Lane Decoupling for Improving the Timing-Error Resiliency of Wide-SIMD Architectures

Evgeni Krimer[1], Patrick Chiang[2], and Mattan Erez[1]

[1]Electrical and Computer Engineering Department, The University of Texas at Austin

[2]School of Electrical Engineering and Computer Science, Oregon State University

krimer@utexas.edu, pchiang@eecs.oregonstate.edu, mattan.erez@mail.utexas.edu

Abstract

A significant portion of the energy dissipated in modern integrated circuits is consumed by the overhead associated with timing guardbands that ensure reliable execution. Timing speculation, where the pipeline operates at an unsafe voltage with any rare errors detected and resolved by the architecture, has been demonstrated to significantly improve the energy-efficiency of scalar processor designs. Unfortunately, applying the same timing-speculative approach to wide-SIMD architectures, such as those used in highly-efficient GPUs, may not provide similar gains.

In this work, we make two important contributions. The first is a set of models describing a parametrized general error probability function that is based on measurements of a fabricated chip and the expected efficiency benefits of timing speculation in a SIMD context. The second contribution is a decoupled SIMD pipeline that more effectively utilizes timing speculation and recovery, when compared with a standard SIMD design that uses only conventional timing speculation. The proposed lane decoupling enables each SIMD lane to tolerate timing errors independent of other adjacent lanes, resulting in higher throughput and improved scalability. We validate our models and evaluate our design using a cycle-based GPU simulator, describe the conditions where efficiency improvements can be obtained, and explore the benefits of decoupling across a wide range of parameters. Our results show that timing speculation can achieve up to 10.3% *improvement in efficiency.*

This research was funded in part by the U.S. Government. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government.

1 Introduction

Energy consumption and power dissipation are two of the most important considerations for future processor designs. In previous technology generations, process advancements such as transistor scaling reduced energy consumption sufficiently to enable continued computational density increases, reduced form factor, and improved battery life. Today, however, transistor density is scaling faster than improvements in energy consumption because supply voltage cannot be scaled as aggressively as in the past. As a result of this slow scaling of efficiency, it is necessary to optimize energy efficiency and power across the entire computing system in order to continue to scale performance.

In this paper we explore one such promising combination of architecture and circuits research. We combine the highly-efficient and high-performance design of a modern graphics processing unit (GPU) with timing-speculation circuit techniques, in order to boost efficiency [7, 6, 2]. Because GPUs rely on massive explicit parallelism, they can support a large number of hardware contexts that tolerate memory latencies and maximize throughput by utilizing a large number of computational units. Hence, control overhead can be kept low while the processor provides very high parallel performance. In order to improve GPU efficiency further, a single controller (sequencer) schedules the operations of a large number of parallel functional units in single-instruction, multiple-data (SIMD) fashion, amortizing the control overhead across many operations. Even with this efficient throughput-based design, overall performance is still bounded by total power consumption.

Further efficiencies can be gained through integrated architecture and circuit-level techniques. One such opportunity arises from the fact that a significant portion of the computational energy is consumed not in performing the arithmetic operation itself, but in providing timing guardbands that ensure reliable execution. By operating at a higher-than-the-minimum-required supply voltage, modern guardbanded systems provide enough slack to tolerate timing un-

certainties due to temperature effects, process variations, supply noise, signal integrity, and others. Furthermore, the maximum clock frequency is set to accommodate the worst (critical) path delay of any computation, even though it may occur rarely. Hence, guard-bands have a negative effect on both maximum clock frequency and power consumption, as the supply voltage cannot be scaled further in order to sustain sufficient timing margin. An alternative design is to trim the guardbands and speculate that the circuit will complete its operation in time for the next cycle, even though such aggressive timing cannot be guaranteed. The circuits are then augmented with a mechanism to detect rare timing violations, with the architecture designed to recover from possible errors resulting from such violations [7]. This architecture-circuit *timing speculation* technique has been shown to significantly boost both the efficiency and variation resiliency of scalar pipelines [6, 2].

The main focus of this paper is improving the energy efficiency of massively parallel (GPU-like) architectures through timing speculation. We explore the tradeoffs associated with timing speculation in the context of SIMD designs, and build on our earlier work [13], to develop and evaluate SIMD- and GPU-specific extensions to the timing speculation technique. In our initial work, we observed that naively applying timing speculation to a SIMD pipeline may perform poorly. Any error that occurs in a single functional unit stalls all the lanes of the entire SIMD pipeline, in effect multiplying the baseline error rate by the degree of parallelism and crippling the benefits of timing speculation [13]. In response to this deficiency, we proposed *decoupled parallel SIMD pipelines* (DPSP) [13]. DPSP allows limited slipping between SIMD lanes and enables each lane to tolerate errors independently, thus overcoming the deficiencies of the timing-speculative SIMD described above. DPSP also enables more efficient error recovery when a timing error is detected, as recovery is localized to a single lane. We extend the DPSP concept to a GPU and discuss GPU-specific implementation details for the first time. We also perform a detailed quantitative evaluation that is based on modeling, simulation, and fabricated circuit measurements.

To summarize our main contributions:

- We demonstrate the potential issues of applying timing speculation to SIMD designs, arising from the fact that an error in *any* SIMD lane stalls *all* SIMD lanes. We also show that despite this problem, efficiency can be improved under some conditions.
- We detail the implementation of DPSP (decoupled parallel SIMD pipeline) and recovery, in the specific context of a GPU. We describe the microarchitecture and its interaction with the GPU execution model, discuss implications and alternatives, and evaluate DPSP with detailed simulations of a GPU.
- We develop a new model for the expected probability of errors that result from timing violations when the supply voltage is reduced. This error rate model is based on a combination of analytical formulation, results from prior work [7], and measurements of a recent test-chip fabricated in a $45nm$ CMOS process [18].
- We develop a new analytical model for the potential efficiency gains of timing speculation. This includes modeling the expected execution time due to recovery overheads. Our model uses the ET^2 metric to isolate the improvements of the architecture. This is necessary because DVFS can be applied in addition to timing speculation, and using metrics such as energy-delay product or energy-per-operation cannot distinguish between these techniques.
- We validate the efficiency model using detailed cycle-level simulations of a GPU architecture augmented with timing speculation. We then draw conclusions and present insights into when timing speculation and DPSP are beneficial, and describe expected future trends. This is the first detailed treatment of timing speculation in the context of an efficient parallel architecture built upon lock-step execution. We conclude that naive timing speculation for a GPU will only improve efficiency (ET^2) by 7.8%, whereas DPSP extends the potential gains to 10.3%.

The rest of the paper is organized as follows: we present closely related work in Section 2, detail the implementation of timing speculation in a GPU in Section 3, describe our extensive methodology in Section 4, introduce a model for the timing-violation-induced error rate in Section 5, develop a model for the efficiency improvements of timing speculation and present model-based results in Section 6, analyze detailed microarchitecture simulation results in Section 7, and discuss implementation issues and summary in Section 8.

2 Related Work

The simplest way to reduce power consumption is to employ clock throttling, known as Dynamic Frequency Scaling (DFS). This improves power linearly, but does not improve performance/watt nor energy-efficiency. Dynamic Voltage Scaling (DVS) attempts to maintain the same throughput while increasing energy-efficiency quadratically by keeping the clock constant while reducing the supply voltage. Finally, Dynamic Voltage-Frequency Scaling (DVFS) incorporates both techniques simultaneously, improving power cubically [5].

While DVFS can be used to improve energy efficiency it does not reduce the guardbands introduced to address input-dependent variations. To reduce such guardbands, prior work [7, 6, 2] speculatively assumed that timing will be met in the vast majority of computations. In the rare situations when the timing speculations were incorrect, the delay violations are detected and corrected by the pipeline. However, these previous mechanisms have only been applied to relatively simple scalar pipelines. Besides the initial work [13], where the authors briefly analyzed the performance impact of timing errors in the context of a SIMD pipeline, there are little other related publications.

