Non-Speculative Double-Sampling Technique to Increase Energy-Efficiency in a High-Performance Processor

Junyoung Park, Ameya Chaudhari, and Jacob A. Abraham
Computer Engineering Research Center
The University of Texas at Austin, Austin, Texas, USA 78712
Email: {jypark, ameyac, jaa}@cerc.utexas.edu

***Abstract*—In the past few years, many techniques have been introduced which try to utilize excessive timing margins of a processor. However, these techniques have limitations due to one of the following reasons: first, they are not suitable for high-performance processor designs due to the power and design overhead they impose; second, they are not accurate enough to effectively exploit the timing margins, requiring substantial safety margin to guarantee correct operation of the processor. In this paper, we introduce an alternative, more effective technique that is suitable for high-performance processor designs, in which a processor predicts timing errors in the critical paths and undertakes preventive steps in order to avoid the errors in the event that the timing margins fall below a critical level. This technique allows a processor to exploit timing margins, while only requiring the minimum safety margin. Our simulation results show that proposed idea results in 12% and 6% improvement in energy and Energy-Delay Product (EDP), respectively, over a Razor-based speculative method.**

I. Introduction

Today's processors have to operate in various operating conditions. In order to ensure correct operation even in the worst-case combination of the conditions, the timing margins of modern processors have to be increased significantly. Therefore, timing margins in modern processors occupy a significant fraction of the clock cycle period. However, the probability of occurrence of the worst-case condition is extremely low. Since most processors, most of the time, operate under normal operating conditions, the worst-case margins incorporated in processor design impose energy and performance penalties on the processor.

In order to utilize these excessive timing margins and improve the processor's Energy-Delay Product (EDP), a metric used to measure energy consumption for a given performance, a number of run-time slack monitoring techniques have been proposed. A class of these techniques monitors the behaviors of critical path replicas [5-8] to predict the timing slack in the processor critical paths. Another popular class of techniques monitors delay changes in the actual critical paths using circuit-level speculative double-sampling [1-4].

However, these two types of recent techniques have a few limitations. The methods which replicate critical paths require large safety margins, as it is not possible to design critical path replicas that exactly replicate the delay behaviors of actual critical paths. The speculative methods are not suitable for high-performance processor designs because the performance overhead due to the speculative operation and the required architectural modifications offset the energy benefit obtained by reducing the timing margins.

In this paper, we propose an alternative, more effective timing slack reduction technique that is suitable for high-performance processor designs. In our technique, we use a new sequential circuit to pre-sample the output of the critical path and estimate the timing slack in the path. The pre-sampling operation enables our technique to operate in a non-speculative fashion, reducing the timing margins to a minimum.

II. Related Research

Some of the popular techniques employed for minimizing timing margins use critical path replicas [5-8]. As the name indicates, instead of monitoring timing slack in the processor's actual critical paths, these methods monitor the slack in critical path replicas. Fig. 1 (a) shows a block diagram of TEAtime [5], an example of the methods that use critical path replicas. These techniques have an advantage of lower area overhead as they need only a few critical path replica blocks to track a large number of timing critical paths. The critical path replica blocks are placed at various locations in the processor die to account for the within-die (WID) variations. In addition, these techniques do not require architectural modifications to the processor designs such as circuit-level speculation techniques do, which we will explain in detail later.

In order to achieve a better EDP for the processor, more accurate replica designs are required for these methods. The path delays estimated by the critical path replicas are not always perfect and hence these techniques require a large safety margin. With variations getting worse with each new process technology, the safety margin has to be increased for the newer generations of fabrication processes, which is the main drawback of this technique.

Another popular technique to detect and minimize the timing slack is that of using a speculative method based on double-sampling [1-4]. A famous example of these techniques, a Razor FF (RFF) [2], is depicted in Fig. 1 (b). This method uses a duplicated sequential circuit, which is composed of a main FF and a shadow latch, and places it in the actual critical path in order to capture its output. In the event that the latched data in the main FF and that in the shadow latch do not match, i.e., the case when a speculative operation fails, the processor re-computes the failed instruction with the data from the shadow latch. This speculative technique enables the processor to eliminate all of the timing margins.

