

# Stretching the Limits of Clock-Gating Efficiency in Server-Class Processors

Hans Jacobson Pradip Bose Zhigang Hu  
Alper Buyuktosunoglu Victor Zyuban  
IBM T.J Watson Research Center

Rick Eickemeyer Lee Eisen John Griswell  
Doug Logan Balaram Sinharoy Joel Tandler  
IBM Systems and Technology Group

## Abstract

*Clock-gating has been introduced as the primary means of dynamic power management in recent high-end commercial microprocessors. The temperature drop resulting from active power reduction can result in additional leakage power savings in future processors. In this paper we first examine the realistic benefits and limits of clock-gating in current generation high-performance processors (e.g. of the POWER4<sup>TM</sup> or POWER5<sup>TM</sup> class). We then look beyond classical clock-gating: we examine additional opportunities to avoid unnecessary clocking in real workload executions. In particular, we examine the power reduction benefits of a couple of newly invented schemes called transparent pipeline clock-gating and elastic pipeline clock-gating. Based on our experiences with current designs, we try to bound the practical limits of clock gating efficiency in future microprocessors.*

## 1. Introduction

Power and power-density limits constitute one of the primary design constraints in future high performance processors. In current CMOS technologies, dynamic (“switching”) power still dominates; but, increasingly, the static (“leakage”) component is threatening to become a major - or even the dominating - component in future technologies [2, 3].

Current generation high-end processors like the IBM POWER4 microprocessor system [14] are performance-driven designs where power limits are still comfortably below the power density limit afforded by the package/cooling solution available in the server markets. In designing and implementing future processors, the power and especially the power-density limits could become a potential “show-stopper” as the areas shrink and the frequencies increase, while supply voltages stop scaling down appreciably beyond 1.0 volt.

As such, techniques like clock-gating (e.g. [13, 6]) have quickly made their way into high-end, general purpose processors, like the POWER5 chip. In fact, it is probably fair to say that clock-gating has been introduced as the primary means of dynamic power management in recent server-class microprocessors. The appeal of clock-gating is that unused resources can be gated off to reduce (average) active power quite significantly, without any IPC loss. Power gating [8] and dynamic adaptation of on-chip resources like caches, queues, fetch and issue bandwidth, etc. [1] are also being actively examined in the research community; but, the potential IPC loss is often a drawback in the high-end server domain. As a method of controlling maximum power, however, dynamic throttling of clocks and data processing rates is now commonly used, even in high-performance processors (e.g. [5]).

In this paper, we first focus on quantifying the realistic benefits and limits of classical clock-gating for a workload suite composed of selections from SPEC<sup>TM</sup> benchmarks, commercial traces and scientific application kernels. We start with a POWER4/POWER5 class machine, and examine the power savings potential across the workload suite in using fine grained clock-gating. The results show that a reduction in total core power of 20-30% is a realistic target for clock-gating in high-performance processors. This is in contrast to what one might infer from early stage modeling where the savings opportunity is much larger.

In the second half of the paper, we look beyond classical clock-gating to see how we can further reduce unnecessary clocking in pipeline latches, without degrading IPC performance. We first extend stage-level clock-gating to register-level clock gating. Subsequently, we examine the potential of a newly invented paradigm called transparent clock gating [9]. We also look at a new elastic pipeline clock gating technique [10] that provides power efficient implementations of pipeline stalling. We show that these advanced extensions of clock-gating have the potential of reducing clock power in pipeline latches (i.e., not including register files and arrays) by an additional 66% on top of stage level clock gating for a floating point unit.

## 2. Conventional Clock-Gating: Fundamentals, Potential and Actual Benefits

In this section, we first provide a review of the logic-level fundamentals behind clock-gating, implemented at various levels of granularity. We then report early stage projections of potential power savings from clock gating in a POWER5 processor. We also report on later stage projections based on more accurate modeling.

