Spare RIBs: Redundant Intermediate Bitslices

David J. Palframan, Nam Sung Kim, Mikko H. Lipasti
Department of Electrical and Computer Engineering
University of Wisconsin–Madison
palframan@wisc.edu, nskim3@wisc.edu, mikko@engr.wisc.edu

ABSTRACT

Delay variations due to dopant fluctuations are expected to become more prominent in future technology generations. To regain performance lost due to within-die variations, many architectural techniques propose modified clocking schemes such as time borrowing or variable latency execution. As an alternative, we propose introducing redundancy along the processor datapath in the form of one or more extra bitslices. This approach allows us to insert a dummy slice into the datapath to avoid a slow critical path created by delay variations. Using the ALU critical path as an example, we demonstrate that our technique can reduce the delay penalty due to random variations by 10% with an area overhead of approximately 2%.

Categories and Subject Descriptors

B.8.0 [**Performance and Reliability**]: General; B.6.2 [**Logic Design**]: Reliability and Testing—*redundant design*

General Terms

Performance, Reliability

Keywords

Process Variation, Doping, Datapath, Bitsliced, Delay, Performance

1. INTRODUCTION

With each successive technology generation, the impact of process variation becomes increasingly significant. In addition to die-to-die (D2D) variations in device characteristics, current and future technologies are increasingly susceptible to within-die (WID) variations. The latter effect is of particular concern, since it can shift the distribution of fabricated chips, decreasing the mean operating frequency [12]. It has even been suggested that the impact of WID variation could counteract the potential gains of technology generations to come [2]. In this paper, we focus on random variation, a type of WID variation that arises primarily due to dopant fluctuations [11]. Random variations become particularly prominent at small device sizes, with the International Technology Roadmap for Semiconductors (ITRS) currently predicting as much as 81% variability in V_t for minimum-sized devices due to doping effects alone in the near future [1].

Many previous proposals to address increased delay due to WID variation are applicable to random variations, though they are arguably less efficient in this case. Many of these techniques are applied at the scope of execution units or pipeline stages. We note, however, that a single slow gate or delay fault in one of these units can limit a processor's maximum operating frequency. With this in mind, we propose a fundamentally different architectural technique intended regain performance lost due to random delay variations. Instead of considering pipeline stages, we propose thinking in terms of datapath bitslices. We create a wider datapath with extra bitslices, allowing us to leave some intermediate slices unused to hide the delay of the slowest paths. For the remainder of the paper, we will refer to these extra slices as Redundant Intermediate Bitslices (RIBs). In addition to avoiding complicated clocking techniques, this approach is not difficult to implement, since it involves widening datapath elements that we are already adept at designing. Note that unlike previous bitsliced architectures that primarily target hard defects [8], our technique does not use fuses or multiplexers between slices since this would increase delays. Using RIBs does offer significant opportunity for hard defect tolerance, but in this paper we focus on the performance implications. It is also worth mentioning that although we emphasize performance improvement at a fixed supply voltage, our concept could also be used to save power through voltage scaling. This paper includes the following contributions:

1. Description of our technique to increase processor clock frequencies when random variation is present;
2. Discussion of the addition of RIBs to various execution units and critical paths;
3. Analysis of this technique when applied to the ALU critical path.

The remainder of this paper is organized as follows: Section 2 discusses related work and motivates our proposal. Section 3 presents our idea for incorporating RIBs throughout the datapath, with particular focus on the ALU critical path. Section 4 includes results from applying our idea to the ALU critical path. Finally, Section 5 concludes the paper and discusses future directions.

2. BACKGROUND AND MOTIVATION

Historically, D2D variations were considered more important and prominent than WID variations. To regain yield lost due to D2D variations, binning can be applied in which parameters that affect the entire chip are modified. For instance, operating voltage can be increased or clock frequency decreased to compensate for slower logic. With subwavelength feature sizes, however, the level of WID variation

has increased. For instance, variation due to lithographic effects can affect gate channel lengths and impact regions of the die. This type of WID variation that affects multiple devices is termed systematic.

