Using a Performance Model to Estimate Core Clock Gating Power Savings

John Griswell, Balaram Sinharoy, Richard J. Eickemeyer
IBM Server Group

Wael El-Essawy[1]
University of Rochester

Abstract
The goal of this study is to estimate the amount of power savings that could be achieved by clock gating if it were applied to the POWER4[2] core design. This goal was accomplished by using an architectural performance model coupled with reasonable estimations of the macro-level power consumption internal to the core. Using this technique, reasonable estimates for the average power saved and maximum power swings over various benchmarks and typical workloads were generated. The paper will cover a brief description of clock gating, an overview of the power modeling methodology, examples of typical clock gating events and some background information on the performance model used. While the study presented here is specific to the POWER4 core design, the methodology used here is applicable to a wide variety of designs and system models.

Introduction
Power consumption is quickly becoming a key factor in the design of modern day processors. Therefore it is important to be able to understand and model power saving techniques during the early stages of a processor's design. The goal of the study detailed in this paper was to estimate the amount of power savings that could be achieved by clock gating in the design of a typical modern day superscaler processor core, namely the POWER4 core [Tendler02]. The paper will cover the problem addressed by this work, the methodology used to solve the problem, a brief description of the clock gating events chosen, background information on the performance model used. While the study presented here is specific to the POWER4 core design, the methodology used here is applicable to a wide variety of designs and system models.

With modern day computers becoming faster and faster and by extension consuming more and more power there is a drive to design new computers with lower power consumption. This can be a problem if there are no power analysis tools available early in the design cycle when decisions which will have a broad impact on power consumption are being made. This problem can be compounded if the design will include clock gating. Clock gating is defined as shutting off the clock to certain areas of a chip during known periods of inactivity. Clock gating raises the stakes because the amount of power saved is highly coupled to the timings and occurrences of the events that trigger the clock gating. Therefore it is very important that tools be created to model clock gating and estimate the power saved by its use.

Methodology
The approach taken was to use an architectural performance model coupled with reasonable estimations of the macro-level power consumption internal to the core to estimate the power saved by clock gating, as in [Brooks00] and the maximum power swings for clock-gating related noise analysis [El-Essawy02]. Using this technique, reasonable estimates for the average power saved and maximum power swings over various benchmarks and typical workloads were realized. In each cycle, it is determined for each macro whether the macro is utilized or could be clock gated based on the activity during the cycle. The gated power or the active power for a macro is its contribution for the cycle.

---

[1] Wael El-Essawy is supported by IBM Fellowship and summer internship.
[2] Power4 is a trademark of IBM.

The power numbers for each macro in POWER4 were obtained by analyzing every circuit and macro schematic with CPAM [Neely00]. CPAM (Common Power Analysis Methodology for microprocessors) is an internally developed IBM design methodology and tool set for the highly detailed analysis of power as well as on-chip noise and reliability of the power distribution network of hierarchically designed, high-performance microprocessors. By performing circuit simulations on an extracted, layout view of a microprocessor design, instantaneous currents are calculated and summed over simulated operational cycles. CPAM takes into account the on-chip decoupling capacitors, wire-to-wire capacitance, intrinsic device capacitance as well as parasitics. With test and control inputs set at functional states, CPAM is run with "random" input vectors applied to the circuit, where CPAM controls the switching factor between consecutive input vectors to a user-specified values. CPAM also takes into account logic orthogonality between inputs (for example control signals for a multiplexer based on transmission gates). The worst-case switching factors are obtained by having the chip architects define worst-case simulation patterns. These patterns were run on both the core and full-chip logic models using the IBM TEXSIM logic simulator. Switching factors were obtained by simply counting the number of times each node toggled on a unit-by-unit basis and dividing by the number of simulation cycles [Warnock02]. From these analysis, a picture of the instantaneous power requirements as well as on-chip power supply noise is calculated.

There are some inherent benefits to using a performance model to estimate power savings. First if the performance model is available early in the design it can give rough estimates for power savings which can then be used to make high level design decisions in cases where power consumption is a concern. It can also give feedback on the level of power savings that can be achieved to ensure that the best methods are used, power saving goal is reached and the power swings due to clock gating will be relatively small. It allows the study of power consumption for various workloads and different modes of operation.

