Chapter 11
Variation and Aging Tolerance in FPGAs

Nikil Mehta and André DeHon

Abstract Parameter variation and component aging are becoming a significant problems for all digital circuits including FPGAs. These effects degrade performance, increase power dissipation, and cause permanent faults at manufacturing time and during the lifetime of an FPGA. Several techniques have been developed to tolerate variation and aging in ASICs; FPGA designers have been quick to adopt and customize these strategies. While FPGAs can use many ASIC techniques verbatim, they have a distinct advantage to aid in the development of more innovate solutions: reconfigurability. Reconfigurability gives us the ability to spread wear effects over the chip which is not possible in ASICs. This chapter examines the impact of variation and wear on FPGAs and notes the benefit that can be gained from variation and aging tolerance techniques that operate open-loop.

11.1 Introduction

Like all scaled semiconductor devices, FPGAs are experiencing dramatic increase in process variability. Every manufactured FPGA is unique in terms of its composition of physical parameters (e.g., channel lengths, oxide thickness, and threshold voltages) for each transistor. FPGAs are also subject to the same aging mechanisms that cause ASICs to degrade in performance and fail over time. However, FPGA manufacturers have the unique challenge that the functionality of an FPGA is not fixed at manufacturing time. Users can map any circuit onto an FPGA, and they expect a fully functioning, high performance design, regardless of the parameter variation imposed on the specific chip they select for mapping. Additionally, because every user design is different, every workload will differ in terms of how it stresses individual circuits within the FPGA. Different temperature and usage pro-

N. Mehta (✉)
Department of Computer Science, California Institute of Technology, Pasadena, CA, USA
e-mail: nikil@caltech.edu

S. Bhunia, S. Mukhopadhyay (eds.), *Low-Power Variation-Tolerant Design in Nanometer Silicon*, DOI 10.1007/978-1-4419-7418-1_11,
© Springer Science+Business Media, LLC 2011

files result in unique aging trends. Without any techniques to combat variation and aging, to meet the demand that every manufactured chip will function with any user configuration and usage pattern over time, manufacturers pessimistically guard band timing parameters.

Fortunately, FPGAs have two distinct advantages over fixed designs like ASICs and CPUs. First, they have spare resources: user designs infrequently utilize all resources, and even when mapped at full logic capacity, FPGAs still have many additional, unused interconnect resources. These extra resources can be used to avoid resources that have been degraded by parameter variation or aging. Second, FPGAs can reconfigure. There are many ways to map a user's design, and these different mappings can be instantiated over time in the field. When a specific user mapping fails on a particular chip due to parameter variation or aging, another configuration may be tried.

These alternate configurations can be identical mappings for all chips deployed in the field over time, or they can be specific to a component such that every chip receives a unique mapping. Component-specific mapping is a challenging but powerful technique for FPGAs that will be explored in depth in the following chapter. This chapter focuses on traditional, component-oblivious techniques for tolerating variation and aging. Many techniques are adopted directly from ASIC and CPU design flows, but customized for FPGAs and are general in that they can be applied to any digital design without reconfiguration. Some, however, exploit configurability, in a open-loop manner, to combat projected aging. The outline for this chapter is as follows: Section 11.2 reviews how variation impacts FPGAs and Section 11.3 describes techniques for variation tolerance; Sections 11.4 and 11.5 discuss the impact of aging and strategies for lifetime extension.

11.2 Impact of Variation

(Chapter 1) described sources of variation in detail. While die-to-die (D2D) variation has historically been the dominant source of variation, in recent technology nodes within die (WID) variation has increased beyond that of D2D. Systematic, spatially correlated, and random WID variations all have a significant impact on FPGAs; random WID variation in particular has been increasing and will be the dominant source of variation in future technology nodes. Compensating for this random WID variation is a significant challenge as neighboring transistors can have completely different characteristics.

To describe the impact of variation on FPGA designs, metrics like timing yield and power yield are often used. Users of FPGAs typically want all chips containing their mapped designs to achieve a set performance target while staying within a power budget. Timing yield and power yield simply express the percentage of devices that meet those targets. Power yield is often termed leakage yield as static power has begun to dominate total power dissipation.

11.2.1 FPGA-Specific Effects

Variation typically impacts ASICs and FPGAs in similar ways; however, commercial FPGAs do have a few key architectural differences that help to mitigate variation. First, while variation in memory is a serious problem in ASICs and CPUs, it is less of a problem for FPGAs. Memory in FPGAs is predominately used as configuration bits; these bits are typically only written once at power on time and area always on – they are never read or written in timing critical paths. This means that the FPGA can afford to use larger thresholds, supply voltages, and skewed cells (Section 11.3.3) to combat variation without impacting power and performance. Modern FPGAs contain large embedded memories that require similar care as embedded memories in ASICs.