Though frequency binning can be applied to improve yield when systematic WID variations are present, it is less efficient since variation is no longer static across the chip. Unlike D2D variations, WID variations decrease the mean chip performance, since a chip can only be run as fast as its slowest path. As an example of this effect, Figure 1 shows the average increase in the delays of the ALU critical paths that we discuss later in this paper. These delay overheads are caused by random variations alone. This phenomenon has motivated a host of techniques that specifically aim to counter the effects of WID variation, including both systematic and random fluctuations in delay.

Techniques that target WID variation range from circuit-level to architectural. For instance, in the circuit domain recent work proposes modifying the body bias to compensate for varying threshold voltages [16]. Even though this technique can be applied to subsections of the die, it is best suited to compensate for D2D or systematic WID variations. Still other techniques propose clocking a processor at its intended frequency, and detecting any errors that may occur. In this case, the error rate can be decreased by speeding up or slowing down specific circuit paths [13].

As an alternative to simply disabling slow units, most architecture-level techniques compensate for slower stages or units with modified clocking schemes. One possibility is to allow variable-latency functional units, so that slow units can take additional cycles to complete [9, 10]. Relaxing constraints on operation latency, however, can potentially complicate scheduling and reduce IPC. To avoid these issues, recent work proposes slowing down the processor only when there is sufficient ILP to require both the fast and slower functional units [5]. Yet another alternative is time borrowing, in which specific clock signals can be skewed to allow a slow stage to use some of the time originally allocated to a faster stage [15].

Just as frequency binning may not be optimal in the case of systematic variations, we suggest that techniques to address systematic WID variations may not be ideal for random variations. Though various fine-grained body biasing techniques have been proposed as in [14], maintaining numerous voltage domains can be complicated. We note that our proposed technique is orthogonal to other techniques that involve clock manipulation. Other proposals demonstrate that multiple techniques can be effectively combined to reduce performance loss due to delay variations [9, 10]. RIBs can be implemented across the majority of the processor datapath. In addition, it does not require a single "fast" copy of a unit, and unlike time borrowing, it is not limited by pipeline loops [5].

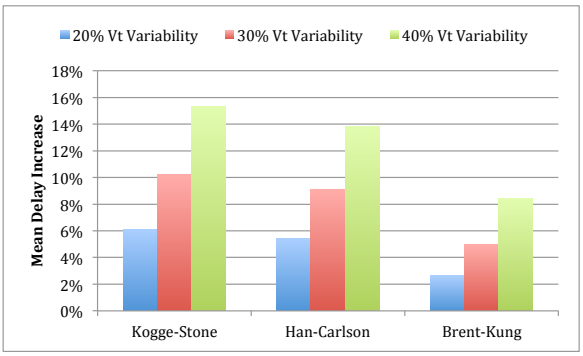


Figure 1: Increase in ALU critical path delays due to random variations.

3. INTERMEDIATE BITSLICES

Addressing random delay variations on a per-unit basis may not be the most efficient approach, since the increased delay may only be caused by a subset of gates. Instead of addressing delay variation on a per-unit basis, we propose a circuit-level bitsliced technique similar to those targeting hard defects [8, 4]. We create a wider datapath that can accommodate an unused RIB at an arbitrary offset. That is, if the original design calls for a 64-bit wide datapath, we could implement a 65-bit datapath, allowing a single bit space to be left at the same location in data registers, latches, wiring, and logic. Though the delay reduction benefits of this technique are derived from applying it to a critical path, keeping this unused bit at the same offset throughout the datapath has the benefit of simplicity with minimal additional multiplexers.

Using RIBs produces two different effects that can combine to decrease the maximum stage delay. Most obviously, once an offset for the unused slice is chosen, the output of any unit at this offset can be ignored while preserving correctness. Any critical paths that fan out only to the RIB therefore no longer contribute to the maximum delay. We also consider that since the input to any unit can be ignored at this offset, these input values can be overridden. Assuming that these inputs are on a critical path that fans out to one or more valid output bits, a secondary effect is that these forced inputs may render some gates static and remove them from the critical path. Of course, these signals must be overridden in a way that preserves or promotes the correct functionality of the unit. This effect is particularly relevant in adders, which we will discuss in the next section.

