Complexities & Challenges of UPF Corruption Model in Low Power Emulation

Progyna Khondkar and Brad Budlong
progyna@cadence.com bbudlong@cadence.com

Abstract- The Unified Power Format (UPF) or IEEE-1801, is not just a language to denote low-power intents or power management specifications for a design – it's a complete command set for verification and implementation of such low-power designs. UPF paved the way to accomplish low power verification with specialized simulator from very early stage of design abstraction from RTL. These low power enabled simulators allows design instrumentation with UPF artifacts to mimic accurate low power behavior at power up-down-cycles where, for example, the inputs and sequential logic in a power down domain needs to be corrupted onto unknown (i.e. x), or propagate x from OFF to ON domain, or analyze x-state initialization of regs after power up. So evidently 4 state logic value (0, 1, x, & z) are essential for accurate low power logic functional verification. However, capacity limit and event driven execution mechanism makes simulator almost impossible to run realistic workloads with multi-pass power sequencing and huge power management architecture at high-speed. Hence hardware emulators comes into play because they are cycle based, runs very fast (in MHz range), accommodate very large designs (~10 Billion Gates SoC) and obviously the best suit for running realistic low power workloads. However, emulators are made of silicon and only allows 2 state logic value (0, 1), which makes UPF based verification very challenging. In this paper, we establish the orchestration of industry's first 4 state logic based hardware emulation technology that not only simplifies UPF based design verification but also significantly shorten the verification turn-around-time with real world power management workloads.

Keywords—UPF; 2 state logic; 4 state logic; Simulator; Hardware Emulator; Hardware Accelarator; LP Verification.

I Introduction

The IEEE 1801 standard specifies the low-power intent, i.e., UPF for any design. The UPF is the ultimate abstraction of low-power methodologies today. It provides the concepts and the artifacts of power management architecture, power aware verification and low-power implementation mechanism for any design. It provides the notions of power architecture from very early stage of design abstraction. Now it's the industry trend and standard for lowering static and in some special cases dynamic power dissipation in every digital design. Overwhelmingly UPF standout as the only alternative choice of lowering power dissipation when fabrication process technology advanced below 65nm.

Almost all SoC today heavily utilizes UPF for power management and low power implications. These SoCs are enormous in size, typically comprise of 8-10 Billion Gates (BG) which consumes 60-70% of design efforts in verification and easily hits the capacity limits of most verification tool and makes design-verification turn-around time miserably long.

On the contrary, todays datacenter-class emulation systems are highly scalable in terms of design capacity (scales up to 10 BG designs) and can run accelerated simulation up to 4 MHz speed on hardware. Furthermore, the emulation allows realistic loads for UPF based designs. For example, emulation allows multi-pass power sequencing workload compared to single-pass in simulation. This enables complex interaction scenarios and repeated power state entry, retain or exit with diversified sets of external stimulus. Figure 1 shows the diagram of today's datacenter-class emulator.

Generally, UPF enabled emulation based low power verification flow is suitable for long running directed tests - that do not require complex testbench interaction. Specifically, the UPF verification comprise of power management, BIOS, firmware, OS, and application-level workloads. Naturally thousands of hierarchical scope based UPF files require to process, overlay supply network circuitry and instrument the design with all types of low power components like power supply network, power switches, isolations, level-shifters, repeaters, retentions, corruption mechanism and so on. Figure 2 shows the UPF enabled emulation verification flow.

UPF LRM defines corruption as "*the rules defining the behavior of logic in response to reduction or disconnection of power to that logic*". Obviously, applying UPF to any design requires additional functionality to model the power supply network, corruption of logics, signal values etc. when insufficient power is applied for normal operation. As well provide additional functionality for propagating, saving and restoring of the system states. Corruption is a situation where the value of signal become unpredictable at power OFF or below predefined threshold value. A corrupted signal is represented by assigning a particular value generally “x” in 4 state and “0” or “1” in 2 state logic representation.