### 2.1. Clock-gating basics and criteria

Figure 1 depicts a typical clocking arrangement used in pipelined data flow logic within a high-end microprocessor, like the POWER5 chip. The bank of latches is clocked via an AND gate that has a valid-bit signal from the previous pipeline stage. A stall-bit from the next pipeline stage is used to recirculate the current data during a pipeline stall. The latches are clocked only when there is valid data available from the previous stage or when the data needs to be held. In alternate designs the stall bit can also be used to gate the clock to further improve clock gating efficiency. Results in section 2 assume recirculated data while those in Section 3.3 assume clock gated stall implementations.

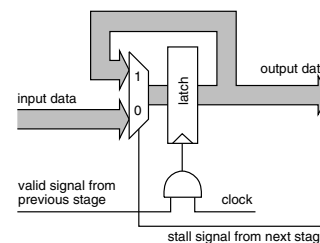


Figure 1. Clock gated latch with data recirculation.

### 2.2. The benefits of clock gating

In modern high-frequency microprocessors, roughly 70% of the active (switching) power is consumed by the clock circuitry and its latch load alone. The major part of the clock power is dissipated close to the leaf nodes of the clock tree that drive latch banks. Gating the clock at the last few levels of the clock buffers is therefore an effective way to reduce active power. Since a clock gated latch keeps its current data value stable, clock gating prevents signal transitions of invalid data from propagating down the pipeline thereby reducing switching power in the combinational logic between latches.

In addition to reducing dynamic power, clock gating can also reduce static (leakage) power. Leakage through CMOS devices is

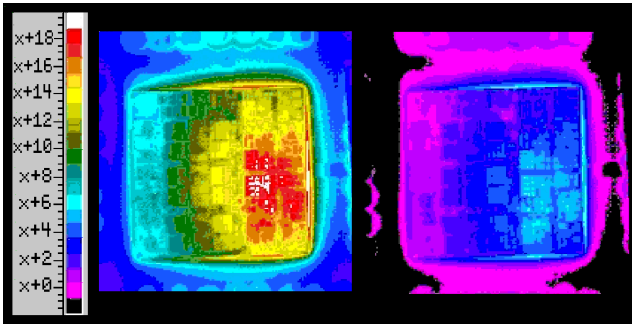


Figure 2. Relative POWER5 processor temperature (Celsius) - without clock gating (left) and with clock gating (right)

exponentially dependent on temperature [7]. The temperature reduction that clock gating provides can therefore significantly reduce leakage power. Figure 2 illustrates the temperature plot of the POWER5 microprocessor without and with clock gating, respectively. (These plots were obtained through direct, temperature-calibrated infra-red imaging of a POWER5 chip executing a pseudo-random mix of test cases). The figure illustrates a temperature drop of over 10 degrees Celsius in the hottest regions of the chip. Figure 3 in turn shows how leakage depends on temperature. From the graph it can be gleaned that at around the 80 degree point, each degree of temperature reduction translates into roughly 1.6% reduction in leakage. Given different temperatures at different points of a processor core, it can be expected that a uniform 10 degree reduction in temperature across the core would result in approximately 10% reduction in leakage.

### 2.3. Clock gating constraints

The main limitations for the application of clock gating is timing on the clock gating signal and the ability to group latches with identical gating conditions. Some latch groups may be too small to be considered for clock gating due to design complexity and power overhead of the associated clock gating logic. With increasing wire delays, placement of latches close to the cone of logic feeding the data input may conflict with the placement necessary to group a set of latches for the purpose of clock gating. In addition, the logic required to compute when a latch should be clock-gated can become quite complex. The clock gating signal may also have to fan out to many clock drivers if the latch group is large. These delays may make it difficult to reach timing closure.

Another problem is the inductive noise ( $Ldi/dt$ ) on supply voltage rails caused by clock-gating. In order to dampen the current surges that result from clock-gating, designers use on-chip decoupling capacitors that can contribute significantly to leakage power, thereby eroding some of the savings achieved through clock-gating.