3.1 Application to High Performance Adders

The ALU and bypass loop can be considered one of the most important critical paths in a processor, since pipelining is usually not considered because of dependency chains. Addition is typically the operation with the longest delay in an integer ALU. Under this assumption, we consider only the adder and bypass network in our critical path analysis. Should a RIB be inserted into the datapath to compensate for delay variation in another unit, it is important that the adder function properly without imposing additional delay due to our modifications.

Correct addition of two numbers that contain an unused bit at the same offset is conceptually simple. The sole requirement is that carry signals must propagate across the gap to produce the correct sum—the output at the RIB offset does not matter. One way to enforce carry propagation is to insert multiplexers between each bitslice. Although this approach might be effective for simplistic carry propagate adders, it would be expensive in terms of area, complexity, and delay when applied to high performance parallel prefix adders that compute carries in parallel. In this paper, we

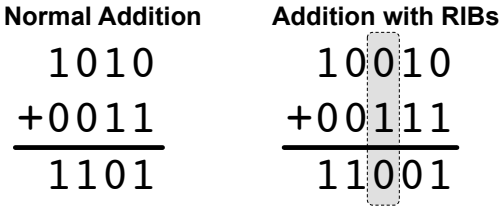


Figure 2: Conceptual example of binary addition with RIBs. Note that the carry traverses the extra slice.

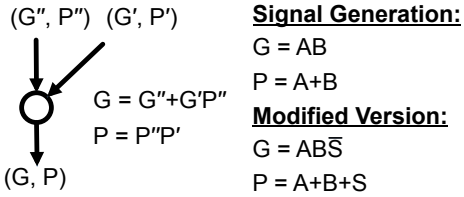


Figure 3: Basic prefix adder operation. Our proposal adds an additional skip input S to the initial propagate and generate logic for each bit position. A and B are operand bits at a given offset.

consider the Kogge-Stone [7], Han-Carlson [6], and Brent-Kung [3] adder architectures to explore trade-offs between complexity, delay, and the number of critical paths. Fortunately, the insertion of multiplexers in this manner would only be necessary to isolate a hard fault in the adder. Regardless of adder architecture and assuming that the bits in the operand RIBs can be manipulated, carries can be made to propagate by setting ones in the unused bits of one operand and zeros in the other, as demonstrated in Figure 2. In a prefix adder, overriding these bits is equivalent to fixing the carry computation signals that correspond to the RIB offset. For carries to traverse the extra bitslice, the *propagate* signal is asserted while the *generate* signal is cleared. To avoid adding extra delay, we propose the addition of an extra control signal input to the gates that generate these signals from the operands. This is a simple modification, since *propagate* is initially calculated with an OR while *generate* uses an AND gate. Figure 3 shows the logic used by the building blocks of parallel prefix adder, as well as our proposed logic modification.

With an adder that is able to accommodate inputs and outputs that have RIBs, it is possible to compensate for slow critical paths in other units. It is also possible to amortize a subset of delay faults within the adder and its critical path. As previously mentioned, since the *propagate* and *generate* bits that correspond to a RIB remain fixed, the delay to generate these signals no longer contributes to the critical path. In addition, slow logic that does not fan out to multiple bits, including some prefix nodes, final sum-calculating XORs, and bypass buffers can be ignored with appropriate RIB placement. Figure 4 shows an example of a Han-Carlson adder with two RIBs. Since we are overriding the *propagate* and *generate* signals at the input of the adder, any excessive delay in the nodes shown in green will no longer contribute to the critical path. Likewise, had the blue path been critical, it also would no longer impact the effective maximum delay.

In widening the datapath, it is clearly not possible to add an arbitrary number of extra bits without increasing delay. We note that for prefix adders, the number of extra bits that can be added for "free" in terms of performance varies with each adder type. For instance, a (power of two wide) Kogge-Stone adder can be widened by only a single bit before delay begins to increase. This is possible because although a wider Kogge-Stone adder requires more prefix nodes to calculate the carry-out bit, the carry-out does not require an XOR, as illustrated in Figure 4. Also, depending on the pipelining configuration, the carry-out is typically not bypassed, and instead sets a flag.

