Reliability-Aware Placement in SRAM-Based FPGA for Voltage Scaling Realization in the Presence of Process Variations

Shahin Golshan, Amin Khajeh, Houman Homayoun, Eli Bozorgzadeh,
Ahmed Eltawil, Fadi J. Kurdahi
Center of Embedded Computer Systems
University of California, Irvine CA, 92697, USA
{golshans, akhajed, hhoumayou, eli, aeltawil, kurdahi}@uci.edu

Abstract

With advances in technology scaling, the configuration memory in SRAM-based FPGA is contributing a large portion of power consumption. Voltage scaling has been widely used to address the increases in power consumption in submicron regimes. However, with the advent of process variation in the configuration SRAMs, voltage scaling can undermine the integrity of a design implemented on the FPGA device as the design's functionality is determined by the contents of the configuration SRAMs. In this paper, we propose to exploit the abundance of homogenous resources on FPGA, in order to realize voltage scaling in the presence of process variation. Depending on the design to be implemented on FPGA, we select the minimal voltage that sustains a reliable placement. We then introduce a novel 2-phase placement algorithm that maximizes the reliability of the implemented design when voltage scaling is applied to the configuration memory. In the first phase, *pre-deployment placement,* we maximize the reliability of the implemented designs considering the a priori distribution of SRAM failures due to process variation and voltage scaling. The second phase, *post-deployment placement,* is performed once the device is fabricated in order to determine a fault-free placement of the design for the FPGA device. Our results indicate significant leakage power reduction (more than 50%) in the configuration memory when our placement technique is combined with voltage scaling with little delay degradation.

Categories and Subject Descriptors:

J.6 [Computer Aided Design (CAD)]

General Terms:

Algorithms, Design, Reliability

Keywords:

FPGA, Process Variation, Voltage Scaling, Placement, Computer Aided Design

1. Introduction

Field Programmable Gate Arrays (FPGA) are widely used in many application domains as they offer high performance, high flexibility and fast time-to-market. The ever-increasing demand for more computations as well as technology scaling is increasing the power densities of FPGA devices. While technology scaling impacts the dynamic power and the leakage power of FPGA devices, the consequences of aggressive technology scaling is more acute for leakage power. As predicted by the ITRS [1], leakage power grows rapidly with technology scaling. In FPGA devices, the configuration memory is the major contributor of leakage power in the system. In fact as much as 45% of the total leakage power is consumed by the configuration memory bits [2]. This ratio becomes even larger as FPGA systems grow more complex. Therefore, in order to cope with the growth of leakage power in the newer generations of FPGA devices, power management solutions must be provided for the configuration bits in FPGA.

Voltage scaling is one of the most effective means of reducing the leakage and dynamic power in digital integrated circuits. In FPGAs, Static Random Access Memory (SRAM) is the dominant technology used for the configuration bits. Therefore, voltage scaling on the SRAMs can be an effective solution for lowering the power consumption in the configuration memories of FPGAs. However, aggressive voltage scaling causes process-variation-induced failures in the SRAM cells such as read access failures, destructive read failures, hold failures, and write failures [3,4]. Concerns on failures due to process variation in SRAMs become more prominent for submicron regimes. Such failures in the configuration SRAMs undermine the functionality and the connectivity of the implemented designs.

The distinctive feature of FPGA architectures is that it provides a large array of logic resources. The inherent redundancy and the reconfigurable nature of FPGA systems provide a unique opportunity to implement the design in the presence of failures in the configuration SRAMs. Fault tolerance can be maintained when the fault-free unutilized resources are used. A judicial placement around the faulty resources can realize the implementation of the design when the configuration SRAMs are operating at lower voltages. By utilizing the spare resources of the FPGA device and using the a priori distribution of SRAM failures due to process variation, we calculate the minimal supply voltage that sustains a fault-free placement. In this paper, we study the potential of power saving by voltage scaling in the configuration

The work in this paper is partially supported by the NSF under the award CNS-CAREER #0846129.

CODES+ISSS '11, October 9–14, 2011, Taipei, Taiwan.
Copyright 2011 ACM 978-1-4503-0715-4/11/10...$10.00.

memory when coupled with placement, in order to address the failures due to process variation.

Since the exact locations failures due to process variation can *only* be determined after the FPGA has been manufactured and deployed, the final fault-free placement has to be determined after the deployment of the FPGA system. Due to within-die and die-to-die variations, the final fault-free placements are also unique for a particular FPGA chip. Therefore, the performance (dictated by the critical path delay and the wire length) of the implementation of the same design on different FPGA chips will vary. Such uncertainty in the performance metrics might not be tolerated for performance-critical applications. Furthermore, since the placement of the design is done after the system is deployed, it is important that the run-time of the placement algorithm be fast, or otherwise, the design cannot be efficiently used in time-constrained applications.

In our work, we split the task of placement into two phases: pre-deployment placement (Sec. 5.2) and post-deployment placement (Sec. 5.3). We first optimize the reliability of the designs implemented on FPGA in conjunction with the critical path delay and wire length using the a priori probability distribution of SRAM failures due to process variation and voltage scaling. It is important to note that we define reliability here as attempting to ensure a fault free post-deployment placement. The relative locations of the resources are determined in the pre-deployment placement. The main idea is to increase the chances that the faulty resources be locally replaced by the spare ones after deployment. Once the FPGA device is deployed and the fault map of the resources is determined, post-deployment placement is performed. Since the resources are already placed in such a way that faulty resources could be replaced with the spare ones locally, the execution time of our post-deployment placement is very fast. In addition, as the relative locations of the resources are determined once during the pre-deployment placement and the faults are resolved locally by using the spare resources in the vicinity of the faults, the performance metrics of the resultant implementation are very close to the ones obtained in the pre-deployment placement.

To the best of our knowledge, this is the first framework that proposes aggressive voltage scaling of the configuration bits in conjunction with reliability-aware placement to provide drastic leakage savings in the configuration memory of FPGA in the presence of failures due to process variation. Our experiments indicate that by employing voltage scaling, more than 50% improvement in the leakage power consumption of the configuration memory can be achieved, while the implemented designs suffer little, if any, performance degradation. The results also suggest that not only is our proposed algorithm very efficient, but it also generates placements that are more robust against variations in the performance metrics, while the delay degradation is insignificant.