Second, connections between logic elements are longer in FPGAs than in ASICs. Although these connections have larger mean delays, they also have smaller percentage variation. Because FPGA connections must traverse several switches, delay variation averages out over the length of the path. The variation in path delay for a path of N identical Gaussian gates with delay distributions of $(\mu_{\tau_{\text{gate}}}, \sigma_{\tau_{\text{gate}}})$ can be written as

$$\mu_{\tau_{\text{path}}} = N\mu_{\tau_{\text{gate}}} \tag{11.1}$$

$$\sigma^2_{\tau_{\text{path}}} = N\sigma^2_{\tau_{\text{gate}}} \tag{11.2}$$

$$\frac{\sigma_{\tau_{\text{path}}}}{\mu_{\tau_{\text{path}}}} = \frac{1}{\sqrt{N}}\frac{\sigma_{\tau_{\text{gate}}}}{\sigma_{\tau_{\text{gate}}}} \tag{11.3}$$

Length N paths have variation reduced by $1/\sqrt{N}$. This means for identical designs, with the way commercial FPGAs are architected (long, unpipelined path lengths), FPGAs may experience less delay variability than ASICs. This can help to improve timing yield.

Another advantage of FPGAs is that they have a small, repeated circuit structure, unlike ASICs which can have a large, arbitrary layout. With the recent explosion of layout design rules, ASIC physical design has become much more difficult. While FPGA manufacturers must also struggle with deep submicron effects, they only have to design, layout, and verify a much smaller structure. Further, because the structure is regular and repeated, it is much easier to understand and potentially avoid the impact of systematic variation. When using optical proximity correction and other resolution enhancement techniques (Chapter 1, Section 1.2.1) to model the effect of diffraction through the mask, a regular, repeated structure means that it is easier to calculate OPC and model its effect on a design.

11.2.2 FPGA-Level Impact

It is difficult to precisely quantify the impact of parameter variation on commercial FPGAs. Because variation data is economically sensitive to vendors, no information characterizing FPGA variability has been officially released. However, several researchers have empirically measured batches of both custom manufactured and commercial FPGAs, characterizing their variation distributions. These studies begin to quantify and document the effect of real variation on FPGAs.

Katsuki et al. measured WID variability for 90-nm custom manufactured FPGAs [2]. To collect variation data they mapped arrays of ring oscillators and measured their frequencies across the chip. Unfortunately, they published data in arbitrary units, so actual delay variation numbers are not available. However, Sedcole et al. measured WID delay variability of 18 commercial 90-nm Altera Cyclone II FPGAs by using a similar ring oscillator measurement technique [10] (Fig. 11.1a). They found that individual logic elements have a random delay variability of $3\sigma/\mu =$ 3.54%; they also found a WID-correlated component of delay variability of 3.66%. Finally, Wong et al. used an at-speed measurement technique to characterize the WID delay variability of both logic and embedded multipliers in a 65-nm Altera Cyclone III FPGA [18] (Fig. 11.1b). They discovered a WID delay variation of $3\sigma/\mu = 5.96\%$. Each of these experiments demonstrate the feasibility of characterizing the delay distribution of individual FPGAs elements. The next chapter will further develop how these measurements can be performed and leveraged to mitigate variation.

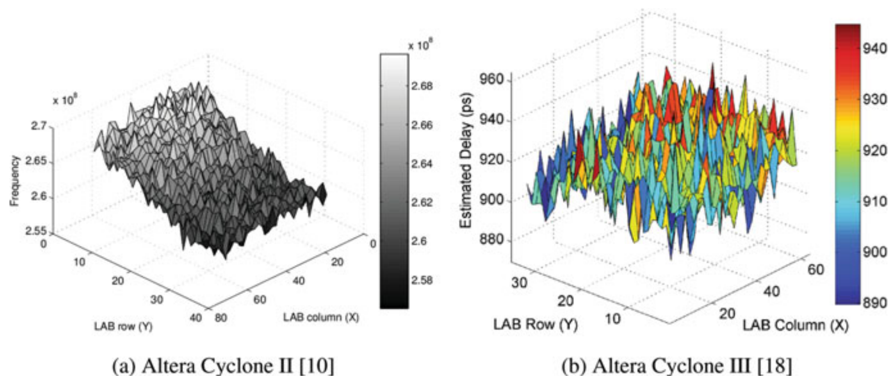

Fig. 11.1 Measured LUT frequency maps for Altera FPGAs

11.3 FPGA Variation Tolerance

To tolerate FPGA parameter variation, researchers have employed many of the same techniques used for ASICs and CPUs. These methods can operate at the CAD level to statistically account for variability when generating a configuration for a design,

or they can operate at the architecture and circuit levels to physically compensate for the effects of variation. Some of these techniques are the same as those used for reducing power consumption but used in a new context.

11.3.1 SSTA CAD

A heavily researched method for dealing with variation during the technology mapping, place and route stages of an ASIC design is the use of statistical static timing analysis (SSTA) (Chapter 4, Section 4.4.3). SSTA attempts to model variability directly in CAD algorithms to help produce circuits that inherently account for variation.

