LATENCY ADAPTATION FOR MULTIPORTED REGISTER FILES TO MITIGATE THE IMPACT OF PROCESS VARIATIONS

Xiaoyao Liang and David Brooks

Division of Engineering and Applied Science
Harvard University
Cambridge MA 02141
{xliang, dbrooks}@eecs.harvard.edu

ABSTRACT

Design variability due to die-to-die and within-die process variations has the potential to significantly reduce the maximum operating frequency and the effective yield of high-performance microprocessors in future process technology generations. This variability manifests itself by increasing the frequency variance and decreasing the mean frequency of fabricated chips. In this paper we develop a model for the impact of variability on the performance of multiported SRAM-based structures such as physical register files which are key architectural components that may encounter variability problems. We find that naively resizing or increasing the access latency of these performance critical datapath resources can have frequency benefits, but may incur a significant IPC loss that limits overall system performance. We propose an extension to latency adaptation called port switching which more efficiently exploits the technique to remedy the IPC loss. We find that even under a conservative, worst case study, 18% mean frequency improvement with less than 5% IPC loss is possible for the 65nm technology node. Finally, we contrast the impact of die-to-die and within-die variations on chip performance and demonstrate that the proposed technique can compensate the frequency loss mainly due to within-die variations.

Key Words: Process variations, Multiported SRAM, Latency adaptation, Port switching

1. INTRODUCTION

Future advanced process technologies will continue to provide transistor density and speed improvements through aggressive feature scaling and novel device topologies. Unfortunately, chip designers will soon be forced to design with the expectation of significant variations in transistor feature sizes and threshold voltages due to sub-wavelength lithography and random dopant fluctuations. Process variations (PV) will manifest in several different ways – through random or correlated variations that may occur within a single die (WID: within-die variations) or across multiple dies (D2D: die-to-die variations) in a production run. The expected increase in within-die variations is especially troublesome for high-performance microprocessor designers because of cell stability and access time problems in SRAMs and the impact on machine cycle time due to increases in the number and criticality of key timing paths. Recent estimates suggest that process variability could impact performance by a full process generation [1].

While the last few years have seen an increased interest in developing probabilistic timing models and circuit-level techniques to address variability, there has been comparably little work at the microarchitectural level. However, key decisions that chip architects make to increase performance (e.g. sizing of architectural resources, etc.) has a substantial impact on the number and distribution of critical paths, and resource sizing benefits can be easily surpassed by the loss of chip frequency under variations. On the other hand, many techniques that can be used to combat process variations, such as chip-by-chip resource resizing in various microarchitectural structures, have inherent costs and must be applied with great caution. Because many of these approaches must be combined with modifications to the chip architecture, microarchitects will have an important role to play in designing future systems that are more tolerant to process variations [2].

Although techniques like adaptive body biasing (ABB) have potential for reducing the impact of variations [7], these approaches incur too much design overhead and complexity to be applied for fine-grain control. Also, large forward body bias incurs exponential leakage power penalties and can not be blindly used for thermal hotspots such as the register files as this will make these units hotter and perhaps lead to thermal runaway.

Approaches based on Razor latches [16] have the potential to correct circuit level errors with double latching. However, there may be difficulties in applying Razor to a multiported register file because there is a very high probability of experiencing variations. Razor may encounter errors and subsequent pipeline flushes very frequently which is very expensive for deeply pipelined, out-of-order machines.

Agarwal et al. propose to use post fabrication resource resizing to combat the impact of PV [3]. This work explores cache memory resizing and only considers within-die random threshold voltage variations. With the expected increase in PV, it is of significant interest to study more performance critical datapath components like register files (RF). Unlike caches, we find that resizing is not suitable for performance critical structures such as the physical register file because resource capacity is very important in these structures. We therefore propose *latency adaptation*. Unlike resource resizing which disables slow entries in the SRAM, we propose to use the slow entries but access them through two clock cycles while keeping the fast entries operating in a single cycle.

In this work, we study the joint impact of both systematic variations, like gate length, and random variations. We also consider differences in compensating for die-to-die and within-die variations due to the proposed techniques and ABB.

This paper takes several steps in the direction of PV-tolerant architecture design using the physical register file as a case study. Specifically, this paper makes three major contributions:

- We develop a performance variation model for multiported SRAM-based structure that considers both systematic and random effects. We model device gate length variations as well as threshold variations.
- We find that naive resizing or latency adaptation for critical datapath components will not bring a significant performance benefit because the recovered frequency is easily compromised by the IPC loss. However, architectural solutions can be used to reduce the IPC loss, and we show that these simple architectural modification can greatly improve the overall performance under PV.
- We show that this method can only help to improve the frequency mean but has little effect on frequency variance. Since within-die variations impact the frequency mean and die-to-die variations affect the frequency variance [1], this method is more effective to combat within-die variations. Fortunately, die-to-die variations can be easily compensated by ABB and we show that a combined approach with full chip ABB can give the best performance.

The remainder of this paper has four sections. Section 2 describes the details of our proposed performance model for the impact of variation on multiported SRAM structures. Section 3 shows the limitation of resizing or naive latency adaptation and proposes an architectural solution to reduce IPC loss while maintaining the frequency benefit. In Section 4, we present our simulation results. Finally, our work is summarized in Section 5.

2. PROCESS VARIATIONS MODEL FOR MULTIPORTED SRAM STRUCTURES

In this section, we introduce our modeling method used to predict the delay distribution of multiported SRAMs. We model both within-die and die-to-die variations with systematic and random effects. While the development of architectural process variation models is in its infancy, this model provides a preliminary model to allow architects to reason about variability in multiported SRAM structures.

2.1. Delay Variations Calculation

Fig. 1 shows a typical 2-read/1-write port SRAM with a detailed single cell design [15]. As indicated in [3], there are three possible types of failure in an SRAM cell. These failures are caused by the mismatch in the device parameters such as gate length and threshold voltage in cells. These device mismatches change the strength of different transistors resulting in different failure mechanisms.