The introduction of timing errors requires mechanisms for detecting errors. Ernst et al. [7] incorporated a shadow latch to sample the logic output at a time delay after the conventional pipeline register. When a discrepancy occurs between the pipeline register and the shadow latch, an error is detected. Several other related techniques based on double sampling include swapping the register components in order to improve meta-stability [2], error-detection sequentials [3], and tunable-replica circuits (TRCs) [23]. Unfortunately, due to min-path race constraints, these double-sampling approaches require that all MIN and MAX path delays be bounded by approximately $\times 1/2$ and $\times 3/2$ of the average cycle time, respectively.

In order to maintain this MIN-MAX timing constraint for a typical pipeline, 'dummy' delay buffers need to be added to short delay paths. These buffers, as well as the double latches in the detectors, result in energy and area overhead regardless of whether supply voltage scaling or timing speculation is used. Hence, in order to minimize this overhead, Razor-based systems are typically used sparingly only on the critical delay paths that set the worst-case timing closure. However, for a well-designed processor implementing timing speculation, the overheads can be very small. RazorII [6], for example, showed a 33% improvement in overall energy consumption using supply voltage scaling, with the RazorII and min-path buffer insertion power overhead only 1.2% and 1.3%, respectively. Similarly, Bowman et. al. [2] demonstrated a 22% improvement in energy-efficiency and 41% improvement in throughput for their resilient processor. Here, the error-detection mechanisms are inserted into only 12% of the critical path registers, resulting in an overall 3.8% area increase and 0.9% power overhead, including min-path buffer insertion. Finally, a more recent Razor-based design using an ARM processor [4] achieves an aggregate 52% power reduction, with the proposed transition detectors and min-delay buffers exhibiting a 9.4% and 6.9% overhead with respect to power and area, respectively.

Prior work on modeling timing violation error probabilities [20] derived their delay distributions from the error probability distribution reported by Ernst et al. [7]. In this work, we also use that data reported in [7]. In addition, following the same modeling principles, we present our own measurement data using a fabricated chip we designed in a modern CMOS process. Moreover, we make a further contribution by extracting an analytical model for the probability of error as a function of V_{dd} that corresponds well with both the published and new measurement results.

3 DPSP Architecture

Prior work, such as Razor [7], proposed to reduce the large overhead of guardbands by speculating that timing will be met in the vast majority of cases. Timing-violation detectors (based on a shadow latch) and a correction mechanism are introduced to enable timing-speculation and improve efficiency. To date, all analysis and experiments have been with a simple sequential pipeline. In initial work [13], we observed that naively applying timing speculation to a SIMD core may work very poorly. An error in any functional unit stalls the entire pipeline, in effect multiplying the baseline error rate by the SIMD width. Therefore even for relatively low error rates, the effective throughput decreases drastically (Fig. 1).

To overcome the sharp decrease in throughput of the SIMD pipeline with timing speculation, we previously proposed the *decoupled parallel SIMD pipeline* (DPSP) microarchitecture [13]. With DPSP, all functional units in the SIMD organization still execute the same instructions in the same order, but parallel pipelines are allowed to slip with respect to one another so that they can tolerate timing violations independently. Thus, the average throughput will be optimal, in that no additional performance overhead is introduced because of the SIMD nature of the pipeline and degradation is on par with sequential (single-instruction single-data, SISD) pipelines. We describe DPSP below, apply it in the context of a GP-GPU, and evaluate the effectiveness and potential of timing speculation in this context.

To enable DPSP in a GPU, we replace the pipeline latch between the decode stage and the register access and execute stages of the sequential parallel pipeline with a set of shallow FIFO *decoupling queues* (one per SIMD lane). When a timing violation is detected in one of the lanes, only that particular lane stalls and initiates local recovery. The sequencer continues to place instructions into the decoupling queues and all non-faulting lanes execute normally. If timing violations are equally distributed between operations and lanes, the average execution rate is identical across the entire DPSP. The entire parallel pipeline stalls only if violations are not balanced between the lanes and one of the decoupling queues fills up.

The DPSP concept is shown in Fig. 2, where the additional DPSP resources are highlighted. Note that additional decoupling queues can be placed between any two pipeline

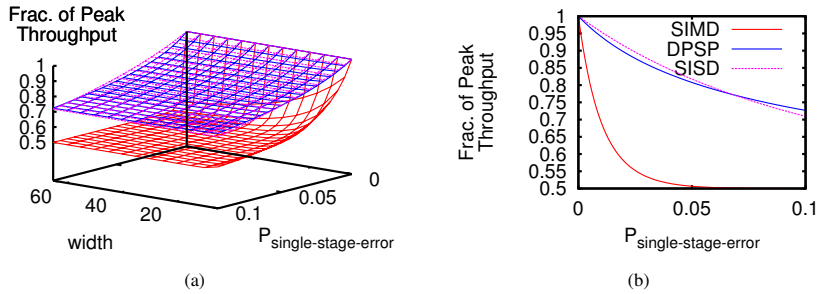


Figure 1: Expected fraction of peak throughput of a 5-stage SIMD pipeline and a 5-stage decoupled parallel SIMD pipelines with decoupling queues as a function of the total probability of error compared to a SISD pipeline (legend for both figures appears in (b)): (a) for varying SIMD width; (b) for 16-wide SIMD. Reproduced from [13].

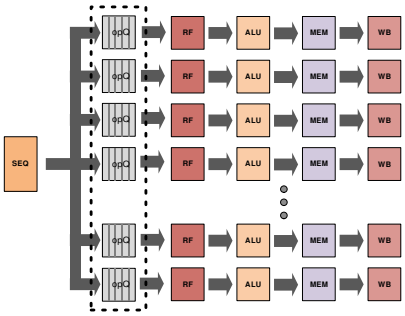


Figure 2: A SIMD pipeline with DPSP. DPSP components are highlighted.

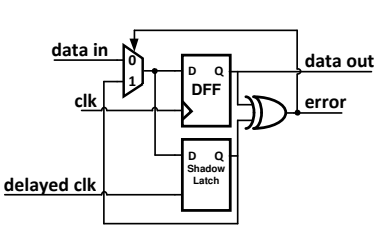
stages to allow for rebalancing of the violations internally within each lane. Our evaluation indicates that a single queue is sufficient to maintain high throughput in the face of input-dependent timing violation errors. Fig. 1.a compares the throughput of SIMD and DPSP for an ideal pipeline that does not stall across a range of random timing-violation error rates (input dependent errors). Fig. 1.b shows a cross-section of the 3D curve for a 16-wide SIMD pipeline and demonstrates how DPSP experiences much lower and more gradual degradation in throughput as the likelihood of a violation grows, maintaining $50-80\%$ better throughput than a simple SIMD with recovery. With the evenly distributed errors used in the analysis shown in the figure, decoupling works nearly perfectly and DPSP matches the degradation curve of a sequential pipeline (SISD). We verify this property of DPSP with detailed architectural simulation and error injection in Section 7.

While the decoupling queues increase the effectiveness of timing speculation, the slip between DPSP lanes requires additional architectural mechanisms to ensure correct execution. In the original SIMD design, all lanes always execute in lockstep and implicitly synchronize after every instruction, but this is not true with DPSP. Any time the lanes explicitly synchronize or potentially communicate the decoupled lanes must be aligned. Current GPUs rely on SIMD pipelines for efficiency and on large degree of data parallelism to hide memory latencies. The execution model is such that communication between lanes is only guaranteed to be correct if an explicit barrier synchronization is executed [16, 21]. A barrier requires all DPSP lanes to internally synchronize to guarantee that all lanes execute the barrier. We adopt a simple solution that we show to be effective in the GPU context. When a barrier is issued, the instruction sequencer is stalled until all decoupling queues are empty. This ensures that no operations in any lane can bypass the barrier regardless of timing violations. It is possible to reduce the overhead associated with stalling the sequencer using *micro barriers* [13], however, our simulations indicate that this complexity is not necessary because barriers are executed rarely.