Despite its ability to accurately monitor the critical path delay and minimize the timing margins, the speculative method has not been widely used for high-performance processor designs. One of the major reasons for this is that high-performance processors usually have a large number of critical paths. The same number of the duplicated sequential circuits needs to be added to the processor to monitor all these critical paths. This increases the processor power consumption due to the increased power of the clock network as well as the storage elements themselves. It also requires major architectural changes to the processor design. Wherever the output of a logic path is being written to a memory block, a non-speculative stabilizing pipeline stage (ST) needs to be added. This additional pipeline stage imposes latency overhead, and of much more concern, a significant power overhead on the processor. Fig. 2 describes the processor pipeline modification that is needed for a speculative based processor design. Since modern high-performance processors adopt architecture-level speculative operations and out-of-order execution techniques that require additional memory arrays such as BTB, BHB, ROB, ISSUE-Q, this speculative method requires a number of additional stabilizing stages to be added before each memory write. Another overhead posed by the circuit-level speculative technique is additional buffers inserted to prevent hold time violations in the speculative clocking paths, which are called short path violations.

An aging prediction technique [10] uses a clocking concept similar to ours, which double-samples data non-speculatively. The aging detection operation is performed based on a duplicated sequential circuit that is composed of two FFs: a main FF and an additional FF that is used as an aging monitor. The additional FF samples the output of a path prior to the main FF due to an aging margin in its data path. This circuit compares the two captured data values in order to check if the aging effects of the path are severe enough to cause a real circuit failure. Even

978-3-9815370-0-0/DATE13/©2013 EDAA

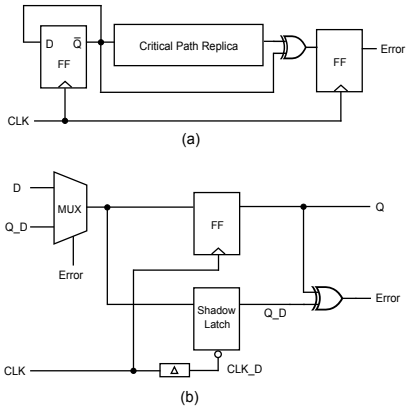

Fig. 1: Two major margin elimination techniques: (a) The critical path replica based (TEAtime [5]), (b) The circuit-level speculative operation based (Razor FF [2])

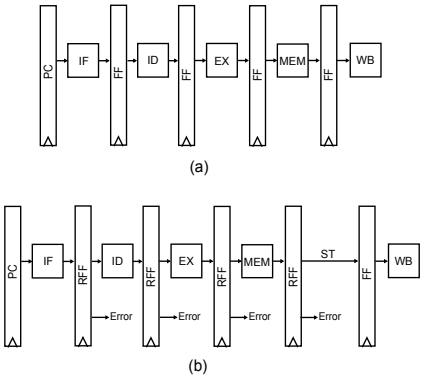

Fig. 2: (a) An original pipeline configuration, (b) A modified pipeline configuration for circuit-level speculative operation (Razor FF [2])

though our proposed circuit has a similar circuit configuration to this aging-aware FF, our idea differs from this as our fundamental idea is built based on adaptive frequency scaling so as to exploit timing slack. In addition, while this approach uses fixed design-time specified margins, we extend the concept to include variable delays, which enables it to be applied for the energy reduction techniques.

III. Design Concept

In this section, we introduce a technique that has the accuracy of in-path delay monitors, like those used in the circuit-level speculative double-sampling, and that is suitable for high-performance processor designs, like the critical path replicating method. The key concept of our technique is sampling the outputs of critical paths prior to normal sampling, which is called non-speculative double-sampling. Fig. 3 shows the conceptual block diagram of a non-speculative double-sampling FF and a timing diagram that is based on it. Here, CLK (clock) signal is generated from CLK_P (the pre-sampling clock) signal with a Δ (the safety margin) delay. Since, due to the safety margin, the timing error rate of FF A is almost zero, this allows us to operate a processor in a non-speculative fashion. The frequency can be increased until FF B encounters an error during run-time. When the two outputs of the FFs are identical, the processor will increase the clock frequency in order to utilize any excessive timing slack. On the other hand, when the outputs of the two FFs are different, the processor should decrease the clock frequency in order to ensure its correct operation. This process makes the processor operate reliably at the maximum possible frequency.

