

Chapter 2

Instruction-Level Tolerance

Abstract Microprocessors manufactured in nanometer processes are beginning to see variation in timing performance of individual instructions. This chapter considers challenges and opportunities in identifying this variation and methods to combat it for predicting and preventing the timing errors in single-core architectures. We start from instruction-level which is the finest granularity to present the processor functionality. We introduce the notion of instruction-level vulnerability (ILV) to parameterize this variation and use it for architectural and compiler optimizations. To compute ILV, we quantify the effect of voltage and temperature variations on the performance and power of a 32-bit RISC in-order processor in 65 nm TSMC technology at the level of individual instructions. Results show 3.4 ns (68 fanout of 4 or 68FO4) delay variation and $26.7\times$ power variation among instructions, and across extreme corners. Our analysis shows that ILV is not uniform across the instruction set. In fact, ILV data partitions instructions into three equivalence classes. Based on this classification, we show how low-overhead monitors and adaptive clocking techniques can be used to enhance performance by a factor of $1.1\times$ – $5.5\times$.

2.1 Introduction

Designers commonly use conservative guardbands into the operating frequency and voltage to handle these variations to ensure error-free operation within the presence of worse case dynamic variations over circuit lifetime that leads to loss of operational efficiency. An alternative is to use sensor circuits to detect dynamic variations coupled with an adaptive recovery methods for quick on-line error detection and compensation.

Further progress in this area requires a careful analysis of the effect of variations on individual instructions. Here we advance the state of the art through following three means:

1. We analyze the effect of a full range of voltage and temperature variations on the performance and power of the 32-bit in-order RISC LEON-3 [1] processor (Sect. 2.2).

2. We introduce the notion of instruction-level vulnerability (ILV) to characterize tolerance of individual instruction to dynamic variations. ILV exposes variation and its effects to the software stack for use in architectural/compiler optimizations. Our results show that ILV is not uniform across the instruction set (Sects. 2.3 and 2.4).
3. Using ILV data, we show the effectiveness of a minimally intrusive and cost-effective fine-grained technique to mitigate the dynamic variations that achieves up to $5.5\times$ performance improvement in comparison to the traditional worst-case design.

2.2 Effect of Operating Conditions

We analyze the effect of operating conditions on the performance and power of the LEON-3 [1] processor compliant with the SPARC V8 architecture. Specifically, we used a temperature range of -40 – 125°C , and a voltage range of 0.72 – 1.1 V. Figure 2.1 shows how the critical path of the processor varies across corners. The higher voltage results in the shorter critical path, while the lower temperature leads to a higher delay in the low-voltage region (voltage ≤ 0.9 V), since MOSFET drain current decreases when the temperature is decreased in the deep submicron technologies [2]. These operating condition (hence dynamic) variations cause the critical path delay to increase by a factor of $6.1\times$ when the operating condition is varied from the one corner to the other. Consequently, a large conservative guardband into the operating frequency is needed to ensure the error-free operation in presence of the dynamic variations.

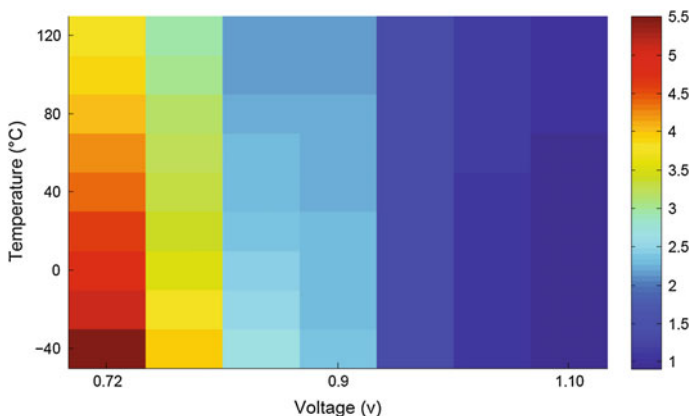


Fig. 2.1 Effect of voltage and temperature variations on the critical path (ns)

2.3 Delay Variation Among Pipeline Stages

We now evaluate the critical paths of each pipeline stage for a given cycle time, while changing the operating conditions. Figure 2.2 shows the number of failed paths with a negative slack for each parallel pipeline stages across three corners. The cycle time is set at 0.85 ns, and voltage varies from 0.72 to 0.88 V, and then to 1.10 V at a constant temperature of 125 °C. As shown in Fig. 2.2, most of the failed paths lie in the execute and memory stages in all three operating voltages. On the other hand, each of the fetch, decode, and register access stages contains less than 40 K failed paths. Furthermore, there is a relatively small fluctuation in their number of critical paths across voltage variations for these stages. Quantitatively, the memory stage at operating voltage of 0.72 V has $1.3\times$, $1.8\times$, $3.8\times$ more critical paths in comparison to the execute, write back, and decode stages, respectively. Memory stage at operating voltage of 1.10 V also faces $1.4\times$, $1.9\times$ more critical paths when the voltage drops to 0.88, 0.72 V, respectively.

