RC22913 (W0304-157) April 16, 2003
Computer Science

IBM Research Report

Integrated Analysis of Power and Performance for Pipelined Microprocessors

Pradip Bose, David M. Brooks, Philip G. Emma, Michael K. Gschwind, Vijayalakshmi Srinivasan, Philip N. Strenski, Victor Zyuban
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Integrated Analysis of Power and Performance for Pipelined Microprocessors

Abstract

Choosing the pipeline depth of a microprocessor is one of the most critical design decisions that an architect must make in the concept phase of a microprocessor design. To be successful in today's cost/performance marketplace, modern CPU designs must effectively balance both performance and power dissipation. The choice of pipeline depth and target clock frequency has a critical impact on both of these important metrics. In this paper, we describe an optimization methodology based on both analytical models and detailed simulations for power and performance as a function of pipeline depth. Our results for a set of SPEC2000 applications show that when both power and performance are considered for optimization, the optimal clock period is around 18 FO4. We also provide a detailed sensitivity analysis of the optimal pipeline depth against key assumptions of these energy models. Finally, we discuss the potential risks in design quality for overly aggressive or conservative choices of pipeline depth.

1 Introduction

Current generation high-end, server-class processors are performance-driven designs. These chips are still somewhat below the power and power density limits afforded by the package/cooling solution of choice in server markets targeted by such processors. In designing future processors, however, energy efficiency is known to have become one of the primary design constraints [8, 2, 23].

In this paper, we analyze key constraints in choosing the "optimal" pipeline depth (which directly influences the frequency target) of a microprocessor. The choice of pipeline depth is one of the fundamental issues confronting the architect/designer during the early microarchitecture definition phase of high performance, power-efficient processors. Even from a performance-only viewpoint, this issue has been important, if only to understand the limits to which pipelining can scale in the context of real workloads [16, 6, 10, 12, 22]. In certain segments of the market (typically desktops and low-end servers), there is often a market-driven tendency to equate delivered end performance with the frequency of the processor. Enlightened customers do understand the value of net system performance; nonetheless, the instinctive urge of going primarily for the highest frequency processor in a given technology generation is a known weakness even among savvy end users and therefore processor design teams. Recent studies [10, 12, 22] seem to suggest that there is still room to increase the pipeline depth, with performance optima in the range of 8-11 FO4 inverter delays[1] per stage (consisting of 6-8 FO4 logic delay and 2-3 FO4 latch delay) for current out-of-order superscalar design paradigms. However, even in these performance-centric analysis papers, the authors do point out the practical difficulties of design complexity, verification and power that must be solved in attaining these idealized limits. Our goal in this paper is to examine the practical, achievable limits when power dissipation constraints are also factored in. We believe that such analysis is needed to realistically bound the scalability limit in the next few technology generations. In particular, power dissipation must be carefully minimized to avoid design points which on paper promise ever higher performance, yet under normal operating conditions, with commodity packaging and air cooling, only deliver a fraction of the theoretical peak performance.

In this paper, we first develop an analytical model to understand the power and performance tradeoffs for superscalar pipelines. From this model, we derive the optimal pipeline depth as a function of both power and performance. Subsequently, the results are validated and further refined using a detailed cycle-accurate simulator of a current generation superscalar processor. The energy model for the core pipeline is based on circuit-extracted power analysis for structures in a current, high performance PowerPC processor. We then derive a methodology for scaling these energy models to deeper and shallower pipelines. With these performance and power models we attempt to determine the optimal pipeline depth for a particular power-performance metric.

Our results based on an analysis of the TPC-C transaction processing benchmark and a large set of SPEC2000 programs indicate that a power-performance optimum is achieved at much shallower pipeline depths than a purely performance-focused evaluation would suggest. In addition, we find there is a range of pipeline depths for which performance increases can be achieved with a modest sacrifice in power-performance efficiency. Pipelining beyond that range leads to drastic reduction in power-performance efficiency.

The contributions of this paper are (1) energy models for both dynamic and leakage power that capture

[1] Fan-out-of-four (FO4) delay is defined as the delay of one inverter driving four copies of an equally sized inverter. The amount of logic and latch overhead per pipeline stage is often measured in terms of FO4 delay which implies that deeper pipelines have smaller FO4.

the scaling of different power components as a function of pipeline depth; (2) an analytical performance model that can predict optimal pipeline depth as well as the shift in the optimal point, when combined with the energy models; (3) cycle-accurate, detailed power-performance simulation with a thorough sensitivity analysis of the optimal pipeline depth against key energy model parameters.

This paper is structured as follows: We discuss the prior, related work in Section 2. In Section 3, we describe the proposed analytical model to study pipeline depth effects in superscalar processors. Section 4 presents the simulation-based validation methodology. In Section 5 we present results using both the analytical model and a detailed simulator. In Section 6 we present a detailed sensitivity analysis to understand the effect of variations in key parameters of the derived power models on the optimal pipeline depth. In Section 7 we discuss workload variability across different design points and its relationship to workload properties, and in Section 8 we analyze the implications of our findings on design decisions. We conclude in Section 9, with pointers to future work.

2 Related Work

Previous work has studied the issue of "optimal" pipeline depth exclusively under the constraint of maximizing the performance delivered by the microprocessor.

An initial study of optimal pipeline depths was performed by Kunkel and Smith in the context of supercomputers [16]. The machine modeled in that study was based on a Cray-1S, with delays being expressed as ECL gate levels. The authors studied the achievable performance from scalar and vector codes as a function of gate levels per pipeline stage for the Livermore kernels. The study demonstrated that vector codes can achieve optimum performance by deep pipelining, while scalar (floating-point) workloads reach an optimum at shallower pipelines.

Subsequently, Dubey and Flynn [6] revisited the topic of optimal pipelining in a more general analytical framework. The authors showed the impact of various workload and design parameters. In particular, the optimal number of pipeline stages is shown to decrease with increasing overhead of partitioning logic into pipeline stages (i.e., clock skew, jitter, and latch delay). In this model, the authors considered only stalls due to branch mispredictions and did not consider data dependent stalls due to memory or register dependencies.

More recently, several authors have reexamined this topic in the context of modern superscalar processor microarchitectures. Hartstein and Puzak [10] treat this problem analytically and verify based on detailed simulation of a variety of benchmarks for a 4-issue out-of-order machine with a memory-execute pipeline. Simulation is also used to determine the values of several parameters of their mathematical model, since these cannot be formulated axiomatically. They report optimal logic delay per pipeline stage to be 7.7 FO4 for SPEC2000 and 5.5 FO4 for traditional and Java/C++ workloads. Assuming a latch insertion delay of 3 FO4, this would result in a total delay of about 10.7 FO4 and 8.5 FO4 per pipeline stage, respectively.

Hrishikesh et al. [12] treat the question of logic depth per pipeline stage empirically based on simulation of the SPEC2000 benchmarks for an Alpha 21264-like machine. Based on their assumed latch insertion delay of 1.8 FO4, they demonstrate that a performance-optimal point is at logic delay of 6.0 FO4. This would result in a total pipeline delay of about 8 FO4.

Sprangle and Carmean [22] extrapolate from the current performance of the Pentium 4 using IPC degradation factors for adding a cycle to critical processor loops, such as ALU, L1 and L2 cache latencies, and branch miss penalty for a variety of application types. The authors compute an optimal branch

misprediction pipeline depth of 52 stages, corresponding to a pipeline stage total delay of 9.9 FO4 (based on a logic depth of 6.3 FO4 and a latch insertion delay of 3.6 of which 3 FO4 are due to latch delay and 0.6 FO4 represent skew and jitter overhead).

All of the above studies (as well as ours) assume that microarchitectural structures can be pipelined without limitation. Several authors have evaluated limits on the scalability and pipelining of these structures [7, 20, 24, 5, 12].

Collectively, the cited works on optimal pipelining have made a significant contribution to the understanding of workloads and their interaction with pipeline structures by studying the theoretical limits of deep pipelining. However, prior work does not address scalability with respect to increased power dissipation that is associated with deeper pipelines. In this work, we aim to build on this foundation by extending the existing analytical models and by proposing a power modeling methodology that allows us to estimate optimal pipeline depth as a function of *both* power and performance.

3 Analytical Pipeline Model

In the concept phase definition studies, the exact organization and parameters of the target processor are not known. As such, a custom, cycle-accurate power-performance simulator for the full machine is often not available or relevant. Therefore, the use of analytical reasoning models supplemented by workload characterization and limit studies (obtained from prior generation simulators or trace analysis programs) is common in real design groups. We present such an analytical model to understand the power-performance optima and tradeoffs during the pre-simulation phase of a design project.

Figure 1 shows a high-level block diagram of the pipeline model used in our analysis. Our primary goal is to derive the optimum pipeline depth for the various execution units by estimating the various types of stalls in these pipes while using a perfect front-end for the processor. Although Figure 1 shows only one pipe for each unit (fixed point, floating point, load/store, and branch), the model can be used for a design with multiple pipes per unit as well.

