part of this paper which were in Power-Aware verification of the SoC:

i. Missing isolation cells:

Let's say design is in Sleep mode (Block A is Power OFF) and PD2 is in Power ON. In the absence of isolation strategy between PD1 and PD2 "X" or "Unknown" values will propagate from the output ports of Block A to Block B.

ii. Incorrect Specification of Save Signal in UPF:

If the states of PD1 did not get restored instead the signals shows "X" When PD1 is powered up after the sleep mode, it could be due to incorrect Save & Re-store Typically this kind of bug can be root caused by UPF modification: UPF developer who defines LOW assuming that the SAVE should happen when there is a transition from HIGH to LOW while the correct interpretation should have been that when there is an active HIGH pulse on SAVE, the values should be preserved. By replacing LOW by HIGH in UPF the problem can be resolved

```
set_retention        retention_latch -domain PD1 -retention_ground_net vss -retention_power_net  vdd
set_retention_control retention_latch -domain PD1 -save_signal {isuspend_ret low} -restore_signal {isuspend_ret
high}
```

Figure 3: Incorrect UPF specification of restore signals can cause unknown value propagation

iii. Missing Retention cell between Power Domain:

EN input is driven from the Block A. During Normal mode of operation, the Block A would drive this signal based on the value programmed in the register. During Sleep mode, PD1 can be off while PD2 can continue to be on. An isolation cell is declared at the output of Block A that clamps the value of EN to HIGH. During NORMAL operation, the value of EN is HIGH. Once isolated, the value continues to be HIGH while the PD1 is powered off.

When Block B is powered on, the isolation is removed and after the clock is enabled, the default value of the register is driven. Since the default value is ZERO, Block B actually gets functionally disabled when the Block A is fully functional. After adding a state retention cell to the UPF for this register, desired functionality is achieved.

iv. Incorrect connections of Power supply to power domains:

If for example there are 2 voltage supplies each connected to Block A and Block B. These supplies aren't present in RTL and specified in the UPF. Corresponding to these supplies there are power switches with the enable to the switches coming from power manager based on the Sleep or Deep sleep mode. In this case, if the UPF definition had an incorrect connection to the power switch for Block B, during power aware simulations, in Sleep mode, instead of turning off Block A alone, the Block B will also turn off i.e., instead of Sleep mode, the chip actually will go to Deep sleep mode.

VI. DEBUG SOLUTIONS

Power–related error during verification can be due to Error found in PA designs "caused by RTL" or the behaviour defined in the "power specification code (UPF)" hence it is critical need for debug tools to understand the power specifications and ease the verification effort by providing additional capabilities. The methodology used in this paper helps us to identify: - Schematic representation of power intents - The power aware signal types (Retention signals, Isolation signals or level shifter signals) in waveform - Trace power related unknown or "X" and help validator to know specific signals belongs to which Power domain. Figure 4 below depicts the steps of traditional debug flow and the new improved flow.

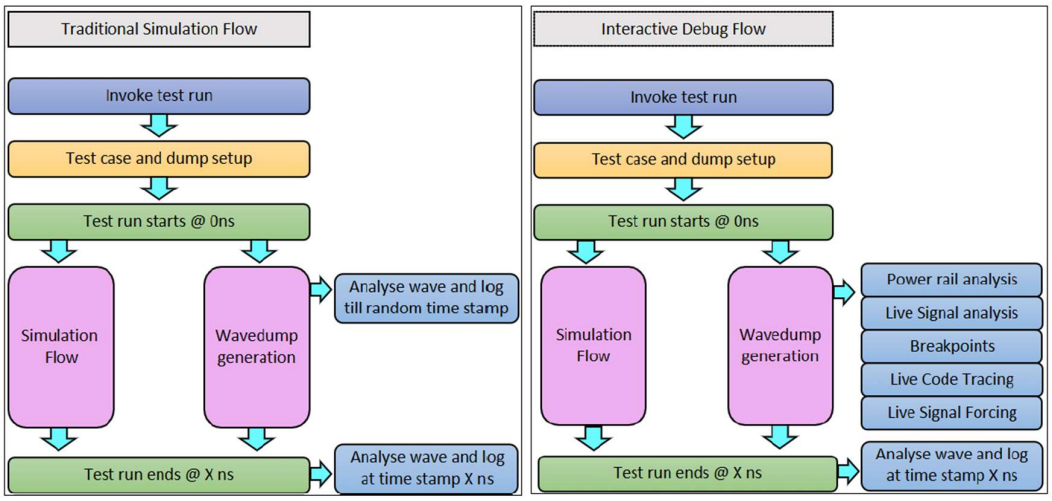

Figure 4 : Comparison of traditional debug flow with live simulation flow

The brief category of the features of interactive debug used as a part of this paper are as below:

- Breakpoints: User can place breakpoints on signals, based on signal values and transitions, enabling accurate control over when a simulation should be paused. This enables the user to pause the simulation right before the point of hang or error.

```
INFO:    2000000 Time Monitor status. Ticks at every 100,000 ns        (incr_top)
INFO:    2100000 Time Monitor status. Ticks at every 100,000 ns        (incr_top)
INFO:    2200000 Time Monitor status. Ticks at every 100,000 ns        (incr_top)
cpucl0_core_power_step11: pass cpucl0_core_power_step11 at 2205953
cpucl0_core_power_step11: pass cpucl0_core_power_step11 at 2205953
cpucl0_core_power_step11: pass cpucl0_core_power_step11 at 2205953
cpucl0_core_power_step11: pass cpucl0_core_power_step11 at 2205953
cpucl0_core_power_step12: pass cpucl0_core_power_step12 at 2236268
2238338858 PS + 24 (stop 1: top.dut.BLK_CPUCL0.nCORERESET[0] = 1)
2238338858 PS + 24 (stop 2: top.dut.BLK_CPUCL0.nCORERESET[0] = 1)
2238338858 PS + 24 (stop 3: top.dut.BLK_CPUCL0.nCORERESET[0] = 1)
xcelium> probe -create -shm -power
```

Figure 5 : Breaking the flow of simulation using breakpoints

- Stepped runs: User can run the simulation in steps of delta time, or based on each edge of the clock, enabling fine grain observation of the simulation flow, signal changes and code tracing.

```
Simulation interrupted at 104918554 PS + 17
xcelium> xcelium> xcelium> run 5 ps
run 5ps
Ran until 104918559 PS + 0
xcelium> Ran until 104918564 PS + 0
xcelium> run 5 ps
Ran until 104918569 PS + 0
xcelium>
```

Figure 6 : Controlling the flow of the simulation via the use of stepped runs

- Live Signal tracing: At each paused step, user can analyse drivers and loads of a signal keeping track of values, which helps understand the root cause of a signal transition failure down to the exact time stamp and error signal.

The problem that VVDD_AN was coming UNDETERMINED was because there were few HDL drivers coming along with the UPF drivers as shown below and those drivers coming from Core Internal Memories have dummy VDDP connected:

```
xcelium> drivers -lps top.dut.BLK_CPUCL0.VVDD_AN
        [Resolution Type]: PARALLEL
      Vol: UNSPECIFIED, St:UNDETERMINED <- [HDL Driver, vct:SV_LOGIC2UPF] top.dut.BLK_CPUCL0.vcpu_all
      Vol: UNSPECIFIED, St:UNDETERMINED <- [HDL Driver, vct:SV_LOGIC2UPF] top.dut.BLK_CPUCL0.vcpu_pre
      Vol: UNSPECIFIED, St:UNDETERMINED <- [HDL Driver, vct:SV_LOGIC2UPF] top.dut.BLK_CPUCL0.vcpu_all
      Vol: UNSPECIFIED, St:UNDETERMINED <- [HDL Driver, vct:SV_LOGIC2UPF] top.dut.BLK_CPUCL0.vcpu_pre
      Vol: 0.750000V, St:FULL_ON <- [UPF Switch] top.dut.BLK_CPUCL0.vcpu_all
      Vol: 0.750000V, St:FULL_ON <- [UPF Switch] top.dut.BLK_CPUCL0.vcpu_pre
```

Figure 7 : Analyzing signal drivers in order to understand root cause of the issues