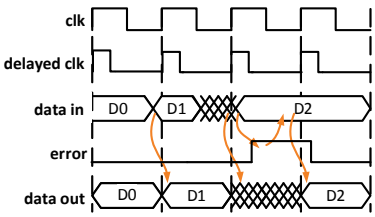
Even though the GPU execution model requires explicit synchronization before any communication between lanes occurs, it is possible that some applications ignore this constraint. We analyzed the applications used in the evaluation and none violate this restriction. Nonetheless, we also evaluate a design in which each memory operation introduces a barrier to guarantee correctness. This barrier is handled as above and stalls the instruction sequencer until every decoupling queue is empty. As we show in our detailed evaluation (Section 7), synchronizing on memory accesses performs well because it maintains the original order and parallelism of memory accesses, to which some applications are sensitive.

3.1 Error Detection and Recovery

We utilize the conventional double-sampling [7] error detection mechanism (Fig. 3), utilizing a shadow latch to sample the logic output at a fixed time delay after the conventional pipeline register flip-flop. In case an error is detected (latched outputs do not match output at delayed clock), the system recovers by stalling the pipeline for one cycle and restarts using the correct outputs of the shadow latch. Stalling for one cycle is sufficient time to allow the pipeline stage that did not meet timing to settle to a correct value. For this type of recovery to succeed, the stall signal must propagate to all pipeline stages before they overwrite the shadow latch. While this is likely not feasible in modern CPUs, and perhaps even in a wide-SIMD design, it is achievable with DPSP. Because of the decoupling queues, the error signal is required to propagate only within a single lane, which is small and can be traversed rapidly. We successfully implemented this recovery policy in our $45nm$ test-chip [18] with DPSP, even though we were unable to accommodate single-cycle stalls with traditional SIMD. Enabling this simple single-cycle recovery mechanism is an important advantage of DPSP, because alternative recovery schemes require energy- and time-consuming pipeline



(a)



(b)

Figure 3: Double sampling error detection mechanism: (a) A shadow latch controlled by a delayed clock usage for error detection; (b) Recovery from error in cycle 2.

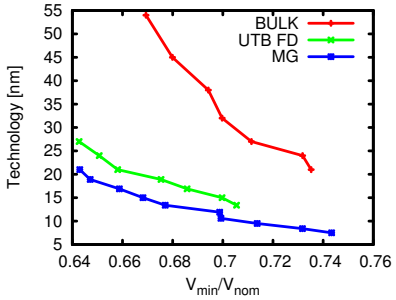


Figure 4: Projected $\frac{V_{min}}{V_{nom}}$ based on ITRS [10] [1]

flushes [7]. When comparing DPSP to SIMD, however, we assume SIMD can recover in a single cycle as well.

Note that double-sampling can detect timing violations only as large as half of the cycle time. Therefore, because we set the nominal supply voltage to be V_{nom}, the supply voltage V_{dd} can be reduced down to only V_{min}, such that $\frac{T(V_{min})}{T(V_{nom})} = \frac{3}{2}$, where T is the critical path delay. We determine V_{min} using the Alpha power-law model [19]. Fig. 4 shows the ratio of $\frac{V_{min}}{V_{nom}}$ for various technologies based on ITRS [10]. This minimum voltage also ensures that recovery is possible by stalling for a single cycle.

3.2 Implementation Overheads

There are two sources of overhead that need to be addressed: the timing speculation mechanism and the DPSP structures. Because there is a range of overhead values for various timing speculation mechanisms reported in the literature (see Section 2), we choose to present the entire range of overheads, up to a conservative 15% energy overhead. To estimate the energy consumption of the DPSP decoupling queues we use Orion 2.0 [11]. A 4 entry 32-bit wide FIFO queue with a single write and a single read port is estimated to consume $\sim 0.285pJ$ in $65nm$ and scales down to $\sim 0.15pJ$ in $32nm$. Compared to $\sim 255pJ/op$ in a modern GPU [17], this overhead is negligible and is not visible in our results.

4 Methodology

The goal of our evaluation is to study the potential energy-efficiency improvements possible with timing speculation in a wide-SIMD architecture, and demonstrate the advantage of using the proposed DPSP technique. To that end, we develop a set of models for estimating timing violation error rates, performance in the presence of such errors, and estimated energy efficiency improvements. The models are based on measurements of a manufactured chip prototype, circuit simulations, and architectural simulations. Using these models, we can quickly explore the design space and gain architectural insight. We then perform detailed architectural simulations to validate our model-based predictions.

We explain our model for error-rate as a function of V_{dd} in Section 5. Our methodology is to measure the distribution of delays of a functional unit, and use this distribution to determine the fraction of computations that will result in an error for a particular supply voltage. We scale the delays using the Alpha power-law model [19] and identify the cutoff based on the nominal operating frequency; we then scale the supply voltage to improve energy-efficiency for a set frequency. The initial delay distributions are based on measurements of a 16-bit multiplier fabricated in a $45nm$ IBM SOI CMOS process. We also use the error-rate results presented by Ernst et al. [7] for an 18-bit multiplier implemented in an FPGA and a 32-bit Kogge-Stone adder simulated in SPICE.

The above methodology is based on a few example measurements, where each measurement is of a different type of implementation (test-chip fabrication, simulation, and FPGA-prototyping). We develop a model that captures the key characteristics of the resulting three very different error-rate functions, enabling us to explore a larger design space than just these three measurements alone.

Our methodology for modeling tradeoffs in energy, explained in Section 6, uses this error-rate model to synthesize an analytical model that combines the impact of timing errors on performance with established models for energy consumption [22, 5]. We use this model to show results across a wide range of parameters. We then validate our model for execution time using a detailed cycle-based architectural simulation (GPGPU-sim [1] ver. 2.1.2), which we enhanced to support DPSP and error injection Section 7. The corresponding energy model for the simulated GPU processor is based on the one presented in [9].

[1] BULK represents planar bulk CMOS, UTB FD represents ultra-thin body fully depleted SOI CMOS, and represents multi-gate CMOS (e.g., FinFETs).

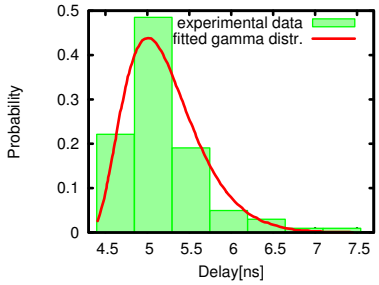


Figure 5: Measured delays and fitted distribution function.

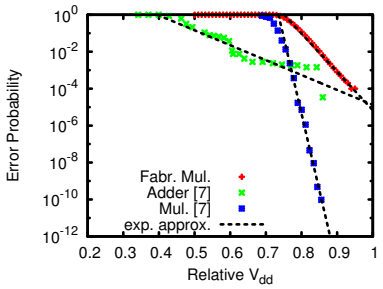


Figure 6: Error rate as a function of supply voltage V_{dd} for various circuits.

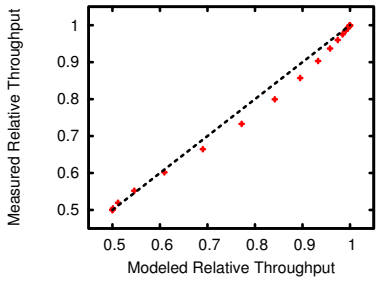


Figure 7: Measured (in GPGPU-sim) and predicted (via model) throughput of DPSP relative to no errors for a synthetic compute-bound kernel.

Table 1: Simulated architecture properties.

Number of Shader Cores	30
Threads in Warp	32
SIMD Pipeline Width*	8×4
Number of Threads / Core	1024
Number of CTAs / Core	8
Shared Memory / Core	16KB
Constant Cache / Core	8KB
Texture Cache / Core	8KB
Number of Memory Channels	8
L1 Cache	none
L2 Cache	none
Main Memory	GDDR3
Bandwidth per Memory Module	4 Bytes/Cycle
DRAM Request Queue Capacity	32
Memory Controller	Out-of-Order (FR-FCFS)
Branch Divergence Method	Immediate Post Dominator
Warp Scheduling Policy	Round Robin
Interconnect Network	Crossbar
Number of MSHRs per / Core	64

Tab. 1 summarizes the parameters of the simulated GPU architecture. This configuration is a good match for the processor used in NVIDIA's Quadro FX5800 and GTX280 GPUs. Note that the SIMD pipeline in each core has 8 lanes and that each operation is repeated 4 times for an instruction vector/SIMD length of 32.