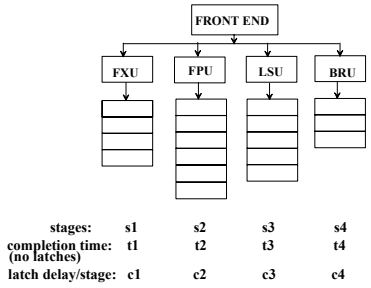

Figure 1. Pipeline Model

In Figure 1, let t_i be the latch-free logic time to complete an operation in pipe i, and s_i be the number of pipeline stages of pipe i. If c_i is the latch overhead per stage for pipe i, the total time per stage of pipe i is $T_i = ((t_i/s_i) + c_i),\ \forall i$. As derived by Dubey and Flynn in [8], and Larson and Davidson (cited in Chapter 2 of [15]), the throughput of the above machine in the absence of stalls is given by $\mathrm{G} = \sum(\frac{1}{\mathrm{T_i}})$.

We now extend this baseline model to include the effect of data-dependent stalls. Workload analysis can be used to derive first-cut estimates of the probability that an instruction n depends on another instruction j for all instructions n, and can be used to estimate the frequency of pipeline stalls. This is

illustrated in the example below, where an FXU instruction $(i+1)$ depends on the immediately preceding instruction i, and will be stalled for $(s_1 - 1)$ stages assuming a register file bypass to forward the results. Similarly, another FXU instruction $(j + 2)$ depends on instruction j, and will be stalled $(s_1 - 2)$ stages. Note that in the above workload analysis, if the source operands of an instruction i are produced by more than one instruction, the largest of the possible stalls is assigned to i.

```
inst (i)   add r1 = r2, r3
inst (i+1) and r4 = r1, r5 -- stalled for (s1 - 1) stages

inst (j)   add r1 = r2, r3
inst (j+1) or r6 = r7, r8
inst (j+2) and r4 = r1, r5 -- stalled for (s1 - 2) stages
```

$$T_{fxu} = T_1 + Stall_{fxu-fxu} * T_1 + Stall_{fxu-fpu} * T_2 + Stall_{fxu-lsu} * T_3 + Stall_{fxu-bru} * T_4$$

where $Stall_{fxu-fxu} = f_1 * (s_1 - 1) + f_2 * (s_1 - 2) + \ldots$

The above equation represents the time to complete an FXU operation in the presence of stalls; f_i is the conditional probability that an FXU instruction m depends on another FXU instruction $(m-i)$ for all FXU instructions m, provided that instruction $(m-i)$ is the producer with the largest stall for instruction m. Similar expressions can be derived for T_{fpu}, T_{lsu}, and T_{bru}, the completion times of an FPU, LSU, and BRU operation, respectively. To account for superscalar (> 1) issue widths, the workload analysis assumes a given issue width along with the number of execution pipes of various types (FXU, FPU, BRU, LSU), and issues independent instructions as an instruction bundle such that the bundle width $\leq$ issue width. Thus, the distance between dependent instructions is the number of instruction bundles issued between them. To account for the dependent instruction stalls due to L1 data cache misses we use a functional cache simulator to determine cache hits and misses. In addition, we split the load/store pipe into two, namely, load-hit and load-miss pipe; thereby, steering all data references that miss in the data cache to the load miss pipeline which results in longer stall times for the dependent instructions. Since the workload analysis is independent of the machine architecture details and uses only the superscalar issue width to determine the different stalls, it suffices to analyze each application once to derive the stalls.

The stalls modeled so far include only hazards in the execution stages of the different pipes. However, these pipes could also be waiting for instructions to arrive from the front-end of the machine. If u_i represents the fraction of time pipe i has instructions arriving from the front-end of the machine, the equation below gives the throughput of the pipeline in the presence of stalls. Note that $u_i = 0$ for unutilized pipelines, and $u_i = 1$ for fully utilized pipelines.

$$G = \frac{u_1}{T_{fxu}} + \frac{u_2}{T_{fpu}} + \frac{u_3}{T_{lsu}} + \frac{u_4}{T_{bru}}$$

Thus far, we focused on deriving the throughput of a pipeline as a function of the number of pipeline stages. In order to optimize the pipeline for both power and performance we use circuit-extracted power models described in Section 4.3.

If P_{s_i} is the total power of pipe i derived from the power models, the energy-delay product for pipe i is given by $\mathrm{ED} = \mathrm{P_{s_i}}/\mathrm{G}^2$. Hence, the optimal value of s_i which minimizes the energy-delay product can be obtained using $\mathrm{d(ED)}/\mathrm{ds_i} = 0$. Note that depending on the class of processors the desired metric for optimization could be $\mathrm{d}((\mathrm{BIPS})^\gamma/\mathrm{W})/\mathrm{ds_i} = 0$ where $\gamma \geq 0$ is an exponent whose value can be fixed in specific circuit tradeoff contexts [29]; BIPS is billions of instructions per second.

4 Performance and Power Methodology

In this section, we describe the performance simulator used in this study as well as the details of our power modeling toolkit and the methodology that we use to estimate changes in power dissipation as we vary the pipeline depth of the machine.

Fetch Latencies		Decode Latencies	
Latency Parms	STD/INF	Latency Parms	STD/INF
NFA Predictor	1/0	Multiple Decode	2/0
L2 ICache	11/0	Millicode Decode	2/0
L3 (Instruction)	85/0	Expand String	2/0
I-TLB Miss	10/0	Mispredict Cycles	3/0
L2 I-TLB Miss	50/0	Register Read	1/1
Execution Pipe Latencies		Load/Store Latencies	
Latency Parms	STD/INF	Latency Parms	STD/INF
Fix Execute	1/1	L1 D-Load	3/3
Float Execute	4/4	L2 D-Load	9/9
Branch Execute	1/1	L3 (Data)	77/0
Float Divide	12/12	Load Float	2/2
Integer Multiply	7/7	D-TLB Miss	7/0
Integer Divide	35/35	L2 D-TLB Miss	50/0
Retire Delay	2/2	StoreQ Forward	4/0

Table 1. Latencies for 19 FO4 Design Point

4.1 Performance Simulation Methodology

We utilize a generic, parameterized, out-of-order 8-way superscalar processor model called Turandot [17, 18] with 32KB I and D-caches and a 2MB L2 cache. The overall pipeline structure (as reported in [17]), is repeated here in Figure 2. The modeled baseline microarchitecture is similar to a current generation microprocessor. As described in [17], this research simulator was calibrated against a pre-RTL, detailed, latch-accurate processor model. Turandot supports a large number and parameters including configurable pipeline latencies discussed below.

Table 1 details the latency values in processor cycles for the 19 FO4 base design point of this study. We assume a 2 FO4 latch overhead and 1 FO4 clock skew and jitter overhead. The 19 FO4 latency values are then scaled with the FO4-depth (after accounting for latch and clock skew overhead). Each latency in Table 1 has two values: the first labeled STD, is for our detailed simulation model, and the second labeled INF, assumes infinite I-Cache, I-TLB, D-TLB, and a perfect front-end. The INF simulator model is used for validating the analytical model described in Section 3.

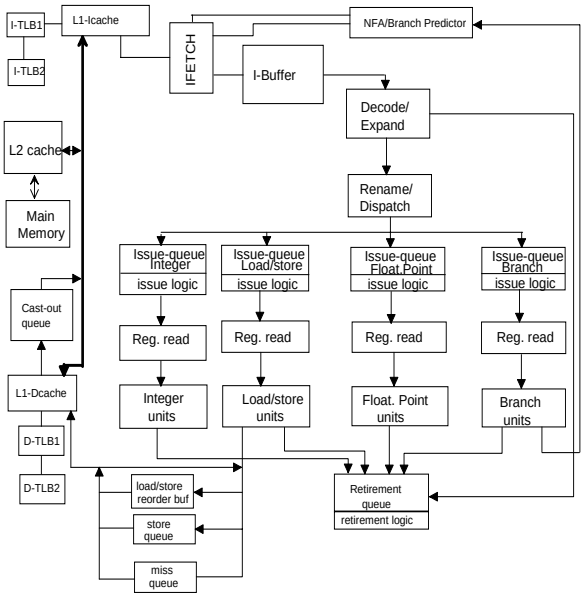

Figure 2. Modeled Processor Organization

4.2 Microprocessor Repipelining

To derive different pipeline design points from the baseline, we consider that the logic effort as measured in logic FO4 delays remains constant regardless of how many pipeline stages the function is partitioned into. However, as logic is divided into more pipeline stages, the overhead of latch FO4increases, as the number of latches inserted increases with pipeline cut points.

We repipeline the latencies shown in table 1 using the re-pipelining equation $\text{Latency}^{\text{target}} = \text{round}(\text{Latency}^{\text{base}} * \text{FO4}^{\text{base}}_{\text{logic}}/\text{FO4}^{\text{target}}_{\text{logic}})$. The baseline rounding approach used is "natural rounding", i.e., $\text{round(x)} = \lfloor \text{x} + .5 \rfloor$, which has several desirable properties, such as dispersing latency discontinuities across a range of pipeline depths, allowing for the fact that paths in the baseline design may have non-zero slack in several instances, and accounting for the fact that resizing of transistors for small negative slack may allow timings to be met.

To estimate impact of the selected repipelining method, we have also used strict upper and lower bounds for repipelining latencies by selecting rounding functions $\lceil x \rceil$ and $\lfloor x \rfloor$. As shown in figure 3, the different repipelining functions track well suggesting that the specific repipelining method does not change the trends of the data. In particular, any deviation is not skewed towards a particular pipeline range, but rather periodic in integer multiples of the original design.

4.3 Power Simulation Methodology

To estimate power dissipation, we use the PowerTimer toolset developed at IBM T.J. Watson Research Center [1, 4, 2] as the starting point for the simulator used in this work. PowerTimer is similar to power-performance simulators developed in academia [3, 25, 26], except for the methodology to build energy models.