There are also a few drawbacks to using a performance model for this type of work. The power savings numbers will always be an estimate because, in general, performance models do not model all of the detailed inter-workings of the processor core; they just produce the same performance as the core, so some approximations must be made. In our case, the model is unable to calculate the power consumed or saved by lower or higher switching factors because the traces used to drive the performance model do not contain data values. This is not a limitation on the methodology, however. These limitations are acceptable given that the data produced is accepted as rough estimate to be used in conjunction with more detailed analysis once the design is farther along.

There are two modes in which the power saved due to clock gating can be estimated using this technique. The first is the average power saved over a long period of time. The second is the power saved on a cycle-to-cycle basis which gives insight into the power swing that can be expected due to clock gating activity. While both use similar methodologies they will be described separately.

The methodology for generating average power savings for entire runs begins with the performance model from which clock gating event frequencies are derived. Next clock gating event descriptions are received from the designers in order to link the events with a list of macros whose clocks will be gated when each event is active. Finally macro power numbers from the CPAM power model were used to estimate what the power used in each macro is. Then all three inputs were fed into a power script which calculates the power saved over the desired interval. Figure 1 depicts this interaction.

While it is nice to see broad power savings over the whole workload run for millions of instructions, there is concern about the power swings on a cycle-to-cycle basis causing noise problems on the chip. To that end the performance model was used to help estimate core power swings due to clock-gating. The fear is that if too many clocks were turned off or on at the same cycle or close to the same time that it could introduce unwanted noise on the chip. In order to calculate the power savings on each cycle the state of all the events had to be known on each cycle. To achieve this the performance model can output a bit vector which represents the state of each event on each cycle for a period of time (typically, over a million cycle) for various workloads. Figure 2 shows how the bit vectors are combined with other inputs to estimate the cycle to cycle power savings. Each bit position in the bit vector represents a separate event and by post-processing these bit vectors with the help of a key which maps each

bit position to an event and a macro list, the tool was able to estimate the power saved/used on any given cycle. Again CPAM macro power data was used for each macro as the third input. For each workload a one million cycle trace was taken and then it was processed to get a histogram of all the power swings for that period of time. During this process when a maximum power swing was detected the tool would create a small ten cycle sub-trace of that peak power swing for later analysis.

Criteria Used to Select the Clock-gating Events

Now that the problem has been defined and the methodology used to address the problem has been covered, a more detailed look at the clock gating events and the criteria used to select them is in order. A key goal for our study is to identify clock-gating events that are relatively easy to verify, does not lead to new critical paths and yet save most of the power. The microprocessors of today are very complex and their verification are becoming increasingly difficult. Clock-gating events add an additional layer of complexity to an already complex modern microprocessor. These events can also create new critical paths that needs to carefully engineered. These aspects of dynamic power management through clock-gating add to the design complexity and schedule risk.

In our power saving study we tried to balance the design complexity, timing and verification cost with the associated power saving benefit. If the clock-gating events are not chosen carefully, they can create various clock-gating related logic window conditions where logic bug can hide, or create a critical path that is extremely difficult to engineer to reach the frequency goal. If not chosen carefully, clock-gating events can also lead to scenarios that are difficult to anticipate and create in simulation or lab bring-up environment and reduce the level of test and coverage.

In our power swing study we tried to identify the clock-gating events that can lead to scenarios where we exceed our noise margin. To reduce the noise margins and still save power, one has to place additional decoupling capacitors very close to the trouble spots. Decoupling capacitors take up area and increases the cost of the chip. It can also lead to new critical paths and difficulty in achieving the targe t frequency, since area for additional decoupling capacitors can increase the distance between communicating logic macros or blocks. In the power swing study, we attempted to tradeoff power saving with noise and area needed for decoupling capacitance.

A few key principles were used to drive the clock gating event selection. First they should be commonly occurring events and the equation to generate them should not be too complex. This will reduce verification cost and chip timing problem. The percentage of cycles an event is active is known as the gating factor. In general the gating factors for various clock-gating events are known from the performance model. For this study, in most cases the events are selected so that the gating factor is high. If the gating factor is low, the event can still be selected if the corresponding power saving is deemed to be substantial. For this study, we assumed that a typical design should have a small number of events from each unit of the chip, to keep the verification cost low. Our goal should be to select 10 to 20 events for most units which save the bulk of the power. Extra caution should be used for clock-gating control macros and clock-gating conditions that gate off multiple stages within a macro or only parts of a macro. These can lead to design problems and increased verification cost. In some control macros only the first layer of latches can be deemed to be candidates for clock-gating, if it leads to substantial power saving. Finally there should not be too many fragmented events. Now here is a brief description of each of the major clock gating events this will give an idea of how detailed the model must be in order to generate the frequencies for these events.