### 2.4. The granularity of clock gating

Clock gating can be performed at many different levels of granularity. At the unit level, all pipeline stages of the unit are clocked as long as there is any instruction present in any stage of the unit. At the stage level, only the pipeline stages where instructions are present are clocked. Intuitively, finer grain clock gating result in larger power savings, but are also more complex to implement.

Many operations in a microprocessor are mutually exclusive. During the execution of commercial workloads, the floating point units experience a very low utilization. Conversely, many scientific workloads are floating-point intensive, leaving the integer units much less utilized. Such mutual exclusiveness has, in the past, made it possible to save considerable clock power through unit-level clock gating alone. In current generation microprocessors, however, si-

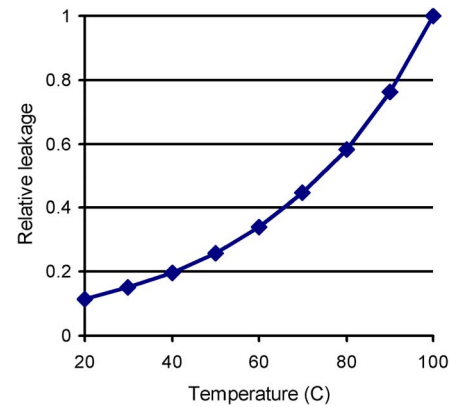


Figure 3. Leakage dependence on temperature.

multaneous multithreading (SMT) [3, 11] has reduced the effectiveness of unit-level clock-gating. This is clearly because SMT increases the average utilization of the execution resources, since the latter is now shared across multiple, diverse threads. Thus, in going from single-threaded to SMT processors it is important to refine the granularity of clock-gating from unit-level to stage-level.

### 2.5. Clock-gating efficiency results

We define clock-gating efficiency (CGE) for a given input workload as:  $CGE = [1 - (average\ clock-gated\ power) / (maximum\ unconstrained\ power)] * 100\%$ , so that higher numbers imply greater levels of power reduction. Figure 4 shows the computed CGE across various single and SMT workloads (SPEC2000 suite) using the Turandot/PowerTimer [4, 12] power-performance simulator. For single-threaded results, we run Turandot in POWER4 configuration; and for multithreaded mode, we run it in a POWER5 configuration, with correspondingly larger resource sizes, where applicable [11]. In each case, we assume a fine-grain, stage-level clock-gating structure. In such early-stage microarchitectural simulation, even though the pipelines are modeled accurately, it is difficult to estimate the per-stage fraction that is ungatable. (As explained in section 2.3, cycle-time constraints often prevent full realization of the clock-gating potential of any given pipeline stage macro). In such early-stage modeling, we added a flat 10% net power overhead to account for leakage and additional "no-load, idle active power" overhead was added in to factor in the effect of ungatable latches; this was deemed to be conservative enough for the target technology. As seen from Figure 4, the early-stage projection of CGE ranged from 38-53% for single-threaded POWER4 core pipelines and 43-54% for SMT-mode POWER5 core pipelines). Note that the CGE for an SMT-mode POWER5 processor is not below that of the single-threaded POWER4 processor, because the POWER5 design [11] increased some of the resource sizes (e.g. the physical register file) in adding SMT to the POWER4 microarchitecture.

In later stage projections, the modeling team used a detailed, cycle-accurate, POWER5 simulator (called the M1 model) to project CGE values, using a much more refined power model. This model tries to account for realistic estimates of ungatable logic and latches on a macro-by-macro basis. Although we do not report the exact CGE estimates obtained from M1-based analysis in this paper, suffice it to say that the effective M1-reported CGE values are lower than first-cut Turandot/PowerTimer based projections. For the very worst-case scientific application kernels, we saw CGE values as low as 19%, while more typical commercial SMT workloads yielded more than 30%. The M1-based numbers generally matched up well with post-silicon measurements in the laboratory.