Figure 4 above depicts the derivation of the energy models in PowerTimer. The energy models are based on circuit-level power analysis that has been performed on structures in a current, high performance PowerPC processor. The power analysis has been performed at the macro level using a circuit-

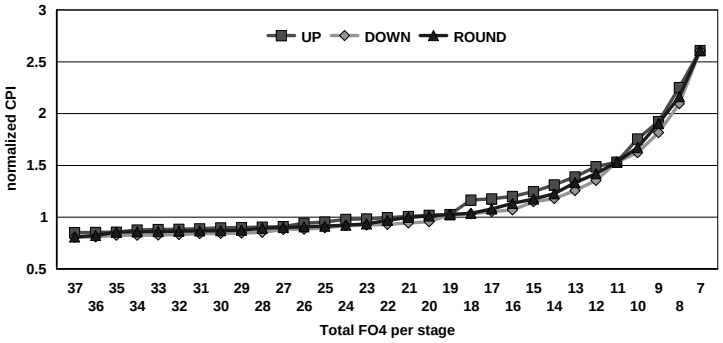

Figure 3. CPI impact of repipelining design and impact of rounding modes.

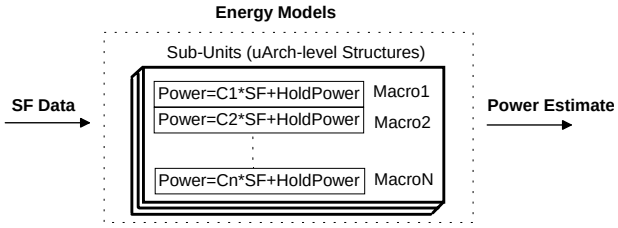

Figure 4. PowerTimer Energy Models

level power analysis tool [19]. Generally, multiple macros combine to form one micro-architectural level structure which we will call a sub-unit. For example, the fixed-point issue queue (one sub-unit) might contain separate macros for storage memory, comparison logic, and control. Power analysis has been performed on each macro to determine the macro's unconstrained (no clock gating) power as a function of the input switching factor. In addition, the *hold power*, or power when no switching is occurring (SF = 0%), is also determined. Hold power primarily consists of power dissipated in latches, local clock buffers, the global clock network, and data-independent fraction of the arrays. The *switching power*, which is primarily combinatorial logic and data-dependent array power dissipation, is the additional power that is dissipated when switching factors are applied to the macro's primary inputs. These two pieces of data allow us to form simple linear equations for each macro's power. The energy model for a sub-unit is determined by summing the linear equations for each macro within that sub-unit. We have generated these power models for all microarchitecture-level structures (sub-units) modeled in our research simulator [17, 18]. PowerTimer models over 60 microarchitectural structures which are defined by over 400 macro-level power equations.

PowerTimer uses microarchitectural activity information from the Turandot model to scale down the unconstrained hold and switching power on a per-cycle basis under a variety of clock gating assumptions. In this study, we use a realistic form of clock gating which considers the applicability of clock gating on a per-macro basis to scale down either the hold power or the combined hold and switching power depending on the microarchitectural event counts. We determine which macros can be clock gated in a fine-grained manner (per-entry or per-stage clock gating) and which can be clock gated in a coarse-grained manner (the entire unit must be idle to be clock gated). For some macros (in particular control logic) we do not apply any clock gating; this corresponds to about 20-25% of the unconstrained power dissipation. The overall savings due to clock gating relative to the unconstrained power is roughly 40-45%.

In order to quantify the power-performance efficiency of pipelines of a given FO4-depth, and to scale the power dissipation from the power models of our base FO4 design point across a range of FO4-depths, we have extended the PowerTimer methodology as discussed below.

Power dissipated by a processor consists of dynamic and leakage components, $P = P_{\text{dynamic}} + P_{\text{leakage}}$. The dynamic power data measured by PowerTimer (for a particular design point) can be expressed as $P^{\text{base}}_{\text{dynamic}} = CV^2 f * (\alpha + \beta) * CGF$, where α is the average "true" switching factor in circuits; i.e., α represents transitions required for the functionality of the circuit and is measured as the switching factor by an RTL-level simulator run under the zero-delay mode. In contrast, β is the average glitching factor that accounts for spurious transitions in circuits due to race conditions. Thus, (α + β) is the average number of transitions actually seen inside circuits. Both α and β are averaged over the whole processor over non-gated cycles with appropriate energy weights (the higher the capacitance at a particular node the higher the corresponding energy weight). CGF is the clock gating factor which is defined as the fraction of cycles where the microarchitectural structures are *not* clock gated. The CGF is measured from our PowerTimer runs at each FO4 design point as described above. The remaining terms C, V, and f are effective switching capacitance, chip supply voltage, and clock frequency, respectively.

Next we analyze how each of these factors scales with FO4 pipeline depth. To facilitate the explanation we define the following variables: $\text{FO4}_{\text{logic}}$, $\text{FO4}_{\text{latch}}$ and $\text{FO4}_{\text{pipeline}}$, to designate the depth of the critical path through logic in one pipeline stage, the latch insertion delay including clock skew and jitter, and the sum of the two quantities, respectively, $\text{FO4}_{\text{pipeline}} = \text{FO4}_{\text{logic}} + \text{FO4}_{\text{latch}}$. We use FO4 and $\text{FO4}_{\text{pipeline}}$ interchangeably. In the remainder of the paper, the qualifier 'base' in all quantities designates the value of the quantities measured for the base 19 FO4 design.

Frequency: FreqScale is the scaling factor that is used to account for the changes in the clock frequency with the pipeline depth. This factor applies to both hold power and switching power:

$$\text{FreqScale} = \text{FO4}^{\text{base}}_{\text{pipeline}} / \text{FO4}_{\text{pipeline}}$$

Latch: With fixed logic hardware for given logic functions, the primary change in the chip effective switching capacitance C with pipeline depth is due to changes in the latch count with the depth of the pipeline. LatchScale is a factor that appropriately adjusts the hold power dissipation, but does not affect the switching power dissipation.

$$\text{LatchScale} = \text{LatchRatio} * \left(\frac{\text{FO4}^{\text{base}}_{\text{logic}}}{\text{FO4}_{\text{logic}}} \right)^{\text{LGF}}$$

where $LatchRatio$ defines the ratio of hold power to the total power and $LatchGrowthFactor(LGF)$ captures the growth of latch count due to the logic shape functions. The amount of additional power that is spent in latches and clock in a more deeply pipelined design depends on the *logic shape functions* of the structures that are being pipelined. Logic shape functions describe the number of latches that would need to be inserted at any cut point in a piece of combinatorial logic if it were to be pipelined. Values of $LGF > 1$ recognize the fact that for certain hardware structures the logic shape functions are not flat and hence the number of latches in the more deeply pipelined design increases super-linearly with pipeline depth. In our baseline model, we assume a LGF of 1.1 and study the sensitivity of the optimal pipeline depth to this parameter in Section 6.1.

Clock Gate Factor: In general, CGF decreases with deeper pipelines because the amount of clock gating potential increases with deeper pipes. This increased clock gating potential is primarily due to the increased number of cycles where pipeline stages are in stall conditions. This in turn leads to an increase in the clock gating potential on a per cycle basis. CGF is workload dependent and is measured directly from simulator runs.

Glitch: The final two factors that must be considered for dynamic power dissipation when migrating to deeper pipelines are α and β, the chip-wide activity and glitching factors.

The "true" switching factor α does not depend on the pipeline depth, since it is determined by the functionality of the circuits. The glitching factor at any net, on the other hand, is determined by the difference in delay of paths from the output of a latch that feeds the circuit to the gate that drives that net. Once a glitch is generated at some net, there is a high probability that it will propagate down the circuit until the input of the next latch down the pipeline. Furthermore, the farther the distance from the latch output to the inputs of a gate the higher the probability of the existence of non-equal paths from the output of the latch to the inputs of this gate. Therefore the average number of spurious transitions grows with $\mathrm{FO4}_{\mathrm{logic}}$ – the higher the FO4 the higher the average glitching factor. Experimental data, collected by running a dynamic circuit-level simulator (PowerMill) on post-layout extracted netlists of sample functional units show that the average glitching factor β can be modeled as being linearly dependent on the logic depth:

$$\beta = \beta_{\mathrm{base}} \frac{\mathrm{FO4}_{\mathrm{logic}}}{\mathrm{FO4}^{\mathrm{base}}_{\mathrm{logic}}}$$

To account for the effect of the dependence of β on pipeline depth, we introduce the following factor which applies only to the switching power:

$$\begin{aligned} \mathrm{GlitchScale} &= (1 - LatchRatio)\left(\frac{\alpha + \beta}{\alpha + \beta_{\mathrm{base}}}\right) \\ &= \frac{1 - LatchRatio}{1 + \beta_{\mathrm{base}}/\alpha}\left(1 + \frac{\beta_{\mathrm{base}}}{\alpha}\frac{\mathrm{FO4}_{\mathrm{logic}}}{\mathrm{FO4}^{\mathrm{base}}_{\mathrm{logic}}}\right) \end{aligned}$$

In this formula, β_{base} is the actual glitching factor averaged over the baseline microprocessor for the base FO4 design point. Notice that β_{base} appears in the formula only in the ratio $\beta_{\mathrm{base}}/\alpha$. This is consistent with our experimental results showing that the glitching factor β is roughly proportional to the "true" switching factor α, for the range $0 < \alpha < 0.3$ (for higher values of α the growth of β typically saturates). For the set of six sample units that we simulated, with the logic depth ranging from 6 FO4 to 20 FO4, the ratio β/α was found to be roughly proportional to the logic depth of the simulated units, $\mathrm{FO4}_{\mathrm{logic}}$, with the coefficient equal to $0.3/\mathrm{FO4}^{\mathrm{base}}_{\mathrm{logic}}$. Based on these simulation results we set $\beta_{\mathrm{base}}/\alpha = 0.3$ for the whole microprocessor in the remainder of this section, and study the sensitivity of the results to variations in the glitching factor in Section 6.4.