- Live Code analysis: User can also trace signal values directly at the HDL, understanding RTL behavior at each step of simulation. User can also walk through the UPF files enabling easier debug of missing power intent specifications. This enables users to identify more uncommon issues like tool misbehavior and UPF misinterpretation.

```
105 create_supply_net VSS        -domain pd_vcpu3 -reuse
106 create_supply_net VVDD_VCPU3 -domain pd_vcpu3 -resolve parallel
107
108 # for CPU4
109 create_supply_net VDD        -domain pd_vcpu4 -reuse
110 create_supply_net VDDM       -domain pd_vcpu4 -reuse
111 create_supply_net VSS        -domain pd_vcpu4 -reuse
112 create_supply_net VVDD_VCPU4 -domain pd_vcpu4 -resolve parallel
113
114 #-------------------------------------------------------------------------------
115 # Connect Supply net
116 #-------------------------------------------------------------------------------
117 connect_supply_net VDD -ports VDD
118 connect_supply_net VDD -ports u_A/VDD
119 connect_supply_net VDD -ports u_A/VDD
120 connect_supply_net VDD -ports u_A/VDD
```

Figure 8 : Analyzing UPF and source code at paused step

- Live Signal forcing: User can force and release signal according to need, enabling fine grain control over signal values, which enables DV engineer to check solutions to hang mid simulation. This also enables engineers to understand effect of different legal values of signals to the simulation behavior.

```
Simulation interrupted at 322639883 PS + 12
> force top.dut.BLK_CPUCL0.GICCLK -delete
322639883 PS + 12 (stop 2: top.dut.BLK_CPUCL0.GICCLK = 1)
> run
322639883 PS + 12 (stop 2: top.dut.BLK_CPUCL0.GICCLK = 0)
> run
322646394 PS + 12 (stop 2: top.dut.BLK_CPUCL0.GICCLK = 1)
> run
322652905 PS + 12 (stop 2: top.dut.BLK_CPUCL0.GICCLK = 0)
> run
322659416 PS + 12 (stop 2: top.dut.BLK_CPUCL0.GICCLK = 1)
> stop -delete 1 2
```

Figure 9 : Forcing signals to legal values mid simulation to understand analyze in behavior

- Power State Network: Power schematics show connections between power rails and switches in visual mode. Leveraging this, engineers can identify broken connections and corrupt power rails and switches.

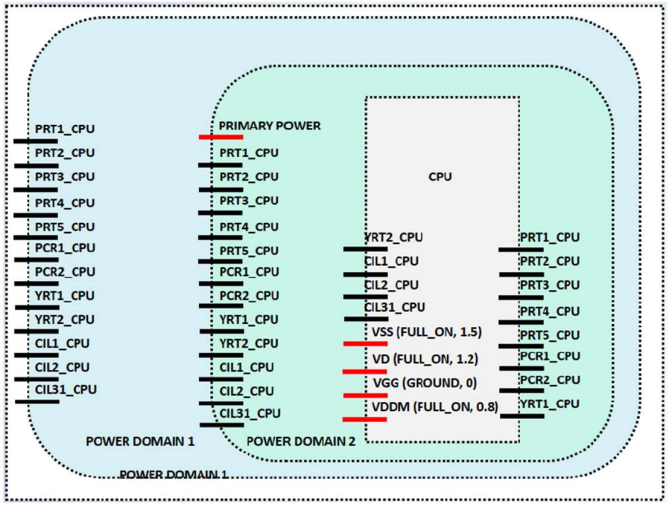

Figure 10 : Power State Networks help analyze UPF connections graphically

VII. EXPERIMENT WITH THE INTERACTIVE DEBUG

As a part of this paper, we have used Interactive debug capabilities of a standard simulator, combined with GUI modes of the Debugger, and we have successfully been able to speed up our debug time and cut down time for iterations. To validate the reduction in run time, we did an estimate of debug time and issue resolution time in a non-interactive debug environment and compared it against the debug time observed by us.

Our experiments were run on the Exynos Mobile SoC and Exynos Automotive SoC to debug real low power simulation hang issues. We were able to debug multiple complex misbehaviors in the RTL using the above-mentioned methodology. Some of the debugs were root caused to issues in the integration of UPF files with the Design environment either to IP-SoC environment tool differences, or the integration issues. We generally start our failure analysis with non-interactive debug methodology and later switch to interactive method when issue is complex to be identified. We picked three such complex cases to analyze the debug time improvement in interactive

debug. We also do a case study of a new low power verification environment bring up time using interactive debug method.