In future leakage-dominated CMOS technologies chip-level effective CGE values may increase somewhat, assuming that the per-

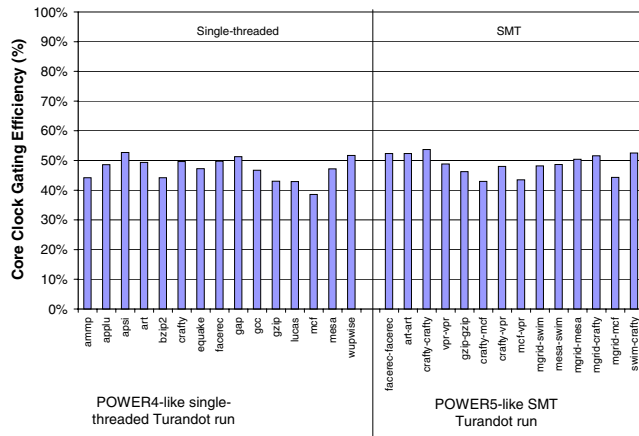


Figure 4. Clock gating efficiency for Turandot simulation.

centage of leakage power is forced to remain constant, because of: (a) additional exploitation of clock-gating opportunities in the non-core storage and interconnect hardware; and (b) additional percentage of leakage power savings due to temperature reduction brought on by clock-gating (see Figure 3); and (c) deeper pipelines, for higher frequency design points. On the other hand, in view of the new trend of multi-core designs using lower frequency cores, the effective chip-level CGE values are expected to saturate to values around 30%, for typical commercial workloads of interest. And, the value may well dwindle to 20% if there is a reverse trend in terms of pipeline depth and core complexity. In observation of this saturating (and possible diminishing effect), in the rest of this paper, we focus on addressing the following question: are there further improvements in clock-gating that can be devised to help increase the effective CGE values beyond current practical limits? Section 3 presents some promising new clock-gating improvements developed by our research team members, that go beyond conventional clock gating in an attempt to achieve this goal.

### 3. Beyond Conventional Clock-Gating

In this section we look beyond classical fine-grain, stage-level clock-gating to investigate the potential of suppressing many other unnecessary (or redundant) clocked latch transitions within a processor.

#### 3.1. Register level clock gating

Register level clock gating takes advantage of pipelines where the flow of data is divided into multiple paths. One such case is the front end of superscalar microprocessors. For example, in an 8-way superscalar pipeline, a pipeline stage processes up to eight instructions in parallel. When fewer instructions are present, e.g., due to hitting a cache line boundary, the registers in the pipeline stage not receiving a valid instruction can be clock gated. A more complex case is when the data of a single instruction can take different paths through the pipeline logic depending on what instruction type and data values are being processed. This case is explored here in the context of a floating point unit of a current generation server class microprocessor. The register level clock gating takes advantage of the fact that, for example, the data of a multiply instruction takes, at least partially, a different path through the pipelined circuit than the data of a compare instruction. Any registers that are not actively used for a certain class of instructions can be gated off when no such instruction is present in the associated pipeline stage.

Figure 5 illustrates the relative clock power savings achievable for different levels of clock gating in the datapath portion of the

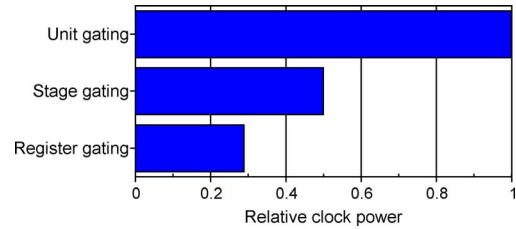


Figure 5. Relative clock power for floating point unit with different levels of clock gating.

floating point unit for a mix of floating point intensive kernels. In this case, stage level clock gating reduces the clock power by close to 50% over unit level clock gating. Register level clock gating in turn reduces the clock power by 44% over stage level clock gating. As illustrated, register level clock gating can significantly reduce the clock power in units where the pipeline has multiple paths for the data flow.

