

Getting productive

Thomas Li describes the issues engineers face in de-bugging power-aware designs and verification flows and considers some solutions

Power has emerged as a major concern in IC design. Emerging power format standards allow one power definition to be used starting from the early stage of design and throughout the complete design, verification and implementation cycle. However, these standards also increase complexity in verification and debug.

For years, designers have implemented power management directly in the gate-level netlist. However, there are many drawbacks to this technique. One is the difficulty to link the gate-level power implementation to the higher level chip behaviour and it is usually too late and too expensive to redesign issues found. Correcting power issues at the gate-level can limit the efficiency of the power management design or, worse, cannot satisfy the required power consumption specification. To verify power behaviour earlier, engineers have begun to model power management in the RTL code. However this technique is becoming impractical as power management is required on more modules, and the required power modes and transitions are growing quickly.

As a result, two standards have emerged to express power intent: the *Common Power Format (CPF)* and the *Unified Power Format (UPF)*. Both use TCL side files that allow engineers to describe the power 'design', such as: creating and managing power domains, specifying

isolation and retention, setting up level shifters, and defining power related rules.

The power design intents described in CPF or UPF are in addition to the original design and verification source code. These descriptions provide the information that allows engineers to develop the power management scheme systematically. The hitch (especially at the Register Transfer Level (RTL)) is that the power intent that is described in CPF or UPF has not been fully implemented into the design yet. While the CPF/UPF specification eases the power design and verification, it increases the complexity.

The challenge for debugging in a power-aware design is that the root causes of bugs could reside not only in the RTL but also in the power intent files. For example, an 'X' in a waveform during power-aware simulation can mean "power off" or it could be a bug caused by structural errors such as missing isolation cells, or control sequence errors such as an incorrect save and restore sequence. A simple "X" value in the waveform does not provide any indication of where it is from or what might cause it. Engineers need to trace back in both the RTL source and the power intent files to locate the root cause. In Figure 1, after investigation of the power intent and RTL source code, the "XX" values found on signals B and C which reside in a "power on" block are caused by a lack of isolation cells for signal A from the

driving "power off" block which propagates the "XX" value to signals B and C.

Design automation tools help engineers save time by automating tasks however debug is not impossible without automation but it is extremely time consuming. Debug automation shortens the time required to fix bugs by helping engineers quickly visualize, trace and understand complex logic. The foundation of a solution for power-aware debug is automation that bridges the gap between RTL and CPF/UPF so that engineers can freely trace behaviour without regard for where it was specified.

The first requirement is visualisation of power domains and power states/modes so the behaviour of power modes or states can be easily grasped. The special views for power design intents can help reveal power intent such as switch, retention, isolation and level shifter rules. Windows that provide a view of the source code and schematic of the CPF/UPF design with extracted hierarchical power domain information are ideal to aid the understanding of the power behavior.

The second requirement is that power intent must be annotated on the traditional RTL, schematic and waveform views so that it can be correlated to design blocks for comprehension of which blocks belong to which power domains and which signals are retention or level shifters to a power domain. The tool should have the intelligence to correlate the power design rules with RTL design. Users can easily get the whole picture of power domains, states and modes as well as visualise how the power management rules will interact with the RTL design. Support for cross-probing between the UPF/CPF power view and the RTL, schematic and waveform views lets engineers locate design or power objects in any other view.

Apart from static design comprehension requirements, there are extra requirements for debugging power-aware simulation results. When the simulator takes the power intent into account during simulation, it adds the sources outside the RTL design that will impact the design behavior. For users to identify the root cause of abnormal behavior, they need to cross-reference the design power domains of interest with the corresponding state during simulation.

Taking this further requires debug automation that can automatically process the combined intent of the

RTL and the CPF/UPF to speed the process of locating, isolating, and understanding the root cause of design behavior. This would include:

- Automatically locating the root cause of a bad value in either the RTL or CPF/UPF code.
- Automatically locating drivers and loads in CPF/UPF as well as RTL.
- Unrolling paths through different power domains and easily identify retention, isolation or level shifter signals in the unrolled paths.
- Isolating the signals driven by CPF/UPF so that debug can be directed to trace into the CPF/UPF.

In order to satisfy the power aware debug automation requirements, there are several techniques needed under the hood. The power design intent files need to be compiled into the same RTL design database. The power domain definitions from the UPF/CPF need to be modeled and properly correlated to the design hierarchy from the RTL. The power management rules, like isolation, retention, etc. need to be "attached" to the right places. While I say "attached" here, that doesn't mean at synthesis of the power rules into the design but rather identification of where the power management rules will impact the design and, based on the knowledge of the control signal values from simulation results, how the design should be impacted.

Even though the debug challenges of power design seem mainly to be at the RTL due to the "virtual design" at the time, there are also challenges to the gate-level netlist after power rules have been implemented into the circuit. The locations where those power-related cells, like isolation and/or retention, are inserted could be improper. The connections of the power/ground pins could be incorrect. The debug solution should provide the capabilities to enable designers to easily debug when these happened.

Although the complexity of designs increases when power requirements are specified using the new CPF and UPF standards, debug automation techniques are advancing to help engineers keep ahead of the comprehension curve. Putting the visualisation and tracing of these formats on the same plane with HDL is the foundation and the minimum requirement for productive debug automation of power-aware designs and their associated verification flows.

SpringSoft | www.springsoft.com
 Thomas Li is a Product Marketing Director for SpringSoft

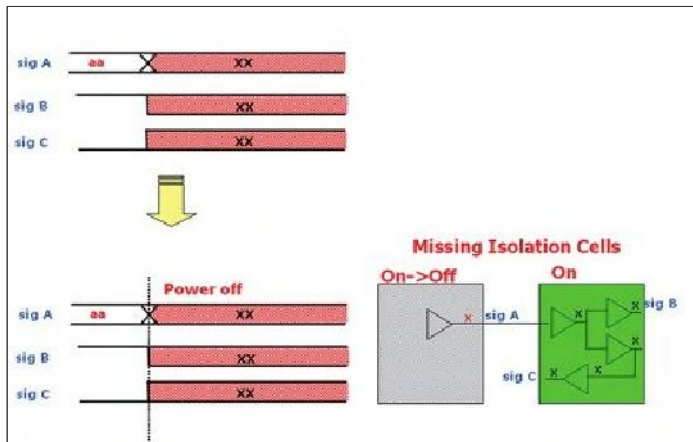


Figure 1: The cause of X is missing isolation cells