- Read Stability Failure: This failure only occurs in traditional 6T SRAM design. During the read operation, the pull down transistors in the cross coupled inverters may not be strong enough to pull the bitline down to a low point so the read may unintentionally flip the value stored in the cell. In our multiported SRAM, we decouple the read bitlines with the internal cross coupled inverters, so as to effectively avoid this type of failure.
- Write Failure: When writing a "0" to a cell storing a "1", if the bitline pull down transistors are not strong enough, it is possible for the stored value not to be toggled resulting in a write failure. In general this type of failure is less frequent than the other failure mechanisms [3], and because register file arrays are usually much smaller than caches, we ignore write failure in this paper. If a chip has write failure, it will be discarded.
- Read Access Failure: This is the key failure mechanism in SRAM structures. Each time the SRAM is read, the read bitlines are required to discharge to a low voltage point. If the discharge is slow and cannot meet the cycle time, a read access failure occurs.

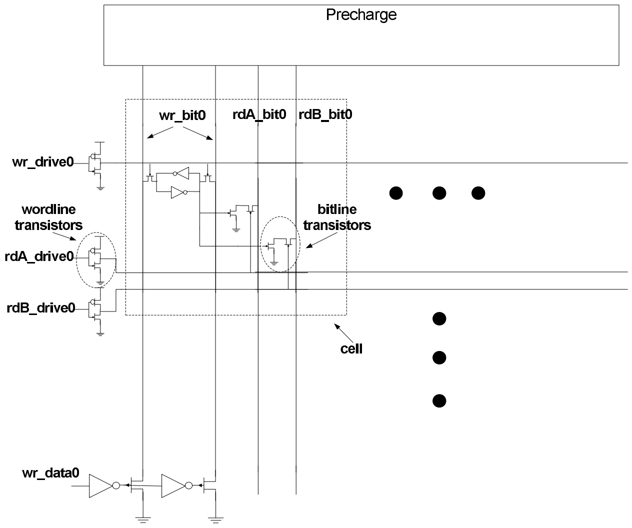


Figure 1: A typical 2-read/1-write SRAM structure.

The read access time is composed of the wordline charge delay and bitline discharge delay. Two wordline and two bitline transistors shown in Fig. 1 determine the access time, and variations in any of these transistors may change the access time. Furthermore, in a multiported SRAM, read ports attached to different entries may incur different access times due to potential variability of these four transistors. The worst case delay on any port/entry combination clamps the maximum SRAM frequency. Thus, multiported SRAM are very susceptible to PV because a single bad transistor can degrade the frequency for the entire structure.

Gate length (L) and threshold voltage (V_{th}) have been identified as the two main source of variations for the next several generations of process technology [4, 5]. In order to extract the delay variations due to the device parameter variations, we use a first order approximation which is widely used in statistical timing analysis [6, 13]. Eq. (1,2,3) shows the relation. The delay for a critical path is the summation of the path nominal delay (D_0) and the additional delay caused by parameter fluctuation of each device on the path, assuming n total devices. For small scale fluctuations, a first order linear approximation is accurate enough and the coefficients (a_k and b_k) can be curved fitted with circuit simulation. Fig. 2 shows the delay and gate length fitting curve for an SRAM, and the linear fit matches the HSPICE simulation for the range under consideration.

$$D_{path} = D_0 + \Delta D_{L_1} + ... + \Delta D_{L_n} + \Delta D_{V_{th_1}} + ... + \Delta D_{V_{th_n}} \tag{1}$$

$$\Delta D_{L_k} = \frac{\partial D}{\partial L_k} \times \Delta L_k = a_k \times \Delta L_k \tag{2}$$

$$\Delta D_{V_{th_k}} = \frac{\partial D}{\partial V_{th_k}} \times \Delta V_{th_k} = b_k \times \Delta V_{th_k} \tag{3}$$

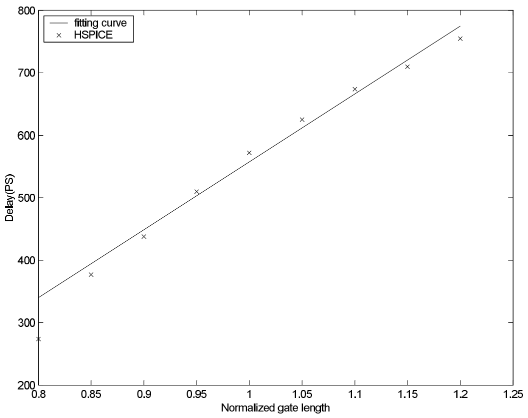


Figure 2: Curve fitting for HSPICE simulation for an SRAM.

2.2. Monte-Carlo Based Modeling

Since there are no widely accepted models for SRAM performance under PV, we adopt the Monte-Carlo method in our model which can be used to closely mimic real chip fabrication. For each transistor, we model the gate length as in Eq. (4) and threshold voltage in Eq. (5), where L_0 and V_{th_0} are the nominal values.

$$L = L_0 + \Delta L = L_0 + \Delta L_{D2D} + \Delta L_{WID} \tag{4}$$

$$V_{th} = V_{th_0} + \Delta V_{th} \tag{5}$$

For the gate length variations, there are die-to-die and within-die components. For die-to-die variations, we generate a random variable for each chip according to a normal distribution and use this variable for ΔL_{D2D} for every transistor on that chip. For within-die variations, there are both correlated and random components. To capture this effect, we use the method introduced in [6]. The SRAM area is divided into a multi-level quad-tree partitioning as shown in Fig. 3. For each quadrant we generate a random variable according to a normal distribution. The ΔL_{WID} of a transistor can be obtained by adding up all the random variables of the quadrants it belongs to. For example, the transistor (TX1) shown in the figure has within-die gate length variations of $\Delta L_{WID} = rand_{0,1} + rand_{1,1} + rand_{2,1}$. Therefore, TX1 and TX2 have strong correlation because they share the variable $rand_{0,1}$ and $rand_{1,1}$, while TX1 has less correlation with TX3, because they only share the variable $rand_{0,1}$.

Threshold voltage variations are mainly incurred due to the random dopant effect [5] as well as gate length variations. Since the impact of gate length on threshold has already been encapsulated in the delay-gate length fitting shown in Fig. 2, we model ΔV_{th} as a random variable which obeys a normal distribution [3] only due to random dopant effect.

