A study on Low Power Check and Formal Verification

Chandana[1], Dr. Srividya P[2], Navaneeth R[3] and Venkata Rangam Totakura[4]

[1,2] RV College of Engineering, Bengaluru, India

[3,4] Cypress Semiconductor, An Infineon Technologies Company, Bengaluru, India

[1] chandana.ec17@rvce.edu.in , [2] srividyap.@rvce.edu.in , [3]navaneeth.r@infineon.com and [4] venkata.rangamtotakura@infineon.com

Abstract: *Low power techniques such as clock gating, power gating, multi-voltage, multi-threshold, etc. are utilized, to decrease the power dissipation in the design. To verify the proper implementation of these techniques, low power check is used. Different special low power cells such as Isolation cell, Level Shifter cell, Retention cell are discussed. Implementation of power aware design is explained using Unified power format (UPF). In an Application Specific Integrated Circuit (ASIC) design flow, from Register-transfer level (RTL) to Graphic Database System (GDSII), netlist undergoes many changes in the process of synthesis, Design for Testability (DFT) insertion, and Placement and Routing (PNR). Hence, formal verification has to performed to ensure whether its functionality remains the same. Basic mathematical algorithms underlying formal Verification such as Binary decision diagram (BDD) and Satisfiability (SAT) solvers are discussed.*

Keywords: clock gating, power gating, multi-voltage, multi-threshold, binary decision diagram, satisfiability solvers.

1. INTRODUCTION

With the increase in number of portable and wireless devices at an ever-increasing pace, the need for low power design is also increasing. Since the battery capacity is limited, in order to get the maximum efficiency, the device must consume least power. As the integration size of the chip reduces due to scaling of technology node and also the chips are clocking at a higher frequency than before, it is becoming more difficult to provide the adequate cooling and more expected battery life to the system. Few years ago, power dissipation was not a factor of concern because of low transistor density and low operating frequency of the devices. But it is an important issue now because of higher device density, higher operating frequency and concerns on environment and energy sources. To meet the requirements, advanced Low power methods are used by designers to save power. These methods do offer power savings, but significantly complicate the verification process. To verify the correct implementation of these Low power design techniques, Low Power Check is done. Incorrect implementation of low power design techniques may lead to design failure. Hence, Low Power Check is required.

The netlist is modified at several stages like synthesis, optimizations, DFT logic insertion, and placement and routing and hence it becomes necessary to verify if the functionality of the design is maintained. Verification of the design for consistency at each stage is a very cumbersome and exhaustive task which would take much time. The traditional methods use input vectors for verification and does not give 100% test coverage as we may not be able to exercise all the paths while testing. To tackle this a new method was developed to perform logic equivalence check without the above-mentioned bottlenecks. It is called Formal Verification. It is basically used to compare netlists at different stages of the

design flow. It checks the equivalence of the modified netlist at different stages of ASIC flow with the reference netlist/RTL.

2. Low Power Design

2.1. Low power techniques

This section includes some of the commonly used low power techniques.

2.1.1. Clock gating: Clock toggles with high frequency, clock tree consumes up to 60%-70% of the total dynamic power dissipation. If we can restrict the switching activity of the clock, we can reduce the dynamic power dissipation by a considerable amount. However, the amount by which we restrict the clock frequency affects the speed of the design. One way to reduce the dynamic power dissipation is by gating the clock, and allowing the clock to reach the design only when it is required. Clock gating is a special case of signal gating. This technique masks the unwanted switching activity of the clock with the help of a control signal, at the cost of extra hardware. Clock gating technique can be implemented at both architectural level and logic level either with the help of basic gates or latches.

2.1.2. Multi-Vt: Multi-Vt optimization technique uses cells with high Vt, low Vt and standard Vt cells to optimize power and timing. High Vt cells does not quickly respond to input changes, but has very less leakage current. Low Vt cells respond quickly respond to input changes, but results in more static power dissipation. Generally, 60% - 70% portions of the entire design are non-timing critical. Hence, high Vt cells can be used in these design portions which will decrease leakage current without affecting timing. Low Vt cells can be used in timing critical portions of the design. This technique reduces static power dissipation without affecting timing, but increases fabrication complexity.