To directly compare results across the various configurations, we use the ET^2 metric, which is the energy dissipated to perform the given computation multiplied by the square of the time it took to execute [15]. The reason we primarily use this metric is that it isolates efficiency improvements to the architecture only and is independent of voltage. A better (lower) ET^2 implies that the system will have superior efficiency for a given performance. Note that ET^2 is the system-level equivalent for the circuit-level ED^2 (energy delay squared product) [15].

5 Error Probability Model

In this section, we present our model for the probability of errors that result from both reducing the voltage guardband and speculating that the circuit will meet timing constraints. We show the error probability as V_{dd} is changed for three functional unit components and then develop a model that generalizes these three profiles. This model allows us to explore a much larger space of tradeoffs between the error rate and the supply voltage.

The first circuit we analyze is a 16-bit multiplier fabricated in the $45nm$ IBM SOI process. The multiplier is part of a chip manufactured to test the DPSP idea [13], but due to a combination of design and fabrication issues we are only able to measure delays with this spin of the chip. We measure the circuit delays at a single supply voltage ($V_0 = 0.53\,V$) and extrapolate the results to the full delay distribution by fitting the data to an analytical Gamma function, as suggested by Kay and Pileggi [12] (Fig. 5 and Tabs. 2–3) . We then use the Alpha power-law model [19] to scale the analytical model of the delays to a range of supply voltages (we use $V_{th} = 480mV$ and $\alpha = 1.3$). We then calculate the error probability as a function of V_{dd} for a fixed target frequency. The target frequency was chosen to exhibit no errors at nominal V_{dd}. Next, the error probability is generated from the fraction of inputs that require a delay greater than the target cycle time (τ) for a varying supply voltage. This is shown in Equations 2 – 3.

$$P(delay = t) = \frac{(t-c)^{a-1}}{b^a \Gamma(a)} e^{-\frac{t-c}{b}} \tag{1}$$

$$\begin{aligned} F_{V_{dd}}(t) &= P(t \le \tau | V_{dd}) \\ &= F_{V_0}\left(t \cdot \frac{V_0}{(V_{dd}-V_{th})^{\alpha}} \cdot \frac{(V_0 - V_{th})^{\alpha}}{V_{dd}}\right) \end{aligned} \tag{2}$$

$$P_{error}(V_{dd}) = P(t > \tau | V_{dd}) = 1 - F_{V_{dd}}(t) \tag{3}$$

Table 2: Gamma distribution parameters

a	4.6124
b	2.10×10^{-10}
c	4.24×10^{-9}

Table 3: Fit quality metrics

Kolmogorov-Smirnov	0.1579
Anderson-Darling	1.8276
Chi-Squared	22.074

Fig. 6 depicts the error probabilities as a function of relative supply voltage for all three functional unit components. We use relative V_{dd} ($\nu \equiv \frac{V_{dd}}{V_{nom}}$), in order to generalize our results across multiple CMOS processes, as each component was designed for a different nominal V_{nom}. The probabilities for the fabricated multiplier were based on the power-law methodology described above ($V_{nom} = 1\,V$). For the 18-bit multiplier and 64-bit adder, we use the probabilities reported by Ernst et al. [7]. These probabilities were derived from measurements of an FPGA implementation of the multiplier ($V_{nom} = 1.8\,V$) and SPICE simulations targeting $0.18\mu m$ for the adder ($V_{nom} = 1.05\,V$).

Fig. 6 also shows the exponential trendline for each of the circuits. This exponential function is a good approximation for the error probability within a range of V_{dd} values. Our model (Eq. 4) includes the "slope" of the probability curve (the exponent) S and the relative voltage V_{maxerr} at which the error probability reaches 1. While this model predicts a non-zero probability of error at the nominal V_{dd}, this error is negligible and we ignore this inaccuracy for simplicity. Tab. 4 summarizes the model parameters for the three circuits we evaluated.

Table 4: Model parameters and R^2 for the circuits evaluated.

	V_{maxerr}	S	R^2
Adder [7]	0.395545254	18.58	0.9491
Multiplier [7]	0.735367316	190.7	0.9935
Measured Fabricated Multiplier	0.751256185	47.82	0.9967

$$P_{error}(\nu) = \begin{cases} 0 & \text{if } \nu = 1 \\ e^{S\cdot(V_{maxerr}-\nu)} & \text{if } V_{maxerr} \leq \nu < 1 \\ 1 & \text{if } \nu < V_{maxerr} \end{cases} \quad (4)$$

6 Model-Based Energy Efficiency Evaluation

We now address the main issue of this paper and evaluate the relationship between the supply voltage, timing speculation errors and recovery, and the resulting energy efficiency of a traditional SIMD pipeline and a DPSP-enabled design. We first present a model for energy efficiency that abstracts many architectural details and enables a rapid exploration of the tradeoff space. We validate this model and show results from a detailed cycle-based architectural simulator in Section 7. We construct this energy-efficiency model by evaluating the impact of fixed-frequency voltage scaling on execution time, dynamic power, and leakage power.

6.1 Execution Time

As the supply voltage is lowered, the probability of an error grows for any given computation. After each error that occurs, the pipeline must recover, thereby increasing execution time. As explained in Section 3, our assumed recovery mechanism inserts a single-cycle bubble into the pipeline and uses the stable and correct computation available at the end of the additional cycle. To ensure that this mechanism is correct, we do not lower the voltage below V_{min}, which is the voltage at which the longest path takes 50% longer than with nominal V_{nom} (Section 3.1).

Another requirement for our recovery mechanism is that the error signal must be propagated in under a cycle. This is relatively easy to do with DPSP, since the error signal is local to a single lane and only needs to propagate up to each lane's decoupling queue. In a traditional SIMD design, on the other hand, it would be extremely difficult to propagate the error signal within the same cycle across all lanes. Nevertheless, we assume that it is possible in order to conservatively estimate the advantages of DPSP.

Accurately modeling the impact of errors on execution throughput requires accounting for the effects of all architecture features, such as memory-related stalls, resource conflicts, and explicit synchronization. To develop a simple model for quickly exploring the design space and drawing insights, we defer the evaluation of these other effects to Section 7, and start with a simple model of executing a single (SIMD) instruction per cycle.

In a traditional lock-step SIMD pipeline with timing speculation, each error event delays execution by one cycle on every lanes. Our initial results [13] indicate that the impact of errors on DPSP throughput can be approximated as the throughput of a scalar pipeline with the same error rate. While we show full simulation results with error injection later in the paper, Fig. 7 shows that approximating a DPSP as a scalar pipeline is a valid assumption. This figure compares the simplistic DPSP model, with the simulated impact of errors, on the throughput of an application that executes a single SIMD instruction/cycle; a compute-bound loop that is not hampered by memory operations and dependencies.