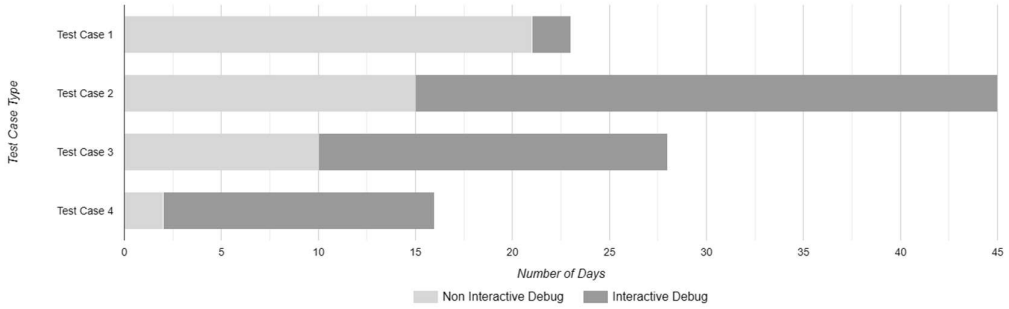

Figure 11 : Interactive and Non-Interactive debug times as components of the complete debug time

Engineers in [1] proved that the debug turnaround time for a typical simulation failure has been reduced by 90%. The total simulation time reduction for the regression run was observed by the engineers to be 25%. We saw a similar observation that, out of 23 days taken to debug the issue in test case 1, the first 21 days were spent on debugging the issue by traditional method. Once we switched to interactive mode, we identified the issue within 2 days. This debug involved stepped runs from a delta timestamp before the point of concern, live signal driver tracing, source code and upf analysis, and PSN analysis at each delta step of the simulation run.

Issues in test cases 2 and 3 were caused by tool misbehaviors which were later fixed by the vendor. When these issues were functionally checked with the waveform at the point of hang, we did not observe any RTL logic that caused error. This made it difficult to explain the cause of the error. Eventually we switched to interactive debug mode, to understand the cause of pain point, step by step by leveraging all the possible solutions provided for interactive debug mode. The issue was later flagged as a tool issue after extensive debugging and support from the tool vendor. We observe that the time taken for interactive debug of the issue is around 65% of the total debug time, but these complex issues could only be identified using interactive method, as the first 35% of debug time did not provide any clue on the issues at hand.

Test case 4 was a new low power test bench that was being set up. Since we had prior experience on debugging through interactive method, we switched to interactive debug right from the beginning of the test bench and design environment bring up. It took about 16 days for the setup to mature and we were able to identify multiple issues in integration of the UPF with design within this timeframe which would not have been possible by traditional methods. Issues such as missing isolation cells, improper voltage source connections were easily identifiable using live signal analysis in interactive mode.

VIII. Results and Conclusion

Because Power aware simulations have large run times owing to the complex designs and integrated united power format files (.upf), they are often plagued by UPF integration issues with design which are hard to debug. Hence low-power simulations are usually the bottlenecks to Verification sign-off. More often than not, DV engineers find themselves in a situation where they cannot afford multiple iterations of the test cases. Hence there is a need to cut down time taken for debug to enable faster verification closure.

We observed a reduced time of debugging as the engineers have better insight on the behavior of the system at each timestamp, enabling better reasoning. Some of the issues with complex root causes could only be debugged by leveraging the interactive mode and it would be difficult to identify them in non-interactive methodology. The ease of debug and better root causing strategy led us to adopt this methodology in subsequent projects which are being executed currently.

REFERENCES

[1] H. Chan, "UVM Interactive Debug Library: Shortening the Debug Turnaround Time", DVCON2017

[2] M. Peryer, "Command Line Debug Using UVM Sequences", DVCON 2011

[3] J.McNeal and B. Morris, "RESSL UVM Sequences to the Mat", SNUG 2015

[4] "IEEE Standard for Design and Verification of Low-Power, Energy-Aware Electronic Systems," in IEEE Std 1801-2018 , vol., no., pp.1-548, 29 March 2019, doi: 10.1109/IEEESTD.2019.8686430