Leakage Currents: As the technology feature size scales down and the power supply and transistor threshold voltages scale accordingly, the leakage power component becomes more and more significant. Since the magnitude of the leakage power component is affected by the pipeline depth, it is essential to include the effect of the leakage power in the analysis of the optimal pipeline depth. Assuming that

the leakage power is proportional to the total channel width of all transistors in the microprocessor, we model the dependence of the leakage power on the depth of the pipeline as follows:

$$P_{\text{leakage}}^{\text{FO4}} = P_{\text{leakage}}^{\text{base}} \left(1 + \frac{w_{\text{latch}}}{w_{\text{total}}} \left[\left(\frac{\text{FO4}_{\text{logic}}^{\text{base}}}{\text{FO4}_{\text{logic}}} \right)^{\text{LGF}} - 1 \right] \right)$$

where LGF is the $LatchGrowthFactor$ defined earlier, $w_{\text{latch}}/w_{\text{total}}$ is the ratio of the total channel width of transistors in all pipeline latches (including local clock distribution circuitry) to the total transistor channel width in the base (19 FO4) microprocessor (excluding all low-leakage transistors that might be used in caches or other on-chip memories). If the technology supports multiple thresholds, or any of the recently introduced leakage reduction techniques are used on a unit-by-unit basis, such as MTCMOS, back biasing, power-down, or transistor stacking, then the above formula for the leakage power component needs to be modified accordingly and we leave the detailed study of these effects for future work. For the remainder of the study we set $w_{\text{latch}}/w_{\text{total}}$ to 0.2 for the base 19 FO4 pipeline. Also, rather than giving the absolute value for the leakage current, $P_{\text{leakage}}^{\text{base}}$ in the base microprocessor, we will count it as a fraction of the dynamic power of the base design, $P_{\text{leakage}}^{\text{base}} = LeakageFactor^{\text{base}} P_{\text{dynamic}}^{\text{base}}$. We set the $LeakageFactor^{\text{base}}$ to the value of 0.1, typically quoted for state of the art microprocessors, and analyze the sensitivity of the results to $LeakageFactor$ in Section 6.5.

Total Power: The following equation expresses the relationship between the dynamic power for the base FO4 design, $\text{P}_{\text{dynamic}}^{\text{base}}$, leakage power and the scaled power for designs with different depths of the pipeline, $\text{P}_{\text{total}}^{\text{FO4}}$, considering all factors above.

$$P_{\text{total}}^{\text{FO4}} = CGF * FS * (LS + GS) * P_{\text{dynamic}}^{\text{base}} + P_{\text{leakage}}^{\text{FO4}}$$

where FS is $FreqScale$, LS is $LatchScale$, and GS is $GlitchScale$.

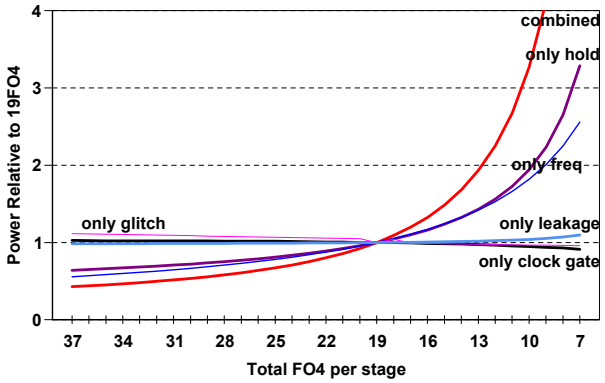

Figure 5. Power Growth Breakdown

Figure 5 shows contributions of different factors in the above formula, depending on the pipeline depth of the design $\text{FO4}_{\text{pipeline}}$. The 19 FO4 design was chosen as a base pipeline.

The line labeled "combined" shows the cumulative increase or decrease in power dissipation. The line labeled "only clock gate" quantifies the amount of additional clock gating power savings for deeper pipelines. The relative effect of scaling in clock gating is fairly minor with slightly more than 10%

additional power reduction when going from the 19 FO4 to 7 FO4 design points. There are several reasons why the effect of clock gating is not larger. First, the fraction of power dissipation that is not eligible to be clock gated becomes larger with more clock gating leading to diminishing returns. Second, some of the structures are clock gated in a coarse-grained fashion and while the average utilization of the structure may decrease it must become idle in all stages before any additional savings can be realized. Finally, we observe that clock gating is more difficult in deeper pipelined machines, because it is harder to deliver cycle-accurate gating signals at lower FO4.

The two lines labeled "only freq" and "only hold" show the power factors due to only frequency and hold power scaling, respectively.[2] Overall, dynamic power increases more than quadratically with increased pipeline depth.

Figure 5 shows that the leakage component grows much less rapidly than the dynamic component with the increasing pipeline depth. There are two primary reasons for this. First, the leakage power does not scale with frequency. Second, the leakage power growth is proportional to the fraction of channel width of transistors in pipeline latches, whereas the latch dynamic hold power growth is proportional to the fraction of the dynamic power dissipated in pipeline latches. Obviously, the former quantity is much smaller than the latter.

4.4 Workloads and Metrics Used in the Study

In this paper, we report experimental results based on PowerPC traces of a set of 21 SPEC2000 benchmarks, namely, *ammp*, *applu*, *apsi*, *art*, *bzip2*, *crafty*, *equake*, *facerec*, *gap*, *gcc*, *gzip*, *lucas*, *mcf*, *mesa*, *mgrid*, *perl*, *sixtrack*, *swim*, *twolf*, *vpr*, and *wupwise*. We have also used a 172M instruction trace of the TPC-C transaction processing benchmark. The SPEC2000 traces were generated using the tracing facility called *Aria* within the MET toolkit [18]. The particular SPEC2000 trace repository used in this study was created by using the full reference input set. However, sampling was used to reduce the total trace length to 100 million instructions per benchmark program. A systematic validation study to compare the sampled traces against the full traces was done in finalizing the choice of exact sampling parameters [13].

We use BIPS^3/W ($energy * delay^2$) as a basic energy-efficiency metric for comparing different FO4 designs in the power-performance space. The choice of this metric is based on the observation that dynamic power is roughly proportional to the square of supply voltage (V) multiplied by clock frequency and clock frequency is roughly proportional to V. Hence, power is roughly proportional to V^3 assuming a fixed logic/circuit design. Thus, delay cubed multiplied by power provides a voltage-invariant power-performance characterization metric which we feel is most appropriate for server-class microprocessors (see discussion in [2, 8]). In fact, it was formally shown in [29] that optimizing performance subject to a constant power constraint leads to the BIPS^3/W metric in processors operating at a supply voltage near the maximum allowed value in state of the art CMOS technologies. As a comparative measure, we also consider BIPS/W (energy) and BIPS^2/W (energy-delay [9]) in the baseline power-performance study.

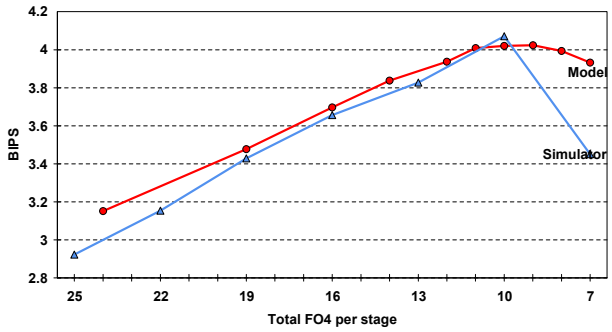

Figure 6. BIPS: Analytical Model vs. Simulator

5 Analytical Model and Simulation Results

5.1 Analytical Model Validation

We now present the performance results using the simple analytical model (see Section 3) and compare it with the results using our detailed cycle-accurate simulator. The results in this section are presented for the average of the SPEC2000 benchmarks described in Section 4.3. For fair comparison, we have modified the simulator to include a perfect front-end of the machine. The model and the simulator use latencies shown in the column labeled "INF" in Table 1.

Figure 6 shows BIPS as a function of the FO4 delay per stage of the pipeline. The BIPS for the analytical model was computed after determining the stalls for the different pipelines using independent workload analysis as explained in Section 3.

From Figure 6 we observe that the performance optimal pipeline depth is roughly 10 FO4 delay per pipeline stage for both the model and the simulator. The BIPS estimated using the model correlates well with the BIPS estimated using the simulator except for very large and very small FO4-depth machines. Since the stalls are determined in the model independent of the pipeline and only once for a workload, it is possible that for shallow pipelines with large FO4 delay per stage, the model underestimates the total stall cycles, and hence the BIPS computed by the model is higher than the BIPS obtained from the simulator. The analytical model currently only uses data dependent stalls to derive the optimal pipeline depth. Hence, for the purpose of the validation experiment, the model and the simulator assume a perfect front-end. However, to estimate the effect of resource stalls we modeled a large but finite register renamer, instruction buffer, and miss queue in the simulator keeping the rest of the front-end resources infinite and perfect. For deeper pipelines ($\leq$ 10 FO4), we observe from the simulator that the stalls due to the resources become appreciable and the analytical model begins to overestimate the BIPS relative to the simulator.