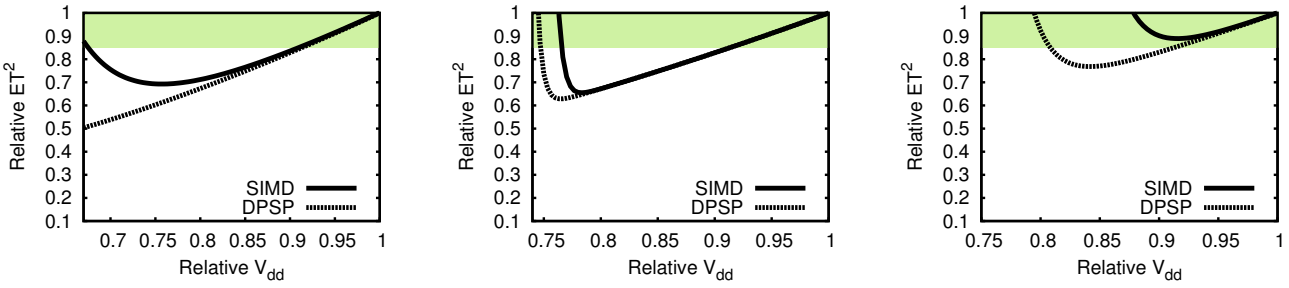
Given our previous model that describes the probability of error as a function of V_{dd} ($P_{error}(V)$), we can now model the throughput and execution time. Eq. 5 summarizes the throughput model, where $N = simdwidth \cdot pipedepth$ is for a traditional SIMD, accounting for the increased error probability when any lane stalls all other lanes. For DPSP, $N = pipedepth$, which is an approximation of DPSP as a scalar pipeline. Next, we derive the execution time as the inverse of throughput (Eq. 6), assuming that the application is perfectly parallelized. Again, note that we validate these assumptions for model simplification in Section 7.

$$\begin{aligned} \Theta(P_{error}, N) &= 1 \cdot (1 - P_{error})^N + \frac{1}{2}\left(1 - (1 - P_{error})^N\right) \\ &= \frac{1}{2} + \frac{1}{2}(1 - P_{error})^N \end{aligned} \quad (5)$$

$$T(V_{dd}) = \frac{1}{\Theta(V_{dd})} \quad (6)$$

6.2 Energy

We derive the relative energy consumed as a function of V_{dd} by considering the dynamic and the static energy components separately. Dynamic energy scales with V_{dd}^2, because no additional computation takes place (the recovery mechanism stalls the pipeline for one cycle). We model



(a) Adder [7] (b) Multiplier [7] (c) Measured fabricated multiplier

Figure 8: ET^2 as a function of V_{dd} for the three error probability functions. Estimated ET^2 for DVFS with no speculation hardware shown by shaded range.

the static energy scaling using the execution time model described above and approximate the change in static power as directly proportional to V_{dd}, as suggested by Su et al. [22]. This linear approximation is reasonable because we vary V_{dd} over a relatively small range that is within 40% of the nominal supply voltage. Eq. 7 represents the energy consumption at V_{nom}, where ϕ is the fraction of dynamic energy versus the total energy for the entire cycle time at V_{nom}. The relative energy as voltage is scaled is shown in Eq. 8, where $T()$ is the delay as a function of V_{dd} (Eq. 6). The scaled time is necessary because static energy increases as the computation time increases, due to integrated leakage energy. When comparing ET^2, we also consider that the baseline design does not require timing speculation. Because we do not precisely know the overhead for a GPU pipeline (see Section 3.1), we consider overheads in the range of $1 - 15\%$.

$$E\left(V_{nom}\right) = E_{nom} = \underbrace{\phi E_{nom}}_{\text{dynamic}} + \underbrace{\left(1-\phi\right) E_{nom}}_{\text{static}} \tag{7}$$

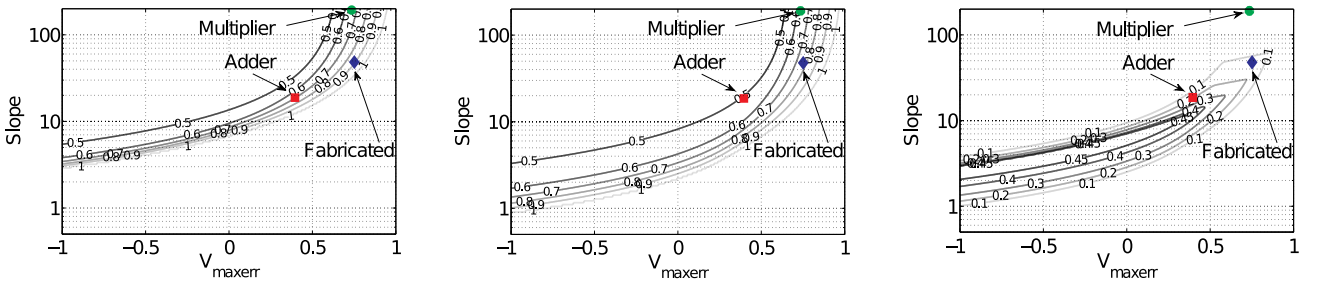
$$\begin{aligned} \frac{E\left(V_{dd}\right)}{E_{nom}} &= \frac{\phi E_{nom} \frac{V_{dd}^2}{V_{nom}^2} + \left(1-\phi\right) E_{nom} \frac{V_{dd}}{V_{nom}} \frac{T(V_{dd})}{T(V_{nom})}}{E_{nom}} \\ &= \phi \frac{V_{dd}^2}{V_{nom}^2} + \left(1-\phi\right) \frac{V_{dd}}{V_{nom}} \frac{T\left(V_{dd}\right)}{T\left(V_{nom}\right)} \end{aligned} \tag{8}$$

6.3 Model-Based Comparison

Fig. 8 compares how ET^2 scales for non-speculative SIMD, speculative SIMD, and DPSP, as a function of V_{dd} with each of the three error probability functions for the three analyzed components (Section 5). We choose $\phi = 0.8$ (80% dynamic energy) as in the McPAT power estimation tool [14]. We normalize ET^2 to that of timing-speculation-enabled SIMD that is operated non-speculatively with DVFS. ET^2 does not change with DVFS as time increases while energy decreases and is a constant in the figure ([15]). Because DVFS can be applied on top of speculation, the ET^2 metric is particularly relevant here. We normalize to a design with timing-speculation circuits to ease comparison between the implementation options, and indicate a range of non-speculative SIMD ET^2 on the figure. The bottom of the range corresponds to a very conservative overhead of 15% for implementing timing speculation (recent work reported overheads of $1-2\%$ [6, 2]). Note that the speculation mechanism limits the lowest operating supply voltage to roughly 70% of V_{dd}, as discussed in Section 3.1. In some cases this lowest possible safe voltage limits the potential benefits of timing speculation.

There are two important observations about these results. The first is that for each of the functional units examined, timing speculation can provide improvements in efficiency. Even without DPSP, the simulated adder and FPGA multiplier exhibit about 15% and 20% improvements respectively compared to a non-speculative baseline, even when considering very conservative speculation implementation overheads of 15%. The measured fabricated multiplier however, exhibits lower gains with no DPSP because its error function has a gentle slope and quickly reaches a probability of 1, leaving less opportunity for beneficial speculation. The large differences in potential arises from the different error function properties. The voltage at which error probability becomes 1 has the strongest influence, with a lower voltage increasing the potential gains.

The second significant result shown in Fig. 8 is that DPSP improves ET^2 on top of speculation, even after accounting for the additional energy overhead of the decoupling queues. The simulated adder has the greatest benefit from decoupling, and ET^2 is improved by an additional 11% on top of SIMD with timing speculation. The benefits are a healthy 12% with the measured multiplier, bringing it below the speculation overhead, but only 3% with the FPGA multiplier. The reason for the small benefit with the FPGA multiplier is that the slope of the error function is very steep, which means that the reduction in effective error rate with decoupling has little impact in terms of energy.

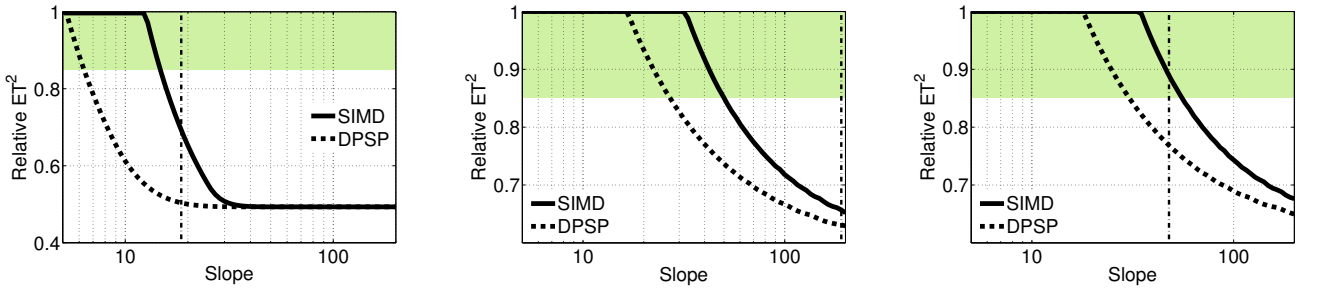


(a) SIMD - lower elevation is better

(b) DPSP - lower elevation is better

(c) Difference of SIMD and DPSP (higher elevation corresponds to bigger improvement of DPSP over SIMD)

Figure 9: Optimal ET^2 with speculation relative to that of SIMD with no speculation.