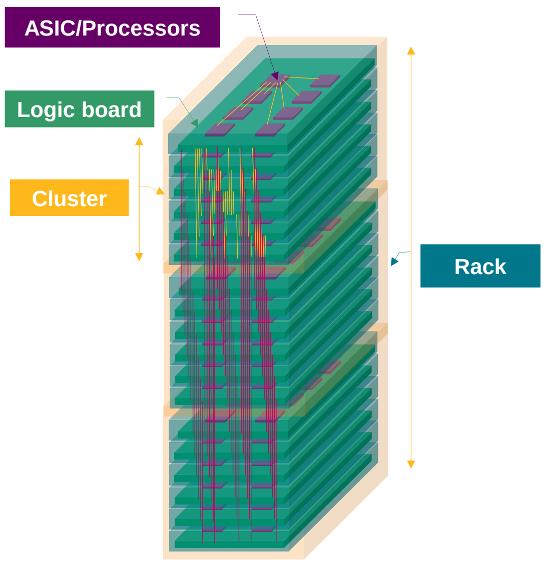

Figure 1 Datacenter-class 3 rack emulator

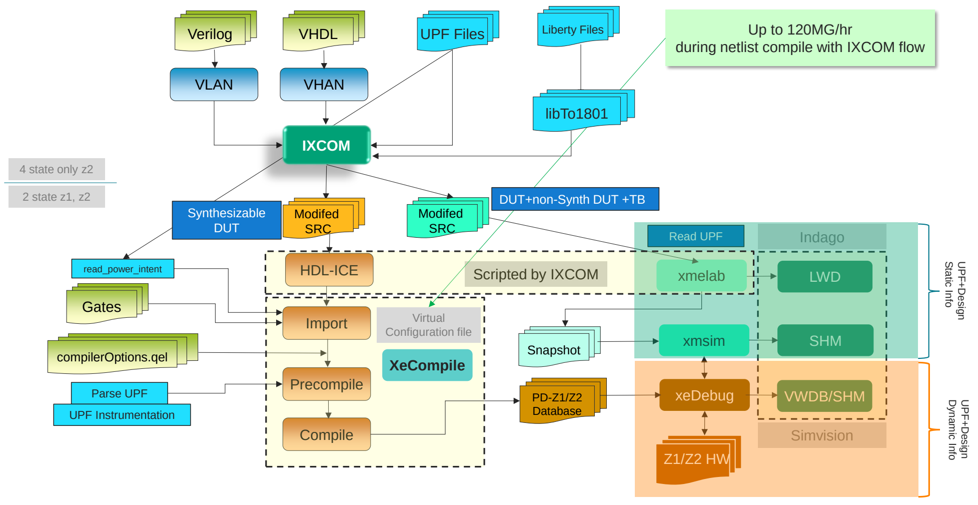

Figure 2 UPF enabled emulation verification flow

It’s important here to reference 4 state and 2 state logic value definitions from System Verilog (SV) LRM. The SV ‘logic value sets’ consists of the following four basic values:

- 0 represents a logic zero or a false condition
- 1 represents a logic one or a true condition
- x represents an unknown logic value
- z represents a high-impedance state

The values 0 and 1 are logical complements of one another. When the z value is present at the input of a gate or when it is encountered in an expression, the effect is usually the same as an x value. SV data types that can have unknown (x) and high-impedance (z) values are called 4-state, which can store all four logic values. All bits of 4-state vectors can be independently set to one of the four basic values. These are logic, reg, integer, and time. The other types do not have unknown values and are called 2-state types, for example, bit and int, which only store 0 or 1 values in each bit of a vector. Type conversion between 4 state and 2 state can be automatic from a larger number of bits to a smaller number of bits involve truncations of the most significant bits (MSBs). When a 4-state value is automatically converted to a 2-state value, any unknown or high-impedance bits shall be converted to zeros.