The goal of most CAD algorithms is to minimize the critical path delay of a circuit. Traditional CAD algorithms do not aggressively minimize near-critical paths since their reduction would not reduce the delay of the circuit. However, near-critical paths are important under variation because there is a probability that they will become critical. As expressed in Equations (11.1), (11.2), (11.3), path delays are not constants but distributions. Hence, with some probability, a path may be faster or slower than its nominal value. Near critical paths that have a probability of becoming critical are deemed to be *statistically critical*.

SSTA is a methodology that identifies statistically critical paths and enables CAD algorithms to optimize those paths. Integrating SSTA into clustering, placement and routing algorithms of the FPGA CAD flow simply involves replacing the nominal STA routine with an SSTA routine. The algorithms then proceed as normal, attempting to reduce the delay of statistically critical paths and improve circuit timing yield (e.g., minimize μ_τ and σ_τ of the circuit).

FPGA CAD algorithms modified for SSTA have been studied for placement [5] and routing [3, 11]. Lin et al. studied a full SSTA CAD flow with clustering, placement, and routing and characterized the interaction between each stage [4]. The only modification made to each algorithm is to replace the criticality of a net in all cost functions with its statistical criticality. In clustering, SSTA is only performed at the start of the algorithm; for placement and routing statistical criticalities are updated at the start of every iteration.

Lin et al. [4] examined a 65-nm predictive technology with $3\sigma_{L_{\text{eff}}}/\mu_{L_{\text{eff}}}$ and $3\sigma_{V_{\text{th}}}/\mu_{V_{\text{th}}}$ of 10, 10, and 6% for D2D, WID spatially correlated, and WID random variation. They observed that SSTA-based clustering alone can improve μ_τ and σ_τ by 5.0 and 6.4%, respectively, and improve timing yield by 9.9%. Placement improves μ_τ, σ_τ and timing yield by 4.0, 6.1, and 8.0%; routing achieves 1.4, 0.7, and 2.2% improvement. All three SSTA algorithms combined yield 6.2, 7.5, and 12.6% improvement. As in the power-aware CAD algorithm case from (Chapter 10, Section 10.3.7), they find that individual enhancements to the three stages of FPGA CAD are not additive, and that stochastic clustering provides the majority of benefit.

These benefits come with a 3$\times$ runtime increase but with minimal changes to FPGA CAD algorithms. Their primary effect may be to provide a better estimate

for the delay of the circuit, reducing some of the conservative margining used to guarantee a target level of performance across almost all chips. Nonetheless, there are several drawbacks to SSTA-based approaches. First, they depend on having accurate models of process variation. Without correct values for the σ, μ, and correlation coefficients of process parameters, SSTA cannot accurately determine which paths are statistically critical. Second, these techniques only improve the probability of a device meeting timing and do not guarantee working parts. SSTA CAD still produces a one-size-fits-all design, and there is non-negligible probability that a particular chip will not yield with a SSTA-enhanced design. Finally, SSTA approaches do not scale well with high random variation, which is expected to dominate future technology nodes. High variation spreads out delay distributions, making many more paths statistically critical. It then becomes difficult for the CAD algorithms to optimize and reduce the delay of many paths at once.

11.3.2 Architecture Optimization

Another strategy for mitigating the impact of parameter variations is to optimize the structure of the FPGA by changing key architectural parameters (Chapter 10, Section 10.4). FPGA logic is parameterized by N, the number of LUTs per CLB, and k, the number of inputs to a LUT. Routing is parameterized by the switchbox type and segment length L_{seg}. Changing these values can have an impact on both timing and leakage yield.

In terms of logic, larger values of N and k increase the area, delay, and leakage of individual CLBs which will hurt leakage and timing yield. However, the total number of CLBs and required routing resources may decrease, which would improve leakage yield. Additionally, the number of CLBs and switches on near critical paths may decrease with larger k and N, improving timing yield. Wong et al. studied the impact of N and k on timing and leakage yield [16]. They observe that while N has little impact on yield, $k = 7$ maximizes timing yield, $k = 4$ maximizes leakage yield, and $k = 5$ maximizes combined yield. These results are not surprising since leakage roughly scales with area and timing yield is directly related to delay; these results largely match the k values that optimize delay, area, and area-delay product, respectively.

Wire segmentation can have an impact on timing yield for similar reasons as N and k. For a fixed distance net, smaller values of L_{seg} increase the number of buffers on the path, increasing mean delay but decreasing variance due to delay averaging (Equation (11.3)). Kumar et al. found that compared to an FPGA with 50% $L_{\text{seg}} = 8$ and 50% $L_{\text{seg}} = 4$ segments in a 45-nm predictive technology with $3\sigma/\mu = 20\%$ variation, using a mix of shorter segments can reduce both mean delay and variance [3]. They find that architectures with between 20 and 40% $L_{\text{seg}} = 2$ segments can improve μ_τ by 7.2–8.8% and σ_τ by 8.7–9.3%.

Work to date has focused on revisiting traditional FPGA architecture parameters with the new concern for variation. The evidence so far suggests that variation does not significantly change the way we would select these parameters. This leaves open

exploration of more significant changes that might introduce new parameters or structures into FPGA architecture design.