(a) Adder [7] – $V_{maxerr} = 0.396\,V$

(b) Multiplier [7] – $V_{maxerr} = 0.735\,V$

(c) Measured fabricated multiplier – $V_{maxerr} = 0.751\,V$

Figure 10: ET^2 of SIMD with timing speculation and DPSP for varying error function slopes for the three V_{maxerr} corresponding to the evaluated functional units. Estimated ET^2 for DVFS with no speculation hardware shown by shaded range. Note the starting point of the y-axis.

Given the dependence on the error function properties, we evaluate the benefits of timing speculation and DPSP across a large space of error functions. We use our model to compute expected ET^2 for each technique varying both of the parameters of our modeled error function (Eq. 4): the relative supply voltage at which error probability is 1 (V_{maxerr}) and the exponential slope of the error probability function ($slope$). Fig. 9 show the results as contour plots with the "elevation" in the three subfigures corresponding to the ET^2 of SIMD with timing speculation, ET^2 of DPSP relative to no speculation, and the difference between the two (subtraction); all ET^2 values are reported relative to those of SIMD with no speculation. Note that V_{maxerr} is a parameter of the error function model and can be negative; this particularly makes sense for error functions with a gentle slope. We also show three 2D cross sections of the 3D surfaces in Fig. 10; each subfigure shows the relative ET^2 of speculative SIMD and DPSP with V_{maxerr} set to one of the three evaluated functional units (points indicated in Fig. 9) and varying slope. Note that we do not show points above relative ET^2 of 1, because at that point speculation should be turned off and DVFS applied.

These results strengthen our insights on when timing speculation and DPSP are beneficial. The steeper the slope, the more potential there is for timing speculation to help. A steep slope indicates that there is a region of V_{dd} in which there are few errors, and decreasing voltage within that region reduces power without significantly impacting performance. The value of V_{maxerr} corresponds to the width of this low-error region. Thus, the greatest benefit from timing speculation is towards the top-left of the surface, with nearly no potential towards the bottom (very gentle slope – many errors in entire range) or right (high probability of error from a high V_{dd}). DPSP provides the most benefit near the diagonal of the surface with potential benefits of above 40% on top of timing speculation with a SIMD pipeline. We confirm our earlier observation that DPSP is most effective when the slope is not extreme. This is clearly shown in the cross-section plots (Fig. 10). We also see that when V_{maxerr} is larger, timing speculation and the DPSP improvement on top of it, are greater for steeper slopes.

This analysis and insights are valuable because the error probability function is very much design dependent. Rules of thumb and good models are necessary for making informed tradeoff decisions. DPSP, for example, is able to provide benefits even when both the slope and V_{maxerr} are high, which is very encouraging. We also believe that the trends in the underlying fabrication technology are likely to

Table 5: Benchmark applications used in simulation.

Benchmark	Abbr.	Instr. Count [M]	Memory Structures Used			Explicit Synchronization
			Shared	Constant	Texture	
AES Cryptography	AES	28	✓	✓	✓	✓
Breadth First Search	BFS	17				
Molecular Dynamics - Coulombic Potential	CP	126		✓		
Discontinuous Galerkin Time-domain Solver	DG	596	✓		✓	✓
3D Laplace Solver	LPS	82	✓			✓
LIBOR Monte Carlo	LIB	907		✓		
DNA Matching	MUM	77			✓	
Neural Network Digit Recognition	NN	68				
N-Queens Solver	NQU	2	✓			✓
Ray Tracing	RAY	71		✓		✓
MD5 Hashing	STO	134	✓			
Weather Prediction	WP	215				

lead to error functions that are closer to the sweet spot of DPSP. In an ideal design, $V_{maxerr} = 1$ and the slope is very steep, which result from perfectly balanced paths. Because of process variation and the challenges in balancing complex designs with the large number of optimization options and uncertainties of a modern process, we believe that gentler slopes and non-ideal lower V_{maxerr} values are likely in the future.

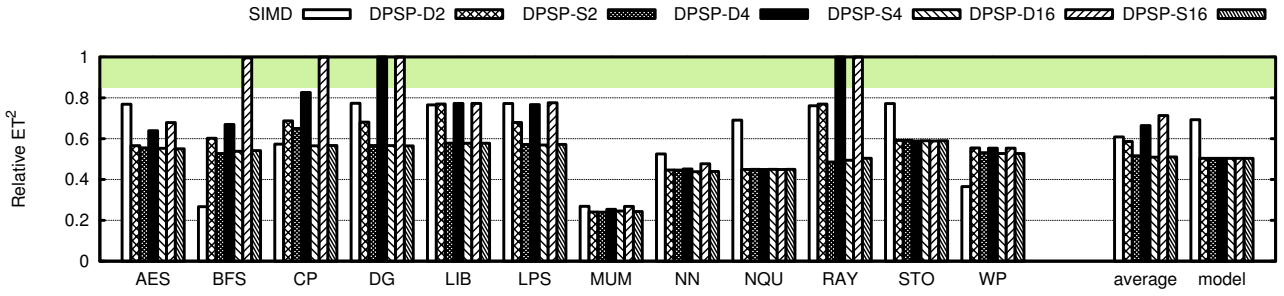
7 Detailed Microarchitecture Simulation-Based Evaluation

In this section we present analysis based on detailed microarchitecture simulations and validate the model-based results presented above. As explained in Section 4, we use the GPGPU-sim [1] detailed microarchitectural GPU simulator, which we modified to support DPSP, timing-speculation, and error injection. We evaluate the set of 12 applications (Tab. 5) included with the GPGPU-sim distribution, which are representative of GPGPU workloads [1]. We do not use the simplified exponential error probability model from Section 5 for injecting errors. Instead, we use the error distributions directly and choose the error rate based on the V_{dd} being simulated. We use the cycle-based simulator to estimate the impact of errors and recovery on execution time. To model power, we rely on the technique presented by Hong and Kim [9]. This power model was shown to match well with real hardware, but its empirical nature requires us to use simulation parameters that are as close as possible to those used to develop the model. To model improvements from timing speculation, we use Eq. 7 (Section 6.2). When presenting the results we discuss how we account for the fact that only a fraction of overall processor power is reduced because timing speculation is mostly applicable to logic.

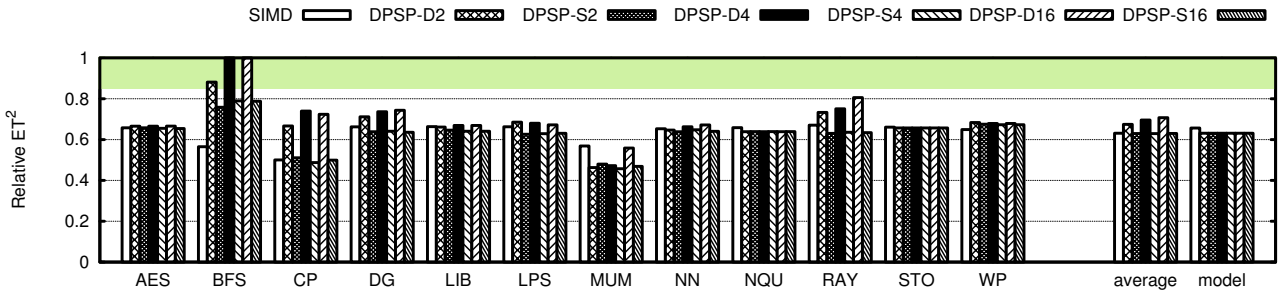
We present different configurations of DPSP, varying the size of the decoupling queues as well as the coalescing behavior. The DPSP-S* variants (where * marks the depth of the queue) all synchronize across the lanes before each memory operation to ensure coalescing behavior that matches the baseline implementation. The DPSP-D* schemes rely on the coalesce buffer and on the fact that slip between lanes is bounded by the shallow decoupling queues, such that most memory operations hit in the coalesce buffer. We also evaluated other options for determining when to synchronize the lanes, but the impact on performance was minimal compared with those presented.