The paper is organized as follows: In Sec. 2 we study the related work. In Sec. 3, we explain the architecture of configuration memory in SRAM-based FPGA. We study the sources of failure in SRAM due to process variation in Sec. 4. In Sec. 5.1, we elaborate on our reliability-aware voltage selection for the configuration SRAMs. In Sec. 5.2(5.3) we study the reliability aware-placement before(after) the deployment of the FPGA system. The experimental results are provided in Sec. 6. We conclude the paper in Sec. 7.

2. Related Work

Voltage scaling has been widely used to lower the power consumption in FPGA logic and routing resources. Dual-Vdd has been used in FPGAs to reduce the power of the resources which are not on the critical paths in the design in order to lower the overall power consumption [5,6]. High-Vth SRAMs have been used to lower the leakage power [6,7], however, such a solution will not reduce the gate leakage, which in low temperatures can account for more than 50% of the total leakage power. In order to effectively reduce leakage power in the configuration memory, our work explores the impact of voltage scaling in the presence of process variation and provides a voltage selection and a placement technique to maintain reliability.

The impact of process variation on timing yield at different stages of the synthesis flow has been studied and optimized in [8]. Novel placement techniques to enhance the timing yield in the presence of process variation have been proposed in [9,10]. In our paper, the main target is to address the reliability aspects of voltage scaling in the presence of process variation for leakage savings in FPGA.

Fault tolerant aggressive voltage scaling on the of SRAM cells has been studied in [3,4] and applied in memory designs in [11]. Our work tries to take advantage of the unique features of FPGA (redundancy and reconfigurability) through placement in order to realize voltage scaling in the presence of process variation.

Fault tolerant placement for FPGA has received a lot of attention in the past decade. These methods usually try to place the design on the fault-free resources using reconfiguration [12,13]. In our work, we provide a novel framework that performs fault-tolerant placement to sustain faults in the configuration SRAMs due to aggressive voltage scaling.

3. Configuration Memory Architecture in SRAM-based FPGA

Modern FPGA devices in general provide an array of logic resources, which may be interconnected, and configured for specific functions. Functionality and connectivity are implemented by Clustered Logic blocks (CLB) and routing blocks. Each CLB contains a number of Block Logic Elements (BLE). Figure 1 depicts the generic structure of a BLE and a CLB. The BLEs implement the logic using *N*-input Look-up Tables (LUTs). The outputs of the LUTs can be programmed to be latched in a flip-flop.

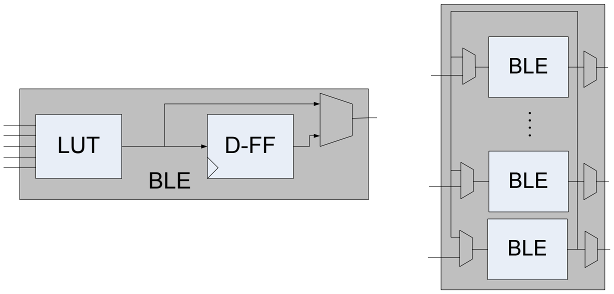

Figure 1: BLE (left) and CLB (right) structure in FPGA architecture

All CLBs and routing blocks are controlled by SRAM cells. Such SRAMs that define the functionality and the connectivity of the implemented designs are referred to as the configuration SRAMs. Throughout this work, we use the terminologies and the features of Xilinx-based FPGA architecture. However, all the concepts provided in this work can be applied to other SRAM-based FPGA architectures (i.e. Altera FPGA devices).

The configuration SRAMs have to be loaded on power-on. If needed, the configuration SRAMs can be reloaded with new values to reflect the changes in the implemented design.

The configuration SRAMs lie close to the CLBs or routing blocks they control and are organized in a regular pattern. The smallest unit of configuration (read/write) is a data frame, which is a 1-bit segment of the configuration SRAMs. A CLB frame controls a portion of a CLB. Figure 2 depicts a CLB frame in the configuration SRAMs. Note that frames lie in a single column of configuration SRAMs.

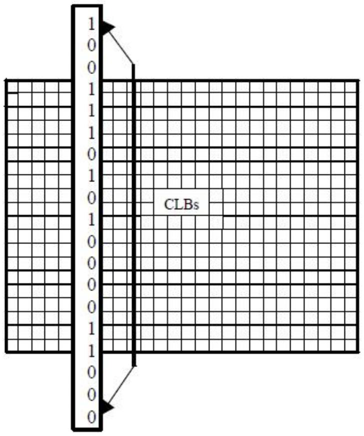

Figure 2: CLB Frame in Xilinx FPGA

The number of CLB frames depends on the device size. The same is true for the number of bits of a CLB frame itself [14].

As mentioned earlier, when voltage scaling is applied, some of the configuration SRAMs might become faulty. In order to detect the faulty SRAMs within the configuration memory, several techniques based on readback feature and dynamic reconfiguration of FPGAs [14,15] have been developed in industry as well as academia. The diagnosis developed in [16,17,18] can be applied to obtain the fault map of the configuration SRAMs. In this paper, we assume that a single error in the configuration bits in a CLB will cause the entire CLB to become erroneous.

4. Impact of Process Variation/Voltage Scaling in Reliability/Power Saving in SRAM

Figure 3 shows the typical six-transistor cell used for CMOS SRAM. The cell consists of two cross-coupled CMOS inverters (NL-PL and NR-PR) that store one bit of information, and two N-type transistors (SL and SR) that connect the cell to the bitlines (BLC and BLT). Process variation-induced failures can be categorized in four major types: Read Access Failure, Write Failure, Destructive Read Failure and Hold Failure (the voltages at node L and R are referred to as $V_L$and V_R respectively)

Read Access Failure (*RAF*): Reading the cell storing $V_L = `0'$ and $V_R = `1',$ begins with precharging BLT and BLC to V_{dd}. Then the wordline, WL, will be set to V_{dd} which results in turning on SL and SR. The ideal case in reading is that NL sinks the current from BLT and in the time $T_{Access} < T_{Max}$ (where T_{Max} is the maximum allowed time to load), a sufficient difference on the bitlines appears which triggers the sense amplifiers to detect the value of the cell. An increase in the access time of the cell during read operation such that $T_{Access} > T_{Max}$ is defined as the Read Access Failure. It has been shown in [11,4] that T_{Access} principally depends on V_{th} of SL and NL and T_{Access} increases with an increase in $V_{th_{SL}}$ and/or $V_{th_{NL}}$.