#### 3.2. Transparent clock gated pipelines

Transparent clock gating (TCG) [9] introduces a new way of clock gating pipelines. It extends upon traditional clock gating techniques and can significantly improve the opportunities available for clock gating. In traditional clock gating, latches are held opaque to avoid data races between adjacent latch stages. Such clock gating is based on the concept of *data propagation*. It takes  $N$  clock pulses to propagate a data item through an  $N$ -stage pipeline.

In a transparent clock gated pipeline, latches are held *transparent* by default. Transparent clock gating is based on the concept of *data separation*. Assume that a pair of data items  $A$  and  $B$  simultaneously move through a TCG pipeline. A data race between  $A$  and  $B$  is avoided by *separating* the two data items by clocking or gating a latch stage opaque, such that the opaque latch stage acts as a barrier separating the two data items from each other. The number of clock pulses required for a data item  $A$  to move through an  $N$ -stage pipeline is no longer only dependent on the number of pipeline stages, but also on the number of clock cycles that separate  $A$  from the closest upstream data item  $B$ . For an  $N$ -stage pipeline, where  $B$  follows  $n$  clock cycles behind  $A$ , only  $\text{floor}(N/n)$  clock pulses have to be generated to move  $A$  safely through the pipeline.

Figure 6 illustrates a transparent pipeline. Consider two data items  $A$  and  $B$  separated by two clock cycles moving through the three stage transparent clock gated pipeline (shadowed latches). The input and output environment (stages 1 and 5) of the transparent pipeline segment is made up of latch stages clock gated at the stage level in traditional opaque fashion. The clock waveforms for  $A$  and  $B$  propagating through a transparent clock gated and opaque clock gated pipeline respectively are illustrated in Figure 7.

When data item  $A$  enters the pipeline its value is held stable by the opaque clock gated latches of the input environment (stage 1).  $A$  therefore does not have to be captured by any of the downstream transparently clock gated latches until  $B$  arrives at stage 1 two clock cycles later and overwrites  $A$ . At this time  $A$  has reached stage 3 of the pipeline and is captured there by enabling the local clock ( $clk_3$  in Figure 7) to transition low. The opaque latches in stage 3 now separate  $A$  from  $B$  and there is no race between the two data items. By the time  $B$  reaches stage 3 and again overwrites  $A$ , data item  $A$  has reached the output environment and is captured in stage 5.

As illustrated in Figure 7, for this example, the TCG pipeline needs to generate the equivalent of only one clock pulse, while the traditional opaque clock gated pipeline needs to generate six clock pulses. In general, in microprocessor pipelines where resource limitations and data dependencies create bubbles in the pipeline,  $\text{floor}(N/n)$  is typically significantly smaller than  $N$ , resulting in

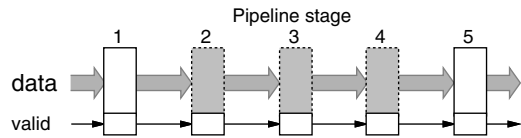


Figure 6. A three-stage transparent clock-gated pipeline.

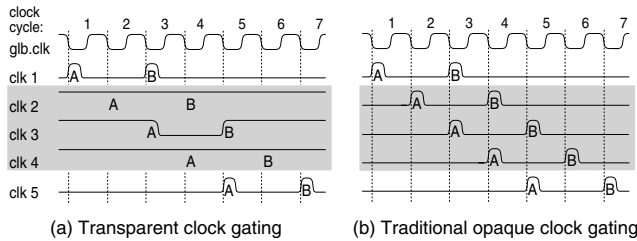


Figure 7. Example waveform trace of the local stage clocks for the pipeline in Figure 6 propagating data items *A* and *B*.

noticeable reduced clocking for TCG pipelines. Detailed circuit implementations of TCG pipelines can be found in [9].