Figure 7 shows that the optimal FO4-depth that maximizes the $BIPS^3/W$ is around 19-24 FO4 for the simulator. We observe that the analytical model tracks the simulator results more closely while optimizing performance alone as seen in Figure 6. Since the analytical model overestimates the performance for shallow pipelines, the cubing of BIPS in Figure 7 compounds these errors.

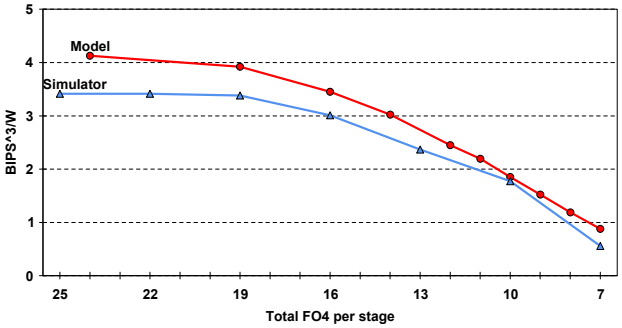

Figure 7. BIPS^3/W: Model vs. Simulator

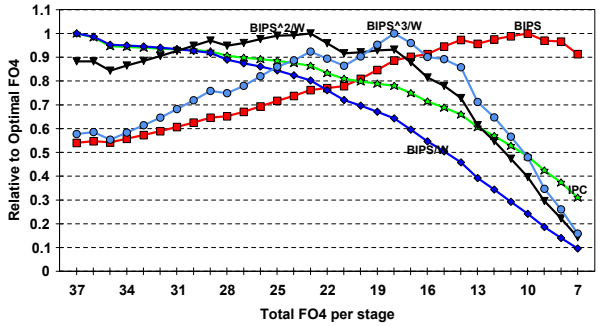

Figure 8. Simulation Results for SPEC2000

5.2 Detailed Power-Performance Simulation

In the remainder of this work we consider the power and performance results using the detailed performance simulator with parameters corresponding to the STD column in Table 1. Figure 8 shows the results for five metrics: BIPS, IPC, BIPS/W, BIPS^2/W, and BIPS^3/W.

Figure 8 shows that the optimal FO4-depth for performance (defined by BIPS) is 10 FO4, although pipelines of 8 FO4 to 15 FO4 are within 5% of the optimal. Because of the super-linear increase in power dissipation and sub-linear increases in overall performance, the BIPS/W always decreases with deeper pipelines. BIPS^3/W shows an optimum point at 18 FO4. BIPS^3/W decreases sharply after the optimum and at the performance-optimal pipeline depth of 10 FO4, the BIPS^3/W metric is reduced by 50% over the 18 FO4 depth. For metrics that have less emphasis on performance, such as BIPS^2/W and BIPS/W, the optimal point shifts towards shallower pipelines, as expected. For the BIPS^2/W, the optimal pipeline depth is achieved at 23 FO4.

Figure 9 presents similar results for the TPC-C trace. The optimal BIPS for TPC-C is very flat from 10-14 FO4 (within 1-2% for all 5 design points which is much flatter than SPEC2000). Using BIPS^3/W, the optimal pipeline depth shifts over to 25-28 FO4. The main reason that the optimal point is shallower for TPC-C is that BIPS decreases less dramatically with decrease in pipeline length (relative to SPEC2000). This is slightly counterbalanced because power increases at a slower rate for TPC-C with deeper pipes, because the additional amount of clock gating is more pronounced due to large increases in the number of stall cycles relative to the SPEC2000 suite.

[2]Although these two factors increase linearly with clock frequency, we are plotting against FO4-depth which is $1/\textit{clock frequency}$.

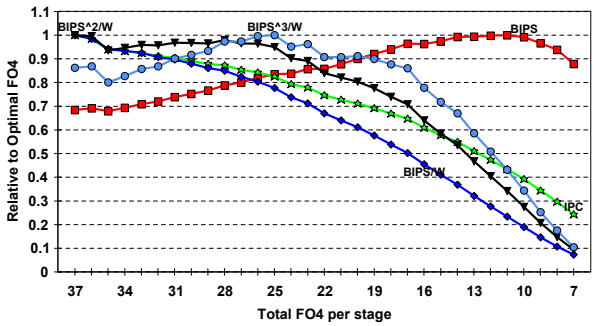

Figure 9. Simulation Results for TPC-C

6 Sensitivity Analysis

The derived equations that model the dependence of the power dissipation on the pipeline depth depend on several parameters. Some of these parameters, although accurately measured for the baseline microprocessor, are likely to change from one design to another, whereas others are difficult to measure accurately. In this section, we perform sensitivity analysis of the optimal pipeline depth to key parameters of the derived power models such as $LatchGrowthFactor$, $LatchRatio$, latch insertion delay ($\mathrm{FO4_{latch}}$), $GlitchRatio$ and $LeakageFactor$.

6.1 Latch Growth Factor

$LatchGrowthFactor$ (LGF) is determined by the intrinsic logic shape functions of the structures that are being pipelined. We have analyzed many of the major microarchitectural structures to identify ones that are likely to have $LatchGrowthFactor$ greater than 1. One structure that we will highlight is the Booth recoder and Wallace tree which is common in high-performance floating point multipliers [14, 21], as shown in Figure 10. Figure 10 shows the exponential reduction in the number of result bits as the data passes through the 3-2 and 4-2 compressors. We have estimated the amount of logic that can be inserted between latch cut points for 7, 10, 13, 16, and 19 FO4 designs by assuming 3-4 FO4 delay for 3-2 and 4-2 compressor blocks. As the 7 and 10 FO4 design points require latch insertions just after the Booth multiplexor (where there are 27 partial products), there would be a large increase in the number of latches required for these designs. We note that the 7 FO4 design also requires a latch insertion after the booth recode stage.

Figure 11 gives estimates for the cumulative number of latches in the FPU design as a function of the FO4 depth of the FPU. For example, the first stage of the 10 FO4 FPU requires 3x as many latches as the first stage of the 19 FO4 FPU because the first latch cut point of the 19 FO4 FPU is beyond the initial 9:2 compressor tree. Overall, the 10 FO4 FPU requires nearly 3x more latches than the 19 FO4 FPU.

There are many other areas of parallel logic that are likely to see super-linear increases in the number of latches such as structures with decoders, priority encoders, carry look ahead logic, etc. Beyond the pipelining of logic structures, deeper pipelines may require more pre-decode information to meet aggressive cycle times, which would require more bits to be latched in intermediate stages. On the other hand, the number of latches that comprise storage bits in various on-chip memory arrays (such as register files and queues) does not grow at all with the pipeline depth, meaning that the $\mathrm{LGF} = 0$ for those latches. Thus, designs with overall $\mathrm{LGF} < 1$ are also possible.

Figure 12 quantifies the dependence of the optimal pipeline depth on LGF using the BIPS^3/W metric.

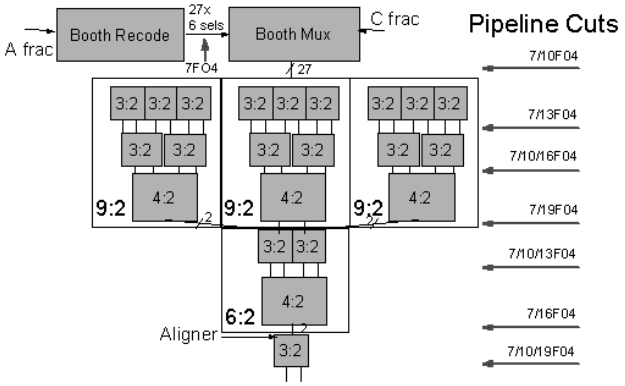

Figure 10. Wallace Tree Diagram and Latch Cut points for 7/10/13/16/19 FO4

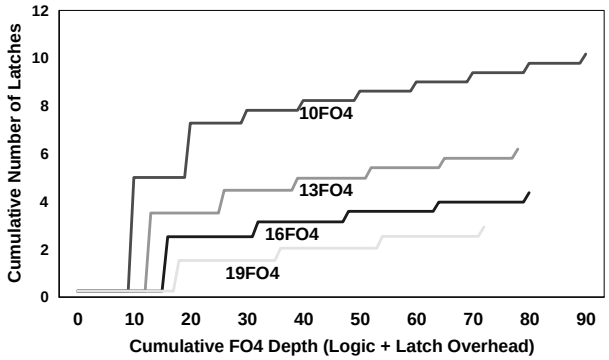

Figure 11. Cumulative latch count for FPU

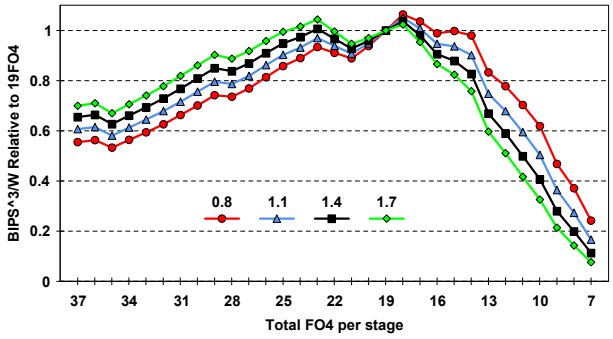

Figure 12. BIPS^3/W varying $LatchGrowthFactor$

It shows that the optimal pipeline FO4 tends to increase as LGF increases above the value of 1.1, assumed in the baseline model. As a point of reference, our estimate for the latch growth factor for the 10 FO4 vs. 19 FO4 Booth recoder and Wallace tree is $\mathrm{LGF} = 1.9$, while for the entire FPU LGF is slightly less than 1.7.