We use Monte-Carlo simulation to generate all the random variables necessary in the model and generate the gate length and threshold voltage for each device on the delay path in the multiported SRAM. For gate length modeling, we apply 6 levels of quadrants with the top quadrant the entire SRAM and the bottom quadrant the devices. We then use the fitting curve and equations introduced in Section 2.1 to calculate the delay of all the possible paths in the SRAM. We simulate 2000 chips which is sufficient for our statistical analysis.

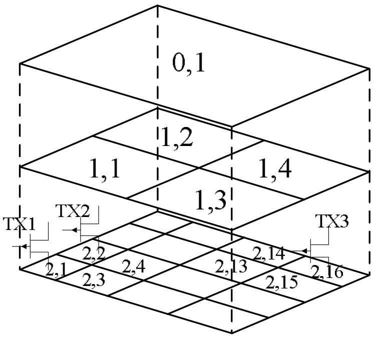


Figure 3: Multi-level partitioning for within-die variations.

3. VARIATION-AWARE ARCHITECTURE FOR PHYSICAL REGISTER FILE LATENCY ADAPTATION

In this section, we first study the frequency benefit of post-fabrication chip-by-chip tuning for several technology generations. We then investigate IPC-frequency trade-offs for the simplest approach. We propose *port switching*, a simple architectural solution to mitigate the IPC loss due to the additional latency added to the physical register file.

3.1. Frequency Benefit of Post Fabrication Tuning

Due to the random nature of PV, the small number of transistors (four in our proposed SRAM structure) on a single delay path, and the large number of critical delay paths (port number × entry number × bit number), the multiported SRAM is very likely to have variations problem in read access time. This requires that the chip mean frequency be significantly shifted to lower values to accommodate for the worst-case delay path [1]. To improve overall yield, it is necessary to apply post fabrication tuning [3]. Post fabrication chip-by-chip test (e.g. BIST [8]) can help identify the delay for each critical path (or entry) in the SRAM. Once we identify the slow paths, we can take either of the two approaches: resource resizing [3], which disables the entries that cannot meet the frequency target, or latency adaptation, which allows the slow entries to operate with two clock cycle latency while the fast entries maintain single-cycle latency. These approaches effectively increase the maximum operating frequency of the SRAM. The frequency will therefore be determined by the longest of the the remaining single-cycle slow delay paths.

To quantify the frequency improvement, we study a multiported physical register file. We build three, 80-entry × 64-bit physical register files on one chip, each with 4 read ports and 2 write ports. As in the Alpha 21264 processor [9], two register files are used for the integer pipe and one for the floating point pipe. Our simulations for 65nm and 100nm technology nodes are based on Berkeley Predictive Technology Models (BPTM) [10] and our 130nm simulations are based on TSMC foundry technology libraries. In our

SRAM fast entry ratio	100%	90%	80%	70%	60%	50%
Mean Freq. 130nm	1	1.06	1.08	1.09	1.10	1.11
Mean Freq. 100nm	1	1.07	1.09	1.10	1.12	1.13
Mean Freq. 65nm	1	1.08	1.10	1.12	1.14	1.15

Table 1: Chip mean frequency improvement after resizing.

Per-port fast entry ratio	100%	90%	80%	70%	60%	50%
Mean Freq. 130nm	1	1.08	1.11	1.13	1.14	1.16
Mean Freq. 100nm	1	1.09	1.13	1.15	1.17	1.19
Mean Freq. 65nm	1	1.11	1.15	1.18	1.20	1.22

Table 2: Chip mean frequency improvement after latency adaptation.

simulations, we assume $\sigma L_{D2D}/L_0 = 5\%$, $\sigma L_{WID}/L_0 = 5\%$. We assume $\sigma V_{th}/V_{th_0} = 10\%$. These assumptions are based on estimates for future variability forecast in [1, 3], and later in the paper we study the sensitivity of these assumptions.

Table 1 and Table 2 shows the chip mean frequency improvement for resizing and latency adaptation under three technology nodes. All the frequency values in the table are normalized to the frequency before tuning is applied. In both tables, 100% means no resizing or latency adaptation is applied. For resizing, a 90% fast entry ratio corresponds to disabling 10% of the entries with the slowest read access time from any port. The frequency after resizing is determined by the longest delay of the remaining 90% entries. For latency adaptation, 90% means for each port in the register file, the slowest 10% of the entries will be accessed in two clock cycles. Because different ports in the register file have different PV characteristics, it is quite likely that the same entry may be slow for one port and fast for another port. The final frequency of the chip is determined by the slowest path delay in the SRAM for the remaining 90% of the entries for all the ports. This conservative approach assumes that there must be 90% fast entries and 10% slow entries for each port. This is a worst case analysis of latency adaptation, because once the chip frequency is decided by the slowest port, it is likely that other ports have more then 90% fast entries. We will see that the worst case study places a lower bound on IPC in Section 4. By marking the slowest paths in the SRAM, both tuning methods increase the operating frequency for each chip and boost the mean frequency of all chips. Latency adaptation is more efficient, because it allows per port entry tunings, while resizing disables an entry if only one port reading that entry is slow, regardless of the other ports.

3.2. IPC Impact of Post Fabrication Tuning

Although post fabrication tuning can boost the chip frequency under PV, it also greatly affects the IPC of the system by changing the size or latency of performance critical datapath resources like the physical register file. The latency of the register file and the number of free registers available for renaming are key component for maintaining parallelism with out-of-order execution and these structures usually are performance bottlenecks [11]. Thus, a detailed IPC-frequency trade-off study must be carried out using $performance = IPC \times frequency$ as the metric to measure the effectiveness of the optimization.

Units	Config
Fetch/Decoder	4 inst
Issue Width	6 inst,4 int 2 fp
ROB	80 entries
Issue Q	20 int 15 fp
Integer RF	80 entries, 41 for renaming, 2-cy
Floating RF	72 entries, 41 for renaming, 2-cy
Exe Unit	int 4, fp 2
LD/ST Q	32 entries
L1 I-cache	64KB, 2-way, 1-cy
L1 D-cache	64KB, 2-way, 3-cy
L2 cache	4MB, 4-way, 7-cy

Table 3: Baseline machine configuration.