2.1.3. Power gating: With power gating, the chip area is divided into power domains and special transistors are deployed to serve as current switches to cut off the leakage current when not in use, and reduce static power dissipation. The basic strategy is to establish two power modes, active mode and sleep mode. The mode in which power is gated off for certain blocks is called sleep/low power mode. When blocks are activated for operations it is called active mode. These modes are switched suitably to minimize power dissipation without impacting performance. Power gating has mainly two trade-offs, architectural trade-offs and delay that occurs while safely switching between the modes.

2.1.4 Multi-V_D: Multi V_D divides the whole design into voltage domains based on power and timing requirements of the blocks. Different blocks run at different functional voltage. Blocks whose performance matters the most, have higher V_D Other blocks have lower V_D to reduce both dynamic and static power dissipation. Level shifter cells are used between blocks operating at different V_D.

2.2 Special low power cells

This section explains the working of special cells used in implementing low power techniques in the design.

2.2.1 Isolation cell: The intermediate output values of the shutdown partition will be near to that of threshold value and will result in short circuit current. Hence, the outputs from shutdown partition driving inputs of active partition must be maintained at predictable signal levels. This is achieved by isolation cell. The isolation logic ensures that all inputs to the active partition are clamped to a fixed value. Isolation cell works in two modes, normal mode and isolation mode. The mode of operation is determined by the control signal from the power management unit. During normal mode of operation i.e., when both the domains are powered ON, isolation cell behaves like a buffer. During isolation mode, isolation cell clamps the output of a powered OFF domain driving an input of the

powered-ON to a stable value. There are three types of isolation cells, clamp-0, clamp-1 and latch-based isolation cell. Either of these can be used according to the requirement.

2.2.2 Level shifter cell: The output of a design partition working at voltage V_A, driving the input of a design partition working at voltage V_B, needs a voltage level translator in domain crossing to translate one voltage level to another voltage for proper functioning of the design. These voltage level translators are called level shifters cells. There are three types of level shifter cells, high to low level shifter, low to high level shifter and bidirectional level shifter. High to low level shifter is not mandatory, but is added to avoid stress on the transistors of lower voltage domain. Low to high level is mandatory for proper functioning of the design. Bidirectional level shifters are used in designs which use dynamic voltage scaling technique.

2.2.3 Enable level shifter cell: This cell is a combination of isolation cell and level shifter cell. Suppose, the output of a power domain A with voltage V_A is driving the input of a power domain B with voltage V_B and both the domains are active, a level shifter needs to inserted in the domain crossing. Also, if power domain A is shut off for a certain period of time, in any power state, an isolation cell also needs to be added. Thus, it is more efficient to add a cell which works as both isolation cell and level shifter cell, enable level shifter.

2.2.4 Retention cell: During sleep/ power shutdown mode, output of the cells in that domain settles to indeterminate values. Once the power domain is in active mode, the cells have to be reset and operation starts from the initial state. Hence, to avoid this the states of control flipflops needs to be retained during power gated mode before the power is shut off, in certain cases. This can be achieved by using a retention cell. A Retention cell has two supplies, a switchable supply which goes off during sleep mode and an always on supply which helps the cell to store the last state. Also, a retention cell has two extra inputs when compared to normal flipflop, save and restore. Save signal is assertion when power is shut off so that the last state is saved, and restore signal is asserted when the design goes into active mode again to restore the contents.

2.2.5 Power switch: In order to switch off power for certain blocks, power switches are used. Either a NMOS or a PMOS can be used as a power switch, which are controlled by power gating controllers. If V_{DD} is gated, the power switch is called header, and if ground is gated, the power switch is called footer. A PMOS is used as a header and a NMOS is used as a footer. Practically, a single switch is not enough for the entire design. Hence, many switches are inserted in parallel. Power switches should satisfy certain parameters, the size must be sufficient to handle the switching current, should have less leakage current. Specially design power switches are used for lesser IR drop.