Overview of Clock-gating Events

POWER4 is a state-of-the-art server processor which employs extensive out-of-order execution to extract higher instruction level parallelism (ILP). For a description of the POWER4 design, please refer to [Tendler02]. Following is only a partial list of potential power saving events that we have studied through our power saving methodology described in this paper.

1. Instructions are fetched from the instruction cache (on a cache and translation hit), and placed in a FIFO queue called instruction fetch buffer. With highly accurate branch prediction algorithm, Instruction Fetch Buffer is frequently full for commercial workloads with large instruction footprint and low instruction level

parallelism. This opens up opportunities to turn off the instruction cache, its directory, instruction side translation mechanism and the branch prediction arrays and associated logic, when the instruction fetch buffer is full.

2. Large percentage of PowerPC code does not generate microcode, so microcode ROM can be clock-gated quite effectively.
3. In POWER4, instructions are predecoded on their way into the instruction cache from the level-2 cache. With an early signal to indicate that L2 will return an instruction cache line, PLAs used to perform the predecode function for the instruction can be kept turned off most of the time. This is highly effective, because the 64KB instruction cache does not suffer cache misses very often.
4. Reorder buffer used in the POWER4 out-of-order execution core is a large structure consisting of several dataflow macros. Most of these macros logically represent tables, and most of the table entries are written only at dispatch time, so an entry can be clock-gated when no new group is being dispatched to it.
5. POWER4 uses a number of large rename pools for register renaming of various architected registers that improves performance. General Purpose Register file and Floating-point Register file are the biggest such rename pools. Large register files, called register mappers, are used to facilitate the register renaming and rollback on a pipeline flush. A mapper entry is written only at dispatch, so the entries can be turned off at other times.
6. After dispatch the instructions are placed in various distributed issue queues. From our performance analysis, we found that the Issue queues are not full most of the time. We can keep track of what is the next entry in the various issue queues that will be written if there is a dispatch and turn the other entries off.
7. When dispatch is blocked because issue queue entry or some other facility is not available, instructions are held in a queue at dispatch. Due to unavailability of such facilities, often we do not dispatch a group, so the dispatch queue can be clock-gated effectively to save power.
8. POWER4 has two fixed-point units and each has multiple subunits, such as ALU, Multiplier, Divider, Rotator, Leading Zero Calculator. At most, one of these are active in a given cycle, providing power saving opportunities for the others.
9. If there is no floating-point instruction to issue, the entire floating-point unit (FPU) can be clock-gated. A valid bit can flow through the pipestage of the FPU in advance of an issue of a floating-point instruction, so that a pipestage can be turned on only when its valid bit is on.
10. From performance analysis, we see that the write ports of GPR/FPR can be clock-gated quite effectively. Since the write operation is known early (at issue time, we know when the instruction will write back), implementation of such a clock-gating event should not have any timing problem. To keep the clock-gating logic simple, the clock-gating does not happen in the event the instruction does not really write back (e.g., a preceding branch mispredicts, interrupts, etc). In POWER4, it is rare that a instruction is issued but flushed before the write-back stage so the loss in power-saving is very small and the simpler logic helps with the verification effort, a good trade-off between power and complexity.
11. Read ports of the GPR/FPR can be turned off when there is no fixed-point or floating-point instructions in the queue and no new instruction is expected to come down in the next cycle.
12. In POWER4, loads and stores can execute out-of-order, but to maintain the PowerPC sequential execution model, facilities such as load reorder queue and store reorder queues are used. These queues are not always full. Most of these queues can be partitioned and partitions can be turned off when they do not have any valid entries.
13. If there is no access on a data cache port, that port can be turned off to save power.
14. Various second level translation mechanism (for example, the Translation Lookaside Buffer and Segment Lookaside Buffer) are usually accessed only when there is a miss at the first level translation structures (known as ERAT, effective to real address translation). Since most translations hit the ERAT, the SLB and TLB structures can stay turned off most of the time.
15. POWER4 has extensive data-prefetching mechanism which is not used when running commercial workload and can be clock-gated for such workloads.
16. POWER4 has extensive RAS features. For example, at power-on-reset, all arrays are tested through extensive ABIST mechanism. After power-on-reset, these ABISTs are usually not needed and can be turned off to save power.