Hence its distinctive that a design under tests with UPF, mandate to coordinate situations for unknown or x initialization, corruption to 'x' and propagation to 'x' as well high-impedance or 'z' condition. Its' also true that emulators are the best choice for realistic UPF or low power workloads specifically because of very quick turn-around-time and very large design capacity accommodation - compared to conventional simulators. However, the problem is - emulators are developed on custom ASIC with hundreds of processing units on physical boards (racks shown in Fig. 1). Essentially, they are hardware on silicon and by default capable of handling only 2 state, i.e. '0' and '1' logic. This signifies the fact that many low power functional verification features like corruption to unknown, initialization to unknown states, propagation of unknown, as well analyze strength resolution will be either impossible to implicate on hardware or have to extrapolate '0' or '1' onto 'x' (i.e. to device unknowns) by some sort of faking mechanism.

However, as logic verification engineer, we know that the inherent meaning of initialization to x, corruption to x and propagation of x – are not just about faking or mapping '0' or '1' onto 'x' for low power implication, visualization or facilitate debugging! Its on one hand properly modeling 'logic' and 'reg' data types (because integer, and time may not involve in low power designing) according to SV and UPF LRM and on other hand properly handling human perception of logic verification and/or implementation mechanism that introduces the notion of 'x pessimism and x optimism duos' in emulation hardware.

Evidently emulation and 4 state logic values are prominent for UPF based low power verification. However, there are multitude of complexities and enormous challenges in realizing 4 state, specifically UPF power related implication like corruption mechanism in hardware.

A. Motivation and Contribution of this Paper:

Our core objective is to accurately define the inherent characteristics of UPF based low power verification flow. We systematically identified today's chip design and verification methodology and approached the limitations of existing highly efficient hardware emulation verification platform. In this paper we identified the requirement of 4-state logic in UPF based emulation flow, identified the complexities, and establish the orchestration of industry's first 4 state logic based hardware emulation technology that not only simplifies UPF based design verification but also significantly shorten the verification turn-around-time with real world power management workloads. We hope this paper, will serve as reference point for innovative research and realization of highly efficient hardware emulation based low power verification platform.

B. Organization of this Paper:

This paper is organized in the following structure. Section I introduces the key concepts and relevant UPF verification concepts and flows. Section II explains fundamentals and complexities of UPF verification on emulation. Section III provides further insight of 4-state logic and orchestration in emulation. The final sections IV draws the conclusion and points further/future research prospective. The references are shown at the end.

II FUNDAMENTALS AND COMPLEXITIES OF UPF VERIFICATION

In section I, we explained how UPF verification concepts and 4-state logic are corelated. We also understand that the inherent meaning of initialization to x, corruption to x and propagation of x - lies in the fact that how SV 'logic' and 'reg' data types are modelled in verification tools and how notion of 'x pessimism and x optimism duos' plays vital roles in realizing them in emulation hardware.

Let's step back and understand what 'x pessimism and x optimism duos are'? First, we have to look for the sources of x's in a design. The primary sources of x in UPF, its power down phenomena on logic, reg etc. and in UPF and non-UPF both, explicit assignment to x, uninitialized state elements ('regs with/without reset') etc. The secondary sources could be incomplete SV assignments, like missing default statement in case block, bus conflicts, out of range bit/part select, indexes in MDA, arithmetic exceptions (divide by zero, modulus 0) etc. Some of these x interpreted in verification tool may differ in actual design implemented in hardware.

The notion of x optimism comes from the fact when verification tool generates '0' or '1' signal values when they are actually unknown ('x'). Similarly pessimism comes when tools produces 'x', whereas they should be known ('0' or '1'). These verification tools results may be far different when the RTL design is actually implemented (synthesized) in gate-level netlists. Which is widely known as simulation synthesis mismatch where x is don't care in synthesis but in simulation, x is logic value distinct from 0, 1 and z.

A. UPF Corruption Model:

In regular HDL based functional verification (simulation) world, its presumed that all logic are powered on at the beginning of simulation and remains powered on throughout the verification cycles. However, the SoC designed today are far more complicated by adopting multi-voltage, power gating, state-data retention, power standby even dynamic voltage scaling techniques on same chip by virtue of UPF as shown in Table 1.