In our processor architecture for resizing, once an entry is identified as slow and disabled, it will not be used to ensure all the remaining entries can run at fast speed. The rename stage will not map any architecture registers to the slow physical register entries. This shrinks the effective size of the register file. For latency adaptation, whenever a slow entry for a certain port is read, the pipe that port belongs to must stall for one clock cycle for register file reading. This adds an additional bubble into the pipeline and thus decreases IPC. For IPC estimation we use *sim-alpha*, a hardware-verified Alpha 21264 simulator [12] with a 21264-like configuration listed in Table 3. The physical register file is pipelined into 2 stages for fast clock rate, with the first stage (RF decode) handling address decoding and second stage (RF read) performing SRAM array access.

We use 11 SPEC2000 benchmarks for our simulations and report average IPC. Phansalkar et al. point out that eight representative benchmarks can cover the SPEC2000 test suites [14]. We use these eight benchmarks (applu, gzip, equake, fma3d, mcf, twolf, mesa, gcc) as well as three randomly picked benchmarks (vortex, swim, perlbmk). For each benchmark, we simulate 100 million instructions.

Table 4 and Table 5 show the IPC impact due to register file resizing and latency adaptation, all normalized to the machine without tuning. From these results we see that the IPC drops significantly when many entries are marked slow or require two-cycles for the SRAM access. Both table also show the chip mean performance for three technology nodes, considering the effect of both frequency benefit and IPC loss. Unfortunately, the frequency benefit due to tuning has been largely compromised by the large IPC drop. Latency adaptation in this study can improve the mean chip performance by 4% for the 65nm technology node, while resizing always degrades performance. The results show that naive approaches for IPC critical datapath resources do not yield significant improvement for overall chip performance.

3.3. Port Switching to Reduce IPC Loss

The measured IPC loss is primarily due to the extra stall cycle caused by latency adaptation. Due to the random nature of de-

SRAM fast entry ratio	100%	90%	80%	70%	60%	50%
IPC	1	0.87	0.61	NA	NA	NA
Mean Perf. 130nm	1	0.92	0.66	-	-	-
Mean Perf. 100nm	1	0.93	0.66	-	-	-
Mean Perf. 65nm	1	0.94	0.67	-	-	-

Table 4: IPC impact and chip mean performance after resizing, "NA" means the IPC simulation fails for that node because of deadlock, extremely low IPC values/long simulation times.

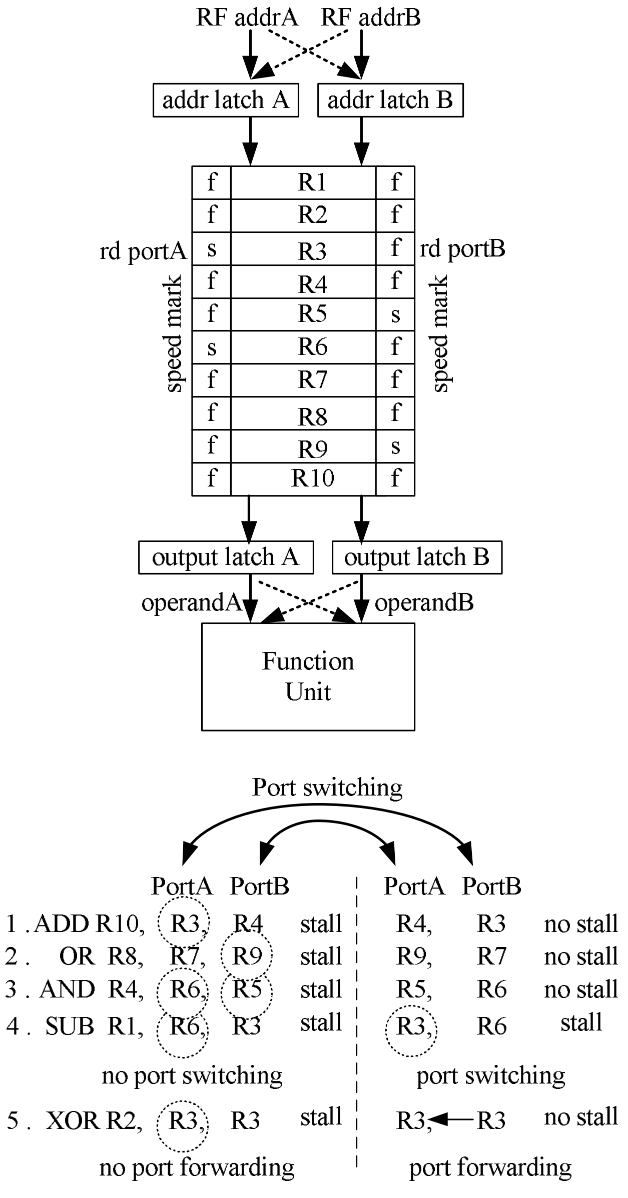


Figure 4: Port switching to reduce stalls. Circled registers means the register causes a stall for reading.

Per-port fast entry ratio	100%	90%	80%	70%	60%	50%
IPC	1	0.94	0.90	0.84	0.83	0.76
Mean Perf. 130nm	1	1.02	1	0.95	0.95	0.88
Mean Perf. 100nm	1	1.02	1.02	0.97	0.97	0.90
Mean Perf. 65nm	1	1.04	1.04	0.99	1	0.93

Table 5: IPC impact and chip mean performance after latency adaptation.

vices on different ports, it is possible that for the same entry, different ports may have different read-access delays, particularly under strong device variations. We leverage this characteristic with *port switching*. With this technique, we propose to opportunistically switch from slow ports to fast ports for register file entry reading in order to greatly reduce the IPC loss.

Fig. 4 provides a simple example to demonstrate the approach. In this example, there are 10 entries in the register file with 2 read ports. For each port, the two slowest entries are marked. The label 'f' and 's' identifies the fast and slow entries for each port, e.g. portA reading R3 and R6 is slow, while portB reading R3 and R6 is fast. Without port switching, every instruction in the code sequence in Fig. 4 incurs a stall, because each instruction has operands that require reading from the slow entries from either or both of the 2 ports. However, if we can identify the problem and steer the register access from the slow port to the fast port before entering the register file read stage, we can avoid stalls. For the first ADD instruction, if we use portA to read R3, it is slow and needs to be stalled, although portB reading R4 has no problem. Instead, we can use portA to read R4 and portB to read R3, since both ports are fast, and hence we can eliminate the stall. By using port switching, we can avoid most of the stalls in the sample code sequence. Port switching cannot help the fourth instruction because there is always one port reading a slow entry whether switching or not. For the fifth instruction, switching cannot help. But because the two operands are reading the same register, and since portB can read faster, it is possible to forward the data to portA so that the stall is avoided.