Introduction to the Core Performance Model

A key component of this tool is the performance model used to generate the clock gating event frequencies. To that end this section will present some background information on the performance model. The model is a cycle accurate performance model of the core and the L2 cache structure and was used for design trade-offs for the POWER4 design. It is trace (not execution) driven which means that instruction traces and addresses (but not data value information) are fed into the performance model to allow it to accurately model detailed processes in the core. The performance model is written in an IBM proprietary modeling language called T. It has undergone extensive performance verification against the actual VHDL which ensures the performance accuracy of both. It allows a detailed look at the performance of the core and is now used for design tradeoffs for future cores. It also has support for varying degrees of speculative execution to cover the activity on incorrect branch paths. The model covers uni-processor performance and corresponding clock gating results. In a multiprocessor configuration, there will generally be more power saved from clock gating because of the lower utilization of the core.

The workloads used to drive the performance model consist of traces from a variety of workloads. These include commercial workloads [Kunke100], SPECint, SPECfp, and some technical workloads. Typically the model is run for 20 million to 40 million instructions when generating these power numbers. This is long enough to overcome trace start-up effects as well as capture the typical behavior of the workloads. The SPEC workloads have been sampled to reduce the length and the sample traces are compared to the full trace to ensure they are a representative subset.

Conclusions

There is a real need to be able model the power savings produced by clock gating early in the design of modern day processor cores. This can be achieved by using macro power data and a cycle accurate performance model. This can help drive high-level design decisions based on power consumption and can also be used to estimate the cycle-to-cycle power swings due to clock gating. In general if a reasonable performance model is being used for design tradeoffs it can be easily extended to make power savings projections early in the design process.

Acknowledgments

We thank our colleagues Doug Logan, who wrote most of the performance model and helped to add statistics for different power events and Joachim Clabes, who supplied us with the CPAM power data and the POWER4 design team for numerous discussions and suggestions with the clock-gating events that we have used for this study.

References

D. M. Brooks, et al., "Power-Aware Microarchitecture: Design and Modeling Challenges for Next-Generation Microprocessors," *IEEE Micro*, vol 20, no. 6, Nov/Dec 2000, pp. 26-44.

W. El-Essawy, D. Albonesi, and B. Sinharoy, "A Microarchitectural-Level Step-Power Analysis Tool," to appear in *Intern. Symposium on Low-Power Electronics and Design*, Aug. 2002.

S. R. Kunkel, et al., "A Performance Methodology for Commercial Servers," *IBM Journal of Research and Development*, vol. 44, no. 6, November, 2000, pp. 851-872.

J. S. Neely, H. H. Chen, S. G. Walker, J. Venuto, and T. J. Bucelot, "CPAM: a common power analysis methodology for high-performance VLSI design," *Proceedings, IEEE 9th Topical Meeting on Electrical Performance of Electronic Packaging*, October 2000, pp. 303-306.

J. M. Tendler, et al., "POWER4 System Microarchitecture," *IBM Journal of Research and Development*, vol. 46, no.1, January, 2002, pp 5-25.

J. D. Warnock, J. M. Keaty, J. Petrovick, J. G. Clabes, C. J. Kircher, B. L. Krauter, P. J. Restle, B. A. Zoric, and C. J. Anderson, "The circuit and physical design of the POWER4 microprocessor," *IBM Journal of Research and Development*, vol 46., no 1., January 2002, pp. 27-52 .