6.2 Latch Power Ratio

Latch, clock, and array power are the primary components of power dissipation in current generation CPUs. This is especially true in high-performance, superscalar processors with speculative execution which require the CPU to maintain an enormous amount of architectural and non-architectural state. One possible reason why the $LatchRatio$ could be smaller than the base value of 0.7 chosen in Section 4, is if more energy-efficient SRAM arrays are used in high-power memory structures instead of latches to reduce the data independent array power (which we include as part of hold power).

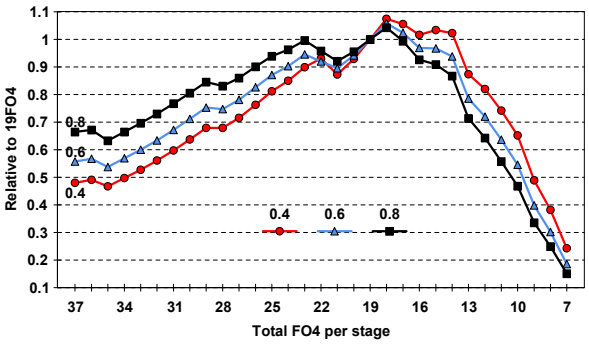

Figure 13. BIPS^3/W varying $LatchRatio$

Figure 13 shows the optimal FO4 design point while varying the $LatchRatio$ of the machine from 80% to 40%. We see that while the optimal FO4 point remains 18 FO4, it is less prohibitive to move to deeper pipelines with smaller latch-to-logic ratios. For example, with a $LatchRatio$ of 0.4, the 13 FO4 design point is only 19% worse than the optimal one while it is 27% worse than optimal with a $LatchRatio$ of 0.6.

6.3 Latch Insertion Delay

Latch FO4	2	3	4	5
Relative Latch Energy	1.0	0.53	0.36	0.29
Relative Clocking Energy	1.0	0.62	0.49	0.43

Table 2. Latch Insertion Delay (excluding skew and jitter) vs. Relative Latch Energy

With the large amount of power spent in latches and clocking, designers may consider the tradeoff between latch delay and power dissipation as a means to design more energy-efficient CPUs. Researchers have investigated latch power vs. delay tradeoff curves both within a given latch family and across latch families [27, 11]. Table 2, derived from [27], shows latch FO4-delay vs. latch energy across several latch styles. The first row of Table 2 shows the latch insertion delay, excluding clock skew and jitter overhead which is assumed to be constant for all latches. The second row shows the relative latch energy, excluding energy dissipated in the clock distribution. The third row shows the relative energy of the clock

system, including both energy of latches (70% of the total) and clock distribution system (30%). It is assumed that the clock distribution energy cannot be completely scaled with reduced latch load. There is significant overhead simply from driving the wires necessary to distribute the clock over a large area.

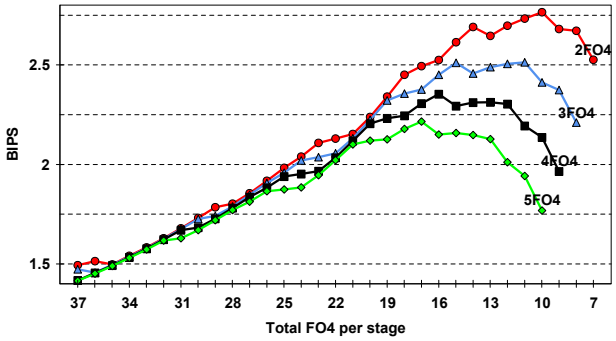

Figure 14. BIPS varying latch power-delay

Replacing the baseline fast 2 FO4 latches with slower, lower power latches increases the latch insertion delay overhead, which impacts both the performance-optimal and power-performance-optimal pipeline depth. Figure 14 shows the processor performance versus pipeline depth for four latches from Table 2. We see that the performance maxima shift towards shallower pipelines as the latch insertion delay increases. For example, with a 3 FO4 latch the performance-optimal FO4-depth is 11 FO4 and with a 4 FO4 latch it becomes 16 FO4.

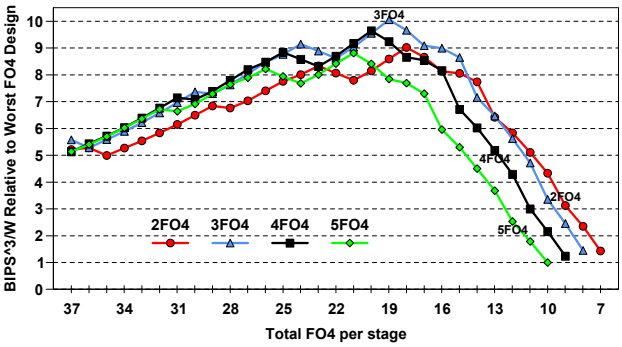

Figure 15. BIPS^3/W varying latch power-delay

Figure 15 shows the impact of the latch insertion delay on the BIPS^3/W rating of the processor for the same range of pipeline depths. In this figure, all of the data points are shown relative to the 10 FO4 design with the base 5 FO4 latch. Unlike curves on all previous sensitivity graphs, curves in Figure 15 do not intersect at the base design point, because different curves represent different designs, with different power and performance levels.

Figure 15 shows that using the fastest 2 FO4 latch results in the best BIPS^3/W rating for processors with stages less than 14 FO4. For processors with pipelines ranging from 15 FO4 to 24 FO4 a lower power 3 FO4 latch is the most energy efficient, whereas for shallower pipelines (25 FO4 or more) the highest BIPS^3/W is achieved with even slower 4 FO4 latches.

The use of the 3 FO4 latch, combined with the choice of the pipeline depth in the range from 15 FO4 to 24 FO4 improves the BIPS^3/W rating of the processor by more than 10%, compared to the base case

of 2 FO4 latches. The graph also shows that the optimal BIPS^3/W design point shifts towards shallower pipelines as high-performance latches are replaced with lower power ones. For example, the 18 FO4 design point is optimal for a processor using 2 FO4 latches, whereas the 19 FO4, 20 FO4, and 21 FO4 design points are optimal for processors using 3 FO4 latches, 4 FO4, and 5 FO4 latches, respectively.

6.4 Glitch Factor

In this subsection we quantify the sensitivity of the optimal pipeline depth to the glitching factor. There are no practical means for accurately measuring the actual value of $\beta_{\text{base}}/\alpha$, averaged over the whole microprocessor. Instead we measured the glitching factor for a selected set of functional units and used the averaged value of $\beta_{\text{base}}/\alpha$ = 0.3 throughout the analysis. In this section we analyze the sensitivity of the optimal pipeline depth to the value of $\beta_{\text{base}}/\alpha$.

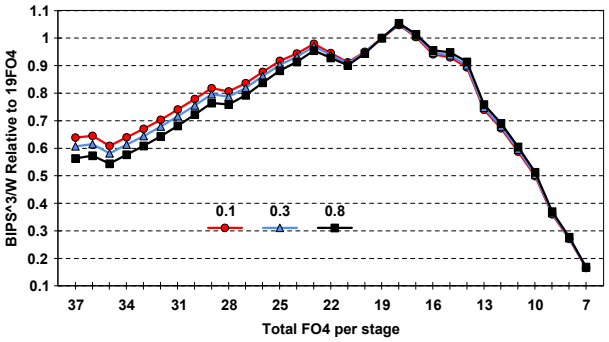

Figure 16. BIPS^3/W varying $\beta_{\text{base}}/\alpha$

Figure 16 shows the dependence of the BIPS^3/W rating of the processor on the pipeline depth for three values of $\beta_{\text{base}}/\alpha$. From this figure we see that higher glitching factors favor deeper pipelines. However, in the base design the decrease in power dissipation related to reduced glitching in deeper pipelines did not have a substantial impact on the optimal pipeline depth, primarily because of the relatively small fraction of power dissipated in combinatorial switching. For designs which have smaller $LatchRatio$ values, this effect could be more significant.

6.5 Leakage Factor

As explained earlier, the leakage power component grows more slowly with the pipeline depth than the dynamic component. Therefore, the optimum pipeline depth depends on the $LeakageFactor$. Throughout the analysis we assumed that for the base 19 FO4 microprocessor the $LeakageFactor$ ($P^{\text{base}}_{\text{leakage}}/P^{\text{base}}_{\text{dynamic}}$) to be 0.1. However, as the technology feature size scales down and the power supply and transistor threshold voltages scale accordingly, the leakage power component becomes more and more significant. To study the effect of the growing fraction of the leakage power component we measured the sensitivity of the optimal pipeline depth to the value of the $LeakageFactor$.