Similarly, in Fig. 2.3 the temperature of processor is varied from -40 to 125 °C at a constant voltage of 1.1 V. As a result, there are no failed paths in the fetch stage when the temperature is varied, and only a small number of failed paths are found in the write back stage at the highest temperature. On the other hand, similar to Fig. 2.2, many paths fail within the execute and memory stages. The execute and memory parts of the processor are not only very sensitive to voltage and temperature variations, but also exhibit a large number of critical paths in comparison to the rest of processor. *Therefore, we would anticipate that the instructions that significantly exercise the execute and memory stages are likely to be more vulnerable to voltage and temperature variations.*

Let us now examine the situation of all paths through the processor under different operating condition and frequency. The Y-axis of Fig. 2.4 shows the proportion of failed paths to nonfailed paths for three corners. We observe that this proportion of failed paths suddenly drops below a certain threshold while the clock is finely scaled with a resolution of 0.01 ns. For instance, the proportion falls below 0.5 with only 0.06 ns clock scaling at (1.10 V, 0 °C); in the other words, the number of nonfailed paths is twice as many as those which fail. Alternatively, the number of nonfailed

Fig. 2.2 Effect of voltage variation on the pipeline stages at 125 °C

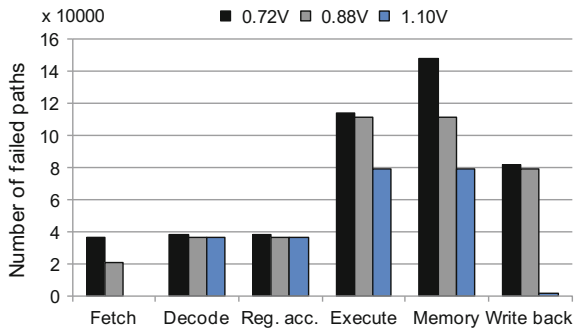


Fig. 2.3 Effect of temperature variation on the number of failed paths among the pipeline stages at 1.10 V

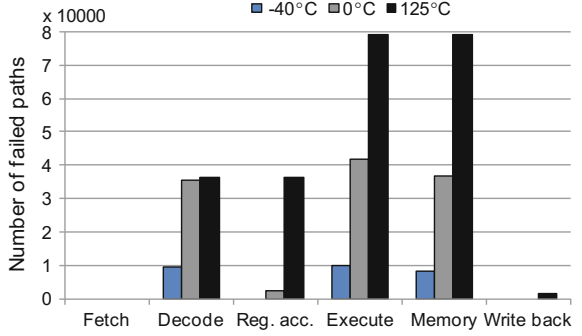
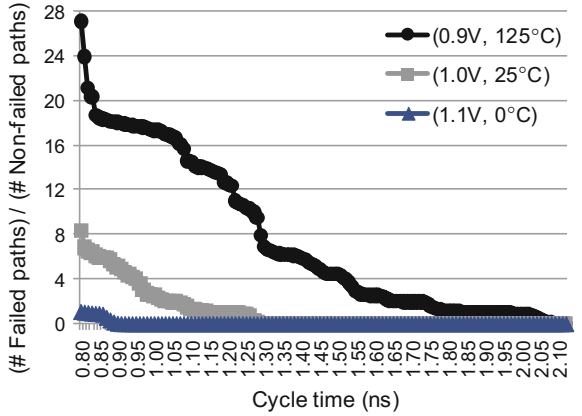


Fig. 2.4 The proportion of failed paths to nonfailed paths versus clock



paths is doubled when the cycle time is increased for 0.3 ns at (0.9 V, 125 °C). These provide an opportunity for an error-free running of some instructions that will not activate those failed paths.

From the previous analysis, we see that instructions will have different levels of vulnerability to variations depending on the way they exercise the nonuniform critical paths across the various pipeline stages. To capture this phenomenon, we define the concept of instruction-level vulnerability to dynamic variations. The classification of instructions is a valuable mechanism to alleviate the guardbanding and improving performance: (i) within a fixed corner, by acquiring the knowledge about which class of instructions is running, the processor can adapt the guardbanding accordingly, without any need for the intrusive variability sensor/observer; (ii) across every corner, processor can adjust its guardbanding for all class of instructions by using a low-overhead variability observer, e.g., phase locked loop (PLL) [3], and ring oscillators (RO) [4].