2.2.6 Always ON cell: Irrespective of where they are placed, these cells are always ON. When dealing with shut-down partitions, there can be situations where certain cells in the shut-down partition need to continuously stay active. Always ON cells should be inserted between the nets connecting sleep mode domain and always ON logic/ active domain for proper functionality of the design. These cells are nothing but special or normal buffers and inverters and are necessary for save and restore pins of retention cells, control pins of switch cells and enable pins of isolation cells and enable level shifter cells.

3. Power Aware Design

3.1. UPF file structure:

UPF file captures the power intent of a design in the form of the structure shown in the Figure 1. UPF captures power distribution architecture, power strategy and usage of special cells. Power distribution architecture includes definition power domains, supply

rails and shutdown logic control. Power strategy includes definition power state table and operating voltages. Special cells are captured by definition of isolation strategies, retention strategies and level shifter strategies based on the requirement.

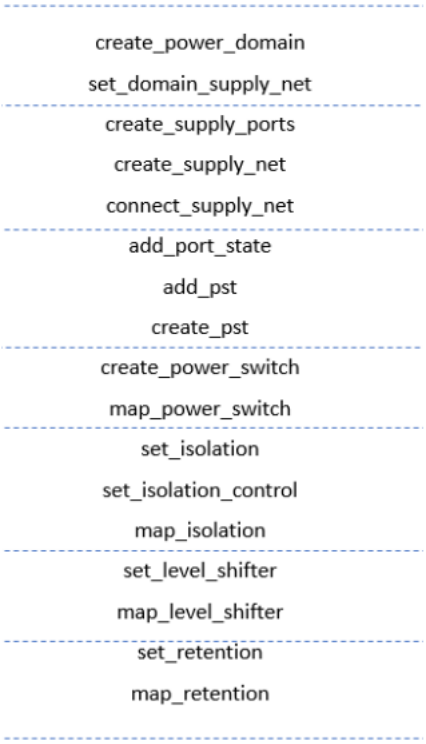

Figure 1. UPF file structure

3.2. Power aware verification

Steps involved in Low power flow,

1. Create an UPF file to define and capture the power intent for System on Chip (SoC). The UPF file can be used to explore power choices while keeping the design integrity documented in golden RTL.
2. Verify the contents of the UPF file using quality checks, which ensure that UPF is syntactically correct, power intent is complete, design and power intent are aligned properly. Finding the missing definitions by using formal techniques will save simulation and synthesis debugging later.
3. Through simulation, verify the correct functionality of the design with low-power behaviour superimposed on top of the functional behaviour, RTL.
4. Power shut OFF is simulated effectively, to ensure that the chip is functioning as intended, with few portions turned off. Control signals are generated from Power Management Block (PMB), which are specified in the isolation strategies, level shifters, etc. When these control signals are asserted, low power behaviour is triggered.
5. Just before shut off, isolation enable signal is asserted. But between isolation insertion and power off, retention signal is asserted by PMB, which causes the simulator to store the current values of all retention flops specified in UPF.
6. Conversely, on power up, retention flops restore the retained values and isolation values forced on outputs will be removed.

4. Low Power Check

4.1. Stages of running low power checks

Lower power checks are performed mainly at three stages,

1. Design/ UPF creation: At this stage, inconsistency between UPF parts such as, there is signal transmission between two different voltage domains but level shifter strategy is not defined, must be resolved by definition of level shifter strategy.
2. Post-synthesis: All the previous checks must be performed and UPF must be changed according to the design changes. Also, proper implementation of low power cells such as, missing isolation cell must be checked at this stage.
3. Post-PNR: Electrical connections of the design with respect to UPF along with previous checks must be performed.