Table 1 Complicated Power Management Schemes on a Chip

Multi-voltage	Power Gating	Power-gating with Retention	Low-power Standby	DVS/AVS
Blocks of chip can run on different supply voltages e.g. 1.5V, 1.0V, 0.8V etc.	Some block of chip is OFF, while others running ON with different supplies	The OFF block require to retain state and data during shut-down and restore them up on powered ON	Some blocks runs at possible low voltage to hold state and data, while rest are OFF	Dynamic voltage and adaptive voltage scaling based on performance demand

These power management features definitely alter and/or instruments the actual design with additional power artifacts – like confinement of different design blocks on to different voltage region (power domain), provide different supply network/sources, power switching and control network, power management cells like isolation or retention, power down corruption semantics and so on. Fortunately todays' low power enabled verification tools (simulators and emulators) allows design instrumentation with UPF artifacts to mimic accurate low power implementation like behavior at functional verification level. Among all, corruption modelling imposes a complete unprecedented requirements from HDL logic values as well impacts on design instrumentation.

In low power, corruption refers to the situation where the value of a signal becomes unpredictable when the power supply for the element driving that signal is disconnected, changes to OFF, or drops below some threshold. Corruption of a signal is represented by assigning a particular value to the signal (it could be x or 0 or 1 (but probably not z/floating). The corruption value depends upon the type of the signal and rare cases may also be user-definable.

It is typically applied to signal drivers and propagates to all sinks of that signal that are not isolated form their sources. When a design instance is turned off, every sequential element within the power down instance and every signal driven from within the power down instance must be corrupted. As long as the power remains off, no additional activity takes place within the powered down instance—all processes within the powered down instance become inactive, regardless of their original sensitivity list. Events that were scheduled before the power was turned off and whose target is inside a powered down instance have no effect.

When a design instance is turned on (restored), corruption of sequential elements and signals within the powered down element ends. Continuous assignments once again become sensitive to changes to their right-hand side expressions, and other combinational processes (such as an always_comb block in SystemVerilog) resume their normal sensitivity list operation. All continuous assignments and other combinational processes are evaluated at power up to ensure that constant values and current input values are properly propagated. Sequential elements are re-evaluated on the next clock cycle after power up.

B. What could be the corruption values:

Signals are corrupted by assigning them to their default initial value (such as x for 4-state types). The SV LRM denotes that 4 state and 2 state data types of default initial value are x and 0 respectively. So any design that have 2 state and 4 state data types.

- 2-state signals (e.g. bit) continue to operate as 2-state: initialize to 0
- 4-state signals (e.g. logic, reg) continues to operate in 4-state: initialize to x

The question may arise for interoperation between 2 state and 4 state data types. The answer can be found in SV LRM as noted in section I, that type conversion between 4 state and 2 state can be automatic from a larger number of bits to a smaller number of bits involve truncations of the most significant bits (MSBs). When a 4-state value is automatically converted to a 2-state value, any unknown 'x' or high-impedance bits shall be converted to zero '0'.

During low power verification, if the driver of a net is powered down, then the output of the driver is corrupted, and this corrupted value propagates to all sinks of that net. To understand how corruption occurs in a given design, it is necessary to recognize the elements of the design that represent or contain drivers. In an RTL code, any statements involving arithmetic or logical operations or conditional execution are interpreted as representing drivers and cause corruption when powered down. Unconditional assignments and Verilog buf primitives do not represent drivers and therefore do not cause corruption when powered down, but they may propagate corrupted signals from upstream drivers.

C. Simulation Emulation Congruency

It is now distinctive that corruption modeling along with other power implications specifically for accurately mimicking low power behavior at power up-down-cycles where, for example, the inputs and sequential logic in a power down domain needs to be corrupted onto unknown value (i.e. x), or propagate x from OFF to ON domain, or analyze x-state initialization for all sequential elements after power up. Even for combinational logic within power down domain needs to evaluate to x.