Write Failure (*WF*): In writing a $`0'$ to the node storing $`1'$ (for example, node L in Figure 3), the voltage at node L (V_L) has to reduce to a value below the trip-voltage of the PR-NR inverter within the time that the wordline is high (T_{Max}). An increase in the T_{Write} of the cell such that $T_{Write} > T_{Max}$ is defined as the Write Failure. T_{Write} increases if the strength of SL reduces ($V_{th_{SL}}$ increases) and/or if that of PL increases ($V_{th_{PL}}$ decreases) [11,4].

Destructive Read Failure (*DRF*): While reading the cell (e.g. $V_L \ = `0'$ and $V_R \ = `1'$) due to voltage divider action between SL and NL the voltage at node L (V_L) will increase to V_{Read}. If due to random variations in the threshold voltages of SL and/or NL, V_L becomes greater than the trip-voltage of PR-NR inverter, the value that the cell is storing will flip and the Read-upset or Destructive Read Failure (DRF) will occur [11].

Hold Failure (*HF*)*:* Inability to hold the cell value at lower voltages than nominal is defined as Hold failure. As a result of hold failure, the cell content will be destroyed.

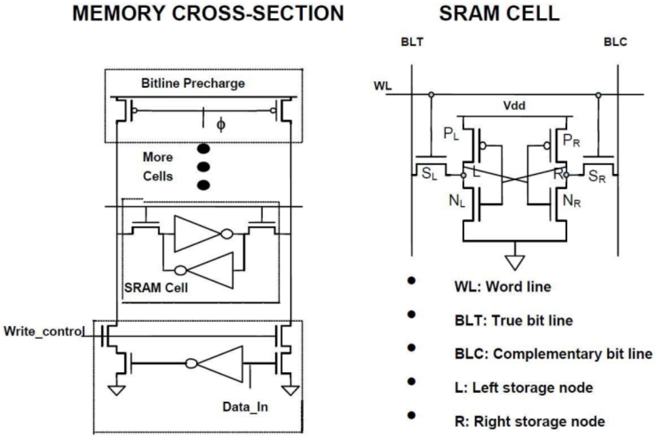

Figure 3: 6T SRAM cell

It is important to note that technology scaling strongly impacts power consumption and delay, while its impact on threshold voltage is a shift in the mean of the distribution with minimal effect on its standard deviation.

In the absence of process variation, the threshold voltage is considered to be a fixed number. However, in the presence of the process variation, V_{th} will follow a distribution which is assumed to be Gaussian with a mean $\mu_{V_{th}} = V_{th0}$ and a standard deviation which can be calculated as [19,20]:

$$\sigma_{V_{th}} = \sigma^0 \ _{V_{th}} * \sqrt{\frac{L_{min}}{L}\frac{W_{min}}{W}} \qquad (1)$$

Where $\sigma^0 \ _{V_{th}}$ is the $\sigma_{V_{th}}$for a minimum sized transistor and is given by:

$$\sigma^0 \ _{V_{th}} = \frac{qT_{ox}}{\varepsilon_0} * \sqrt{\frac{N_a W_d}{3L_{min}W_{min}}} \qquad (2)$$

where *Na* is the effective channel doping, W_d is the depletion region width, T_{ox} is the oxide thickness, L_{min} and W_{min} are the minimum channel length and width respectively. To factor in process variations, one needs to consider the effect of RDF, oxide thickness (T_{ox}) variation, and effective length (L_{eff}) variation on the threshold voltage variation. In advanced technology nodes, the number of the atoms in the channel reduces from a few thousands to a few hundreds. While it is possible to control the average number of the atoms in the channel, N_{avg}, it is almost impossible to assign the exact same number of atoms to each device in the same location. In fact, the number of the atoms in the channel, N, follows a Poisson distribution and can be represented as [21]:

$$f\left(N; N_{avg}\right) = \frac{{N_{avg}}^{N} e^{-N_{avg}}}{N!} \tag{3}$$

Where N_{avg} is the average number of the atoms in the channel. On the other hand, the variation of T_{ox} and L_{eff} can be represented as a Gaussian distribution and are given by [20]:

$$\begin{gathered} L_{eff} \sim N\left(\mu_{L_{eff}}, \sigma_{L_{eff}}\right) \\ T_{ox} \sim N\left(\mu_{T_{ox}}, \sigma_{T_{ox}}\right) \end{gathered} \tag{4}$$

The variation in these three parameters, results in a Gaussian distribution for the threshold voltage V_{th}, which can be expressed in terms of the nominal V_{tho} with the effects of the supply voltage V_{dd} and the source-to-body voltage V_{SB} reflected through the drain induced barrier lowering (DIBL) and body effects, as shown in the following equation:

$$V_{th} = V_{th0} + \gamma\left(\sqrt{\Phi_s - V_{sb}} - \sqrt{\Phi_s}\right) - \theta_{DIBL} V_{dd} + \Delta V_{NW} \tag{5}$$

where V_{th0} is the zero-bias threshold voltage, θ_{DIBL} represent the DIBL effect, V_{SB} is the body voltage, ΔV_{NW} models the narrow width effect, γ is the body effect coefficient and Φ_s is the surface potential.

From these fundamental equations it can be shown that the first and second moments of the Gaussian distribution for the threshold voltage are a linear function of $\mathrm{V_{dd}}$ and can be expressed as follows:

$$\begin{gathered} V_{th} \sim N\left(\mu_{V_{th}}, \sigma_{V_{th}}\right) \\ \mu_{V_{th}} = aV_{dd} + b \\ \sigma_{V_{th}} = c \end{gathered} \tag{6}$$

The constants $a, b,$ and c are fixed for a given technology and only need to be found once by either Monte Carlo simulations or polynomial fitting. For example, for a 32nm technology (using PTM [22]), the following data set can be generated:

$$\begin{gathered} V_{dd} \in [0, +1.0] \\ a = -0.116,\ b = 0.453 \\ c = 0.04 \end{gathered} \tag{7}$$

In order to find the probability of failure for the SRAM cell, the distribution of access time, T_{Access} , write time, T_{Write} and the storing node voltage, $\mathrm{V_{R/L}}$ needs to be calculated as a function of supply voltage , $\mathrm{V_{dd}}$. To do so, one can represent the access time as:

$$\begin{gathered} V_{th} \sim N(\mu_{V_{th}}(V_{dd}), \sigma_{V_{th}}(V_{dd})) \xRightarrow{\forall(V_{dd})} \\ T_{Access} \sim Dist.(\mu_{T_{Access}}(V_{dd}), \sigma_{T_{Access}}(V_{dd})) \end{gathered} \tag{8}$$