3.2 Logic Operations

We also propose the extension of this concept to other integer ALU logic that may not be on the critical path. Wider bitwise operations with RIBs, for instance, are trivial to implement. Shifts, on the other hand, do introduce additional complications. Without modification, any RIB in the shifted result would not be aligned with the rest of the datapath. To remedy this, we operate under our previous assumption that addition is the slowest ALU operation. The slack present allows the insertion of an additional multiplexer stage to "fix" the shifted result and return the unused bit to the proper offset.

Multiplication introduces a similar problem, since the partial products to be added are essentially shifted versions of an operand. Even if RIBs in all partial products could be aligned, there would be additional issues when summing, since the carry select addition would need to allow a non-predetermined number of carries to traverse a RIB. To avoid this complex issue, one solution is to add logic to "com-

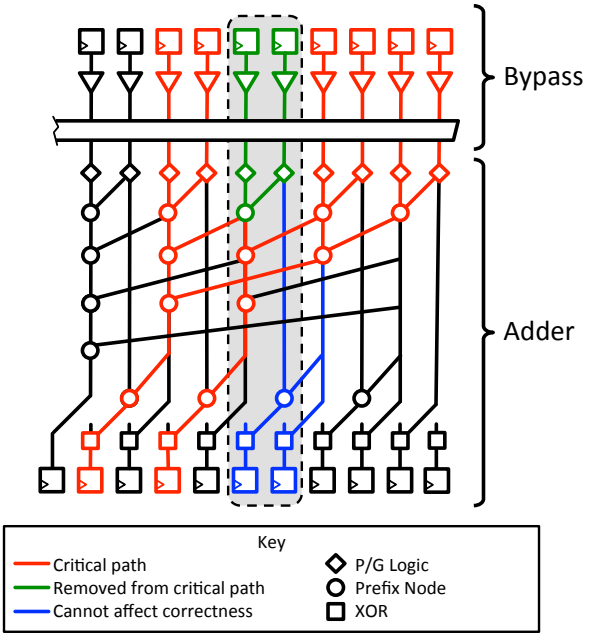
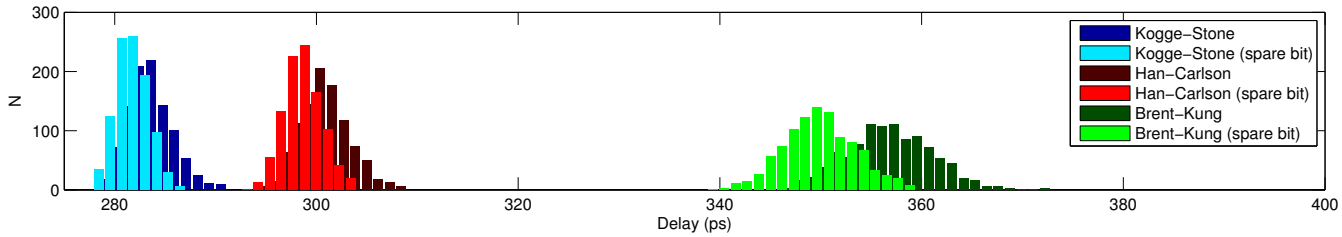
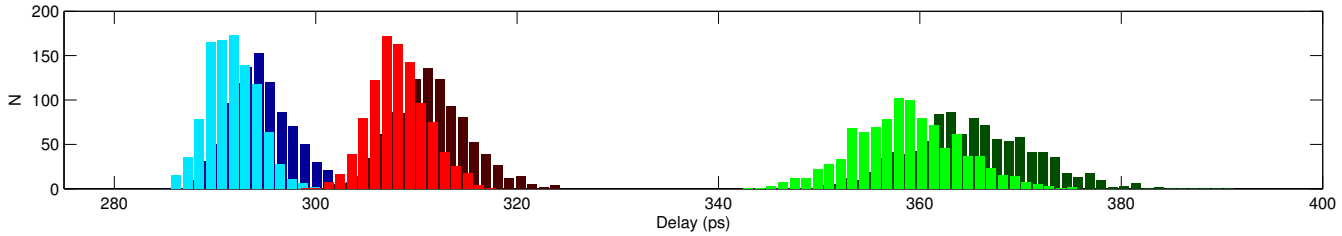