Conventionally corruption modeling is based on 4 state logic and easily accommodate in event driven low power simulation environment. However, emulation is devised on actual silicon hardware (custom ASIC) and there is no unknown or x state in hardware. This makes low power corruption modeling, as well low power verification (i.e. wave based debug) extremely difficult. For example, we utilizes "0 or 1 or random or inverted" mechanism for 2 state UPF emulation flow. This may be applicable for both power up and power down state as follow:

- Power down states {high | low | random | inverted}
- Power up states {high | low | random | inverted}

Consequently users have to assign and maintain specific power up down values through out the entire verification flow. Arguably it's possible that the simulator can tag, or debugger can fake a power down corruption value to x, but that's completely for the sake of visualization. Actual x corruption, propagation, initialization or evaluation is not possible in 2 state environment. Figure 3 shows the default 2 state corruption and figure 4 shows 4 state corruption on debugger waveform window as well figure 5 shows

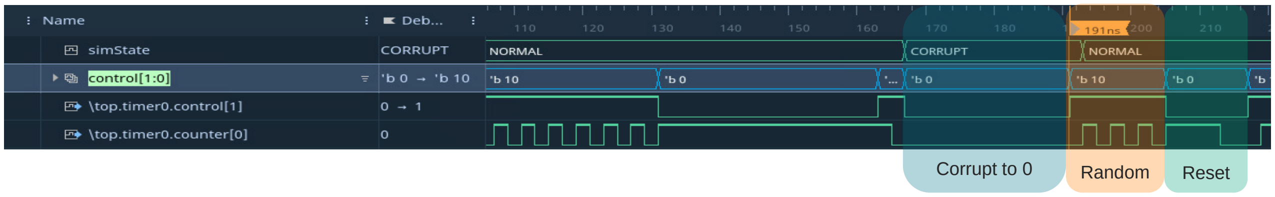

Figure 3. Emulation (2 State) showing 0 during power OFF and random after power ON

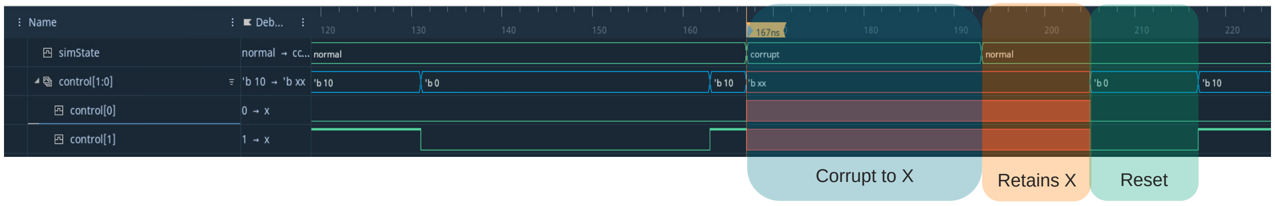

Figure 4. Emulation (4 state) showing x during power OFF and design reinitializes as needed after power ON

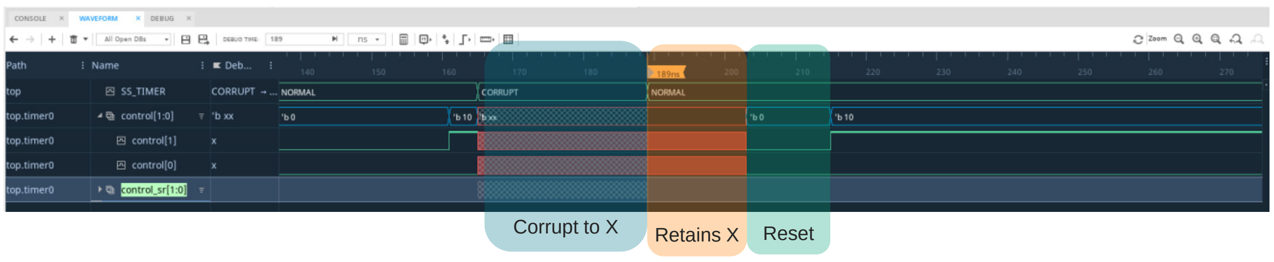

Figure 5. Simulation (which are generally 4 state) Showing x during power OFF and design reinitializes as needed after power ON

