Verification of Low Power Designs using CPF

Noah Bamford, *Freescale Semiconductor*, Saji George, *Freescale Semiconductor*, Milind Padhye, *Freescale Semiconductor*

***Abstract*— The power wasted by leakage current can no longer be ignored in sub-micron designs. As the size of transistors shrink, the amount of current due to leakage rises exponentially. In order to minimize the power lost due to leakage, several design techniques have been developed. Functional defects can be introduced if these features designed to reduce power are either specified or implemented incorrectly. The problem is not only being able to detect these functional defects, but also being able to detect them early enough in the design cycle to avoid costly delays. This paper explains how CPF (Common Power Format) enabled tools can be used functionally verify that the features added to save power have not introduced defects. The paper will illustrate how the specification of design features such as Power Switch Off (PSO), Save Restore Registers (SR) and Isolation (ISO) can be verified at a RTL level and the advantages of using a CPF based flow over an ad-hoc solution.**

***Index Terms*—Low Power Verification, Common Power Format, Logic Equivalency**

I. Introduction

In 1965, Gordon Moore wrote an article for Electronics Magazine in which he observed that the number of transistors on a chip was doubling every two years and he hypothesized that this growth would continue perpetually. The growth predicted continues today, fueled by advances in design automation and shrinking transistor sizes. The advances in computer aided design have allowed engineers to create larger designs using limited resources. Shrinking lithography has resulted in the package size growing smaller even as circuits have gotten more complex. As the design elements entered the sub-micron space, the leakage current began to grow to a point where it could no longer be ignored. New methodologies and design techniques were invented to diminish the effects of leakage. Many of these techniques, such as power switch off, can change the functionality of a design if either specified or implemented incorrectly. Because of this, functional verification must address the low power methodologies and prove that the design elements inserted to reduce leakage power have not altered functionality. This paper will address the verification of power switch off and the design practices associated with removing power from a domain such as isolation and state retention registers. It will show the advantages of using a complete solution utilizing common power format (CPF) over ad-hoc methods. In order to understand the need for verification, first the design practices will be explained and some of the possible defects will be revealed.

Noah Bamford has worked for 8 years as a verification engineer in the wireless group of Freescale Semiconductors. He received his BS in ECE from Carnegie Mellon University

Milind Padhye is Low Power Design Manager at Freescale Semiconductor, Wireless design organization. He has been working in the field of low power design for last six years and has multiple patents on power reduction techniques and integration. He has leaded multiple chips for low power architecture and design. Milind holds MS-EE from IIT Kharagpur

Saji George has led the integration of numerous wireless and transportation SoC products at Freescale Semiconductors. He received his MS in EE from UTEP

II. Power Switch Off and Isolation

One of the most effective means of reducing leakage current is to switch off the power or ground rails in portions of the design during low power modes. After disconnecting the power rail, nets that were driven from powered down logic will float. In order to ensure that these floating outputs do not corrupt logic that is still powered on and active, the powered off domain must be isolated from the rest of the system. To isolate the domain, outputs from the switch off domain are driven to a known value by elements that remain powered on. There are three primary types of isolation: isolation one, which will drive the output to one, isolation zero, which will drive the output to zero, and an isolation keep, which will latch the output of the powered off domain and continue to drive the value after the power has been removed. After the power rail is reconnected to the supply, the isolation must be made transparent so that logic in the newly powered up domain can once again drive the outputs.

There are four major stages of a low power mode involving power switch off and isolation that are relevant from the verification perspective. These stages, shown in Fig. 1, are Power Down (operation of device while domain is powering off), Power Off (operation will domain is off), Power Up (operation while domain is powering up), and Power On (operation *after* power has been restored). Functional coverage points need to be identified in each of these stages, and stimulus needs to be written to achieve the coverage.

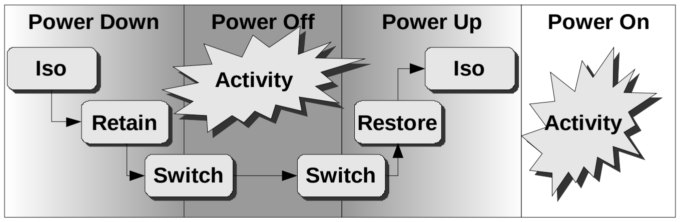

Fig. 1. Power scenarios that must be tested

The first stage, “Power Down”, isolation is asserted and the power is switched off. Correct operation is dependent on both the proper values being used for isolating the outputs and the sequence in which the logic is powered down. If the power is switched off before the isolation begins driving outputs, it is possible for unknown values to be propagated from powered off domain. For example, a zero may be sense coming from an active low interrupt which was to be isolated high. This zero will be treated as an interrupt, which will in turn wake the system out of the low power mode. In simulation, the X representation of the powered of domain will cause the logic in the interrupt controller to be corrupted.

