Power Optimization in a High Performance Microprocessor Design

Mitch Dale

Calypto Design Systems

Introduction

To effectively reduce dynamic power, hardware designers must understand a multitude of clock gating transformations and have the practical experience to know when they should be applied. The tradeoff between power reduction and verification cost is not always clear so designers tend to be cautious, leaving power savings on the table. That is, until they don't have a choice.

Clock gating in all shapes and sizes

Power has become a primary consideration during hardware design. Dynamic power can contribute up to 50% of the total power dissipation. Clock gating is the most common RTL optimization for reducing dynamic power. Effective clock gating implementation requires skillful application and comprehensive verification.

There is a vast array of clock gating techniques available to designers. Clearly not all of these are equal when it comes to reducing switching activity. Many transformations are simple, while others are highly guarded, patented algorithms. Most clock gating is done at the Register Transfer Level (RTL). RTL clock gating algorithms can be grouped into three categories: system-level, sequential and combinational. System-level clock gating stops the clock for an entire block, effectively disabling all functionality. On the contrary, combinational and sequential clock gating selectively suspend clocking while the block continues to produce output.

Combinational clock gating is a straightforward substitution to the RTL code. It reduces power by disabling the clock on registers when the output is not changing. Opportunities to insert combinational clock gating can be found by looking for conditional assignments in the code. Clock gating logic is substituted when code like "if (cond) out <= in" is present. (See figure 1). Combinational clock-gating is now a feature in most RTL compilers. Power aware synthesis tools identify RTL coding patterns and make the appropriate substitution. Hardware designers only need to understand some simple RTL coding guidelines to gain the benefits of combinational clock gating.

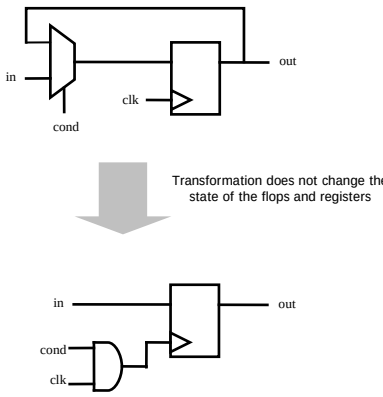

Figure 1 – Combinational clock gating

Since combinational clock gated flops maintain a one to one state mapping with the original RTL, Combinational Equivalence Checking Tools can be used for functional verification. This makes verification simple to setup and comprehensive. On the other hand because switching activity is eliminated only when data is not changing, the actual power savings is limited. In typical designs, combinational clock gating can reduce dynamic power by about 5%-10%.

Sequential clock gating alters the RTL micro-architecture without affecting design functionally. Power is optimized by identifying unused computations, data dependent functions and don't-care cycles in the original code. There are many types of sequential clock gating transformations. Identifying opportunities for sequential clock gating is difficult, requiring sequential analysis. One example of a sequential optimization is turning off subsequent pipeline stages based on a propagated valid condition. (See figure 2) Because of the additional logic this transformation only makes sense if the datapath is multiple bits wide.

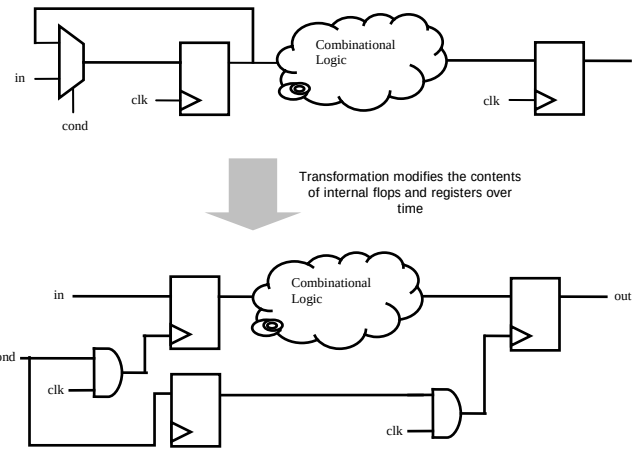

Figure 2 – Sequential Clock Gating

Sequential clock gating is a multi-cycle optimization with multiple implementation tradeoffs and RTL modifications. Consequently there is a greater demand on functional verification resources. On the other hand sequential clock gating can save significant power, typically reducing switching activity by 15-25% on a given block.