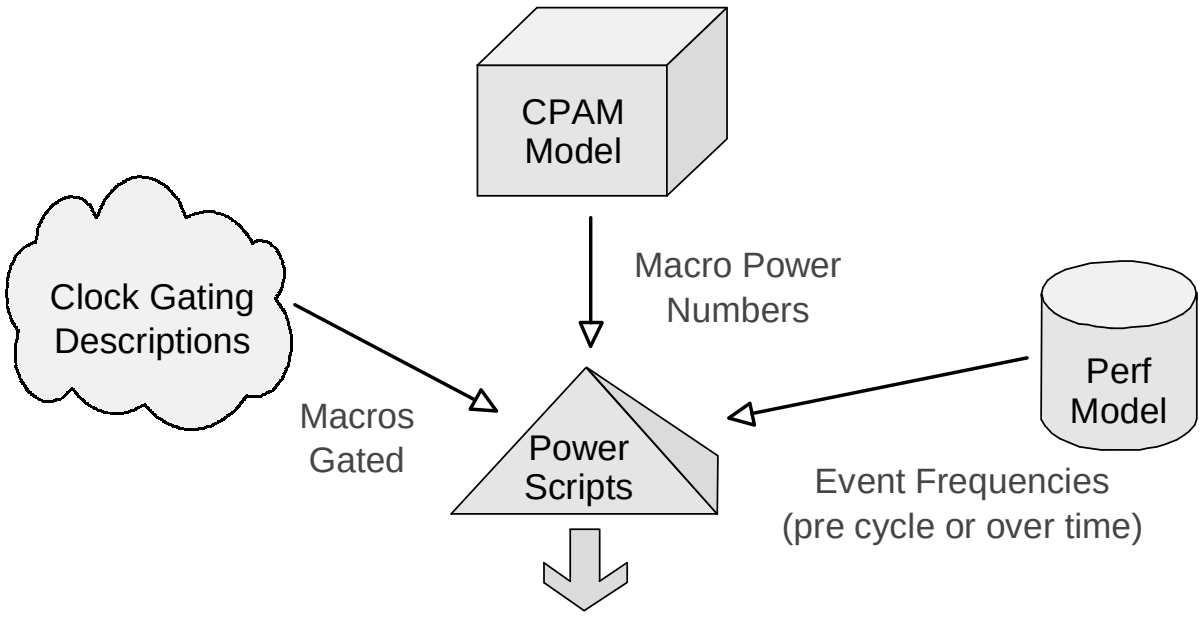
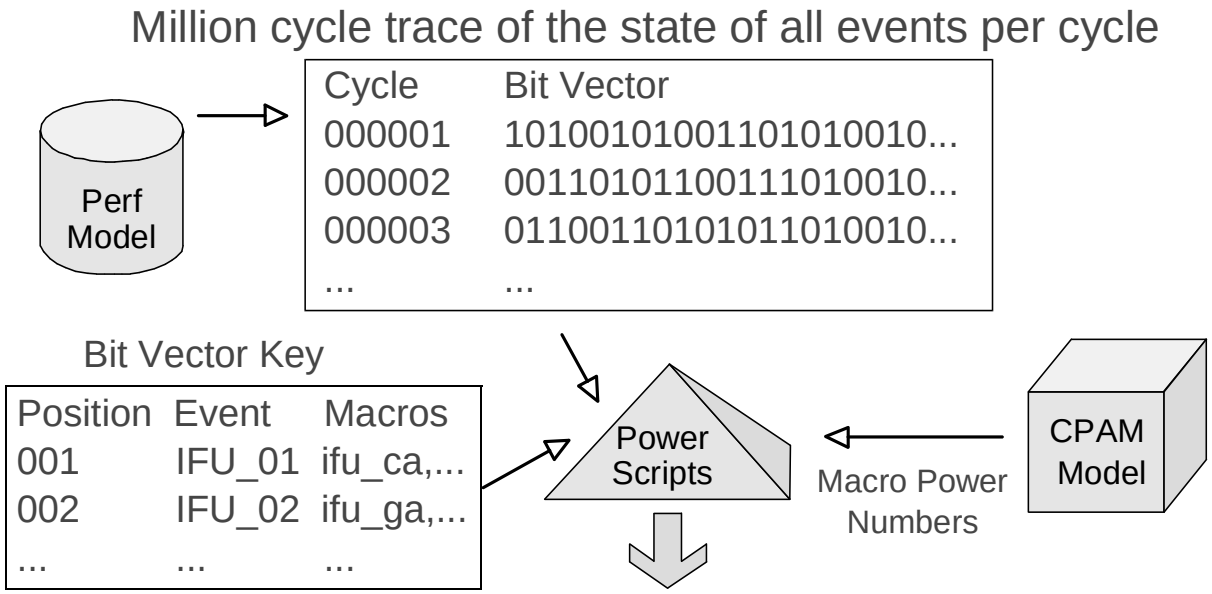

Power Modeling Methodology
CPAM Model
Macro Power Numbers
Clock Gating Descriptions
Perf Model
Macros Gated
Power Scripts
Event Frequencies (pre cycle or over time)
Total power saved over whole run and for each cycle, per unit and for the core

Figure 1

Power Swing Methodology



Histograms of the each cycles power swings for the core and each unit.

Figure 2