For a given maximum allowed access time, T_{Max}, according to the failure definitions, we will have:

$$\begin{aligned} P_e(V_{dd}, T_{Max}) &= P[T_{access} > T_{Max}] \\ &= Q(\tau)|_{\tau=t} \end{aligned} \tag{9}$$

Where

$$t = \frac{T_{Max} - \mu_{T_{Access}}(V_{dd})}{\sigma_{T_{Access}}(V_{dd})} \tag{10}$$

and $Q(\cdot)$ is the Gaussian Error Integral or Q −function and is given by:

$$Q(\tau) = \frac{1}{\sqrt{2\pi}} \int_{\tau}^{+\infty} e^{\left(-\frac{x^2}{2}\right)} dx. \tag{11}$$

For this study, we assume the write operation is performed at the nominal voltage and therefore we exclude the write failure from the total probability of failure. Similar analysis can be done for storing node voltage distribution and the hold failure and destructive read failure can be calculated accordingly. Thus, the total probability of failure can be calculated as

$$P_e = P[RAF \cup DRF \cup HF] \tag{12}$$

5. Reliability-aware voltage scaling and placement in the presence of process variation

So far, we have studied the impact of voltage scaling on the reliability of SRAM cells in the presence of process variation. In this section, we first study the impact of voltage scaling on the reliability of the implemented design across the whole FPGA area. Depending on the design to be implemented on FPGA, we select the proper voltage range that sustains the reliability constraints. Once the proper voltage has been selected, we attempt to maintain the high reliability through placement, so that CLB faults due to process variation can be confined locally. Once the FPGA device is deployed and the fault map is generated for the FPGA device, the local placements can be modified and the updated placement can be reconfigured on FPGA. In this section, we will study our framework in details. Since the operating condition (such as supply voltage) dependent failure in SRAMs are due to mismatches in the device strength and only intra-die variation causes mismatch, we assume that the faults caused by reducing the supply voltage are independent. This means that the probability of failure for all SRAMs for a given supply voltage V_{dd_0}, is equal to P_{e0} and are independent of each other.

5.1 Reliability-aware application-based voltage Selection for maximum power saving

As mentioned in Sec. 4, voltage scaling can save a significant amount of power at the expense of reliability in the presence of process variation in SRAM cells. In the context of CLB configuration SRAM cells, such faults will manifest themselves as functional errors of the implemented design. Therefore, it is desirable to formulate the extent of such faults based on the supply voltage applied to the SRAM cells. Throughout this section, we refer to the probability of errors in a single SRAM cell due to process variation for a certain supply voltage (V) as $P_e^{bit}(V)$.

Given the number of configuration SRAM cells (n) used to program a single CLB, we can calculate the probability of error for a single CLB as:

$$P_e^{CLB}(V) = 1 - (1 - P_e^{bit}(V))^n \cong n.P_e^{bit}(V) \qquad (13)$$

In the equation above, we count a fault in a single SRAM cell as an error in the whole CLB. Since the probability of error for a single SRAM cell is minute in comparison to the number of configuration bits for the CLB, we can use a first-order approximation to simplify Eq. (13).

The implemented designs on FPGA rarely occupy the whole resources available on chip. Since FPGA architecture comprises a large array of CLBs, it is very likely that faults in the configuration SRAMs due to process variation could be prevented from emerging as errors in the design through relocation of the faulty CLBs (reconfiguration) to the spare resources. Of course, the opportunities for having an error-free design on FPGA depends on the number of available CLBs on the FPGA dedicated to the design (N_R), the number of erroneous CLBs (N_E), the number of CLBs utilized by the design (N_U) and the number of spare CLBs ($N_S = N_R - N_U$). The following equation formulates the reliability of the implemented design against process variation:

$$R_G = \Pr\{N_E \le N_S\} =$$

$$\sum_{i=0}^{N_S} \binom{N_S}{i} . P_e^{CLB}(V)^i . (1 - P_e^{CLB}(V))^{N_S - i} \qquad (14)$$

We refer to the probability derived above as the *global reliability*. In Eq. (14), we have applied the binomial expansion to calculate the reliability. Note that Eq. (14) enumerates all the cases in which the errors in CLBs can be recovered through reconfiguration. In order to find the proper supply voltage range for the configuration SRAM cells for a given global reliability, we take the inverse of Eq. (14):

$$V_{min} = f^{-1}(R_G, N_R, N_U) \qquad (15)$$

Using Eq. (15), we find the lower bound of the supply voltage that can sustain the design with the given global reliability value. We use V_{min} to eliminate the voltages resulting in intolerable faults in the design.

Global reliability indicates the probability of recovery when all the CLBs in the designated area on FPGA are for granted. It does not discriminate on where to place the faulty CLBs. However, the replacement of CLBs can increase the critical path delays in the design. In order to consider the connectivities between CLBs in the implementation, we introduce a pre-deployment placement method to plan the placement in such a way that minimal increases in delay will be incurred in case relocations are needed.

5.2 Reliability-aware pre-deployment placement in the presence of process variation

Conventionally, the main objective of placement optimization algorithms has been the reduction of the wire length and/or the critical path delay of the implemented design [23]. These algorithms do not take into consideration the potentials of faults in CLB configuration SRAMs. In this section, we present a reliability-aware placement of the CLBs on FPGA based on the a priori fault distribution due to process variation. We refer to this placement as the pre-deployment placement as the actual fault locations on the FPGA fabric has yet to be determined.

As mentioned earlier, global reliability (Eq. (14)) is used to determine the reliability of the implemented design when all the spare CLBs can be reconfigured to avoid the faulty ones. However, the critical path delay of the fault-free design might become unacceptable. Note that we are interested in local relocations with minimal changes to the placement once the system is deployed. Therefore, we introduce the concept of *detailed reliability*, which indicates the potentials to replace the faulty CLBs with the fault-free ones in the close vicinity of the fault. A simple example is depicted in Figure 4. The CLBs of the design are labeled as A, B, C and D. In this example, we assume that A has a direct connection with B and C and there is a connection between C and D. Since the number of spare CLBs and the number of utilized CLBs are the same for the two placements, the global reliability values for the two placements are the same. However, once the faulty CLB is discovered, the relocation of A will incur an increase in the critical path delay in Figure 4 (a), whereas in Figure 4 (b), there is no delay penalty in relocating A.