4.2. Common low power checks

Some of the common low power checks are,

1. Missing isolation strategies at OFF to ON crossings.
2. Missing isolation cells at OFF to ON crossings.
3. Isolation cell/strategy is present in design/UPF when it is not required.
4. Clamp value of the isolation cell toggles clock/reset pin.
5. Missing level shifter strategies at domain crossings.
6. Missing level shifter cells at domain crossings.
7. Level shifter cell/strategy is present in design/UPF when it is not required.
8. Retention cell/strategy is missing when it is required.
9. Retention cell/strategy is present when it is not required.
10. Some of the design's cells have missing liberty attributes.

5. Formal Verification

Formal verification (FV) is the use of tools that mathematically analyze the space of possible behaviors of a design, rather than computing results for particular values. Rather than attempting specific values, FV tool will look at the entire space of possible simulations. Of course, it won't run all potential simulations, but it will utilize advanced mathematical approaches to consider all possible outcomes.

5.1. Advantages of Formal Verification

There are many advantages of FV,

1. It proves whether a design satisfies its specifications mathematically.
2. It analyses all possible design behaviours w.r.t its capacity and constraints.
3. It uncovers interesting corner cases that user might have not thought.
4. It is output driven analysis, it enables us to reason about the behaviors of a design at any point in the design.
5. With the help of FV, we can understand infinite behaviours, such as model behaviour over an unbounded time span.

FV can be used in several ways for design validation, one of the uses is formal equivalence check. Formal equivalence check refers to comparison of two models and determine whether they are equivalent. The models might each be RTL models, or schematics, or physical netlist, or this might involve the comparison between any of these levels of abstraction.

6. Basic Algorithms used in Formal Verification

6.1 Binary decision diagram (BDD)

Generally, a lot of redundancy is observed in truth tables, we need to take advantage of the sparseness of the actual data to generate much more compact representations of truth tables. BDD is a data structure which provides a compact representation of a Boolean function. BDDs have numerous beneficial qualities, such as providing a concise representation of a Boolean function, easy to manipulate, and being particularly powerful for FV applications, since they are canonical once the order in which the variables appear is fixed. A unique BDD is constructed, if we apply a set of standardized rules. So, any other equivalent design will have the same BDD representation. This means that a FV tool can confirm equivalence between two formulas just by converting both to BDD form and check if they are identical.

Theoretically, we can construct the BDD for a Boolean function as follows,

- Build a binary decision tree for the provided Boolean function, keeping in mind that no variable appears more than once along any path from root node to leaf, and that the variables always occur in the same order along every path from root node to leaf.
- Apply the two reduction techniques,
    1. Merge all the duplicate nodes.
    2. Delete a node, if both child pointers of that node point to the same child, because it is redundant.

A BDD can be created using a truth table given for a design. However, a design/specification is not expressed frequently as a truth table. It is more likely to be in the form of RTL. Fortunately, there is a well-defined set of standards for creating primitive BDDs that correspond to basic logic operations, as well as for combining primitive BDDs to build compose operations. To create a BDD for a design, we can use primitive BDDs to incrementally construct more complex BDDs.

Model checking is an algorithm that analyses the behaviour of a sequential system over time with the help of BDDs. A model checking algorithm can search over the possible future states and determine whether a property is violated, given a set of requirements defined as temporal logic properties and a finite state machine. A state space BDD is created for the initial state, as the search begins, and it is expanded to include all possible next states. The limitation of this method is space and time blowups.

6.2. Satisfiability solvers

The problem of determining if a Boolean formula is satisfiable or unsatisfiable is called Boolean SAT. If a Boolean formula turns out true for any values of Boolean variables, it is satisfiable or unsatisfiable. Consider showing that the implementation satisfies the requirement. (Implementation implies requirement) is a tautology, in a tautology, for any inputs, the formula evaluates to true. If it evaluates to true, implementation satisfies the requirement. However, it is more efficient to focus on searching for any assignment of variables where the implementation does not satisfy its requirements, since we are trying to find bugs quickly. If !(Implementation implies requirement), does not evaluate to true, then it can be concluded the implementation does satisfy the specification.