2.4 Instruction Characterization Methodology and Experimental Results

We use integer pipeline of LEON-3 processor with hardware multiplier/divider units as well as the instruction/data caches to characterize instructions. First, we synthesized the open-source VHDL code of LEON-3 with the TSMC 65 nm technology library (general purpose process) to generate gate-level netlist. The signoff stage for accurate analysis of the operating conditions has been made with Synopsys PrimeTime, using its voltage-temperature scaling features for the composite current source approach of modeling cell behavior. Mentor Graphics' ModelSim is also used for detail gate-level simulations.

2.4.1 Gate-Level Simulation

In the gate-level simulation, for each individual instruction, we apply the Monte Carlo method to observe instruction behavior. To accurately exercise each instruction, we use a normal distribution for the sources, destination, and immediate operands. A large sample of the SPARC ISA is evaluated, including the logical/arithmetic instructions, memory access instructions (load/store), multiply/divide instructions. To quantify the ILV to voltage and temperature variations, we define the **probability of failure (PoF)** for each instruction_{*i*} in Eq. 2.1, where N_i is the total number of clock cycles in Monte Carlo simulation which takes to execute instruction_{*i*} with random operands; and Violation_{*j*} indicates whether there is a violated stage at clock cycle_{*j*} or not.

$$\text{PoF}_i = \frac{1}{N_i} \sum_{j=1}^{N_i} \text{Violation}_j$$

$$\text{Violation}_j = \begin{cases} 1 & \text{If any stage violates at cycle } j \\ 0 & \text{otherwise} \end{cases} \quad (2.1)$$

In other words, PoF_i defines as the total number of violated cycles over the total simulated cycles for the instruction_{*i*}. If any of the analyzed stages have one or more violated flip-flop at clock cycle_{*j*}, we consider that stage as a violated stage at cycle_{*j*}. Intuitively, if instruction_{*i*} runs without any violated path, PoF_i is 0; on the other hand, PoF_i is 1 if instruction_{*i*} faces at least one violated path in any stage, in every cycle.

Table 2.1 Probability of failure of ISA at 1.1 and 1.0 V, while varying temperature and frequency

Corners		(1.1V, -40°C)			(1.1V, 0°C)			(1.1V, 125°C)					(1.0V, 25°C)					
Cycle time (ns)		0.74	0.76	0.78	0.74	0.76	0.78	0.80	0.82	0.84	0.86	0.88	0.90	1.08	1.10	1.12	1.14	1.22
Logical & Arithmetic	add	1	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0
	and	1	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0
	or	1	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0
	sll	1	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0
	sra	1	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0
	srl	1	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0
	sub	1	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0
	xnor	1	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0
	xor	1	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0
Mem	load	1	0	0	1	0	0	1	0	0	0	0	0	1	0.786	0	0	0
	store	1	0	0	1	0	0	1	0	0	0	0	0	1	0.814	0	0	0
Mul. & Div	mul	1	0	0	1	0.967	0	1	0.042	0.015	0.012	0.002	0	1	0.998	0.976	0.074	0
	div	1	0.837	0	1	0.948	0	1	0.991	0.991	0.984	0.984	0	1	0.964	0.993	0.990	0

2.4.2 Instruction-Level Delay Variability

Tables 2.1 and 2.2 summarize the PoF of each evaluated instruction across various corners. We finely change the clock cycle to observe the paths failure for every exercised instruction, and then consequently evaluate its PoF. As shown, instructions exhibit a very wide range of delay under different operating conditions ranges from 0.76 to 4.16 ns. More precisely, the PoF values shown in tables evidence two important facts. First, for their vulnerability to variations, instructions are partitioned into three main classes: (i) the logical/arithmetic instructions, (ii) the memory instructions, and (iii) the multiply/divide instructions. The 1st class shows an abrupt behavior when the clock cycle is slightly varied. Its PoF switches from 1 to 0 with a slight increase in the cycle time (0.02 ns) for every corner, mainly because the path distribution of the exercised part by this class is such that most of the paths have the same length, then we have a all-or-nothing effect, which implies that either all instructions within this class fail or all make it. The 2nd class, the memory instructions, needs much more relaxed cycle time to be able to survive across conditions. For instance, as shown in Table 2.2, only 0.04 ns more guardbanding on the cycle time of the 1st class instruction can guarantee the error-free execution of the memory instructions while they are experiencing 40 °C temperature fluctuation. The 3rd class is the multiply/divide instructions which need higher guardbanding in comparison to the 1st class instruction, ranges from 0.02 ns at (1.1 V, -40 °C) to 0.30 ns at (0.72 V, 125 °C). Since this class highly exercises the execution unit,¹ it has a higher PoF in comparison with the rest of classes in the same clock cycle, for every corner.