11.3.3 Transistor Sizing

Lower level, device, and circuit parameters can also be optimized to mitigate the impact of parameter variation. Transistor sizing is a common strategy used in ASICs to directly reduce the magnitude of variation. Larger transistors have increased numbers of dopants, and, due to law of large numbers effects, the magnitude of dopant variation will reduce. V_{th} variation can be expressed as a function of transistor dimensions as:

$$\sigma_{V_{th}} \propto \frac{1}{\sqrt{WL}} \tag{11.4}$$

Increasing W in logic transistors can increase variation tolerance, but at the cost of area and energy. There has been no published work quantifying the tradeoffs in resizing FPGA logic transistors, which may be a promising avenue of exploration.

Another option is to adjust the W of transistors located in memory, which can reduce the probability of failure of FPGA SRAM bits [8]. SRAM bits can fail in one of four ways: read upsets, write upsets, hold failures, or access time failures. FPGA configuration memory will typically fail with read or hold upsets; writes only happen at configuration time and timing is irrelevant as they are always in the read state. FPGA data memory failures are dependent on application characteristics; Paul et al. [8] study applications where memories were used as logic such that reads dominate writes and hence design a cell that increases read and access time failure tolerances significantly.

Figure 11.2 shows the skewed SRAM cell. Increasing W for the pull-up transistors in the cross coupled inverters decrease read failure probability by increasing the voltage at which a read upset occurs. Decreasing W of the access transistors makes it more difficult for the internal storage node to be flipped, but increases access time failure probability by slowing down reads. Increasing W of the pull-down transistors compensates for this effect. Paul et al. show two orders of magnitude read upset

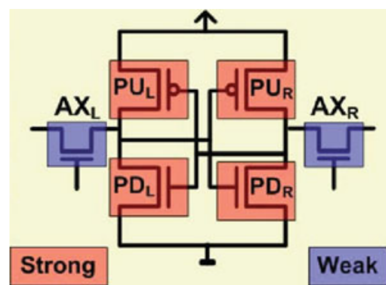

Fig. 11.2 Skewed SRAM cell [8]

and access time failure probability reductions at the cost of a four orders of magnitude increase in write failure probability. The write failures can be addressed with component-specific mapping techniques (Chapter 12).

11.3.4 Asynchronous Design

Choosing the right timing target to achieve high performance and acceptable timing yield is a significant concern with synchronous designs under process variation. If a timing target is chosen too pessimistically, performance of manufactured designs will suffer; if it is chosen too aggressively, many devices may fail. One way to avoid this problem is to design circuits that are not clocked and do not run at a uniform frequency. Instead of using a global clock for synchronization, sequencing can be performed by using handshaking circuitry between individual gates. These self-timed or asynchronous circuits have been studied in depth by ASIC designers and have been explored by FPGA designers as well. Asynchronous FPGAs have been demonstrated in [6, 15] and have even begun to be commercially manufactured [1].

The primary advantage of asynchronous FPGAs in terms of process variation is their high tolerance to delay variability. In the most robust class of asynchronous circuits, quasi-delay insensitive (QDI) circuits [7], only a single timing assumption must be validated to ensure correct operation. This assumption is called an isochronic fork, shown in Fig. 11.3. If the adversary path is faster than the primary path due to delay variation, circuit B may receive an incorrect value. To ensure correctness delay elements can be inserted in the adversary path to margin against error.

Figure 11.4 depicts an asynchronous FPGA LUT circuit from [6]. There are two important characteristics of how asynchronous QDI logic is commonly implemented. First, dual-rail precharge logic is used for high speed circuits. Second, data communication between circuit blocks is done via one-hot encoded signals. We see that in the LUT, precharge signals charge up the dual-rail output bits. During