Fig. 11 summarizes the optimal ET^2 achieved in simulation for each application when using the three empirical error probability profiles developed in Section 5. To find the optimal ET^2, we simulated each application with multiple V_{dd} settings (each with its appropriate error rate and power improvement) and chose the best one. Note that the ET^2 values reported in Fig. 11 are under the assumption that the entire processor benefits from reduced V_{dd}. Unfortunately, some structures, such as the I/O drivers and SRAMs, cannot be improved with timing speculation. The results should be derated by a factor corresponding to the fraction of total processor power of the circuits for which margin can be trimmed. When reporting absolute numbers in the discussion below, we assume that 75% of the power is dissipated by structures amenable to guardband reduction, which is our best estimate based on our previously described power models [14, 9]. Fig. 11, however, compares the techniques without this derating factor.

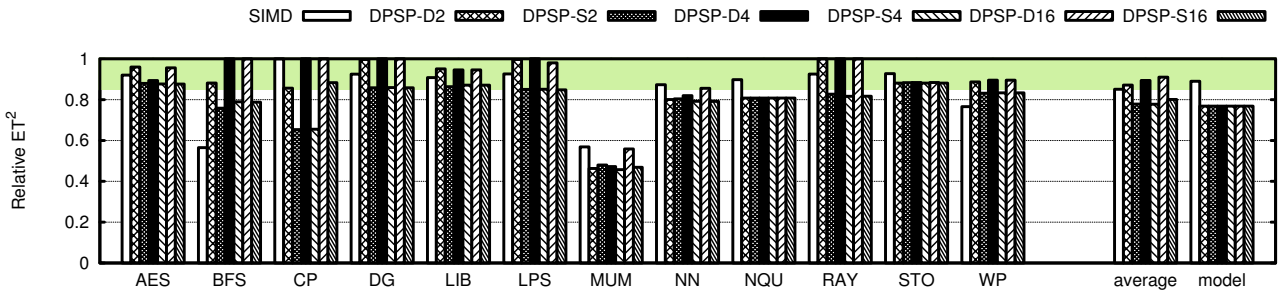
The results show that the analytical model we developed in Section 6 correlates well with the overall (average) behavior of both SIMD and DPSP for all error profiles. We also present the behavior of each application across a range of supply voltage values (rather than just the optimal) in Fig. 12. This figure shows that application behavior also matches the model qualitatively. Because of space constraints, we only show the detailed behavior for the error function corresponding to the measured multiplier circuit from our test-chip.



(a) Adder [7] error profile



(b) Multiplier [7] error profile



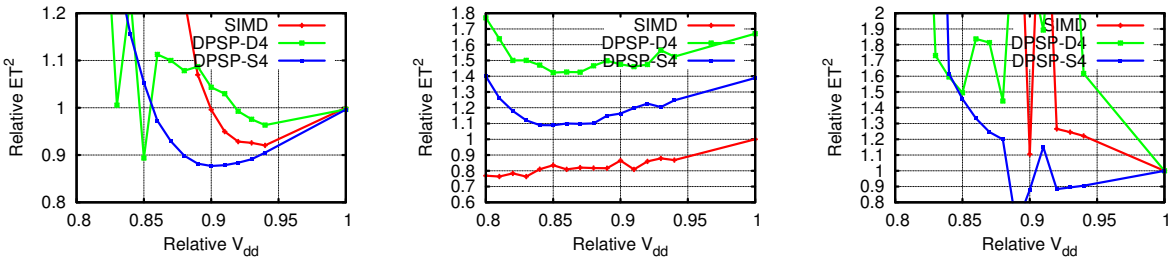
(c) Measured fabricated multiplier error profile

Figure 11: Optimal simulated and model-based ET^2 with different configurations of timing speculation and DPSP. Estimated ET^2 for DVFS with no speculation hardware shown by shaded range.

As expected, our model overestimates the ET^2 of timing-speculative SIMD because the natural synchronization points and memory-related stalls of real execution. Surprisingly, however, DPSP with deep queues and minimum synchronization (only on explicit barriers) performs poorly. The reason is that decoupling results in more non-coalesced memory accesses, which reduces performance [16]. Moreover, the number of non-coalesced accesses is higher with deeper decoupling queues. The DPSP configurations that synchronize before every memory operation do not break software optimization for coalescing. We expect newer GPU architectures, such as NVIDIA's Fermi [17], to be much less sensitive to this coalescing issue because of the introduction of a first-level cache. Unfortunately, the energy model we use does not include a cache and its empirical nature prevents us from adding one. It is also interesting to observe that with all configurations, there is no advantage to deeper queues, and a low-cost 4-deep decoupling queue is sufficient.

Several applications stand out in behaving differently from the average and we discuss each in detail below. AES (Fig. 12.a) shows sensitivity to coalescing that is not sufficiently covered by the coalesce buffer with DPSP-D*, which allows slip on memory operations. For some specific and relatively high error rate, however, increasing the number of stalls, in effect, decreases the slip (all lanes have errors) resulting in a corner-case that improves coalescing. Behavior is smoother with DPSP-S* because coalescing is guaranteed as all loads synchronize.

BFS (Fig. 12.b) suffers from a very high control divergence rate [1]. The method used by GPUs to handle control divergence in their SIMD pipelines requires synchroniza-



(a) AES (b) BFS (c) CP

(d) DG (e) LPS (f) LIB

(g) MUM (h) NN (i) NQU

(j) RAY (k) STO (l) WP

Figure 12: ET^2 as a function of V_{dd} for various benchmarks using the error profile measured for our fabricated multiplier.

tion each time multiple control paths reconverge [8]. The high divergence rate of BFS results in many more barrier operations than in a typical application, which limits the effectiveness of decoupling. Note that our mechanism for synchronizing the decoupled lanes stalls the instruction sequencer and thus significantly degrades performance. Notice that the deeper the queues, the higher the penalty for synchronization because the queues are drained before instruction sequencing resumes.

CP, DG, and RAY (Fig. 12.c, Fig. 12.d, and Fig. 12.j) show extreme cases of sensitivity to memory coalescing. Timing speculation with SIMD works quite well and DPSP-S* enables even higher efficiency.

MUM (Fig. 12.g) has unusually low SIMD occupancy with 60% of SIMD instructions (warps in NVIDIA's terminology) having fewer than 5 operations out of a maximum of 32 [1]. As a result, decoupling provides little advantage because the error rate of SIMD is not amplified as much as with other applications. Furthermore, the performance of this application is bound by memory bandwidth, further decreasing the benefits of decoupling. Being memory bandwidth bound, on the other hand, provides a large opportunity for timing speculation to improve efficiency. Even a large number of errors can be tolerated without increasing execution time.

WP (Fig. 12.l) is very sensitive to memory performance and latency. The register file in our GPU configuration is not large enough to support the levels of locality and parallelism required by this application and memory accesses cannot be effectively overlapped with computation [1]. The strong dependence of WP's performance on memory accesses and memory scheduling results in unusual ET^2 behavior as V_{dd} is decreased. SIMD with timing speculation reaches its highest efficiency at relative V_{dd} of 0.9 and further decreases in voltage result in reduced performance and worse ET^2. Decoupling exacerbates the memory scheduling and coalescing issue and is even less effective than lockstep SIMD execution.

8 Discussion and Conclusions

This paper, for the first time, presents a detailed discussion and evaluation of applying timing speculation to an efficient wide-SIMD parallel pipeline. We demonstrate that there is significant potential in extending this circuit/architecture technique to a GPU pipeline. We describe a GPU-specific implementation that accounts for the execution model and synchronization/memory-access mechanisms of GPUs. We also describe the use of low overhead decoupling queues to minimally augment the SIMD design, improving both DPSP execution and the tolerance to timing violations. DPSP provides an additional benefit of simplifying the recovery mechanism to enable higher efficiency and performance. Specifically, DPSP enables each SIMD lane to recover from errors independent of other lanes. This implies that any error signal can only be propagated within each simple SIMD lane, which is realizable in a single cycle. Such single-cycle signaling enables stall-based recovery that is most energy and performance efficient. We further evaluate our GPU-based design and show that the peculiarities of its memory access architecture result in the non-intuitive conclusion that a very small degree of decoupling with synchronization for every memory operation yields the best efficiency.