Instead of formulating the reliability for the entire FPGA as a whole, we introduce a model which captures the reliability locally. We use a grid to divide the CLBs into several grid cells. Note that the grid has no actual hardware implications and it is only used for the purpose of reliability computation. The *detailed reliability* of the FPGA is then defined as the combination of the reliabilities of the individual grid cells:

$$R_D^k = \sum_{i=0}^{N_S{}^k} \binom{N_S{}^k}{i} . P_e^{CLB}(V)^i . (1 - P_e^{CLB}(V))^{N_S{}^k - i} \qquad (16)$$

$$R_D = \prod_{k=1}^{grid\_cells} R_D^k \qquad (17)$$

In Eq. (16), we calculate the reliability of a single grid cell based on the number of erroneous CLBs ($N_E{}^K$), the number of total CLBs within the grid cell ($N_R{}^K$), the number of utilized CLBs in the grid cell ($N_U{}^K$) and the number of spare CLBs ($N_S{}^K = N_R{}^K - N_U{}^K$), provided that all the CLB relocations have to take place within the same grid cell. In Eq. (17), we calculate the total reliability of the design implemented on the FPGA. It is easy to verify that the local reliability value for Figure 4 (b) is greater than that of Figure 4 (a). Accordingly, the faulty CLB at the bottom-right CLB can be relocated within the same grid cell as depicted in Figure 4 (b). The main advantage of the detailed reliability formulation is that it can be easily incorporated into the conventional placement tools, which are based on simulated annealing. In order to simplify Eq. (17), we rewrite it as:

$$R_D{}' = -\ln(R_D) = \sum_k -\ln R_D^k \qquad (18)$$

Note that the minimization of $R_D{}'$ will in fact maximize the detailed reliability. Eq. (18) is easier to be implemented into the simulated annealing engine. The fundamental move performed on the CLBs in simulated annealing engines is swapping. Based on the cost function associated which each move, the annealing engine decides whether to reject or accept the swapping. It is crucial that the cost function used to evaluation the benefit of the move be calculated very efficiently, or otherwise it would take a long time to obtain the final placement. Interestingly, detailed reliability computation (Eq. (18)), can be performed in $O(1)$ time:

***Theorem* I.** *The time complexity of the calculation of the detailed reliability upon CLB swapping is O*(1).

Proof. The two CLBs to be swapped either lie on the same grid cell (e.g. CLBs C and D in Figure 4 (a)), or they belong to two different grid cells (e.g. CLBs C and D in Figure 4 (b)). In case the swapping does not change the utilization of the grid cells, the detailed reliability remains the same. Also, when we swap the contents of two occupied CLBs in two different grid cells, the detailed reliability stays unchanged. The only time there is an increase/decrease in the detailed reliability is when we relocate a CLB in one grid cell (denoted by *i*) to a spare location in another grid cell (denoted by *j*). In this case, we can calculate the detailed reliability difference (*i-new* and *i-old* refer to the grid cells *i* after and before the swapping, respectively) as:

$$\Delta R'_D = \ln(R_D^{i-NEW}) + \ln(R_D^{j-NEW}) - \ln(R_D^{i-old}) - \ln(R_D^{j-old}) \quad (19)$$

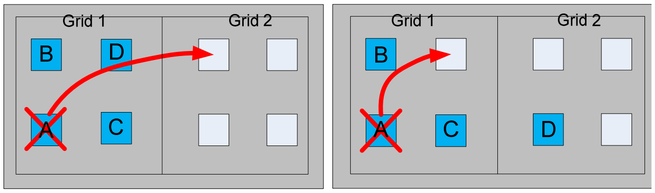

Figure 4: (a) sample placements when detailed reliability is not used (b) when detailed reliability is used

Rather than recalculating the detailed reliability for each grid cell at every swap, we calculate the detailed reliability of each grid cell once for all the possible utilizations, from 0 up to the number of CLBs in the grid cell (i.e. 4 for the grid cells in Figure 4), given the a priori probabilities of CLB reliability. Assuming that the detailed reliability for the initial placements has been calculated at the beginning, it takes *O*(1) to calculate the difference in the detailed reliability □

We have developed a simulated annealing search engine which simultaneously optimizes the critical path delay, wire length and the detailed reliability. In our annealing engine, we have adopted the adaptive annealing schedule of VPR, the state-of-art FPGA placement tool. Features of VPR adaptive annealing schedule are explained in details in [23]. We used the following cost function for our simulated annealing engine:

$$\Delta cost = \alpha \cdot \frac{\Delta R'_D}{R'^{\,PREV}_D} + (1-\alpha)\left(\beta \frac{\Delta Delay}{Delay^{PREV}} + (1-\beta)\frac{\Delta wl}{wl^{PREV}}\right) \quad (20)$$

Where α is the coefficient denoting the tradeoff between the combination of the critical path delay, the wire length, and the detailed reliability. Details on critical path delay calculation and wire length computation inside the simulated annealing engine have been provided in [23].

5.3 Reliability-aware post-deployment placement in the presence of process variation

In order to have an error-free design on the FPGA once the FPGA device is deployed, the fault map for the CLBs must be determined. For the remainder of this section, we assume that fault diagnosis techniques have been applied on the FPGA [16,17,18] and the fault map is available.

The basic idea underlying our solution is to obtain new placements for each grid cell on the FPGA rather than trying new placements for the entire FPGA chip. Since we have already planned and optimized the placements in the pre-deployment placement phase of our proposed solution, we are able to modify the placement in a finer granularity. We can avoid the faulty CLBs by changing the placement of the grid cell in which the faulty CLB lies on. As the relative locations of the CLBs have been planned during the pre-deployment phase, replacing CLBs within a grid cell will have negligible impact on the performance of the design. On the other hand, we can save a great deal of time optimizing the placements for individual grid cells on the FPGA as opposed to optimizing the placement on the FPGA as a whole. Therefore, in our post-deployment placement technique, we perform placements on the individual grid cells independently to obtain fault-free placements with optimized performance (wire length and critical path delay). An alternative to our proposed solution is to apply the conventional placement techniques (e.g. simulated-annealing based) to obtain the fault-free placements, once the fault map of the FPGA device is known. However, as we demonstrate later in the section, our solution will always find the fault-free placement more efficiently (faster convergence).

The highlights of our reliability-aware post-deployment placement in the presence of process variations are shown in Figure 5. The solution starts by finding the minimal dimensions of the grid cells that can sustain a fault free placement, given the initial placement (obtained by pre-deployment placement) (FMGCD) and the fault map. Then, for each individual grid cell, we place the CLBs using the simulated annealing engine in such a way that no CLB designated within the grid cell is assigned to a faulty resource (PDP).