Fig. 4 (a) illustrates our non-speculative double-sampling FF, composed of a main FF and a shadow latch that performs pre-sampling operations. Similar to the speculative double-sampling technique, we use our duplicated sequential circuit in the same way as in-path delay monitors. While the speculative approach uses two clocks, CLK (clock) and CLK_D (the delayed clock), which trails CLK by a delay of Δ (the

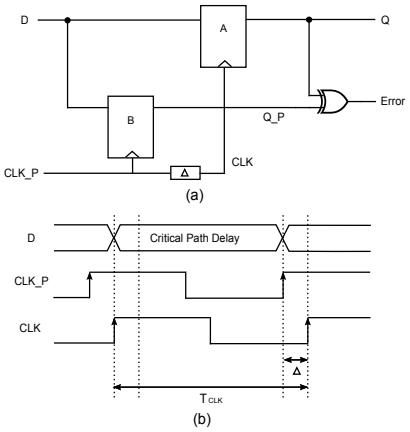

Fig. 3: (a) A conceptual model of pre-sampling FF, (b) A timing diagram

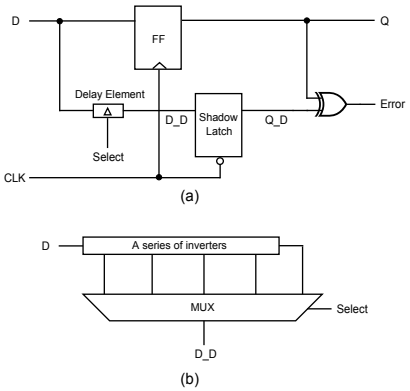

Fig. 4: (a) Our non-speculative double-sampling FF, (b) A Delay Element block

safety margin), our scheme uses a single phase clock. Since adding this delay on the clock path adds skew to the clock and requires additional design effort as well as a great deal of power consumption by the clock networks, we add the delay on the data path of the shadow latch instead, in order to design the pre-sampling function in our single phase clocking system.

We use tunable delay for the Delay Element so it can provide different safety margins for different operating points. Fig. 4 (b) shows a block diagram of our tunable delay, the Delay Element block. Since the tunable delay on the shadow-latch has a lower activity factor, this leads to a lower power overhead for a processor when compared to the speculative approaches that use delayed clocks.

However, the clock frequency of the non-speculative based processor is lower than that of the speculative based one, as it must operate with the safety margin. Fig. 5 illustrates how the maximum clock frequency is determined for two different techniques. Here we can see that the clock

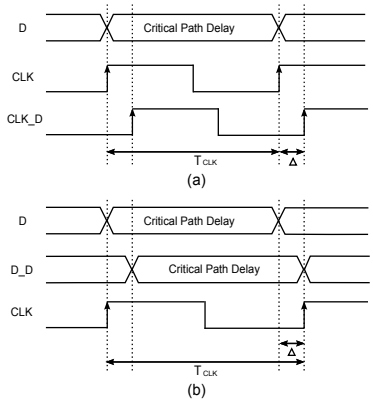

Fig. 5: The maximum clock frequency of (a) the speculative based processor, (b) the non-speculative based processor

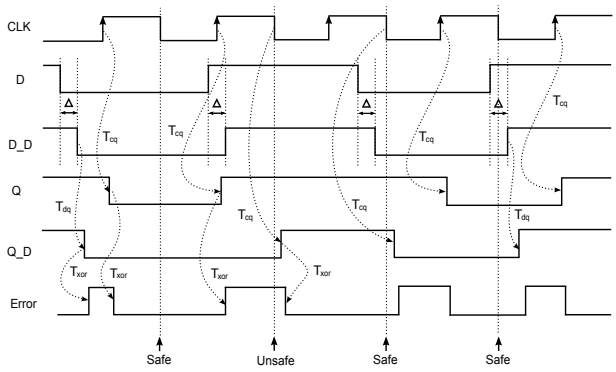

Fig. 6: Non-speculative double-sampling FF timing diagram

cycle time, T_{CLK}, of the non-speculative technique exceeds T_{CLK} of the speculative technique by Δ.