Some architectural modifications are necessary to implement port switching. First, we assume that the SRAM port speed information can be collected off-line (during BIST) and that the chip operating frequency is pre-selected. The information is configured into a ROM at test time. The per-port speed information will accompany the standard register entry IDs. For example, with our 80-entry register file with 4 read ports, we use an extended address of 11 bits, with the lowest 7 bits storing the standard physical RF entry number and the highest 4 bits storing the speed information (fast or slow) for each of the four ports for that entry. When a source operand is renamed in the renaming stage, it grabs the physical RF address as well as the four port speed bits (from the ROM) and creates an address bundle. If the address is not a true register file address, (normally used for non-register file reading immediate operands and other special registers), all the ports speed bits will be set to fast. The 11-bit address bundle is propagated through the pipeline.

Fig. 5 shows the hardware implementation details. In this figure, we assume port0 and port1 provide two operands for a functional unit. In the RF decode stage, the lowest 7 bits are decoded as

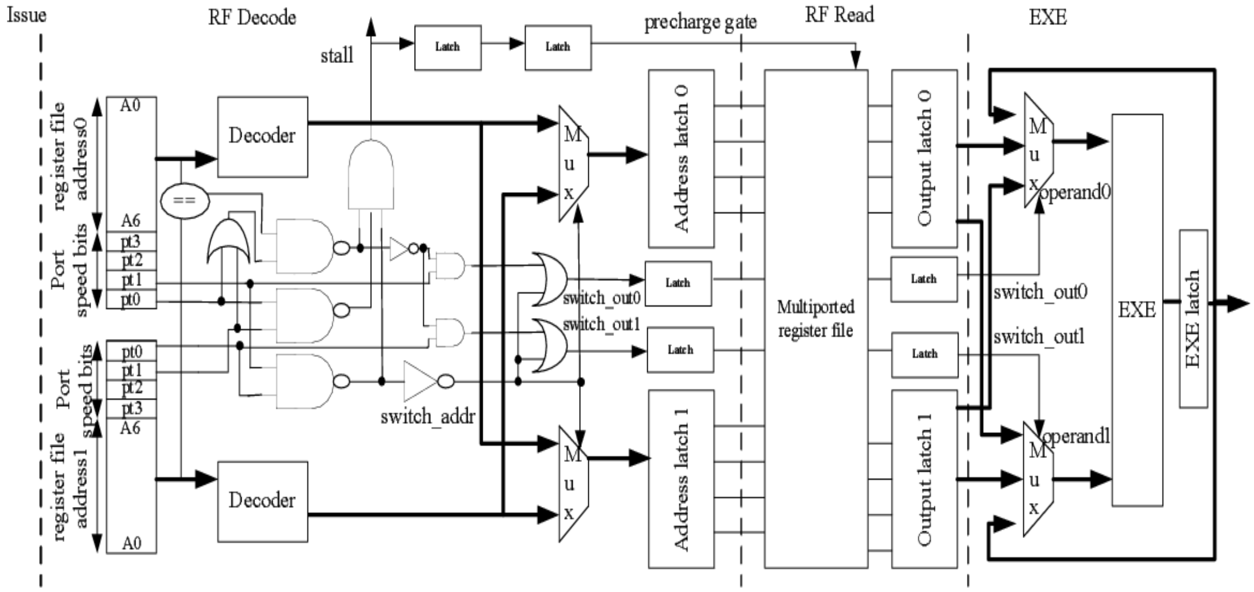
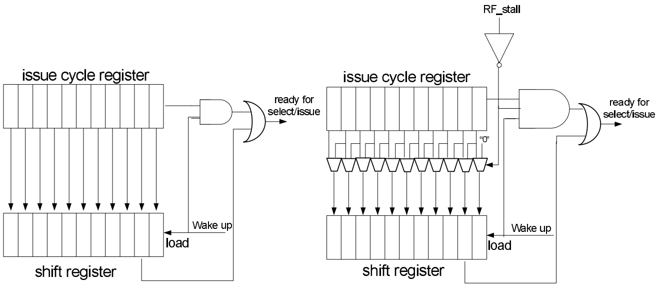


Figure 5: Hardware implementation for port switching, a '1' in bit pt0 means port0 reading this address is fast, while '0' means slow.

usual. However, we also check the four port speed bits. If address0 can run fast with port1 and address1 can run fast with port0, port switching occurs. The "switch_addr" signal controls the MUX to properly steering the decoded address to the address latches, which will then be used to access the register file in the next RF read stage. At the same time, the logic compares the two addresses and decides if a forwarding operation is necessary. The forwarding and switch signal will be combined to generate "switch_out0" and "switch_out1" signals and these signals will be latched twice to control the output multiplexers and switch the operands back or forward the operands before execution. Since switching cannot avoid all the stalls, stall detection logic is implemented. If any of the two ports is slow and there is no switch or forwarding, a stall signal is generated. The stall signal passes to the previous issue stage so that all dependent pending instructions must wait for an extra cycle and no instruction can be issued to this pipe for the next clock cycle. In order to allow the register file to take two clock cycles for access, the stall signal must be latched twice to prevent precharge for the second cycle of the access.

All the switch, forwarding, and stall signals are generated in parallel with register address decoding so they will not add to the critical path delay. The output multiplexers before the execution stage can be merged with the execution bypassing multiplexers to minimize impact on the execution delay. Only the multiplexers in the RF decoder stage may affect critical datapaths. To keep this overhead small, we limit the use of switching. Only two ports connecting to the same functional unit can be switched or forwarded requiring a small 2-1 MUX.