In order to clarify how the size of the grid cells influence the behavior of the post-deployment placement, refer back to the example in Figure 4. As depicted in Figure 4(a), a 2 × 2 grid cell cannot sustain a fault-free placement. The reason is that all the four CLBs in the 2 × 2 grid cell on the left are utilized and there is a faulty CLB to be avoided. Therefore, no arrangement of CLBs within the grid cell will yield a fault-free placement. However, as depicted in Figure 4(b), a 2 × 2 grid cells can sustain a fault-free placement. In this case, there will be two 2 × 2 grid cells: one on the left with four available CLBs, three utilized and one faulty and another on the right with four available CLBs, one utilized and no faulty one.

The function FMGCD (Figure 5) starts with 2 × 2 grid cells and given the initial placement and the fault map, examines each grid cell to figure out whether a fault-free placement can be sustained depending on the number of available CLBs in the grid cell, the number of erroneous CLBs and the number of CLBs utilized within the grid cell (line 11). If 2 × 2 grid cells cannot sustain a fault-free placement, the algorithm tries 3 × 3 grid cells and eventually if no smaller grid cell size can sustain a fault-free placement, the whole FPGA is treated as a single *M* × *M* grid cell (without loss of generality, the FPGA is assumed to be an *M* × *M* array of CLBs). The boundaries of each grid cell are computed in lines 4-11. The function outputs the minimal grid cell dimensions (line 14) or returns *M* (line 15).

```
Function Find-Minimal-Grid-Cell-Dimension (FMGCD)
Input: Initial Placement(IP), fault map (F), FPGA dimensions ($M \times M$)
Output: Minimal grid cell dimensions ($i \times i$)
[1]  for $i \leftarrow 2$ to $M$ {
[2]     $g.area \leftarrow i \times i$
[3]     minimal-grid-cell-dimension ← true
[4]     for $x \leftarrow 1$ to $M / i$
[5]        for $y \leftarrow 1$ to $M / i$ {
[6]           $g.x\_min \leftarrow x \times ((M / i)-1) + 1$
[7]           $g.y\_min \leftarrow y \times ((M / i)-1) + 1$
[8]           $g.x\_max \leftarrow g.x\_min + M / i - 1$
[9]           $g.y\_max \leftarrow g.y\_min + M /i - 1$
[10]          $g \leftarrow IP(g.x\_min, g.y\_min, g.x\_max, g.y\_max)$
[11]          if g.num-faults > g.area - g.num-utilized-CLBs
[12]             minimal-grid-cell-dimension ← false}
[13]    if minimal-grid-cell-dimension == true
[14]       return i}
[15] return M
Function Post-Deployment Placement (PDP)
Input: Initial Placement(IP) , fault map (F), FPGA dimensions ($M \times M$)
Output: Fault-free placement (P)
[16] $i \leftarrow$ FMGCD(IP, F, M)
[17] for $x \leftarrow 1$ to $M / i$
[18]    for $y \leftarrow 1$ to $M / i$ {
[19]       $g.x\_min \leftarrow x \times ((M / i)-1) + 1$
[20]       $g.y\_min \leftarrow y \times ((M / i)-1) + 1$
[21]       $g.x\_max \leftarrow g.x\_min + M / i - 1$
[22]       $g.y\_max \leftarrow g.y\_min + M /i - 1$
[23]       P.g ← fault-free-simulated-annealer(g) }
[24] return P
```

Figure 5: Outline of reliability-aware post-deployment placement in the presence of process variation

Our post-deployment placement (function PDP in Figure 5) first calls FMGCD to find the minimal grid cell dimensions (line 16). Then, once the boundaries of the grid cells are determined (lines 19-22), the fault-free placements of the CLBs within a grid cell are calculated using a simulated annealing-based placement algorithm. Our simulated annealing search engine simultaneously optimizes the critical path delay and the wire length. In our annealing engine, we have adopted the adaptive annealing schedule of VPR [23]. The distinctive feature of our annealing engine is that we only try the legal moves within the grid cells. A legal move is a move (CLB swapping) that does not utilize a faulty CLB in case a CLB is moving from a location to a new one. The cost function to be minimized is the same as the one used in VPR [23]:

$$\Delta cost = \beta \frac{\Delta Delay}{Delay^{PREV}} + (1-\beta)\frac{\Delta wl}{wl^{PREV}} \qquad (21)$$

Note that since we have performed the preprocessing in FMGCD and found the minimal grid cell dimensions, we always obtain a fault-free placement. Our post-deployment placement in fact tries to optimize the performance by trying new placements locally (within grid cells), rather than considering the whole FPGA. The main advantage of modularizing the problem of fault-free post-deployment placement can be determined through the mathematical analysis provided in the rest of this section.

***Theorem* II.** *The worst case time complexity of* FMGCD *is* $O(n^{3/2})$*, where n refers to the number of CLBs in the design.*

Proof. As shown in Figure 5, in the worst case, we need to examine the grid cells with all the possible sizes (line 1) to find out the minimal grid cell size. Therefore, we can formulate the run-time of FMGCD as:

$$\sum_{i=2}^{M}(M/i)^2.i^2 = \sum_{i=2}^{M} M^2 = O(M^3) = O(n^{3/2}) \qquad (22)$$

In the equation above, since we need to check every CLB within the grid cell to determine if it is faulty or not, and there are M^2/i^2 grid cells of size $i \times i$, we can conclude that the time complexity of the algorithm is $O(M^3)$. Since the number of CLBs is often comparable to the area of the FPGA device (M^2), we can equate the formulation above to $O(n^{3/2})$ □

It is important to note that the average time complexity of FMGCD depends on the detailed reliability of the placement (Eq. (17)) and the grid cell size (k × k) used to calculate the detailed reliability in the pre-placement algorithm. In fact, the probability that the grid cell sizes returned by FMGCD be equal to the grid cell size (k) used in the pre-placement algorithm is equal to the detailed reliability.

$$\Pr\{\mathrm{FMGCD}(IP, F, M) \le k\} = R_D \qquad (23)$$

For example, if the detailed reliability of the pre-deployment placement is 0.9 and the grid cell sizes used to compute the detailed reliability is 2 × 2, then for 90% of the time, FMGCD will return 2, which is much faster than the worst case. Using the equation above, we can rewrite Eq. (22) as:

$$\sum_{i=2}^{k}(M/i)^2.i^2 = \sum_{i=2}^{k} M^2 = O(M^2.k) = O(k.n) \qquad (24)$$

Note that the probability that the run time of FCMGD is as expressed in Eq. (24) is equal to the detailed reliability (R_D). Given that the detailed reliability is maximized in pre-deployment placement, the run-time of FMGCD is almost a linear function of *n*.