The next stage is “Power Off”. Portions of the design are still active in this stage, and certain activities are still required to be performed. The correct isolation values are required to insure operation of the powered on portion of the device. For example, if a wait signal is isolated active, an erroneous access to the domain could cause the system to lock up. As in the example shown during power off, isolating an interrupt to the active state may cause the low power mode to be exited prematurely. In addition, it is important to verify that logic is not powered off erroneously. If logic is required to exit from the low power mode, for example, one must be certain that it is in the powered on domain.

The third stage is “Power Up”. During power on, it is important to ensure that isolation is indeed removed, and also that the correct sequence of switch and making the isolation transparent is followed: the isolation should not be removed until after power has been restored. In addition, care must be taken to insure that the primary outputs of the powered on domain are being driven to the correct state by the logic inside the domain. The power up sequence may need to reset the logic to ensure that the powered up logic are driving good values. In the forth stage, “Power On”, the logic that was powered off is once again being used in the system. The logic needs to be in a known state that enables all functionality required to be performed. The reset that drove the primary outputs to a known state before power up may be required to reset other portions of the design. An example of the sequencing of reset, isolate, and switch, along with the ideal response of a flop and output

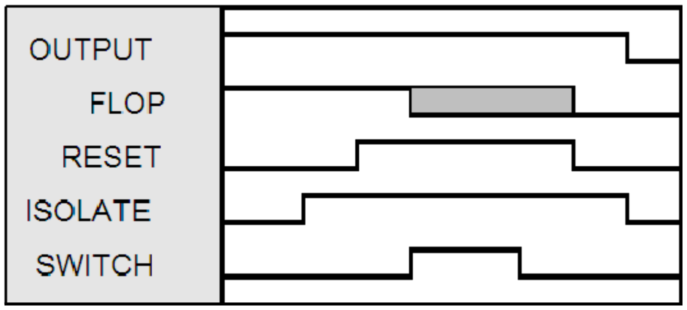

Fig. 2. Behavior of a register with a reset value of zero, an isolation value of one and a value of one at the time of power off

can be seen in Fig. 2.

III. State Retention

The system may require the state of a powered down domain to be equivalent both before and after the low power mode. If the logic in the domain is not required during the low power mode, it can still be powered down. One of the methods of retaining the state of the domain is to have software save register values to memory, and then write the saved values back after exiting the low power mode. This method can only be used if the requirements of the system only call for memory mapped registers to be saved. Another method is to use state retention registers for the logic whose state is required to be maintained. A portion of these registers remain powered on in order to retain the value while the vast majority of the logic is powered off. The result is a significant saving in leakage current during the low power mode while also maintaining the value of registers in the domain.

Retention may be implemented in either all the registers in a domain, or only in a portion of them as required. The verification effort changes for some of the four stages of power down that were discussed in Power Switch Off. During “Power Down”, the value of the registers being retained must be saved off by the system prior to power being switched off. There are no changes for the “Power Off” stage. For the “Power Up”, the register must restore the value after power is restored, but before isolation is removed. For a design were only a portion of the logic is being retained, the restoration of value must occur after the non retained portion of the design has been reset. Once again, one must be certain that primary outputs are being driven to a known good value before removing isolation. In the forth stage, “Power On”, the activities being tested now span power off. The activities should test that the correct portion of the logic has been retained. Operations that require the pre and post power mode state of the design to be maintained should be run. An example of the sequencing of retention, isolate, and switch, along with the ideal response of a flop and output can be seen in

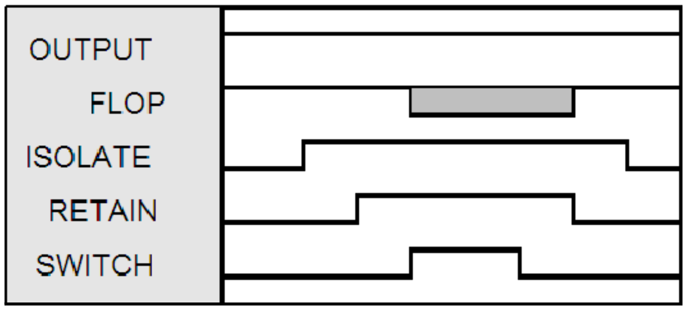

Fig. 3. Behavior of a state retaining register with a an isolation value of one and a value of one at the time of power off

Fig. 3.