From these three figure it's evident that 4 state emulator corruption model to x is expected and matches the corruption behavior of simulators. Hence 4 state corruption model also possesses the advantages of directly compare the results between simulation and emulation. Apparently the 4 state emulation and simulation results appears congruent in UPF based regular SV 'reg' related corruption mechanism. However, actual implementation and/or realization of 4 state logic in emulation could have differences - primarily because of x pessimism and optimism and subtle difference how RTL level and gate-netlist level design interpret these x pessimism optimism duos. Because emulation turns every RTL design to sub gate level design (through special synthesis process), which is more close resemblances of the design to its' actual hardware chip implementation. Hence emulation and simulation will not be congruent in the presence of x's and the reason is further explained below.

Simulation has 3 modes of operation

- ❖ SV LRM mode: Default
- ❖ CAT (Compute As Ternary) mode: Optimistic and similar to how hardware propagates x's
- ❖ FOX (Forward Only X) mode: More pessimistic forwarding of x's

Emulation will only adopt a similar mechanism of simulation CAT mode because

- ❖ Emulation transforms RTL design on to gate-netlist by synthesizing
- ❖ In addition, SV LRM strictly applicable for on the RTL designs

Note that, two equivalent RTL implementations will handle x's differently in LRM simulation mode. However, two equivalent RTL that results in the same gates will behave the same in emulation.

D. Experimental Analysis of UPF Based Simulation Emulation Congruency

Although the previous section provides detail in differences based on x optimism pessimism duos, this sections will summarize the UPF based simulation and emulation similarities or dissimilarities based on 4-state logic corruption models. Table 2 and Table 3 shows our empirical data for corruption mechanism comprising of simulator, 2 State and 4 State emulator.

Table 2: Comparative studies for sequential elements and power sequence comparison for Simulator and Emulator

DESIGN OBJECTS IN POWER DOMAIN	SIMULATOR	2 STATE EMULATOR	4 STATE EMULATOR
Input port of Power Domain	No	No	No
Output port of Power Domain	Propagate the value	Propagate the value	Propagate the value
Latches	Corrupts to x	Corrupts to 0/1	Corrupts to x
FF/Regs	Corrupts to x	Corrupts to 0/1	Corrupts to x
Memory	Corrupts to x	Corrupts to 0/1	Memory Wrapper Model to protect Read/Write and Contents of Memory from Corruption

The experiments in Table 2, considers all design objects are part of a power domain in UPF based verification setup as shown in first column. The sequential elements - latches, FFs, and memory instances emulation 2 state corrupts sequential elements based on special UPF command/options

- ❖ create_power_domain <domainName> -power_down_states -power_up_states

that allows users to select corruption and normal values at power up and down shown in the 3rd column.

However, the 4 state results for every sequential elements - latches, and FFs, as well every signal elements driven from within power down domain will be corrupted to x. There are little difference in corruption mechanism in memory between simulation and emulation (Corrupt to x vs Memory Wrapper Model) because memory is modeled, treated and implemented differently in simulation and emulation. The subsequent section explains detail on UPF based corruption mechanism in emulation from modeling and implementation perspective.

Table 3: Comparative studies of logic states and Corruption Semantics for Simulator and Emulator

Run Time Constraints			**Observation Outcome**		
Run Condition	**Power Status**	**Design Object**	**Simulator**	**2 state Emulator**	**4 state Emulator**
time 0 after swap-in (**xt0**)	ON	Reg (Not RET)	x until reset asserted	Default 0	0
time 0 after swap-in (**xt0**)	OFF	Reg (Not RET)	x until reset asserted	Default 0	x until reset asserted
Async. Reset is Asserted	ON	Reg (Not RET)	Initialize to known value	Initialize to known value	Initialize to known value
Async. Reset is Deasserted	ON	Reg (Not RET)	No change	No change	No change
Power up (any time)	OFF->ON	Reg (Not RET)	remain x	Default random	remain x
Power down (any time)	ON->OFF	Reg (Not RET)	Goes to x	Default to 0	Goes to x
ISO Association	ON->OFF	Power Domain IO	Clamp value	Clamp value	Clamp value
	OFF->ON	Power Domain IO	Propagated value	Propagated value	Propagated value
RET Save/Restore	ON->OFF	Reg (RET)	Save last data	Save last data	Save last data
	OFF->ON	Reg (RET)	Restore saved data	Restore saved data	Restore saved data