Since sequential optimizations change the state of the design, Combination Equivalence Checking Tools cannot be used for verification. This is not the case for Sequential Equivalence Checking (SEC). SEC tools can comprehensively verify sequential changes to RTL like clock gating.

Implementing clock gating schemes

System-level clock gating is designed into the original hardware architecture and coded as part of the RTL functionality. For example, sleep modes in a cell phone may strategically disable the display, keyboard or radio depending on the phones current operational mode. System-level clock gating shuts off entire RTL blocks. Because large sections of logic are not switching for many cycles it has the most potential to save power. On the other hand these modifications are invasive to the design function. The enable logic is part of an overall power management strategy and sometime includes consideration for software control. Verification of system-level power optimizations must be thought through in the system-level test plan.

Most hardware engineers understand how to write RTL in such a way that synthesis tools can recognize and automate combinational clock gating. Likewise hardware architects recognize and build in system-level clock gating opportunities. Even with these optimizations place there is substantial dynamic power saving opportunities remaining in the RTL [1] if designers understand the cost / reward tradeoffs of sequential clock gating.

Today there are no tools to automate sequential clock gating. However with SEC the verification problem is solved so designers can focus finding and implementing efficient sequential optimizations.

Sequential Clock Gating in the Design Flow

A standard practice is for design teams to create a block-wise power budget at the beginning of a project. As blocks are implemented, designers optimize those blocks that are over budget. Accurate power analysis for technologies 90nm and below depends on physical place and route information. Unfortunately this information is not available until late in the design flow. This means sequential clock gating is done late in the project, further highlighting the importance of comprehensive verification.

Identifying the enable condition is difficult for sequential clock gating. The enable logic can become very complex. Multi-cycle analysis of the design is needed, making it nearly impossible to ensure correctness by construction. To adequately verify sequential clock gating with simulation, testbenches must be monitored and modified to cover all enable/disable conditions. Further complicating the verification is the fact that clock gating typically cross hierarchies and interacts with neighboring blocks. Previously the cost of verification has limited the designer's ability to make sequential clock gating. Only mandatory sequential optimizations where allowed, that has changing with the availability of Sequential Equivalence Checking (SEC) tools.

SEC functionally verifies sequential optimizations by comparing the clock gated RTL to the corresponding original design. SEC uses formal sequential analysis to verify all possible input sequences that enable and disable clocks without testbenches or assertions. This has the advantage of saving the time of modifying testbenches and running regressions. Additionally, SEC efficiently verifies clock gating schemes that cross hierarchies and block boundaries with out having to conceive specific testbench sequences.

Case Study - Power Optimization in a High Performance Microprocessor

A hard requirement for the super-scaler, high performance, PowerPC design was it must use a standard package with cooling fan while operating at rates up to 667 Mhz. From the beginning of the project minimizing design power was a fundamental design goal. Every trick was used, including dynamic voltage scaling, multi V^{th}, sleep modes, low power memories and sequential clock gating.

Sequential clock gating optimizations were performed by senior engineers on "hot areas" of the design. One such design block was the instruction pipeline, about 1100 lines of RTL code.

The first challenge was to identifying the "don't care" states in the pipeline. Using this information and analyzing backwards across three to four pipeline stages, an opportunity for reducing power was identified. In this specific optimization, previous pipeline stages

are disabled in earlier cycles when it is determined that the output is not used in the current cycle. Figure 3 shows a simplified diagram of this logic optimization. As a side note, it would be simpler to only clock gate the final flop with vld_2. However this would not reduce the switching activity associated with PKT1 and PKT2 flops and combinational logic.