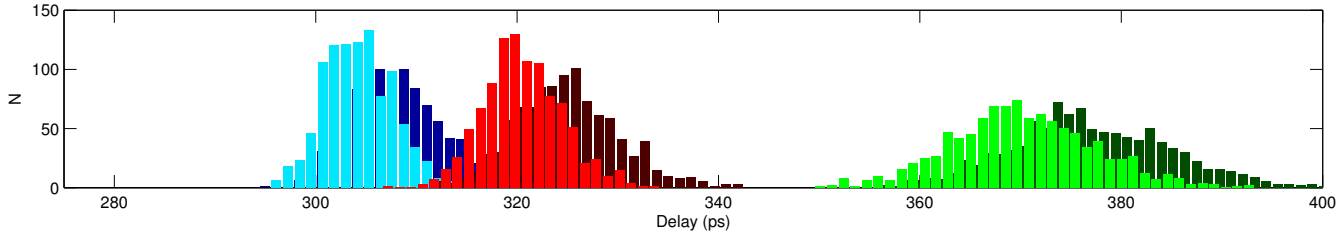
Figure 4: Example of adding two extra bits to an 8-bit Han-Carlson adder. The shaded region indicates the location of the two RIBs.



(a) Threshold voltage variability (3σ) set to 20%



(b) Threshold voltage variability (3σ) set to 30%



(c) Threshold voltage variability (3σ) set to 40%

Figure 5: ALU critical path delay histograms. Each distribution is comprised of 1000 circuits, each with randomized delays. Our scheme with one spare bitslice consistently shifts the delay distribution.

pact" the multiplicands and reinsert the gap into the product. We argue that this approach has minimal impact on performance, since the multiplier is already a pipelined unit and the extra delay required is minimal. Alternatively, dedicated "shuffle" logic in the multiplier could be avoided by leveraging the already-modified shifter unit. Custom "compact" and "expand" instructions could be executed prior to and following multiplication. The advantage of utilizing the modified shifter in this manner is that there is no additional multiplication latency if all RIBs are in the MSBs, but otherwise multiplication will take more clock cycles.

Though the analysis in this paper focuses on the integer datapath, floating point arithmetic plays an important role in many applications. Floating point logic introduces considerable additional complexity, though at its core it is fixed-point. For this reason, we defer consideration of floating point logic to future work.

3.3 Other Datapath Logic

This paper focuses on the ALU critical path, but the register file is another critical path that would likely benefit from our proposal. As noted in [9], the register file is particularly susceptible to variation, since the critical path is comprised of very few gates. Also, it only takes a single slow bitcell to increase the maximum delay of a bitline. Application to the register file is conceptually simple, since each bitslice is independent. We plan to undertake this analysis in future work.

It stands to reason that in our design, there must be some bound on where data is stored with RIBs and where it is stored in the traditional compacted form. Assuming that not every cycle in a multi-cycle L1 cache access is critical, there may be slack enough for a small amount of logic to make this transition between the datapath and L1 cache. Alternatively, our scheme could be merged with redundancy in the cache, and data saved including extra bits.

4. RESULTS

In this section, we present results from our simulations of random variations in ALU critical loops with different adder types. We begin by discussing our simulation methodology and framework. Finally, we present and discuss the results obtained with and without RIBs.