The experiments in Table 3, considers all design objects are SV "reg" data types, some of them are ordinary reg and some of them are special reg with UPF data and state retention capabilities. It also have design objects as primary input and outputs (IO) of power domains. The "Power Status" column shows the power ON or OFF or ON to OFF or OFF to ON situation. The results clearly shows there is no differences in results between simulation and 4 state emulation except there is a x to 0 deposition at time 0 after swap in in row 1. Here deposition only works at emulation hence reg remains x at time 0 for simulation until reset is asserted while 0 for 4 state emulation. So its obvious congruency works in simulation and emulation 4 state in UPF corruption modeling to represent unknown.

III UPF 4-STATE LOGIC ORCHESTRATION IN EMULATION

In previous section we noticed that the experiment data in Table 1 shows little difference in corruption mechanism in memory between simulation and emulation (Corrupt to x vs Memory Wrapper Model) because memory is modeled, treated and implemented differently in simulation and emulation. In this section we will identify the difference and explain the reason with experiments and analytical data.

A. Impact of Memory and FIFO in UPF Based Emulation

When a power domain is shut down, the content of all logics including memories and FIFOs are set to 'x' and it's comparatively easy to implement in simulation. However, as noted earlier, corruption or x modeling in emulation is very costly because it needs to model in hardware. The alternative choice could be to setting all powered-down logics to some random value. While it is relatively easy to set registers to random values inside a powered-down domain, it is quite difficult to set memories and FIFO contents to random values.

One approach is to load the contents to memories and FIFOs with predefined randomized values, but the load operations can be time consuming due to the large number of memories and FIFOs that are in powered-down domains. Another approach is to create external logic that is triggered when the power domain is in shutdown mode. This external logic will update the memory and FIFO contents at emulation speed. This approach, though re-usable, requires additional hardware resources, requires code development for the emulation platform, and also requires additional debug to make sure that it functions correctly.

Ultimately none of these worked as efficient and flowless mechanism for memory and FIFO corruption. In UPF 4 state we introduce an automatic memory wrapper in hardware and do not have runtime cost to implement. Memory wrappers help model the behavior of design memories correctly, specifically when there are x's on the address or control signals of memories. Without the wrappers, x's on these control signals of memories are interpreted as 0 for non-bit-blasted memories.

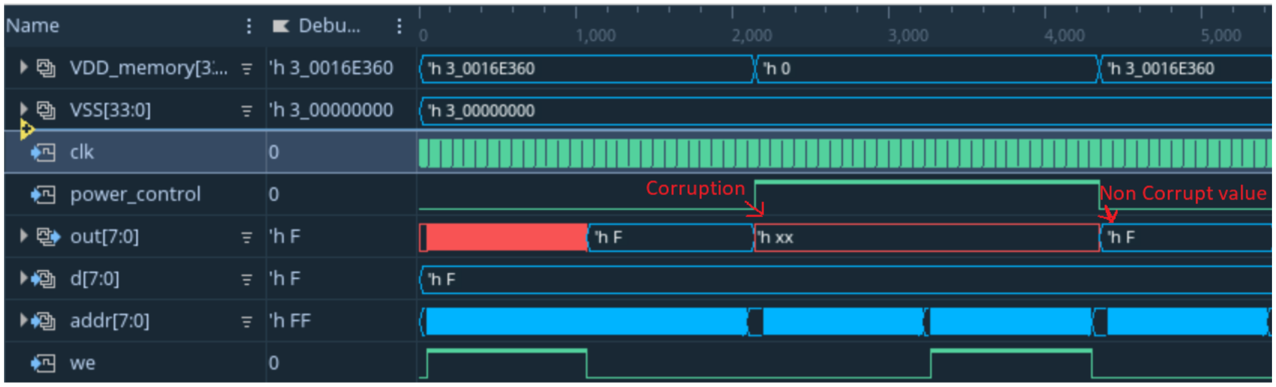

Figure 6. Memory corruption results at power down without memory wrapper

Here in Fig. 6, the memory was written to 00001111. During power OFF, the "out" shows x however, dumping the memory contents shows 0:ff/00001111, which signifies that the contents of memory is not actually corrupted.