BDD based FV techniques incrementally build a representation of all possible design next states until a fixed point is reached when the structure contains all possible future outcomes. Then the algorithm can check whether requirements are violated in any of the reachable states. SAT solvers instead proceeds by iteratively building an expression to describe all possible execution paths from the design state and concurrently checking that

requirements are satisfied at each step. Eventually, either a counterexample is found or the expansion gets too big and the FV engine can no longer determine whether the requirements are satisfied for the current step. SAT solvers require less memory, can check much larger models, and does not require BDD ordering. Hence, SAT based techniques are better than pure BDD techniques.

SAT solvers represent the Boolean formula in conjunctive normal form, so that they can quickly determine a large formula is 0 without checking every term. SAT strategies focuses on the following insights,

- Divide and conquer approach.
- Identify critical problems to solve.
- Use known characteristics of realistic RTL designs.
- Leverage information from earlier failures in later stages.

These ideas along with some additional creativity have led to significant algorithm improvements enabling SAT to tackle industrial sized problems.

7. Conclusion

Power aware verification is necessary for today's designs. Common low power techniques along with special low power cells are explained. Power aware design with UPF along with common low power checks are discussed. Formal verification saves time and effort compared to traditional verification techniques. Basics algorithms used by formal verification tools such as BDD and SAT solvers are discussed.

REFERENCES

[1] *Seligman, Erik, Tom Schubert, and MV Achutha Kiran Kumar,"Formal verification: an essential toolkit for modern VLSI design", Morgan Kaufmann,* ***(2015)****.*

[2] *Grimm, Tomás, Djones Lettnin, and Michael Hübner. "A survey on formal verification techniques for safety-critical systems-on-chip." Electronics 7, no. 6* ***(2018)****.*

[3] *Bailey, Stephen, Gabriel Chidolue, and Allan Crone. "Low power design and verification techniques." Mentor Graphics, white paper* ***(2007)****.*

[4] *Klavakolanu, SR Sastry, M. Kama Raju, Fazal Noorbasha, and B. Raghu Kanth. "A review report on low power VLSI systems analysis and modeling techniques.", International Conference on Signal Processing and Communication Engineering Systems, pp. 142-146. IEEE,* ***(2015)****.*

[5] *Varadharajan, Senthil Kumaran, and Viswanathan Nallasamy. "Low power VLSI circuits design strategies and methodologies: A literature review.",Conference on Emerging Devices and Smart Systems (ICEDSS), pp. 245-251. IEEE,* ***(2017)****.*

[6] *Hazra, Aritra, Sahil Goyal, Pallab Dasgupta, and Ajit Pal. "Formal verification of architectural power intent." IEEE transactions on very large scale integration (VLSI) systems 21, no. 1* ***(2012)****: 78-91.*

[7] *Gourisetty, Venkatesh, Hamid Mahmoodi, Vazgen Melikyan, Eduard Babayan, Rich Goldman, Katie Holcomb, and Troy Wood. "Low power design flow based on Unified Power Format and Synopsys tool chain.", 3rd Interdisciplinary Engineering Design Education Conference, pp. 28-31. IEEE,* ***(2013)****.*

[8] *Swamy, Gitanjali. "Formal verification of digital systems.", Proceedings Tenth International Conference on VLSI Design, pp. 213-217. IEEE,* ***(1997)****.*

[9] *Xu, Zhan, Xiaolang Yan, Yongjiang Lu, and Haitong Ge. "Equivalence checking using independent cuts [logic design verification].",Test Symposium, pp. 482-485. IEEE,* ***(2003)****.*

[10] *Goldberg, Evguenii I., Mukul R. Prasad, and Robert K. Brayton. "Using SAT for combinational equivalence checking." Proceedings Design, Automation and Test in Europe. Conference and Exhibition 2001, pp. 114-121. IEEE,* ***(2001).***