The difficulty implementing this optimization is keeping track of the signals that crossed pipeline stages and contribute to the enable condition. There are no automated tools that can provide this information so this analysis is done manually by reading the RTL source code and waveforms. To simplify the analysis, a methodology for naming signals based on pipeline stages was adopted.

From previous design knowledge, this particular clock gating transformation is only effective when more than sixteen flops can be gated. This is where the reduced switching activity is meaningful compared to the cost of additional enable logic. To eliminate glitching on the clock line, special clock generator cells were added to output of the enable logic.

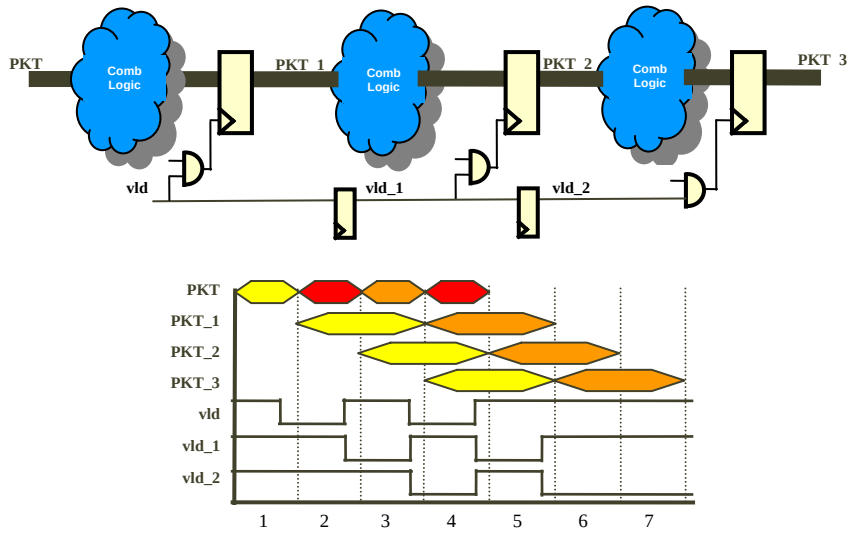

Figure 3 – Disabling pipeline stages when the output is not used

Since this transformation is a sequential RTL change, it cannot be verified with combinational equivalence checking tools. Previous projects showed that simulation regressions are biased towards testing active states in the design, whereas this transformation takes advantage inactive "don't care" states. With clock gating changes there is always a question about test coverage because it is difficult to test for all combinations of gating states and "don't care" assumptions.

On this project, Sequential Equivalence Checking (SEC) was used for functional verification. SEC identified designs differences as short (4 cycles or less) counter examples. This made it easy to quickly debug and locate errors. During the project, five

bugs design bugs where found using SEC. They ranged from the valid condition being a cycle late, to missing a term in the enable logic.

This project was successful on several accounts: power for this block was reduced, a methodology for analyzing and verifying sequential clock gating was developed and the project was completed ahead of schedule. In the past this type of optimization would take 1 to 2 weeks of creating testbenches, debugging and running regressions. With SEC the optimization was identified, completed and verified in 3 days.

Conclusion

RTL clock gating is a common technique for reducing dynamic power. Today there are no automated tools to identify or make sequential RTL clock gating optimizations. Such optimizations require experienced engineers that know when and how to apply the appropriate sequential change. Since this is a manually transformation, verification is critical. Sequential Equivalence Checking can verify clock gating, giving designers the confidence to make aggressive power optimizations late in the design process. The result is a lower power, higher quality design.

References

[1] Hai Li, Swarup Bhunia, Yiran Chen, T. N. Vijaykumar, and Kaushik Roy, "**Deterministic Clock Gating for Microprocessor Power Reduction**", Feb 2003