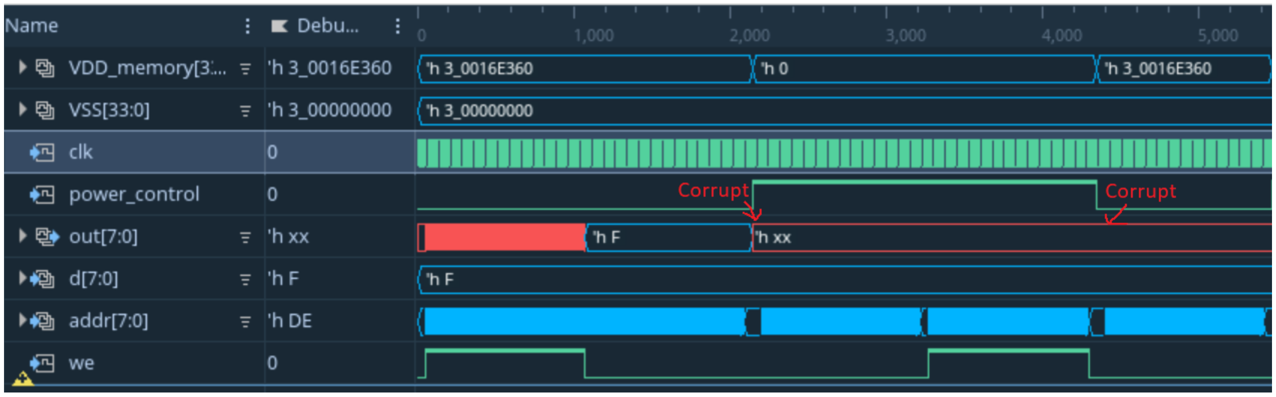

Figure 7. Memory corruption results at power down with memory wrapper

Here in Fig. 7, the memory was written exactly same to 00001111. During power OFF, the "out" shows x as well, dumping the memory contents also shows 0:ff/xxxxxxxx, which signifies that the contents of memory is properly corrupted. This results matches with simulation and desirable verification mechanism for memory during power down.

IV Concluding Remarks

UPF based low power designs are prevalent today in every chip designed and manufactured today. The demands and expectation for 4 state emulation in the low power design verification industry is eternal. This paper identifies the complexities & challenges of UPF based low power design verification on emulation platform through corruption modeling. In the abstract and extended abstract sections, we approached the rudimentary parts of low power verification, we explained the requirements of 4 state logic and primary bottleneck for implication the 4 state logic in hardware emulation platform. We also presented empirical data that shows comparative studies of logic states and corruption semantics as well sequential elements and power sequence comparison between simulator and emulator. These evidently shows the potential and importance of 4 state in hardware emulation platform.

In our future research plan for "UPF Corruption Model in Low Power Emulation", we want to extend our studies to further understand the capacity and performance impact of corruption in 4 state vs 2 state logics. More specifically the influence and contribution of "Corruption Model" in terms of emulation capacity or gate-size during build-time and performance in clock-frequency during run-time.

Acknowledgment

The authors would like to express sincere gratitude to the reviewers from Cadence Design Systems for providing useful insight and guidance in conducting the research and preparing this paper.

References

[1] Progyna Khondkar, "Low-Power Design and Power-Aware Verification", Hard Cover ISBN: 978-3-319-66618-1, October, 2017, Springer International Publishing.

[2] Design Automation Standards Committee of the IEEE Computer Society, "IEEE Standard for Design and Verification of Low-Power, Energy -Aware Electronic Systems", IEEE Std. 1801™-2018.