Figure 8 illustrates the clock power savings achievable when using TCG as compared to traditional stage level clock gating. The results were obtained through microarchitecture level simulation of a server class microprocessor over a range of commercial, TPC- $C^{TM}$ , and floating point loop benchmarks. The graph shows clock power reductions close to 50% over traditional stage level clock gating for the fixed point and load-store units under commercial and TPC-C workloads. Even under heavy floating point workloads where fewer bubbles are available in the pipeline due to better branch prediction and a higher degree of instruction level parallelism, the clock power in the floating point unit can be reduced by 34%. Note that the clock power is normalized individually to each entry in the graph and not across units or workloads.

It is clear that in pipelines where bubbles are present between valid data items, transparent clock gating techniques can significantly reduce the number of clock pulses compared to traditional opaque clock gating techniques.

### 3.3. Elastic pipeline clock gating

A major concern in modern microprocessors is the problem of stalling high-frequency pipelines. Stalling occurs frequently in the front end of the processor core in pipeline stages preceding the instruction issue stage due to data dependencies between instructions. The main problem with stalling a high-frequency pipeline is the short cycle time available to propagate a stall signal, indicating the need to hold the current data, to upstream pipeline stages. The stall signal is typically heavily loaded and needs to propagate over long

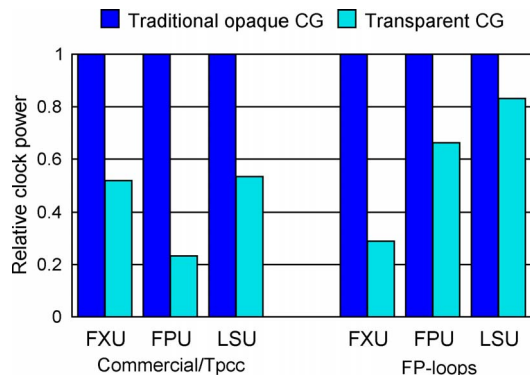


Figure 8. Clock power reduction for TCG vs. traditional stage level clock gating.

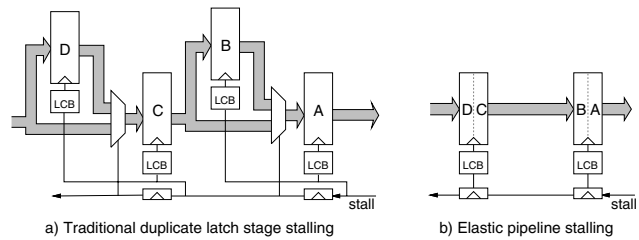


Figure 9. Different stall approaches.

distances to the many latches of several pipeline stages. This problem gets worse with deeper pipelining and increasing wire delays due to technology scaling.

Due to limited cycle reach it has become necessary to latch the stall signal after a certain distance as it propagates backward along the pipeline. At each such latch point the stall signal is delayed by one clock cycle. A stalled stage needs to hold its current data and cannot receive new data. To avoid losing data, an extra stall buffer is therefore needed to capture arriving data items until the upstream stages have seen the stall signal and stop pushing new data.

A possible solution for progressively stalling high-frequency pipelines is to implement the stall buffer through an additional latch stage set in parallel to the original latch stage as illustrated in Figure 9(a) (the LCBs implements the local clock buffers and clock gating). A multiplexor is needed to enable the latch stage to read its data either from the upstream stage when there is no stall, or from the stall buffer when the pipeline restarts after a stall. This solution is expensive in terms of circuit timing as well as area and power. The duplicate latch stage and stage wide multiplexor introduce extra clock and leakage power. In addition, the extra capacitive load and delay through the multiplexor require up-sizing the transistors of the data- and control-path to recover the delay, resulting in additional dynamic and static power.