Figure 17 shows the BIPS^3/W rating of the processor versus pipeline depth for three values of the $LeakageFactor$: a value of 0 that represents older CMOS technologies, a value of 0.1, assumed in the current model, and values of 0.5 and 1.0, projected for future generation CMOS technologies (arguably, extreme values). The results in Figure 17 show that unless leakage reduction techniques become the standard practice in the design of high-end microprocessors, the high values of the $LeakageFactor$ projected for future generations of CMOS technologies may tend to shift the optimum pipeline depth

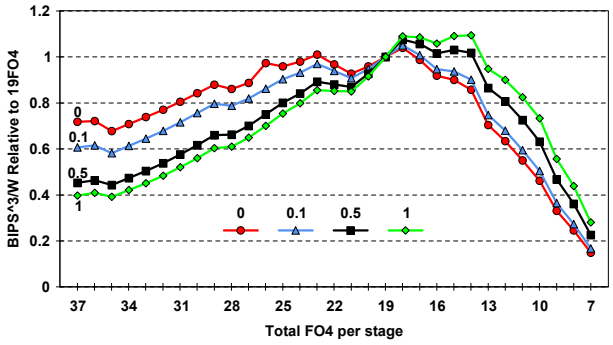

Figure 17. BIPS^3/W varying $LeakageFactor$

towards slightly deeper pipelines. For current generation technologies the result for the optimal pipeline depth is sufficiently stable with respect to reasonable variations in the $LeakageFactor$.

Summary of the Sensitivity Analysis: In this section we considered the sensitivity of optimal pipeline length to five key parameters in the power models using the BIPS^3/W metric. We did not observe a strong dependence of the results on the assumptions and choices of any of these parameters, which demonstrates the stability of the model, and its applicability to a wide range of designs. To summarize the results, higher values of the $LatchGrowthFactor$ favor shallower pipelines, lower values of the $LatchRatio$ favor deeper pipelines, the use of lower-power latches favors shallower pipelines, higher values of the $GlitchFactor$ favors deeper pipelines, and, finally, higher leakage currents favor deeper pipelines.

7 Workload variability

We have studied the distribution of optimal pipeline depth as a function of the workload characteristics. In figure 18 we show how the workload optima are distributed.

We note that the optima are clustered at a few preferred configurations. This clustering is a response to the discretization introduced by dividing the data path into several pipeline stages. The (microarchitecturally) optimal points are then those which utilize a given FO4 delay best by having minimal slack, i.e., just before a new pipeline stage must be introduced to achieve the next decrement in FO4 delay per pipeline stage.

At this point, the frequency gain for reducing the delay per pipeline stage is FO4/(FO4-1), ranging from (8/7) = 14% in the very high frequency regime to (19/18) = 6% in a moderately high frequency design. On the other hand, CPI degrades as a step function response to the increment in latency of a particular unit, making that design point less attractive from a performance perspective. Finally, the deeper pipeline also increases the number of latches in the pipeline and hence also increases power dissipation.

Examples of this behavior can be seen in figure 18 at the transition from 18 to 17 FO4 which increases the latency of the FP unit by 1 cycle, and hence degrades CPI for floating-point intensive benchmarks. This causes significant clustering of optima for FP-intensive benchmarks at 18 FO4. Similarly, other key microarchitectural latencies change at the transition from 17 to 16 FO4 and 14 to 13 FO4, leading to optima of benchmarks with high sensitivity to those particular parameters at 17 and 14 FO4, respectively.

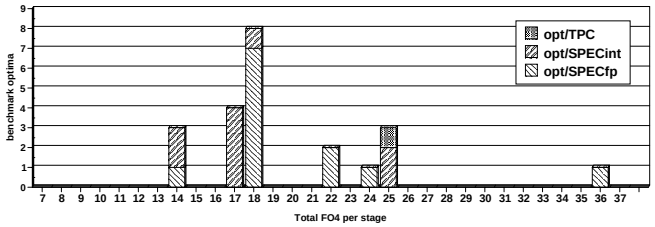

Figure 18. Distribution of power/performance optimal pipeline stage depth

We note that the *lucas* benchmark has a power/performance optimum at a pipeline logic complexity of 36 FO4 as its performance curve is essentially flat across any pipeline depth. However, power increases almost as badly as with any other workload due to the increase in latches and clock frequency. While there is some saving due to increased ability to gate the clock at a finer level, this improvement is measured on the order of 20% over the polynomial growth in power shown in figure 5. The *lucas* benchmark is an extreme case in that it is dominated by L2 misses. As such, more aggressive pipelining has no impact on performance as L2 behavior scales with memory system performance, not core architecture.

8 Design Implications

Figure 19 gives a graphical interpretation of optimizing the pipeline depth in the power-performance space. Power, plotted on the y-axis, is the average power dissipated on the set of benchmarks, measured in relative units. Delay, plotted on the x-axis, is the average execution time for the same set of benchmarks, measured in relative units. Microarchitectural configurations with different pipeline depths are represented by points on the bold curve (denoted with □). Thus, the bold □ curve shows how the processor design point moves in the power-performance space as the depth of the pipeline changes, assuming unchanged power supply and circuit tuning point (hardware intensity[3]). The deeper the pipeline (less FO4) the steeper the part of the curve where the corresponding design point sits.

The curves denoted by the circle symbol (○) show the circuit energy-delay tradeoff curves which represent implementations of the microarchitecture with a fixed pipeline depth for varying power supply and circuit hardware intensity (tuning point), such that the condition for the optimal hardware intensity defined in our previous work is observed at each point. The base hardware intensity is set to the value of 2 for each circuit energy-delay tradeoff curve, which corresponds to the power supply voltage of 1.3V in a 0.13μm CMOS technology. Thus, for any fixed pipeline depth we can move the power-performance point of the processor along one of the hardware tuning curves (○) in the vicinity of the base point. Notice that on any of these curves (○) every 1% in performance (frequency) achieved through changing the power supply and retuning the circuits costs 3% in power.

[3]We use *hardware intensity* η as a quantitative measure of how aggressively the circuits in a processor are tuned to meet a target clock frequency [28]. Hardware intensity shows the energy cost (in $\%$) required to improve the delay D of a hardware macro by 1% through restructuring the logic and retuning the circuits, at a fixed power supply v assuming that the curvature of the energy-delay curve is sufficiently high, so that $\frac{D^2}{E}\frac{\partial^2 E}{\partial D^2} > \eta(\eta+1)$ is satisfied at every point:

$$\eta = -\left.\frac{\%E}{\%D}\right|_{\text{through retuning}}$$

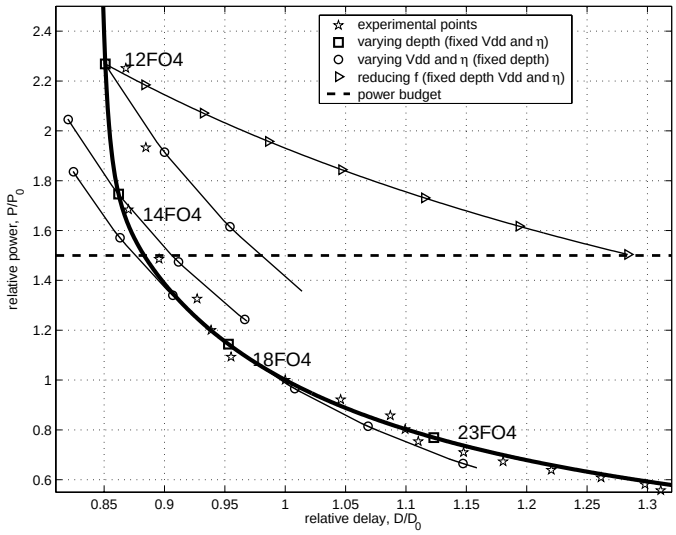

Figure 19. Power efficiency impact of microarchitectural and circuit tuning to achieve high performance implementations of microprocessors.

The 18FO4 point on the bold □ curve that minimizes the performance-cube per watt metric is the point where a 1% gain in performance through increasing the pipeline depth also costs 3% in power. Then, the 18FO4 point is the only point on the bold □ curve at which the tangent to the architectural energy-delay tradeoff curve has the same value as the tangent to the circuit energy-delay tradeoff curve, or the point at which the architectural and circuit energy-delay curves touch.

At this pipeline depth, the processor delivers the required performance while dissipating the minimum power among implementations with any depth of the pipeline. To graphically demonstrate this, suppose that the original design point with 15FO4 pipeline delivers the required performance of $0.9D_0$. This pipeline depth minimizes the BIPS to the power of 10 per watt metric, which means that at this point every percent in performance achieved through increasing the pipeline depth costs 10% in power. Thus the initial design point of 15FO4 sits on a steeper part of the curve than the 18FO4 design point. Then by reducing the pipeline depth we can move the processor power-performance point down the bold □ curve, saving 10% in power for every 1% loss in performance. Then by raising the power supply and tuning up the circuits accordingly we can move the design point up the corresponding hardware tuning curve (○) to recover the performance, spending approximately 3% in power for every 1% gain in performance. Obviously, the resulting design with 18FO4 is a better design point than the initial 15FO4, because it delivers the same performance at a lower power.

Similarly, if the original pipeline has a depth of 30FO4 which minimizes the MIPS-squared per watt metric, then by increasing the pipeline depth we can move the design point up the bold □ curve, spending approximately 2% in power per every percent in gained performance. If the higher performance resulting from the increase in the pipeline depth is not needed, the design point can be moved down the corresponding hardware tuning curve (○) by reducing power supply and tuning down the circuits, thereby saving 3% in power per every percent in performance. Since the circuit energy-delay tradeoff curve at this point is steeper than the architectural energy-delay tradeoff curve, the resulting design will deliver the same performance as the initial one, but at a lower power.