The RF stall signal is generated, if necessary, during the beginning of the cycle after the instruction is issued. So all dependent instructions, including back-to-back instructions, will see the stall signal at the beginning of their schedule cycle at the same time they are awakened and deciding whether to issue. Thus, there will not be any speculative issue problems requiring replays. There will be some necessary modification to the scheduling logic. Since the issue logic can see the stall signal at the same time the instructions are awakened, the issue logic should take this stall into consideration. Fig. 6 shows the hardware modification of the scheduling logic. In a conventional design [17], shown in Fig. 6a, each instruction operand in the issue queue includes an issue cycle register that records the issue cycle after the operand wakes up. For example, "111111" in the issue cycle register refers to a back-to-back instruction that can be issued the same cycle when it is waken up. A string of "111110" means that the instruction can be issued one cycle after wake-up. When the instruction sees the wake up signal, the information in the issue cycle register is loaded to a shift register, and the shift register will shift right to decide when the instruction is ready for selection/issue. For back-to-back instruc-



a. Conventional issue/select logic

b. modified issue/select logic with RF stall

Figure 6: Scheduling logic modifications for RF stall.

tions, the wake up signal requires a logical AND with the issue cycle register to ensure consecutive issue. The modified version of this logic that includes the register stall signal is shown in Fig. 6b. If there is no RF stall, everything behaves exactly the same. If there is a stall, the back-to-back instruction is gated from issuing during the wake-up cycle. Also, the shift register is loaded with a biased version (left shift 1 bit) of the issue cycle register to count in the extra one cycle delay for the register file access.

The RF stall signal is generated early in the phase of the issue cycle, while the wake up signal is usually performed near the end of the phase because of the need to perform a CAM-lookup for address matching. So the stall gating logic will not add much to the issue critical path delay (only requiring a 2-input AND to change to a 3-input AND). The additional multiplexers before the shift register is not on the issue critical path so has no impact on the delay.

4. DETAILED SIMULATION RESULTS AND SENSITIVITY ANALYSIS

In this section, we highlight the performance benefits of port switching, discuss the impact of the technique on the chip performance distribution, and perform sensitivity analysis.

Our simulations shows that the IPC loss due to RF latency adaptation can be greatly reduced by port switching. We plot SPEC2000 benchmark results in Fig. 7. This figure shows the effectiveness of port switching with 70% fast entries per port, and we see that port switching only incurs a small IPC drop compared with the standard register file.

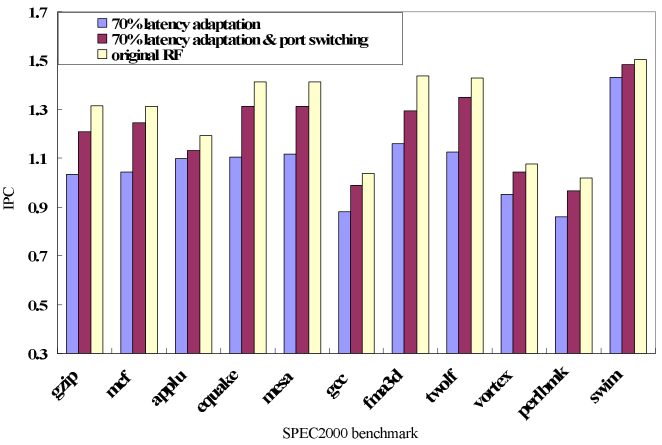


Figure 7: SPEC2000 benchmark results.

Table 6 shows the average IPC with register file latency adaptation utilizing port switching and the chip mean performance improvement. With port switching and a 70% fast entry ratio per port, only a 5% IPC drop is observed. Because port switching reduces the IPC loss, the frequency benefit translates directly into a significant increase in system performance. The results demonstrate that the optimal configuration includes 70% -80% fast entries depending on the process technology. Essentially this means that for these configurations we can avoid the slowest 20%-30% entries for each port and increase frequency. The chip mean performance is significantly improved compared with naive latency adaptation, and a 12% mean performance improvement can be obtained for the 65nm node for the worst case.

Per-port fast entry ratio	100%	90%	80%	70%	60%	50%
IPC	1	0.99	0.97	0.95	0.89	0.87
Mean Perf. 130nm	1	1.07	1.08	1.07	1.01	1
Mean Perf. 100nm	1	1.08	1.10	1.09	1.04	1.04
Mean Perf. 65nm	1	1.10	1.11	1.12	1.07	1.06

Table 6: IPC for latency adaptation with port switching and chip mean performance improvement.

4.1. Performance Distributions and Full-chip ABB

Fig. 8 shows the chip performance distribution before and after RF latency adaptation for the 70% point, including port switching, using 65nm technology. The performance mean as well as the whole distribution is shifted to the faster end. So for a certain performance point (e.g., the “Performance 1” point, shown by the dashed line), the yield increases, from 25% to 83%. Alternatively, for a certain yield requirement (90% yield in the figure), the performance point grows from 0.90 to 0.98.

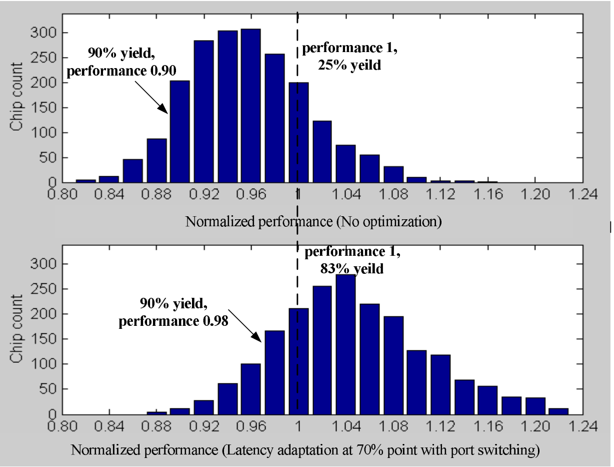


Figure 8: Chip performance distribution before and after latency adaptation at the 70% point with port switching.

Fig. 8 also shows that although latency adaptation can help to improve the chip frequency mean, it cannot help to reduce the frequency variance. This can be seen in the figure as the distribution shifts to higher performance, but the variance stays almost unchanged. Bowman et al. show that within-die variations directly impact the frequency mean and die-to-die variations impact the frequency variance [1]. This can be used to explain this phenomena. The SRAM tuning is carried out chip-by-chip so it can only reduce the slow paths on a single chip. If we assume there is no within-die variations but only die-to-die variation, then all the paths on a single chip share the same variation and there is no delay difference between paths, and this approach will not have any benefit.