We have developed a significantly more cost-effective approach to implementing the stall buffer, through an idea we call *Elastic Pipelining* [10]. The elastic pipeline does not require any additional latches or multiplexors in order to implement stall buffers in master/slave pipelines. Instead, the technique takes advantage of the fact that master/slave latches have the capacity to store two distinct data items, one in the master latch and one in the slave latch (see Figure 9(b)). While a master/slave latch cannot store more than one data item while the pipeline is actively propagating data, during a stall condition the current data is held still and only occupies the slave latch. This leaves the master latch free to act as a stall buffer and capture data arriving from upstream. The idea of the elastic pipeline is to dynamically increase its effective depth, and subsequently its storage capacity, in segments of the pipeline where no data needs to move. In high-frequency pipelines elastic clock gating techniques can provide a pipeline architecture with significantly improved timing and power characteristics.

Figure 10 illustrates simulated results for four types of stall implementations in an experimental high-frequency 32-bit multiply-accumulate (MAC) unit. The “Elastic” entry represents the implementation of an elastic clock gated pipeline. The “Unit” entry represents stalling at the unit level, while “Stage” and “Stage-II” represent stalling at the stage level with additional latch stages as stall buffers (Figure 9(a)). The “Stage” implements optimal clock gating (only clocked when capturing stalled data), while “Stage-II” implements non-optimal clock gating (clocked each time a valid data item arrives) in an attempt to improve stall signal delay by reducing stall signal distribution for the first stall cycle. The elastic pipeline approach has an 18% reduced delay on the worst case stall signal, a 27% reduction in dynamic stall power under worst case data switching, an estimated 44% reduction in leakage power, and a 33% re-

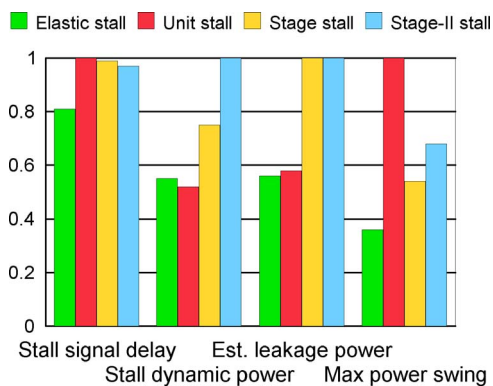


Figure 10. Relative worst-case stall signal delay, dynamic stall power for a stall/unstall event, area-based leakage estimate, and maximum power swing (di/dt) for a stall/unstall event for different stall implementations of a MAC unit.

duction in worst case power swing compared to the optimally clock gated “Stage” implementation. Detailed circuit implementations of elastic pipelines can be found in [10].

#### 4. Summary of advanced clock-gating methods

To assess the improvement in clock gating efficiency when the three presented techniques are combined, we evaluated their compound impact on clock power savings in the context of a floating point unit (FPU) excluding the register file. In the evaluation, only the datapath was clock gated with register level techniques as control logic is far less amenable to such type of clock gating. For the transparent clock gating technique, latches with feedback (about 30% of the latches in the control logic) were not gated in transparent mode to avoid combinational loops. For the elastic clock gating technique, the stall signal is latched at every third pipeline stage, and stalls occur on average 5% of the time. In the non-elastic implementation the first stage to receive the stall signal uses “Stage-II” type stalling while the other stages use “Stage” type stalling (see Section 3.3) due to timing constraints on the stall signal.

Figure 11 shows the calculated compound effect of the three clock gating techniques when they are all applied to an FPU. The elastic clock gating can save about 5% clock power over traditional stage level clock gating and stalling techniques. The register level clock gating can provide an additional 25% clock power savings on top of the elastic clock gating. Finally, the transparent clock gating technique can add another 36% clock power savings on top of the elastic and register level clock gating techniques. Under the given assumptions, these techniques subsequently have the combined potential to reduce clock power by 66% over a traditional stage level clock gated and buffer stalled design in FPU-like processor units.