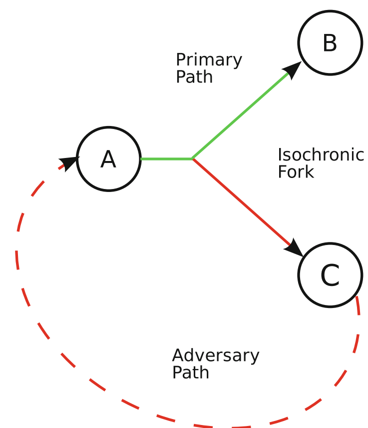

Fig. 11.3 Isochronic fork assumption in QDI asynchronous circuits

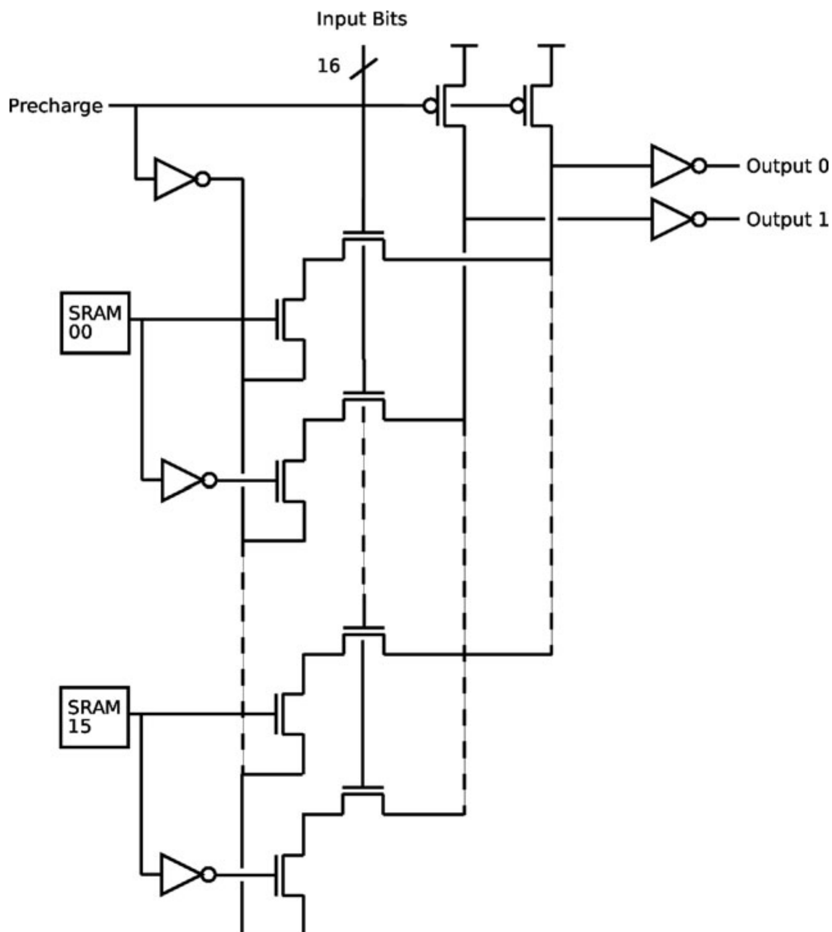

Fig. 11.4 Asynchronous FPGA 4-LUT [6]

evaluate (precharge low), a single LUT input case is selected via the 16-bit, one-hot encoded input, and one of the dual-rail outputs of the LUT is pulled down. The precharge signal is controlled via handshake circuitry from adjacent blocks. When the next block is ready for data and the previous block has sent valid data (as determined by the handshake logic), the LUT will fire.

Figure 11.5 shows a switch circuit. At the heart of the switch circuit is a weak-condition half-buffer, which essentially acts as a register for asynchronous logic. Handshaking signals (enable signals from each direction) control data movement through the switch in the same manner as described for the LUT. Enable and data signals are selected via pass transistor multiplexers as in synchronous switch boxes. One of the primary demonstrated advantages of asynchronous FPGAs is that by using register-like circuits at every switch, interconnect is effectively pipelined for

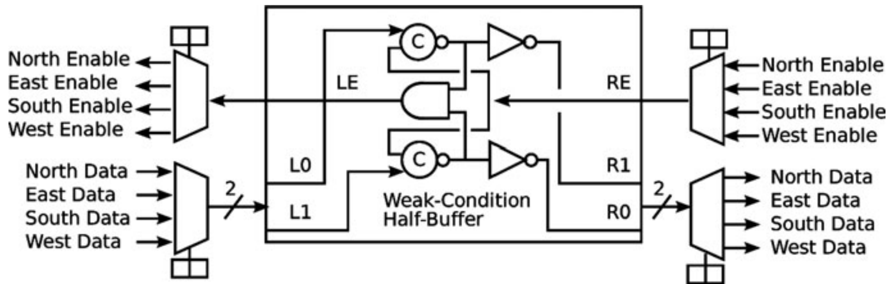

Fig. 11.5 Asynchronous FPGA switch [6]

high speed. Some synchronous FPGA designs have demonstrated highly pipelined interconnect [14], but commercial FPGA vendors have yet to adopt this design style.

If one adopts a fully asynchronous system model, the system will always work regardless of the variability, which reduces the design burden of achieving timing closure in synchronous circuits. The asynchronous design decouples the delay of unrelated blocks allowing each to run at their natural speed. Nonetheless, the throughput and cycle-time of asynchronous circuits are impacted by variation, and high-variation can prevent the asynchronous circuit from meeting a performance yield target as well. The main drawback of asynchronous circuits is that they require area and energy overhead for handshaking circuitry, although energy is simultaneously reduced by eliminating the clock network and through implicit clock gating (as only circuit elements that are performing computation are switching). There has been no published work quantifying the advantages of asynchronous FPGAs under variation, which may be a promising area for future work.

11.4 Impact of Aging

(Chapter 1, Section 1.4) described sources of aging in detail. We will briefly discuss how the primary sources of aging (NBTI, HCI, TDDB, and electromigration) impact FPGAs.