Our post-deployment fault-free placement algorithm (Figure 5) uses simulated annealing in order to find the fault-free placements within each grid cell. The time complexity of simulated annealing search algorithms generally depends on the annealing schedule. The annealing schedule specifies the number of moves to attempt per temperature, how annealing temperature varies throughout the annealing, and when the annealing should terminate. In our post-deployment placement algorithm, we adopted the same annealing schedule as the one described in [23]. The number of moves per each annealing temperature is in the order of $O(n^{4/3})$, where *n* is the number of CLBs in the design. In order to compare and contrast our post-deployment placement with the alternative placement which uses simulated annealing on the whole FPGA device, we provide the following theorem:

***Theorem* III.** The number of moves per annealing temperature in the PDP solution is asymptotically smaller than the number of moves per temperature in conventional simulated annealing placements.

Proof. The number of moves per temperature can be formulated as $O(n^{4/3})$. Now, since we are confining our algorithm to find the placements within the grid cells, and the area of each cell is $i \times i$ (line 16 in Figure 5), the number of utilized CLBs within the grid cells are at most i^2. The number of grid cells in the FPGA is proportional to (M^2/i^2), therefore we can formulate the number of moves per temperature for the post-deployment placement as:

$$O((M^2/i^2).(i^2)^{4/3}) = O(n.i^{2/3}) \ll O(n^{4/3}) \qquad (25)$$

Note that in the worst case when the whole FPGA becomes one grid cell ($i = M$), the number of moves per temperature for the two techniques become the same. However, since in our pre-deployment placement (Sec. 5.2), we have already planned the relative locations of CLBs and have considered the potentials of faults in the neighborhood of CLBs, the dimensions of the grid cells (i) is much smaller than the dimensions of the FPGA itself (M). Therefore, our post-deployment placement always performs faster than the conventional placement □

It is worth mentioning that not only does our post-deployment placement reduce the number of moves per annealing temperature, it also drastically reduces the range of moves in the simulated annealing. Since the relative locations of the CLBs have already been planned in the pre-deployment placement, our post-deployment placement only fine-tunes the locations of the CLBs within the grid cells. Therefore, the simulated annealing engine used in our post-deployment placement tends to converge faster to the fault-free solution.

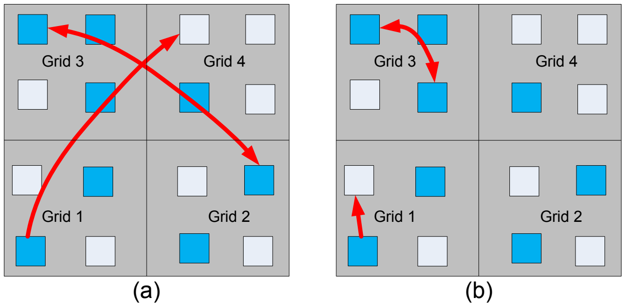

Figure 6: Simulated annealing moves in (a) the conventional placement and (b) post-deployment placement

Figure 6 demonstrates the difference in the range of moves attempted in our framework in comparison to conventional placements. As shown in Figure 6(a), the moves range across the FPGA when conventional simulated annealing placements are used. However, the moves in our post-deployment placement are confined to the grid cells. Therefore, post-deployment placement reaches the final solution faster. Note that the relative locations of CLBs are planned during pre-deployment placement.

6. Experimental Results

In this section we first explain the experimental flow used to evaluate our reliability-aware placement solution for voltage scaling in the presence of process variation. Then, we present the results of our proposed technique in details.

6.1 Experimental Flow

The experimental flow used in this work is depicted in Figure 7. As the first step, we generate the libraries which contain information about the reliability of the SRAM used in the configuration bits in FPGA. In this work, we ran Monte-Carlo simulations on 32nm SRAM technology to obtain the distribution functions for different SRAM failures due to process variation. Predictive Technology Models (PTM [22]) are used for the device models in HSPICE simulation. Then the distribution of read time, hold time and storing node voltage is obtained. For each Monte Carlo run, 5000 points are simulated to find the distributions. Then the probability of failure is calculated based on the equations presented in Sec. 4.

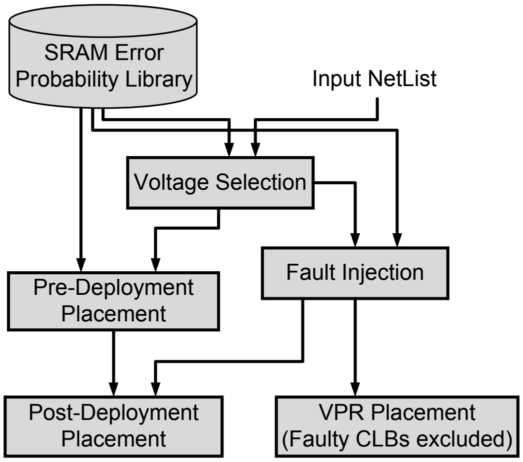
SRAM Error Probability Library
Input NetList
Voltage Selection
Pre-Deployment Placement
Fault Injection
Post-Deployment Placement
VPR Placement (Faulty CLBs excluded)

Figure 7: Experimental Flow

Once the libraries are ready, depending on the number of utilized CLBs for the FPGA device and the yield point (reliability probability) of desire, the minimal supply voltage for the configuration SRAMs is selected. In our experiments, the architecture of the configuration bits is assumed to be similar to architecture of Virtex II devices (128 bits for CLBs). We used MCNC benchmarks for the experiments. In order to keep the utilization of the FPGA high, in our experiments, for each benchmark, the FPGA area is assumed to only accommodate 10% and 30% more CLBs than the number of CLBs utilized by the benchmark.

Based on the number of available CLBs in the FPGA the yield point and the number of utilized CLBs, the minimal permissible supply voltage is selected for the configuration bits (Sec. 5.1.).

Once the minimal safe supply voltage is selected based on the global reliability constraints (Sec. 5.1.), we use the detailed reliability metric, expressed in Eq. (19) inside our reliability-aware pre-deployment placement algorithm (Sec. 5.2). In the cost function used in our pre-deployment placement algorithm, we set the weights α and β to be 0.5 (Eq. (20)). We used 2×2 grid cells to compute the detailed reliability metric for our pre-deployment placement algorithm.