Now what if the design with the optimal 18FO4 pipeline does not deliver the advertised frequency, and a business decision is made to go with a deeper pipeline (say 13FO4). Then if the final product still meets the power-dissipating capability of the package (shown by the power budget line in figure 19), that company will just end up with a energy-inefficient design. However, if the final product exceeds the power-dissipating capabilities of the package when running at maximum speed (which is a more likely scenario), then stepping down the bold □ curve after the processor is in the development (or shipment) phase is not an option. One way to reduce the processor power is to run it at a lower power supply and frequency . This leads to a shift of the power-performance point down the hardware tuning curve (○) that originates at the initial 13FO4 power-performance point. As we can see, eventually such a processor will deliver much lower performance than could be achieved if the depth of the pipeline had been chosen according to the BIPS-cube per watt optimum. Notice that if such a processor is set to run at a lower than maximum frequency without reducing the power supply, then the power-performance design point will shift down the frequency scaling curve (denoted by the triangle ▷), which will lead to even higher performance loss in the final product. Also, recently reported execution and fetch throttling techniques to restrain the maximum power lead to the shift of the design power-performance point down a similar curve.

Thus, planning the pipeline depth according to the optimal power-performance balance leads not only to the energy-efficient processor, but also results in higher final performance in product for which meeting the package-set power budget is an issue.

9 Conclusions

In this paper, we have demonstrated that it is important to consider both power and performance while optimizing pipelines. For this purpose, we derived detailed energy models using circuit-extracted power analysis for microarchitectural structures. We also developed detailed equations for how the energy functions scale with pipeline depth. Based on the combination of power and performance modeling performed, our results show that a purely performance-driven, power-unaware design may lead to the selection of an overly deep pipelined microprocessor operating at an inherently power-inefficient design point.

As this work is the first quantitative evaluation of power and performance optimal pipelines, we also performed a detailed sensitivity analysis of the optimal pipeline depth against key parameters such as latch growth factor, latch ratio, latch insertion delay, glitch, and leakage currents. Our analysis shows that there is a range of pipeline depth for which performance increases can be achieved at a modest sacrifice in power-performance efficiency. Pipelining beyond that range leads to drastic reduction in power-performance efficiency with little or no further performance improvement.

Our results show that for a current generation, out-of-order superscalar processor, the optimal delay per stage is about 18 FO4 (consisting of a logic delay of 15 FO4 and 3 FO4 latch insertion delay) when the objective function is a power-performance efficiency metric like $BIPS^3/W$; this is in contrast to an optimal delay of 10 FO4/stage when considering the BIPS metric alone. We used a broad suite of SPEC2000 benchmarks to arrive at this conclusion.

The optimal pipeline depth depends on a number of parameters in the power models which we have derived from current state-of-the-art microprocessor design methodologies, e.g., higher leakage parameters for a process may favor deeper pipelines as area and hence leakage scales with transistor width and not the number of latches. Also, as already established through recent prior work, such optimal de-

sign points generally depend on the input workload characteristics. Our simulation-based experiments on a typical commercial application (TPC-C) shows that although the optimal pipeline depth is around 10-14 FO4 for performance-only optimization, it increases to 24-28 FO4 when we consider power and performance optimizations.

References

[1] D. Brooks, P. Bose, V. Srinivasan, M. Gschwind, P. Emma, and M. Rosenfield. Microarchitecture-level power-performance analysis: The PowerTimer approach. *IBM Journal of Research and Development*, 2003. to appear.

[2] D. Brooks et al. Power-aware Microarchitecture: Design and Modeling Challenges for the next-generation microprocessors. *IEEE Micro*, 20(6):26–44, Nov./Dec. 2000.

[3] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A framework for architectural-level power analysis and optimizations. In *Proceedings of the 27th Annual International Symposium on Computer Architecture (ISCA-27)*, June 2000.

[4] D. Brooks, J.-D. Wellman, P. Bose, and M. Martonosi. Power-Performance Modeling and Tradeoff Analysis for a High-End Microprocessor. In *Power Aware Computing Systems Workshop at ASPLOS-IX*, Nov. 2000.

[5] M. Brown, J. Stark, and Y. Patt. Select-free instruction scheduling logic. In *Proceedings of the 34th International Symposium on Microarchitecture (MICRO-34)*, pages 204–213, December 2001.

[6] P. Dubey and M. Flynn. Optimal pipelining. *J. Parallel and Distributed Computing*, 8:10–19, 1990.

[7] P. G. Emma and E. S. Davidson. Characterization of branch and data dependencies in programs for evaluating pipeline performance. *IEEE Transactions on Computers*, C-36(7):859–875, 1987.

[8] M. J. Flynn, P. Hung, and K. Rudd. Deep-Submicron Microprocessor Design Issues. *IEEE Micro*, 19(4):11–22, July/Aug. 1999.

[9] R. Gonzalez and M. Horowitz. Energy dissipation in general purpose microprocessors. *IEEE Journal of Solid-State Circuits*, 31(9):1277–84, Sept. 1996.

[10] A. Hartstein and T. R. Puzak. The optimum pipeline depth for a microprocessor. In *Proceedings of the 29th International Symposium on Computer Architecture (ISCA-29)*, May 2002.

[11] S. Heo, R. Krashinsky, and K. Asanovic. Activity-sensitive flip-flop and latch selection for reduce energy. In *19th Conference on Advanced Research in VLSI*, March 2001.

[12] M. Hrishikesh, K. Farkas, N. Jouppi, D. Burger, S. Keckler, and P. Sivakumar. The optimal logic depth per pipeline stage is 6 to 8 FO4 inverter delays. In *Proceedings of the 29th International Symposium on Computer Architecture (ISCA-29)*, pages 14–24, May 2002.

[13] V. Iyengar, L. H. Trevillyan, and P. Bose. Representative traces for processor models with infinite cache. In *Proc. 2nd. Symposium on High Performance Computer Architecture (HPCA-2)*, Feb. 1996.

[14] R. Jessani and C. Olson. The floating-point unit of the PowerPC 603e microprocessor. *IBM J. of Research and Development*, 40(5):559–566, Sept. 1996.

[15] P. Kogge. *The Architecture of Pipelined Computers*. Hemisphere Publishing Corporation, 1981.

[16] S. R. Kunkel and J. E. Smith. Optimal pipelining in supercomputers. In *Proceedings of the 13th International Symposium on Computer Architecture (ISCA-13)*, pages 404–411, June 1986.

[17] M. Moudgill, P. Bose, and J. Moreno. Validation of Turandot, a fast processor model for microarchitecture exploration. In *Proceedings of the IEEE International Performance, Computing, and Communications Conference (IPCCC)*, pages 451–457, Feb. 1999.

[18] M. Moudgill, J. Wellman, and J. Moreno. Environment for PowerPC microarchitecture exploration. *IEEE Micro*, 19(3):9–14, May/June 1999.

[19] J. S. Neely, H. H. Chen, S. G. Walker, J. Venuto, and T. Bucelot. CPAM: A common power analysis methodology for high-performance VLSI design. In *Proc. of the 9th Topical Meeting on the Electrical Performance of Electronic Packaging*, pages 303–306, 2000.

[20] S. Palacharla, N. Jouppi, and J. Smith. Complexity-Effective Superscalar Processors. In *Proceedings of the 24th International Symposium on Computer Architecture (ISCA-24)*, 1997.

[21] P. Song and G. De Micheli. Circuit and architecture trade-offs for high-speed multiplication. *IEEE Journal of Solid-State Circuits*, 26(9):1184–1198, Sept. 1991.

[22] E. Sprangle and D. Carmean. Increasing processor performance by implementing deeper pipelines. In *Proceedings of the 29th International Symposium on Computer Architecture (ISCA-29)*, May 2002.

[23] V. Srinivasan, D. Brooks, M. Gschwind, P. Bose, V. Zyuban, P. Strenski, and P. Emma. Optimizing pipelines for power and performance. In *Proc. of ACM/IEEE 35th International Symposium on Microarchitecture*, November 2002.

[24] J. Stark, M. Brown, and Y. Patt. On pipelining dynamic instruction scheduling logic. In *Proceedings of the 33rd International Symposium on Microarchitecture (MICRO-33)*, pages 57–66, Dec. 2000.

[25] N. Vijaykrishnan, M. Kandemir, M. Irwin, H. Kim, and W. Ye. Energy-driven integrated hardware-software optimizations using SimplePower. In *Proceedings of the 27th Annual International Symposium on Computer Architecture*, June 2000.

[26] V. Zyuban. *Inherently Lower Power High Performance Superscalar Architectures*. PhD thesis, University of Notre Dame, March 2000.

[27] V. Zyuban and D. Meltzer. Clocking strategies and scannable latches for low power applications. In *Proc. of Int'l Symposium on Low-Power Electronics and Design*, 2001.

[28] V. Zyuban and P. Strenski. Unified Methodology for Resolving Power-Performance Tradeoffs at the Microarchitectural and Circuit Levels. In *Proc. of Int'l Symposium on Low-Power Electronics and Design*, 2002.

[29] V. Zyuban and P. Strenski. Unified Methodology for Resolving Power-Performance Tradeoffs of the Microarchitectural and Circuit Levels. In *Proc. of Int'l Symposium on Low-Power Electronics and Design*, pages 166–171, 2002.