While this is a marked improvement in clock gating efficiency over traditional clock gating, this power saving potential must be put in a perspective of the whole microprocessor core. Many resources on the core cannot take advantage of further refined clock gating techniques (such as register files and arrays). Assuming that

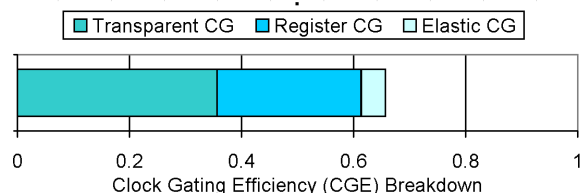


Figure 11. Compound clock gating efficiency for elastic, register, and transparent clock gating of a floating point unit.

all latches in a processor core can be clock gated to the same degree as those in the FPU, the additional clock power savings from using the presented clock gating techniques can be expected to be in the range of 8-10% of the total core power. Under these assumptions, given a base line core with a 20-30% CGE obtained from traditional stage level clock gating, the effective upper limit of core CGE is expected to lie in the range of 30-40%. Note that these numbers may change as the degree of gatability of various units are taken into account. We are currently working on quantifying the second order effects that these improvements in the clock gating efficiency bring due to reduction in switching, temperature, and device sizes.

#### 5. Conclusions

We provide an industrial perspective of clock-gating based on initial experiences gathered during the design of clock-gated designs within IBM’s server-class microprocessor offerings. Although clock-gating is now entrenched into the basic design methodology and the overall savings in power are significant, the increasing effects of leakage power and tighter timing paths make it hard to go beyond or even sustain the net 20-30% core power savings that is achievable from this paradigm alone. While the advanced clock gating techniques presented in this paper show promise in improving the clock gating efficiency further, the additional chip power savings achievable through these techniques may not be enough in future high-end microprocessors dominated by leakage power.

#### References

- [1] ALBONESI, D. H., ET AL. Dynamically tuning processor resources with adaptive processing. *IEEE Computer* 38, 12 (Dec 2003), 49–58.
- [2] BORKAR, S. Design challenges of technology scaling. *IEEE Micro* 19, 4 (Jul/Aug 1999), 23–29.
- [3] BROOKS, D. M., ET AL. Power-Aware Microarchitecture: Design and Modeling Challenges for Next-Generation Microprocessors. *IEEE Micro* 20, 6 (November/December 2000), 26–44.
- [4] BROOKS, D. M., ET AL. New methodology for early stage, microarchitecture-level power-performance analysis of microprocessors. *IBM Journal of R&D* 47, 5/6 (2003), 653–670.
- [5] COLWELL, B. *The unabridged Pentium 4: IA32 processor genealogy*. Addison-Wesley, 2005.
- [6] GOWAN, M., ET AL. Power Considerations in the Design of the Alpha 21264 Microprocessor. In *DAC* (June 1998), pp. 726–731.
- [7] HEO, S., BARR, K., AND ASANOVIC, K. Reducing Power Density through Activity Migration. In *Proc. International Symposium on Low Power Electronics and Design (ISLPED)* (Aug 2003).
- [8] HU, Z., ET AL. Microarchitectural Techniques for Power Gating of Execution Units. In *Proc. International Symposium on Low Power Electronics and Design (ISLPED)* (Aug 2004).
- [9] JACOBSON, H. M. Improved Clock-Gating through Transparent Pipelining. In *Proc. International Symposium on Low Power Electronics and Design (ISLPED)* (Aug 2004).
- [10] JACOBSON, H. M., ET AL. Synchronous Interlocked Pipelines. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems* (Apr 2002).
- [11] KALLA, R., ET AL. IBM POWER5 Chip: A Dual-Core Multithreaded Processor. *IEEE Micro* 24, 2 (Mar/Apr 2004), 40–47.
- [12] MOUDGILL, M., ET AL. Environment for PowerPC Microarchitecture Exploration. *IEEE Micro* (May/Jun 1999), 15–25.
- [13] RABAEY, J. M., AND PEDRAM, M. *Low Power Design Methodologies*. Kluwer, 1996.
- [14] TENDLER, J. M., ET AL. POWER4 System Microarchitecture. *IBM Journal of R & D* 46, 1 (Jan 2002), 5–26.