Fig. 6 shows a timing diagram of the non-speculative double-sampling operations. The D signal path shows the timing diagram for the input port of the main FF. D_D is a delayed version of the D signal and is the input of the shadow latch. Q and Q_D are the outputs of the main FF and the shadow latch, respectively. The error, which is computed by the XOR operation of the Q and Q_D signals, is sampled at every negative clock edge in order to detect any timing violation in the shadow latch. We can see that timing violation of the shadow latch is detected when different Q and Q_D signals are captured at the same rising clock edge as shown in the second case of Fig. 6.

TABLE I compares our non-speculative double-sampling technique with the two previously introduced techniques. Here, we can see that the methods that use critical path replicas are suitable for high-performance processor designs but require a large safety margin due to their inherent inaccuracy. While the speculative double-sampling methods do not require any safety margin, they are somewhat difficult to use in high-performance processors. On the other hand, our non-speculative method is quite attractive with respect to almost all of the aspects noted in TABLE I except for the need for a small safety margin.

IV. Simulation and Evaluation

In order to verify the effectiveness of our non-speculative technique, in this section we compare the energy consumption and EDP of our non-speculative based processor with those of the speculative based processor that is based on Razor FFs.

A. RT-level and circuit-level simulations

We implemented the speculative as well as the non-speculative technique in the 5-stage ARM processor design, which is our design base. First, we synthesized the design in order to generate a base netlist using a IBM 45nm slow-corner process library. We then carried out several iterations of P&R (Place and Route) steps in order to optimize the synthesis result. The optimization steps led to the emergence of a large number of critical paths in the processor.

TABLE I: Comparison between previous research and our non-speculative double-sampling technique

	(a) Critical Path Replicas (CPRs)	(b) Speculative Double-Sampling	(c) Non-Speculative Double-Sampling
Architectural Modification	Not required	Required	Not required
Performance Penalty	No	Yes	No
Short Paths Constraints	No	Yes	No
Critical Path Tracking Accuracy	Inaccurate	Accurate	Accurate
Additional Power Overhead	CPMs	CPMs + Recomputing + Additional pipes + Buffers	CPMs
Area Overhead	Several CPMs	CPMs + Buffers + Additional FFs for Stabilize pipes	CPMs
Critical Path Monitor (CPM) Size	Big	Small	Small
CPM Design Effort	High (CPRs design)	Low	Low
Timing Margin	Large safety margin	No	Small safety margin
Suitability for High-Performance Processor Designs	Yes	No	Yes

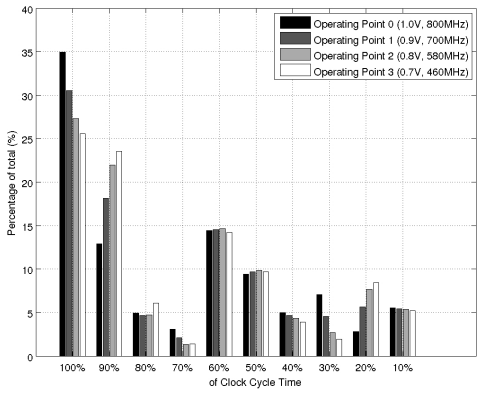

Fig. 7: A distribution of the processor logic paths as a percentage of the clock cycle time

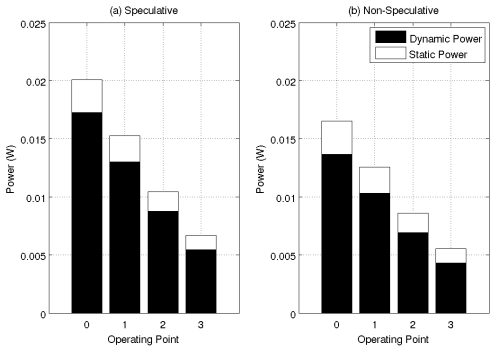

Fig. 8: Power consumptions of the speculative and non-speculative based processors at each operating point (SS, 125°C)

We estimated the delays along all of the timing paths in the processors and determined the potential critical paths by performing post-layout static timing analysis. Fig. 7 illustrates a distribution of the timing path delays as a percentage of the clock cycle time at each operating point. Here, the frequency value at each operating point does not include timing margins. For these critical paths, we replaced the original FFs in the netlist with each type of the target FFs: the Razor FFs (for a speculative based processor) and our non-speculative FFs (for a non-speculative based processor), respectively.