Thus, latency adaptation can only mitigate within-die parameter variations. As forecast in [1], within-die variations will become the dominant factor for the next several generations of technology. Die-to-die variations, on the other hand, can be overcome by applying full chip adaptive body bias [7]. ABB cannot be applied at the fine granularity required to mitigate within-die variations, such as the entry-to-entry performance variations experienced within multiported SRAMs, without incurring substantial design complexity and area overheads.

We propose to combine full chip ABB with latency adaptation to achieve the best performance improvement under strong PV. In our simulations, ABB will be used to correct die-to-die variation by sampling different corners on a chip and comparing the average delay with nominal. If the delay is larger (meaning die-to-die device variations degrade chip frequency), full chip forward body bias is used to accelerate the chip. If the delay is smaller (meaning die-to-die device variations increase the frequency), full chip reverse body bias is used to slow down the chip to save leakage power. Full chip ABB makes all the chips behave more uniformly and eliminates the die-to-die variations. Fig. 9 shows the approach. Full chip ABB can largely reduce the variance caused by die-to-die variations and make the distribution sharper while latency adaptation shifts the mean to the higher performance end reducing the impact of within-die variations. The combined approach helps to increase the 90% yield performance point from 0.9 to 1.03. Compared with the result in Fig. 8, the additional improvement (from 0.98 to 1.03) is due to the application of full chip ABB. Also, now the yield for the "Performance 1" point is 100%, compared with 83% in Fig. 8 without full chip ABB.

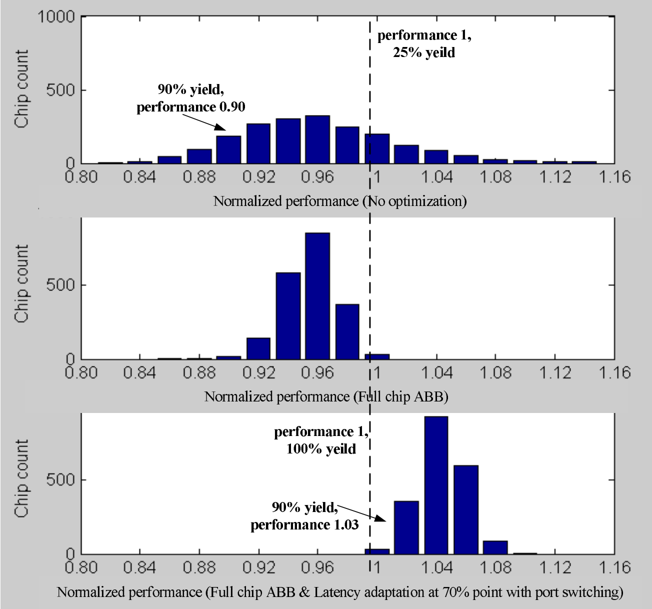


Figure 9: Chip performance distribution with full chip ABB and latency adaptation at 70% point with port switching.

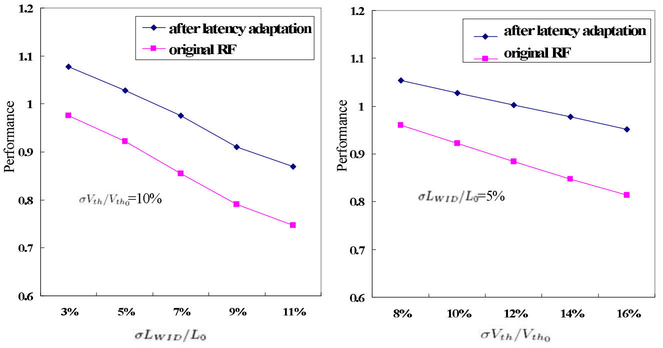


Figure 10: Mean chip performance under different variation scenarios. Assumes full chip ABB and latency adaptation with a 70% adaptation setting with port switching.

4.2. Sensitivity to Amount of PV

All the previous simulations assume that $\sigma L_{WID}/L_0 = 5\%$ and $\sigma V_{th}/V_{th_0} = 10\%$. We now perform a sensitivity analysis on these within-die variation parameters by sweeping $\sigma L_{WID}/L_0$ from 3% to 11%, and $\sigma V_{th}/V_{th_0}$ from 8% to 16%. We believe the range swept can cover most of the variation situations for the next several technology generations. We also assume full chip ABB can correct die-to-die variations. In Fig. 10, we plot the mean performance of the chips at the 70% adaptation point with port switching. The simulation is at the 65nm technology node. The approach works almost equally well for different variation scenarios. The plot shows that the approach is more effective at correcting random V_{th} variations, because the more random the critical delay paths appear to be, the more efficiently the approach can identify and correct slower paths. While within-die gate length variations have correlated components, the approach is less effective at correcting large scale systematic variations. In large SRAM structures, threshold voltage variations due to random dopants are the key source of failure and this approach is well-suited to reducing the impact of these variations.

4.3. Impact of Pessimistic Assumptions

As mentioned before, we base our simulations on a worst-case study. In particular, we always assume a fixed and equal fast-slow entry ratio for each port in an SRAM. For example, with an 80-entry SRAM with 70% fast ratio, we always assume 56 fast entries and 24 slow entries for each port. But in our Monte-Carlo simulation, we find that there is usually only one port that is reduced to this number of fast entries while the other ports can have more fast entries once the chip frequency is determined. Table 7 shows the true fast entry number for three randomly picked chips as well as our worst case assumption. For each chip shown in the table, only 1-2 ports have a 70% fast ratio while other ports can have more fast entries. We simulated the IPC for the three chips and show the results in Fig. 11. It is clear that our worst case assumption places a lower bound on the IPC values for all the chips. Since it is not possible to simulate IPC for each chip, our current results are a conservative estimation in evaluating the efficiency of latency adaptation. Most chips should have higher IPC and thus higher performance than reported in this paper. In our future work,

Port	chip1	chip2	chip3	worst case
RF1, Port0	56	73	61	56
RF1, Port1	57	74	63	56
RF1, Port2	59	68	68	56
RF1, Port3	56	72	70	56
RF2, Port0	75	56	61	56
RF2, Port1	72	60	65	56
RF2, Port2	72	57	57	56
RF2, Port3	71	56	68	56
RF3, Port0	62	69	56	56
RF3, Port1	67	62	64	56
RF3, Port2	60	67	62	56
RF3, Port3	66	66	63	56

Table 7: Number of fast entries for each port for three random chips and the worst case. Each register file is an 80-entry, 4-read port SRAM.

we plan to develop methods to better estimate the true IPC without exhaustive simulation.