Since random variation affects each device separately, it is potentially very time consuming to simulate at a low level. To address this challenge, we characterize the delay of each gate type at a certain level of V_t variability using SPICE Monte-Carlo simulations. This initial step allows us to obtain a probability density function for the delay of each type of gate. All of our simulations use the 22nm predictive technology model [17]. We then synthesize a logic netlist for the desired adder and bypass network width. Adders are implemented with inverting CMOS gates, and our bypass network model includes latches, buffers, and multiplexers as shown in Figure 4. We choose the bypass wire capacitance such that bypass delay is roughly one third of the cycle time when a 64-bit Kogge-Stone adder is used.

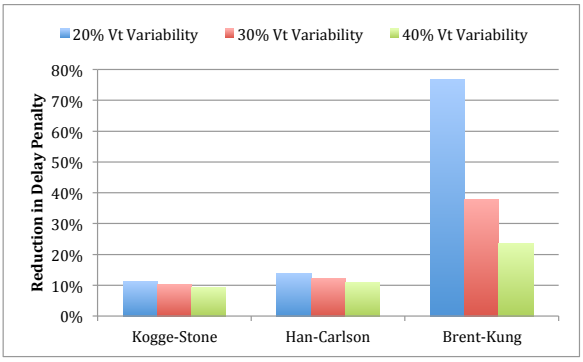


Figure 6: Reduction in the delay penalty imposed by threshold voltage variation.

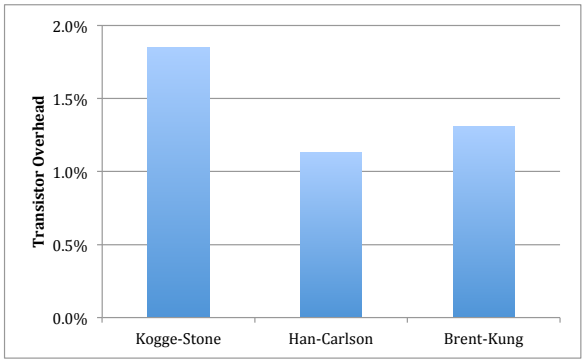


Figure 7: Transistor overheads for critical paths augmented with one bitslice.

To generate a circuit with randomized delays, we use the previously-obtained delay PDFs for each gate in the circuit. After all gate delays have been generated, we calculate the maximum delay of the circuit. If a RIB is to be included, the offset is found that will minimize the circuit's maximum delay. Note that for this experiment, we ignore both D2D and WID systematic variations, and focus only on random variations.

Figure 5 shows the delay distributions when 1000 critical path circuits are generated for each configuration. The baseline cases use a 64-bit wide datapath, and we add a single unused bitslice when applying our scheme. We repeat the experiment for 20%, 30%, and 40% variability in threshold voltages. We consider these levels of variability to be reasonable, since although the ITRS predicts quite high variability for minimum-sized devices, random variation is highly dependant on device size and not all devices will be minimum-sized. We first observe the expected shift in the mean delay with increasing variability, validating our initial motivation. We also see that with the addition of a spare bitslice, we are able to consistently regain some of the performance lost. In addition to decreasing the mean delay, adding a RIB also reduces the standard deviation of the delay distribution.

Figure 6 summarizes the benefits of applying our technique. Using circuit delays when no variation is present for reference, we see that we can reduce the delay overhead of random variation by about 10%. The Brent-Kung adder dominates the plot because it is actually a special case in this experiment. Since there is only a single critical path at the adder output, it is possible to decrease the adder delay by placing a RIB at this offset even in the absence of delay variations.

Figure 7 shows the estimated transistor overheads for the critical paths studied. Though this estimate excludes control logic, we believe that it demonstrates the overall low footprint of our design. Our scheme may also require testing logic to determine optimal RIB offsets, but we leave this to future work.

5. CONCLUSIONS

In this paper, we have presented a bitsliced technique to improve processor performance in the presence of within-die random variations. Specifically, we have performed analysis of this technique when applied to the ALU critical path consisting of the adder and bypass logic. Our technique is capable of reducing delay and has a low overhead. We expect that extending analysis to more of the datapath will show an increase in potential performance gains. In addition to this, we note that while outside the scope of this paper, our technique also has potential to improve yield lost due to hard defects.

6. REFERENCES

[1] International technology roadmap for semiconductors, 2009 edition. *Semiconductor Industry Association*, 2009.