NBTI degrades the threshold of PMOS transistors and is heavily dependent on how long the transistor is held on. The degradation caused by NBTI can be reversed while the PMOS transistor is turned off; this is perhaps the simplest way to combat the induced V_{th} degradation. Reconfiguration potentially allows the FPGA to change the on-profile seen by individual transistors without additional circuitry. NBTI threshold degradation slows down PMOS transistors in logic and significantly reduces the static noise margin (SNM) of SRAM cells. FPGAs typically have more NMOS than PMOS transistors; most architectures use NMOS pass transistors for multiplexers in LUTs and switches that comprise the majority of area. PMOS transistors are used in configuration SRAM, buffers, and embedded structures (e.g., memories and multipliers) in modern parts. Configuration bits are the

primary source of NBTI degradation in FPGAs [9] as these transistors are the largest fraction of PMOS devices, and when configured as conducting, they remain on for the entire time the part stays powered on. NBTI can potentially cause severe configuration cell instability through SNM degradation, resulting in early lifetime failures.

HCI degrades the threshold of NMOS devices and is not reversible like NBTI. It is heavily dependent on transistor usage and current density. The most direct technique to combat HCI is to ensure uniform wear leveling across the chip and not use particular NMOS transistors for the entire operational lifetime. Electromigration causes wire faults and is similar to HCI in that it is usage dependent, not reversible, and best avoided through uniform wearing.

TDDB is largely dependent on gate leakage. Hence, the best approach to mitigating TDDB is to reduce leakage current. The prior chapter outlined several techniques to reduce leakage, such as body biasing (Chapter 10, Section 10.4.6), dual V_{th} (Chapter 10, Section 10.4.6), power gating (Chapter 10, Section 10.4.4), and bit inversion (Chapter 10, Section 10.3.3 and Chapter 11, Section 11.5.1).

Finally, each source of aging is heavily dependent on temperature. When compared to CPUs, FPGAs are much more power efficient and have a lower power density. In identical environments for identical workloads, this will translate to lower operational temperatures and longer lifetimes. However, when compared to an equivalent ASIC, FPGAs are much less power efficient, which could mean faster aging.

As in the case of parameter variation, no commercial data about aging has been made public. However, Stott et al. performed accelerated aging experiments on a pair of Altera Cyclone III FPGAs to estimate the impact of aging on FPGAs [13] (Figures 11.6 and 11.7). Using measurement technique from [17], they tested chips overvolted from 1.2 to 2.2 V and heated to 150°C. They observed a frequency degradation of 15%, with NBTI being the primary cause.

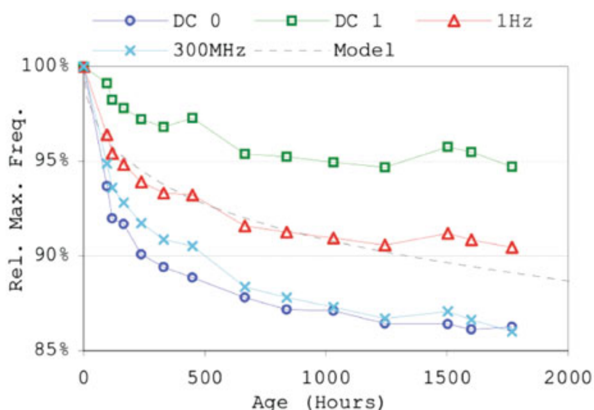

Fig. 11.6 Measured LUT frequency degradation for Altera Cyclone III under accelerated aging [13]

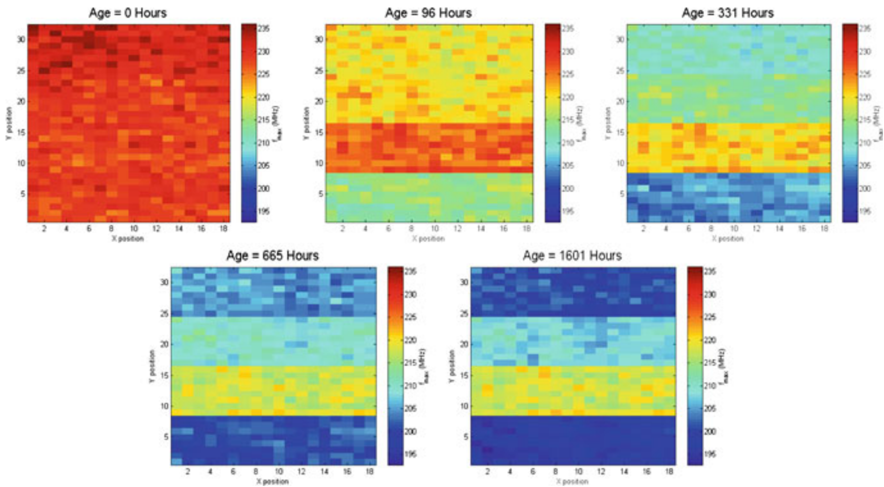

Fig. 11.7 Measured LUT frequency map of Altera Cyclone III under accelerated aging [13]

11.5 FPGA Aging Tolerance

FPGAs are an excellent substrate for compensating for lifetime degradation because of their spare resources and in-field reconfigurability. These characteristics make it easy to change a mapped FPGA design over time to more uniformly spread out wear over the part. The following techniques begin to take advantage of this configurability.