Further, based on these results, we can define an adaptive clock cycle for each class of instructions to mitigate the conservative guardbanding, not only within a fixed process corner, but also across corners. All instruction classes act similarly across the wide range of operating conditions: as the cycle time increases gradually, the PoF becomes 0, firstly for the 1st class, then for the 2nd class, and finally for the 3rd

¹Moreover, 64–82% (depends on the corner) of the failed paths in the execution stage lie in the hardware multiplier and divider units.

Table 2.2 Probability of failure of ISA at constant voltage 0.72V, while varying temperature and frequency

Corners		(0.72V, -40°C)				(0.72V, 0°C)				(0.72V, 125°C)							
Cycle time (ns)		4.10	4.12	4.14	4.16	3.58	3.60	3.62	3.64	3.66	2.88	2.90	2.92	2.94	2.98	3.00	3.20
Logical & Arithmetic	add	1	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0
	and	1	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0
	or	1	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0
	sll	1	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0
	sra	1	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0
	srl	1	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0
	sub	1	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0
	xnor	1	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0
	xor	1	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0
	Mem																
Mul. & Div	load	1	0.823	0.823	0	1	0.823	0.823	0	0	1	0.823	0.823	0.823	0.796	0.796	0
	store	1	0.847	0.847	0	1	0.847	0.847	0	0	1	0.847	0.847	0.847	0.823	0.823	0
	mul	1	0.995	0.995	0	1	0.996	0.994	0	0	1	0.998	0.997	0.996	0.996	0.996	0
	div	1	0.995	0.995	0	1	0.995	0.995	0.812	0	1	0.994	0.994	0.993	0.991	0.991	0

class. A processor can benefit from this classification by adapting its guardbanding for each class of instruction by acquiring the knowledge about which class of instructions is/will be running.

2.4.3 Less Intrusive Variation-Tolerant Technique

All intrusive techniques [5–7] try to avoid timing failure for instructions that activate the critical paths by dynamically switching to two-cycle operation. These expensive, instruction by instruction timing adjustment techniques do not expose opportunity for further software-level optimizations especially for sequences and classes of instructions. Therefore, we could have an advanced dynamic clock speed adaptation technique, possibly compiler driven, which can quickly decide on the clock speed of the processor at a very fine-grained [8], just looking at the fetched instructions and keeping track of their entry into the stages, and at the same time monitoring the current corner with a low-overhead monitoring hardware [3, 4]. This technique not only provides great performance enhancement for processor, but also is a step forward toward a less intrusive circuit monitoring and cost-effective robust design.

Table 2.3 shows how a program consisting of various classes of instructions can benefit by this technique under different operating conditions: the performance improvement when processor runs a program only consists of specific classes, in comparison to the traditional worst-case design. For instance, at the typical operating condition (1.0V, 25°C) processor can decrease the cycle time from 4.16ns (Table 2.2) to 1.22ns (Table 2.1), and consequently achieves 3.4× speed improvement, when its running program consists of all three classes. It can further reduce the cycle time to 1.12ns (3.7× speedup) when only the 1st, and 2nd classes of instructions are used in its program. As shown, the proposed solution can greatly achieve 1.1 × –5.5 × performance improvement depends on the type of instruction and the operating condition.

Table 2.3 Performance improvement for different classes of instructions

Vol. (V)	Temp. (°C)	1st and 2nd class	1st, 2nd, 3rd class
1.10	−40	5.5×	5.3×
1.10	0	5.5×	5.3×
1.10	125	5.1×	4.6×
1.00	25	3.7×	3.4×
0.88	−40	3.9×	3.7×
0.88	0	3.9×	3.7×
0.88	125	3.9×	3.5×
0.72	0	1.1×	1.1×
0.72	125	1.3×	1.3×

2.4.4 Power Variability

From delay variability of instructions, we examine now variation of power consumption across and within process corners. The total power consumption of the instruction classes under four operating conditions is shown in Fig. 2.5, when the cycle time is adjusted for each class accordingly, i.e., the best frequency for each class is applied. As a result, all three classes of instructions experience a wide range of total power variability (0.1–2.6 mW), $1.15\times$ intra-corner power variation (across the three classes) due to exercising various parts of processor, and $26.7\times$ inter-corner power variation, at maximum. This implies that ILV could potentially expose opportunity for further software-level optimizations for both performance and power.