We observe that the general trend in GPU architectures is to reduce the dependence on exact memory optimizations, and therefore expect cache-based GPUs to show greater gains with larger amounts of decoupling. We also describe a potential problem involving high synchronization penalties for highly control divergent applications. While we do not evaluate a solution in this paper, recent work on mechanisms to mitigate the negative impact of control divergence (e.g., Fung and Aamodt [8]) are likely to improve DPSP as well.

An important contribution of our work is the new detailed models we provide for analyzing timing-speculative SIMD and DPSP designs and the insights we draw from these analyses. We present a novel model for deriving a timing-speculation induced error probability. This model is based on empirical measurements from a chip fabricated in a modern $45nm$ CMOS process, as well as on previously reported results [7] that have also been used in other related models (VARIUS [20]). We generalize the model using simple and intuitive parameters that allow us to explore the large implementation space. We also develop a new model for the behavior of relative ET^2, and validate both models with detailed cycle-based simulations of a GPU [1].

Using this comprehensive evaluation framework leads to interesting insights with respect to timing speculation and SIMD. We note that current trends point to increasing SIMD width for improved efficiency. Based on our analysis, successfully applying timing speculation to such designs requires this new DPSP mechanism, because a naive implementation results in a much higher effective error rate, eliminating much of the advantages provided by operating with reduced margins. We describe how to intuitively interpret the different error probability functions of different circuits on the potential benefits of timing speculation. Finally, we conclude that timing speculation is likely to remain effective in future technologies, as the trends of increasing process variation should lead to a gentler error function slope and a lower relative voltage where the error probability approaches 1. Based on this discussion, we conclude that the average of 10.3% improvement in ET^2 we observe in our evaluation is conservative rather than optimistic.

Acknowledgements

This work is supported, in part, by the following organizations: Department of Energy under Early Career Program, DARPA under contract HR0011-10-9-0008, Intel Corporation, and The Texas Advanced Computing Center. We thank Robert Pawlowski, Joseph Crop, and Jacob Postman from the Oregon State University for their effort to implement the test-chip and provide the measurements.

References

[1] A. Bakhoda, G. Yuan, W. Fung, H. Wong, and T. Aamodt. Analyzing CUDA workloads using a detailed GPU simulator. In *Performance Analysis of Systems and Software, 2009. ISPASS 2009. IEEE International Symposium on*, pages 163–174. IEEE, 2009.

[2] K. Bowman, J. Tschanz, N. Kim, J. Lee, C. Wilkerson, S. Lu, T. Karnik, and V. De. Energy-Efficient and Metastability-Immune Resilient Circuits for Dynamic Variation Tolerance. *IEEE J. Solid-State Circuits*, 44(1):49–63, 2009.

[3] K. Bowman, J. Tschanz, C. Wilkerson, S. Lu, T. Karnik, V. De, and S. Borkar. Circuit techniques for dynamic variation tolerance. In *Proceedings of the 46th Annual Design Automation Conference*, pages 4–7. ACM, 2009.

[4] D. Bull, S. Das, K. Shivashankar, G. Dasika, K. Flautner, and D. Blaauw. A power-efficient 32 bit arm processor using timing-error detection and correction for transient-error tolerance and adaptation to pvt variation. *Solid-State Circuits, IEEE Journal of*, 46(1):18–31, 2011.

[5] A. Chandrakasan, M. Potkonjak, J. Rabaey, and R. Brodersen. Hyper-lp: A system for power minimization using architectural transformations. In *Computer-Aided Design, 1992. ICCAD-92. Digest of Technical Papers., 1992 IEEE/ACM International Conference on*, pages 300–303. IEEE, 1992.

[6] S. Das, C. Tokunaga, S. Pant, W. Ma, S. Kalaiselvan, K. Lai, D. Bull, and D. Blaauw. RazorII: In situ error detection and correction for PVT and SER tolerance. *Solid-State Circuits, IEEE Journal of*, 44(1):32–48, 2009.

[7] D. Ernst, S. Das, S. Lee, D. Blaauw, T. Austin, T. Mudge, N. Kim, and K. Flautner. Razor: circuit-level correction of timing errors for low-power operation. *IEEE MICRO*, pages 10–20, 2004.

[8] W. W. Fung and T. M. Aamodt. Thread Block Compaction for Efficient SIMT Control Flow. In *17th International Symposium on High Performance Computer Architecture (HPCA-17)*, February 2011.

[9] S. Hong and H. Kim. An integrated gpu power and performance model. In *Proceedings of the 37th annual international symposium on Computer architecture ISCA 10*. ACM Press, 2010.

[10] ITRS. Process Integration, Devices, and Structures (PIDS) Update, 2010. URL `http://www.itrs.net/links/2010itrs/home2010.htm`.

[11] A. Kahng, B. Li, L. Peh, and K. Samadi. Orion 2.0: A fast and accurate noc power and area model for early-stage design space exploration. In *Proceedings of the conference on Design, Automation and Test in Europe*, pages 423–428, 2009.

[12] R. Kay and L. Pileggi. Primo: probability interpretation of moments for delay calculation. In *Proceedings of the 35th annual Design Automation Conference*, pages 463–468. ACM, 1998.

[13] E. Krimer, R. Pawlowski, M. Erez, and P. Chiang. Synctium: a near-threshold stream processor for energy-constrained parallel applications. *IEEE IEEE Computer Architecture Letters*, pages 21–24, 2010.

[14] S. Li, J. Ahn, R. Strong, J. Brockman, D. Tullsen, and N. Jouppi. Mcpat: an integrated power, area, and timing modeling framework for multicore and manycore architectures. In *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on*, pages 469–480. IEEE, 2009.

[15] A. Martin. Towards an energy complexity of computation. *Information Processing Letters*, 77(2-4):181–187, 2001.

[16] J. Nickolls, I. Buck, M. Garland, and K. Skadron. Scalable parallel programming with CUDA. *Queue*, 6(2):40–53, 2008.

[17] NVIDIA Corp. Fermi CUDA Architecture, 2009. URL `http://www.nvidia.com/object/fermi_architecture.html`.

[18] R. Pawlowski, E. Krimer, J. Crop, J. Postman, N. Moezzi-Madani, M. Erez, and P. Chiang. Synctium-I: A 530mV, 10-lane SIMD Processor with Variation Resiliency in 45nm-SOI. In *Solid-State Circuits Conference, 2012. ISSCC 2012. Digest of Technical Papers. IEEE International*, feb. 2012.

[19] T. Sakurai and A. Newton. Alpha-power law MOSFET model and its applications to CMOS inverter delay and other formulas. *Solid-State Circuits, IEEE Journal of*, 25(2):584–594, 1990.

[20] S. Sarangi, B. Greskamp, R. Teodorescu, J. Nakano, A. Tiwari, and J. Torrellas. VARIUS: A model of process variation and resulting timing errors for microarchitects. *Semiconductor Manufacturing, IEEE Transactions on*, 21(1):3–13, 2008.

[21] J. Stone, D. Gohara, and G. Shi. OpenCL: A parallel programming standard for heterogeneous computing systems. *Computing in science & engineering*, 12(3):66, 2010.

[22] H. Su, F. Liu, A. Devgan, E. Acar, and S. Nassif. Full chip leakage estimation considering power supply and temperature variations. In *Proceedings of the 2003 international symposium on Low power electronics and design*, pages 78–83. ACM, 2003.

[23] J. Tschanz, K. Bowman, S. Walstra, M. Agostinelli, T. Karnik, and V. De. Tunable replica circuits and adaptive voltage-frequency techniques for dynamic voltage, temperature, and aging variation tolerance. In *VLSI Circuits, 2009 Symposium on*, pages 112–113. IEEE, 2009.