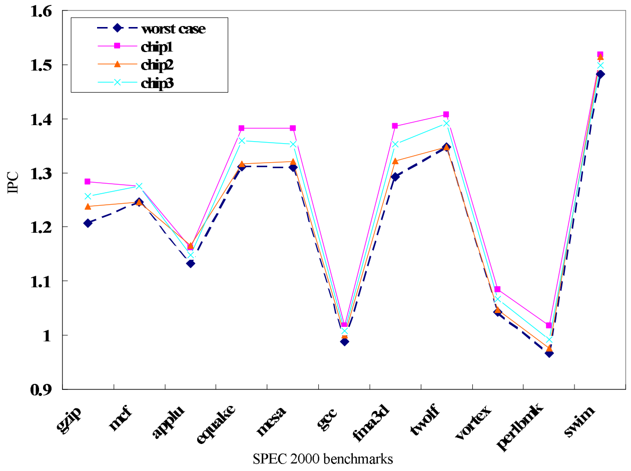


Figure 11: IPC for three chips and for the worst case. The worst case study places a lower bound on the IPC for all chips.

5. CONCLUSION

In this paper we study an approach to mitigate the impact of process variations within the design of multiported register files. We describe a modeling method to consider PV for multiported SRAM structures. We find naive resizing or latency adaptation cannot help significantly because of large IPC loss, and we propose to use port switching to reduce the IPC loss and boost the overall system performance. We study the impact our approach has on die-to-die variations and within-die variations and find that the approach is well-suited to random, within-die variations. Combining this technique with full chip ABB can give satisfactory results even under significant PV.

This paper implicitly assumes that the register file is the only source impacting the overall chip frequency. Our future research plans to explore whether similar techniques can be used for other delay critical components like the issue queue and other storage units. This work effectively reduces the PV impact on the physical register file and can be combined with other approaches like cache resizing [3]. We hope that this work can encourage architects to begin to integrate the impact of PV and associated optimization techniques into the early architecture design phase.

6. REFERENCES

[1] K. A. Bowman and S. G. Duvall and J. D. Meindl. "Impact of Die-to-Die and Within-Die Parameter Fluctuations on the Maximum Clock Frequency Distribution for Gigascale Integration", *IEEE Journal of Solid-State Circuits, Vol. 37, N0. 2*, February 2002.

[2] S. Borkar. "Microarchitecture and Design Challenges for Gigascale Integration", *Keynote Speech, 37th International Symposium on Microarchitecture*, December 2004.

[3] A. Agarwal, B. C. Paul, Hamid Mahmoodi, etc. "A Process-Tolerant Cache Architecture for Improved Yield in Nanoscale Technologies", *IEEE Transactions on Very Large Scale Integration Systems, Vol. 13, NO. 1*, January 2005.

[4] S. G. Duvall. "Statistical Circuit modeling and optimization", *5th International Workshop Statistical Metrology*, June 2000.

[5] K. A. Bowman, X. Tang, and J. C. Eble, etc. "Impact of Extrinsic and Intrinsic Parameter Fluctuation on CMOS Circuit Performance", *IEEE Journal of Solid-State Circuits, Vol. 35*, August 2000.

[6] A. Agarwal, D. Blaauw, S. Sundareswaran, etc. "Path-based Statistical Timing Analysis Considering Inter and Intra-die Correlations", *In Proceedings of TAU*, June 2002.

[7] J. W. Tschanz, J. T. Kao, S. G. Narendra, etc. "Adaptive Body Bias for Reducing Impacts of Die-to-die and Within-die Parameter Variations on Microprocessor Frequency and Leakage", *IEEE Journal of Solid-State Circuits, Vol. 37*, November 2002.

[8] M. H. Tehranipour, Z. Navabi, S. M. Falkhrai. "An efficient BIST method for testing of embedded SRAMs", *Proceedings of IEEE International Symposium on Circuits and Systems, Vol. 5*, May 2001.

[9] R. E. Kessler. "The Alpha 21264 microprocessor", *IEEE/ACM International Symposium on Microarchitecture*, 1999.

[10] Berkeley Predictive Technology Model, UC Berkeley Device Group. http://www-device.eecs.berkeley.edu/ ptm/

[11] T. Monreal, A. Gonzalez, M. Valero, etc. "Delaying physical register allocation through virtual-physical registers", *IEEE/ACM International Symposium on Microarchitecture*, November 1999.

[12] R. Desikan, D. Burger, etc. "Sim-alpha: a Validated, Execution-Driven Alpha 21264 Simulator", *Technical Report TR-01-23, Department of Computer Sciences, University of Texas at Austin*, 2001.

[13] M. Orshansky, L. Milor, P. Chen, etc. "Impact of Spatial Intrachip Gate Length Variability on the Performance of High-Speed Digital Circuits", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 21, No. 5*, May 2002.

[14] A. Phansalkar, A. Joshi, L. Eeckhout, etc. “Measuring Program Similarity: Experiments with SPEC CPU Benchmark Suites”, *IEEE International Symposium on Performance Analysis of Systems and Software* , March 2005.

[15] N. H. E. Weste and D. Harris. “CMOS VLSI Design: A Circuits and Systems Perspective, Third Edition ”, *Addison Wesley ISBN: 0-321-14901-7*, 2004.

[16] D. Ernst, N. S. Kim, etc. “Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation”, *IEEE/ACM International Symposium on Microarchitecture*, 2003.

[17] J. Stark, M. D. Brown, etc. “On Pipelining Dynamic Instruction Scheduling Logic”, *IEEE/ACM International Symposium on Microarchitecture*, 2000.