11.5.1 Bit Inversion

Bit inversion was discussed in the previous chapter (Chapter 10, Section 10.3.3) as a way to lower leakage power in LUTs. Because certain LUT configurations leak more than others, by flipping LUT configuration bits (and inverting the input select signals to the LUT to ensure correct output), leakage can be reduced. The same technique can be applied to reducing lifetime degradation in LUTs from NBTI. Because NBTI is reversible, by turning off PMOS transistors that were previously on, degraded values of V_{th} can begin to return to normal.

Ramakrishnan et al. proposed a scheme for relaxing PMOS LUT transistors by loading in a new, bit-inverting bitstream during the life of an FPGA [9]. Inverting configuration bits in the interconnect is more challenging because routes between CLBs may be disrupted. To address this challenge they classified switch configuration bits in three ways: dead, inactive, and used. Dead switches have undriven inputs and unused outputs and can safely be flipped without affecting any routes.

Inactive switches have driven inputs and unused outputs, meaning that inversion could cause a short. They propose a method for flipping these bits in a round-robin fashion to avoid creating a short. Finally, used switches have driven inputs and used outputs. The switches contain valid routes that must be rerouted to allow for bit flipping. They propose a rerouting scheme described in [12]. With these techniques they report a failure-in-time (FIT) decrease of 2.5%.

Bit inversion was also used as a compensation strategy for TDDB [12] by the virtue of it reducing leakage current. Using simulations, they found that bit inversion improved mean-time-to-failure (MTTF) due to TDDB by 24% on average.

11.5.2 Reconfiguration

The most promising technique to use for FPGA lifetime extension is reconfiguration. As an FPGA ages, new configurations can be mapped to a part, avoiding aged resources and utilizing fresh ones or simply rearranging the assigment of logic to resources. Srinivasan et al. explored using reconfiguration to compensate for both HCI and electromigration [12]. Both effects are irreversible and usage dependent, and require avoidance of over-utilization of resources over time.

To avoid HCI, Srinivasan et al. propose to re-place a design over time to previously unused portions of the FPGA. Figure 11.8 shows an example. Using switching activity estimates, they determine the extent to which a region has aged during its operational lifetime. They then perform re-placement using an algorithm that can place blocks with regional restrictions, avoiding areas estimated to be degraded by HCI. Because placement is regionally constrained potentially in a smaller space, the new placement may have a longer critical path. They report that using this method to avoid HCI, part life can be increased by 28% at the cost of 1.53% delay.

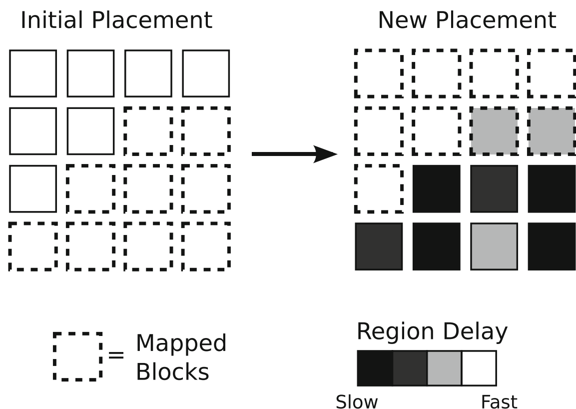

Fig. 11.8 Region relocation for uniform wear

Since electromigration targets wires, simply re-placing logic may not ensure avoidance of specific interconnect segments. Therefore, to mitigate electromigration Srinivasan et al. propose to re-route nets over time. Only nets that have high activity factors and are most likely to fail are chosen for re-routing. Similar to re-placement, re-routing incurs a frequency penalty as re-routed nets may not be timing optimal. By using this technique to combat electromigration they report a lifetime extension of 14% at the cost of 1.3% delay.

An important point to note about these techniques is that they assume that every component is the same and ages in the same way. Hence, every chip will be remapped in the same manner over its lifetime. As manufactured chips are all unique due to process variation and age in unique ways, it would be more effective to create customized configurations for chip. However, this scheme would require ways to measure and characterize each chip separately, in addition to generating a custom mapping per chip. The next chapter will explore these ideas and challenges in depth.

11.6 Summary

Table 11.1 summarizes techniques for variation tolerance; Table 11.2 summarizes techniques for aging tolerance. As in the case of low power techniques, benefits may not be cumulative.