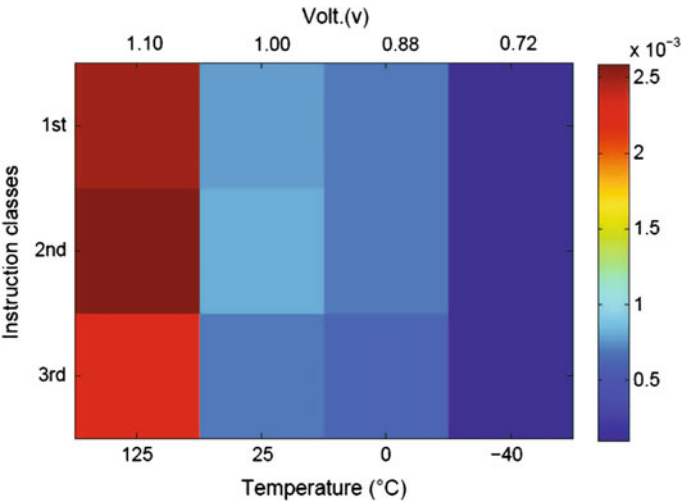


Fig. 2.5 Intra- and inter-corner total power (W) variability of the instruction classes

2.5 Chapter Summary

The concept of instruction-level vulnerability to dynamic voltage and temperature variations is defined. Based on that, all exercised instructions in the integer pipeline of LEON-3 are partitioned into three classes for the full range of operating condition: (i) the logical and arithmetic instructions, (ii) the memory instructions, and (iii) the multiply and divide instructions. Using this classification in conjunction with less intrusive variability observers provides architectural/compiler optimizations a great opportunity to enhance processor performance by $1.1 \times$ – $5.5 \times$, in TSMC 65 nm technology. It is also a step forward toward a low-overhead, efficient, and cost-effective robust design.

References

1. Leon3. <http://www.gaisler.com/cms/>
2. R. Kumar, V. Kursun, Reversed temperature-dependent propagation delay characteristics in nanometer cmos circuits. *IEEE Trans. Circuits Syst. II Express Briefs* **53**(10), 1078–1082 (2006)
3. K. Kang, S.P. Park, K. Kim, K. Roy, On-chip variability sensor using phase-locked loop for detecting and correcting parametric timing failures. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **18**(2), 270–280 (2010)
4. M. Bhushan, A. Gattiker, M.B. Ketchen, K.K. Das, Ring oscillators for CMOS process tuning and variability control. *IEEE Trans. Semicond. Manufact.* **19**(1), 10–18 (2006)
5. D. Ernst, S. Das, S. Lee, D. Blaauw, T. Austin, T. Mudge, N.S. Kim, K. Flautner, Razor: circuit-level correction of timing errors for low-power operation. *Micro, IEEE*, **24**(6), 10–20 (2004)
6. K.A. Bowman, J.W. Tschanz, S.L. Lu, P.A. Aseron, M.M. Khellah, A. Raychowdhury, B.M. Geuskens, C. Tokunaga, C.B. Wilkerson, T. Karnik, V.K. De, A 45 nm resilient microprocessor core for dynamic variation tolerance. *IEEE J. Solid-State Circuits* **46**(1), 194–208 (2011)
7. D. Bull, S. Das, K. Shivshankar, G. Dasika, K. Flautner, D. Blaauw, A power-efficient 32b ARM ISA processor using timing-error detection and correction for transient-error tolerance and adaptation to PVT variation, in *2010 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)* (2010), pp. 284–285
8. J. Tschanz, N.S. Kim, S. Dighe, J. Howard, G. Ruhl, S. Vangal, S. Narendra, Y. Hoskote, H. Wilson, C. Lam, M. Shuman, C. Tokunaga, D. Somasekhara, S. Tang, D. Finan, T. Karnik, N. Borkar, N. Kurd, V. De, Adaptive frequency and biasing techniques for tolerance to dynamic temperature-voltage variations and aging, In *IEEE International Solid-State Circuits Conference, 2007. ISSCC 2007. Digest of Technical Papers* (2007), pp. 292–604

From Variability Tolerance to Approximate Computing in
Parallel Integrated Architectures and Accelerators

Rahimi, A.; Benini, L.; Gupta, R.

2017, XV, 197 p. 86 illus., 48 illus. in color., Hardcover

ISBN: 978-3-319-53767-2