IV. An Ad-Hoc Verification Solution

Faced with the challenge of verifying low power logic, an engineer may try to piece together an ad-hoc verification solution. If the domain being powered off is small, the verification may begin using the RTL netlist. Most likely, however, the number of memory elements will delay the verification effort until a gate level netlist is available.

RTL simulations require test bench elements to mimic the state retention, power switch off, and isolation. For isolation, the value to be isolated is forced on the primary outputs by a test-bench element. In some cases, an isolation wrapper may be inserted by the design team and the test-bench element used to mimic isolation will not be required. In order to simulate state retention, the verification or design engineer must first identify all registers whose state is to be retained. A test-bench element must be created to store values from the registers when the retention sequence occurs. The values are deposited back to the registers by the test-bench element after restore sequence take places. For power switch off, all nodes must be forced to ‘bX by test bench elements. One can see as the design grows this task becomes overwhelming.

Gate level simulations have isolation cells and state retention registers in the netlist. Therefore the test bench is not required to model these elements. The test bench is still required to model the power switch off unless a physical netlist with power and ground is used (and the verilog library elements support power down). Because there is no way to verify that the elements i.e., isolation and retention, identified in RTL are the same as are present in the gate netlist, the full verification suite must be re-run.

The need for gate level simulation means that design defects may not be identified until late in the cycle. An incorrect isolation value may not be found until days before tape-out. One may not identify a register that needed to be retained until after the masks have been created. With the cost of masks growing exponentially as the lithography shrinks, these can be extremely costly mistakes.

V. CPF Verification solution

There are many problems with the ad-hoc solution: a large effort to create test bench elements, reliance on gate level simulations, and the numerous opportunity to introduce human error. A CPF based solution addresses many of these concerns and enables an enhanced verification solution.

A CPF file can be considered a netlist or specification of power related information. Just as the RTL verilog netlist is read by both the design and verification tools, the CPF file is also read by multiple tools in the flow such as the synthesis tool and the RTL simulator. The file contains a description of the low power architecture of design: which domains can be powered off, which values to isolate, and which registers to retain. In addition, the power control signals that govern when to retain, restore and isolate are contained within the file.

When a simulation is run with the CPF file, the simulator takes care of isolating the output of the powered off domains. It will also drive powered down nodes to ‘bX, and is responsible for retaining and restoring the correct register values. Written in Tcl, wild cards enable registers and isolation ports to be identified easily. This removes the burden of writing cumbersome test bench code that existed with the ad-hoc solution. With the weight lifted, verification engineers can concentrate their efforts on ensuring that the four stages of the low power modes are adequately tested.

The second major benefit of CPF is that it can be read by formal tools. While using CPF in simulation enables one to accomplish the same tasks faster and more efficiently then the ad-hoc solution, using it in a formal environment allows one accomplish things that were impossible in the ad-hoc flow. The logical equivalency between what used for RTL simulation, the netlist and CPF file, and the gate level netlist can be proven. This removes the need to re-run every simulation from the RTL level on the gate level netlist.

In order to get the most from a CPF based verification flow, the same file which is verified by simulation should be used by synthesis. The chance of human error being introduced is decreased by having one central location for the power specification. An error present in the CPF used for synthesis is caught early by the RTL verification. In the ad-hoc solution, the error would not be found until gate simulations had begun. In addition, the CPF can be checked formally against low power rules at on the RTL level, rather than having to wait for a structural netlist.

VI. Conclusion

New techniques and methodologies introduced into submicron design can not be ignored by verification. While it is possible to use ad-hoc methods to verify these design elements, the methods are often far from ideal and can be slow, cumbersome and incomplete. CPF enabled simulators and formal verification tools enable a superior solution. A solution that is easier to implement and can let engineers spend their time where the return on investment is greatest. In addition, CPF enables a complete solution, allowing one not only to simulate in RTL, but also to formally prove equivalency between what was simulated and what was taped out.