Table 11.1 Roundup of FPGA techniques for variation tolerance

	$3\sigma/\mu$ V_{th} variation		Timing improvement		Yield improvement	
Technique	Regional	Random	μ	σ	Timing	Leakage
SSTA clustering	10%	6%	5.0%	6.4%	9.9%	–
SSTA placement	10%	6%	4.0%	6.1%	8.0%	–
SSTA routing	10%	6%	1.4%	0.7%	2.2%	–
Logic block architecture	?	?	–	–	9%	73%
Interconnect architecture	–	20%	7.2–8.8%	8.7–9.3%	–	–

Table 11.2 Roundup of FPGA techniques for aging tolerance

	Target			
Technique	Aging mechanism	Location	Lifetime metric	Lifetime increase (%)
Bit inversion	NBTI	Configuration Bits	FIT	2.5
Bit inversion	TDDB	LUTs	MTTF	24
Re-placement (open loop)	HCI	LUTs	MTTF	28
Re-routing (open loop)	Electromigration	Interconnect	MTTF	14

These techniques show that FPGAs can begin to tolerate variation by using very similar techniques as those used by ASICs. However, most of these techniques do not use the most powerful tool at the disposal of an FPGA: reconfiguration. FPGAs have a valuable opportunity to surpass ASICs in reliability at a time when variation and aging effects are predicted to increase substantially. The fundamental problem caused by variation is that every manufactured chip is degraded differently; component-specific reconfiguration is the natural response to compensate for this unique degradation on a per-chip basis. For aging we saw that open-loop configurability can increase lifetime; however, by closing the loop and measuring characteristics of individual components, reconfiguration should be able to extend life even further. The challenges and potential benefits of component-specific mapping will be addressed in the next chapter.

References

1. Achronix Semiconductor Corporation, Inc. (2010) Achronix Speedster Data Sheet, preliminary (v1.0) edn http://www.achronix.com/
2. Katsuki K, Kotani M, Kobayashi K, Onodera H (2005) A yield and speed enhancement scheme under within-die variations on 90 nm LUT array. In: Proceedings of the IEEE custom integrated circuits conference, pp 601–604
3. Kumar A, Anis M (2010) FPGA design for timing yield under process variations. IEEE Trans VLSI Syst 18(3):423–435
4. Lin Y, He L, Hutton M (2008) Stochastic physical synthesis considering prerouting interconnect uncertainty and process variation for FPGAs. IEEE Trans VLSI Syst 16(2):124
5. Lucas G, Dong C, Chen D (2010) Variation-aware placement for FPGAs with multicycle statistical timing analysis. In: Proceedings of the international symposium on field-programmable gate arrays. ACM, New York, NY, pp 177–180
6. Manohar R (2006) Reconfigurable asynchronous logic. In: Proceedings of the IEEE custom integrated circuits conference, pp 13–20
7. Martin A, Nystrom M (2006) Asynchronous techniques for system-on-chip design. Proc IEEE 94(6):1089–1120
8. Paul S, Mukhopadhyay S, Bhunia S (2009) A variation-aware preferential design approach for memory based reconfigurable computing. In: Proceedings of the international conference on computer-aided design. ACM, New York, NY, pp 180–183
9. Ramakrishnan K, Suresh S, Vijaykrishnan N, Irwin M (2007) Impact of NBTI on FPGAs. In: Proceedings of the international conference on VLSI design. IEEE Computer Society, pp 717–722
10. Sedcole P, Cheung PYK (2006) Within-die delay variability in 90 nm FPGAs and beyond. In: Proceedings of the international conference on field-programmable technology, pp 97–104
11. Sivaswamy S, Bazargan K (2008) Statistical analysis and process variation-aware routing and skew assignment for FPGAs. Trans Reconfig Technol Syst 1(1):1–35. doi: http://doi.acm.org/10.1145/1331897.1331900
12. Srinivasan S, Mangalagiri P, Xie Y, Vijaykrishnan N, Sarpatwari K (2006) FLAW: FPGA lifetime awareness. In: Proceedings of the ACM/IEEE design automation conference. ACM, p 635
13. Stott EA, Wong JSJ, Pete Sedcole P, Cheung PYK (2010) Degradation in FPGAs: measurement and modelling. In: Proceedings of the international symposium on field-programmable gate arrays. ACM Press, New York, NY, p 229

14. Tsu W, Macy K, Joshi A, Huang R, Walker N, Tung T, Rowhani O, George V, Wawrzynek J, DeHon A (1999) HSRA: high-speed, hierarchical synchronous reconfigurable array. In: Proceedings of the international symposium on field-programmable gate arrays, pp 125–134
15. Wong C, Martin A, Thomas P (2003) An architecture for asynchronous FPGAs. In: Proceedings of the international conference on field-programmable technology, pp 170–177
16. Wong H, Cheng L, Lin Y, He L (2005) FPGA device and architecture evaluation considering process variations. In: Proceedings of the international conference on computer-aided design. IEEE Computer Society, p. 24
17. Wong JSJ, Sedcole P, Cheung PYK (2008) A transition probability based delay measurement method for arbitrary circuits on FPGAs. In: Proceedings of the international conference on field-programmable technology, pp 105–112
18. Wong JSJ, Sedcole P, Cheung PYK (2009) Self-measurement of combinatorial circuit delays in FPGAs. Trans Reconfig Technol Syst 2(2):1–22