Processor power values were obtained and averaged from post-layout simulation using PrimeTime-PX. Fig. 8 shows the dynamic and static power of the speculative and non-speculative based processor in each operating point described in Fig. 7. The power consumption of the speculative based processor is slightly higher than that of the non-speculative one due to its additional pipeline stage and the high activity factor of the clock network.

In order to measure the path delay variations, we extracted SPICE netlists for the timing critical paths and performed Monte-Carlo simulations. The delay variation data obtained from the Monte-Carlo simulations were used to decide the error probability and value of Δ (the safety margin), which will be added to the clock cycle time of the non-speculative based processor. According to our Monte-Carlo simulation results, the highest 10-20% of the timing paths should be considered to be the potential critical paths.

B. Architectural simulations

In order to implement both the speculative based as well as the non-speculative based processors, we modified the simplescalar simulator [9]. For the speculative based processor simulator, we added an extra

pipeline stage called 'pre-commit'. This stage is non-speculative and is used to stabilize the data before committing it to the processor's memory. This simulator was modified to inject faults in the instruction execution. When an error was triggered, the processor pipeline was flushed and the program counter was rolled back to the first instruction in the commit stage. The simulator for the non-speculative based processor was similar to that of the speculative based one, except it did not have the additional pipeline stage and the mechanism to roll back.

In order to incorporate the effect of random timing violations, we injected random timing errors during the simulations. The error probability was computed using equations (1) and (2). As noted in the circuit-level simulation section, the mean and the variance parameters of the path delay variations were obtained from Monte-Carlo simulations.

$$P_{err}(t) = 1 - \phi\left(\frac{t-\mu}{\sigma}\right) = \frac{1}{2}\left[1 - erf\left(\frac{t-\mu}{\sigma\sqrt{2}}\right)\right] \quad (1)$$

$$erf(x) = \frac{2}{\sqrt{\pi}}\int_{-\infty}^{x} e^{t^2}\,dt \quad (2)$$

In equation (1), $P_{err}(t)$ is the probability of any critical path delay exceeding the value t. μ is the mean critical path delay, and σ is the standard deviation of the delay. Thus, a FF that latches data t seconds after the clock edge will have an error probability of $P_{err}(t)$.

The error rate for the main FF, which operates speculatively in the speculative based processor, was maintained at 0.04%. The error rate for the shadow latch, which pre-samples data in the non-speculative based processor, was also maintained at 0.04%. The safety margin between the shadow latch and the main FF in both methods was fixed in such a manner that, on average, the latching time for the main FF was the 6σ delay point. This ensures an almost zero error probability with respect to the operations of both processors. In addition, our simulation results indicate a 0.16% IPC overhead for the speculative based processor. This verifies the reported overhead in Razor [2].

In order to compare the two methods in the same condition, we applied the same clock frequency control scheme to both cases. As discussed in the previous section, the clock frequency of the non-speculative based processor is lower than that of the speculative based processor since the non-speculative technique requires a safety margin even if it is small. We estimated the total run-time of the two processors for the "SHA" Mibench benchmark program using frequency modulation. In order to evaluate the system, rather than using random instruction streams, we used a processor benchmark code with a sufficiently long run-time. The total run-time was computed by adding the clock cycle periods for all of the clock cycles. The clock period was incremented (or decremented) in steps of 10ps to control the processor frequency, and thus, the error rate.

C. Evaluation

TABLE II shows the energy consumption and EDP values obtained at different operating points and the percentage benefit in using our non-speculative technique over the speculative approach. As can be seen from the results, the non-speculative based processor fared better than the speculative based processor in terms of both energy and EDP. This is because the power overhead for the speculative processor offsets its benefit of short execution time due to higher operating frequency than that of the non-speculative processor.