[2] K. Bowman, S. Duvall, and J. Meindl. Impact of die-to-die and within-die parameter fluctuations on the maximum clock frequency distribution for gigascale integration. *Solid-State Circuits, IEEE Journal of*, 37(2):183–190, February 2002.

[3] R. P. Brent and H. T. Kung. A regular layout for parallel adders. *IEEE Trans. Comput.*, 31:260–264, March 1982.

[4] Z. Chen and I. Koren. Techniques for yield enhancement of vlsi adders. In *Proceedings of the IEEE International Conference on Application Specific Array Processors*, ASAP '95, Washington, DC, USA, 1995. IEEE Computer Society.

[5] E. Chun, Z. Chishti, and T. N. Vijaykumar. Shapeshifter: Dynamically changing pipeline width and speed to address process variations. In *Proceedings of the 41st annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 41, pages 411–422, Washington, DC, USA, 2008. IEEE Computer Society.

[6] T. Han and D. Carlson. Fast area-efficient vlsi adders. In *Proc. 8th Computer Arithmetic Symp*, pages 49–56, 1987.

[7] P. M. Kogge and H. S. Stone. A parallel algorithm for the efficient solution of a general class of recurrence equations. *IEEE Trans. Comput.*, 22:786–793, August 1973.

[8] R. Leveugle, Z. Koren, I. Koren, G. Saucier, and N. Wehn. The hyeti defect tolerant microprocessor: A practical experiment and its cost-effectiveness analysis. *IEEE Trans. Comput.*, 43:1398–1406, December 1994.

[9] X. Liang and D. Brooks. Mitigating the impact of process variations on processor register files and execution units. In *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 39, pages 504–514, Washington, DC, USA, 2006. IEEE Computer Society.

[10] X. Liang, G.-Y. Wei, and D. Brooks. Revival: A variation-tolerant architecture using voltage interpolation and variable latency. In *Proceedings of the 35th Annual International Symposium on Computer Architecture*, ISCA '08, pages 191–202, Washington, DC, USA, 2008. IEEE Computer Society.

[11] H. Mahmoodi, S. Mukhopadhyay, and K. Roy. Estimation of delay variations due to random-dopant fluctuations in nanoscale cmos circuits. *Solid-State Circuits, IEEE Journal of*, 40(9):1787–1796, September 2005.

[12] M. Orshansky, L. Milor, P. Chen, K. Keutzer, and C. Hu. Impact of spatial intrachip gate length variability on the performance of high-speed digital circuits. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 21(5):544–553, May 2002.

[13] S. Sarangi, B. Greskamp, A. Tiwari, and J. Torrellas. Eval: Utilizing processors with variation-induced timing errors. In *Proceedings of the 41st annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 41, pages 423–434, Washington, DC, USA, 2008. IEEE Computer Society.

[14] R. Teodorescu, J. Nakano, A. Tiwari, and J. Torrellas. Mitigating parameter variation with dynamic fine-grain body biasing. In *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 40, pages 27–42, Washington, DC, USA, 2007. IEEE Computer Society.

[15] A. Tiwari, S. R. Sarangi, and J. Torrellas. Recycle: pipeline adaptation to tolerate process variation. In *Proceedings of the 34th annual international symposium on Computer architecture*, ISCA '07, pages 323–334, New York, NY, USA, 2007. ACM.

[16] J. Tschanz, J. Kao, S. Narendra, R. Nair, D. Antoniadis, A. Chandrakasan, and V. De. Adaptive body bias for reducing impacts of die-to-die and within-die parameter variations on microprocessor frequency and leakage. In *Solid-State Circuits Conference, 2002. Digest of Technical Papers. ISSCC. 2002 IEEE International*, volume 1, pages 422–478, 2002.

[17] W. Zhao and Y. Cao. New generation of predictive technology model for sub-45nm design exploration. In *Proceedings of the 7th International Symposium on Quality Electronic Design*, ISQED '06, pages 585–590, Washington, DC, USA, 2006. IEEE Computer Society.