Voltage (V)	Temp (°C)	Energy (x10^{-4}J) / EDP (x10^{-5})		
		Speculative	Non-Speculative	% Improvement
0.7	25	5.97 / 5.31	5.34 / 5.19	10.48 / 2.44
	75	5.99 / 5.36	5.36 / 5.23	10.52 / 2.47
	125	5.36 / 5.99	5.23 / 5.36	10.52 / 2.47
0.8	25	7.07 / 4.81	6.27 / 4.57	11.39 / 5.04
	75	7.15 / 4.92	6.34 / 4.68	11.35 / 4.97
	125	7.11 / 4.87	6.30 / 4.63	11.35 / 4.96
0.9	25	8.45 / 4.67	7.40 / 4.35	12.47 / 6.97
	75	8.51 / 4.74	7.45 / 4.40	12.52 / 7.09
	125	8.57 / 4.81	7.50 / 4.47	12.52 / 7.08
1.0	25	9.58 / 4.56	8.27 / 4.14	13.64 / 9.15
	75	9.65 / 4.64	8.30 / 4.63	13.56 / 8.98
	125	9.72 / 4.71	8.41 / 4.29	13.44 / 8.73

TABLE II: Energy consumption and EDP of the speculative and non-speculative based processors

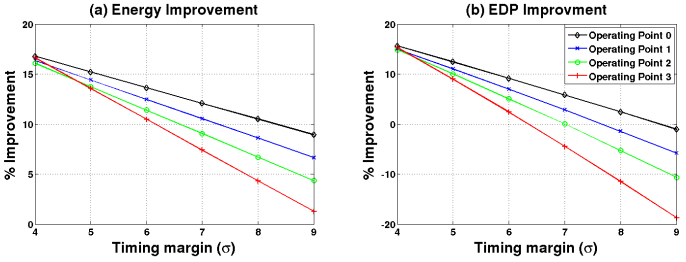

Fig. 9: Percentage improvement in (a) energy consumption and (b) EDP, over the Razor-based speculative technique

Our technique shows a 13.5% improvement in energy consumption and 9% improvement in EDP at the high voltage operating points. On average, our technique shows a 12% energy and 6% EDP improvement across all operating points. As shown in the table, the improvement is more pronounced at the higher voltage levels. As we go down to lower voltages, the variability of the critical path delays increases and a larger safety margin is needed to ensure a low system error rate. This increases the processor execution time reducing the energy efficiency of the processor.

Fig. 9 shows the percentage improvement in energy consumption and EDP over the Razor-based speculative technique as a function of the timing margin (in terms of path delay variance (σ)). As we increase the timing margin, the execution time for the applications increases, decreasing the percentage performance benefit of our technique. As previously noted, the path delay variation occurs more often for the lower voltages, leading to a faster drop in EDP for such cases.

V. Conclusions

In this paper, we have introduced a novel technique that utilizes a processor's excessive timing slack during run-time. Our double-sampling technique enables the processor to execute in a non-speculative fashion without requiring any modifications of the original processor architecture. This in-path non-speculative approach provides high timing-error prediction accuracy and is suitable for high-performance processor designs. On average, when compared to a Razor-based speculative technique, this technique results in 6% lower EDP and 12% lower energy consumption.

References

[1] K. Bowman, et al., "Circuit Techniques for Dynamic Variation Tolerance," *DAC*, pp. 4-7, 2009.

[2] D. Ernst, et al., "Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation," *MICRO*, pp. 7-18, 2003.

[3] D. Blaauw, et al., "Razor II: In Situ Error Detection and Correction for PVT and SER Tolerance," *ISSCC*, pp. 400-622, 2008.

[4] K. Bowman, et al., "Energy-Efficient and Metastability-Immune Resilient Circuit for Dynamic Variation Tolerance," *JSSC*, vol. 44, no. 1, pp. 49-63, 2009.

[5] A. K. Uht, "Uniprocessor Performance Enhancement Through Adaptive Clock Frequency Control," *TC*, vol. 54, no. 2, pp. 132-140, 2005.

[6] A. Drake, et al., "A Distributed Critical-Path Timing Monitor for a 65nm High-Performance Microprocessor," *ISSCC*, pp. 398-399, 2007.

[7] C. Lefurgy, et al., "Active management of timing guardband to save energy in POWER7," *MICRO*, pp. 1-11, 2011.

[8] J. Park, et al., "A Fast, Accurate and Simple Critical Path Monitor for Improving Energy-Delay Product in DVS Systems," *ISLPED*, pp. 391-396, 2011.

[9] T. Austin, et al., "SimpleScalar: An infrastructure for computer system modeling," *TC*, vol. 35, no. 2, pp. 59-67, 2005.

[10] M. Agarwal, et al., "Optimized circuit failure prediction for aging: Practicality and promise," *ITC*, pp. 1-10, 2008.