Our reliability-aware post-deployment placement is applied once the FPGA device is deployed and the actual fault map is present. We generated 100 fault maps based on the SRAM error probabilities for the configuration bits on the design. Note that in our fault map, we mark a CLB as faulty if it contains at least a faulty bit in its CLB frame.

In order to measure the performance of our reliability-aware post-deployment placement technique, we run post-deployment placement for each fault map and record the performance metrics (wire length and delay), the leakage power savings, the number of successful placements as well as the execution time of the post-deployment placement technique. In order to highlight the benefits of our framework, we use a modified version of VPR [23], which only tries fault-free CLBs for the moves inside the simulated annealing engine. All the placement techniques are carried out on the same machine, a 2.99 GHz Pentium IV with 1 GB of RAM running Microsoft Windows XP.

6.2 Experimental Results

In the first set of experiments, we find the minimal supply voltage permissible for a given reliability constraint. The nominal supply voltage is set to be 1 v. We reduce the supply voltage by increments of 0.05. For each supply voltage, we measured the probability of error due to process variation for SRAM, CLB, and the whole design when the FPGA is accommodating 10% and 30% extra CLBs for the design. In order to calculate the probability of error for the whole design, we used the formulation of global reliability Eq. (14) and complemented the probability to obtain the probability of error for the design. Figure 8 presents the results obtained for the benchmark ALU4 from the MCNC suite. The same trend is seen for all the other MCNC benchmarks.

The minimal safe supply voltage is calculated based on the extra area available for the benchmarks. As mentioned in Sec. 5.1, we use our model as formulated in Eq. (15) to calculate the minimal safe voltage for the given reliability constraint. The maximum probability of error for the designs is assumed to be 10^{-2}. As shown in Figure 8, we highlighted the limit of 10^{-2} with a dashed line. The results in Figure 8 indicate that 0.8 v is a suitable supply voltage that realizes the design with a probability of error of 10^{-2}. Note that when we have more spare CLBs in our design, the design becomes more reliable.

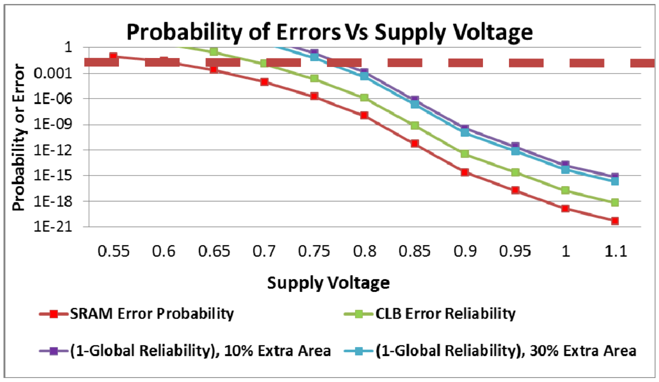

Figure 8: The impact of voltage scaling on the probability of errors due to process variation for SRAM, CLB and the whole design

In Figure 9, we report the leakage power consumed by ALU4 for different supply voltages and different FPGA areas. The leakage reported in Figure 9 is relative to the leakage of the FPGA accommodating the benchmark with the minimal area. We observe the same trend for all the other MCNC benchmarks. The results indicate that by using voltage scaling, we drastically save on leakage. The leakage power savings are more for the case with 10% extra area, since more configuration bits are used in the FPGA. On the other hand, more spare CLBs will allow more supply voltage reduction. Therefore, in order to find the right combination of area, reliability and power saving, all the parameters have to be considered simultaneously.

Once the supply voltage for the configuration bits is determined, we use our pre-deployment placement (Sec. 5.2). We then inject faults in the CLBs according to the probability of errors in our library for specific supply voltages. We perform two placement methods to obtain fault-free placements: our post-deployment placement (Sec. 5.3) and a variant of VPR which excludes faulty CLBs in the random moves. For each benchmark, we generate 100 fault maps and evaluate the performance of the two techniques. In Table 1 we report the averages and the standard deviations of the performance metrics obtained by running the placement techniques on the 100 fault maps generated for each benchmark. Note that all the values reported in Table 1 are relative to the results obtained by VPR under the same conditions, when the FPGA is fault-free. The third and fifth rows of Table 1 report the relative average critical path delay and the relative wire length respectively. These numbers are gathered for all the MCNC benchmarks and they are averaged out. We also report the standard deviation of the critical path delay and the wire length related to the average values (the fourth and sixth row in Table 1).

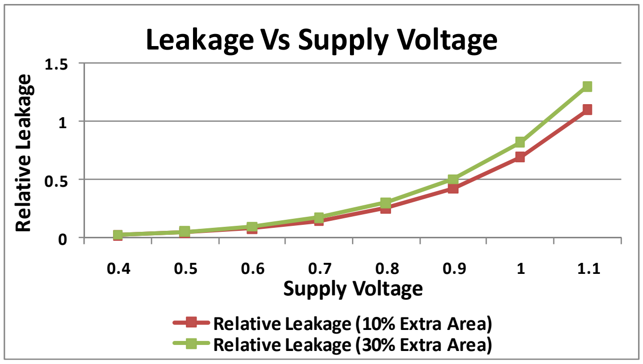

Figure 9: Relative Leakage for different supply voltages for SRAM

As shown in Table 1, there is a little degradation in the delay (D) and the wire length (WR) of the pre-deployment placement when compared with VPR. The main reason is that in our pre-deployment placement we also try to maximize the detailed reliability (Sec. 5.2) in the cost function. However, such little degradation is justified when we compare the results of our post-deployment placement with the results obtained from the VPR with faulty CLBs excluded. The average critical path delays and the wire lengths are very close for the two techniques. However, as the standard deviations reported for the critical path delays and the wire lengths suggest, the critical path delays and the wire length are more predictable when our post-deployment placement is used. The main reason is that in our post-deployment placement, we only try to resolve the CLB faults by replacing CLBs within grid cells, rather than drastically change the placement globally (as done in VPR when faulty CLBs are excluded). Therefore, we expect to see that the results obtained by our post-deployment placement to be closer to the average values of the critical path delay and the wire length, which makes our proposed framework more robust and much faster when compared to the other placement solution (VPR when faulty CLBs excluded).

We reported the run time of our proposed framework in Table 1. There is degradation in the run time of our pre-deployment placement compared to VPR. However, when our pre-deployment placement is coupled with our post-deployment placement, the main speed-up advantage of our proposed framework is clear. While the variant of VPR which excludes faulty CLBs takes more time to execute, our post-deployment placement technique is very fast. Note that post-deployment placement is run for each deployed system, and therefore the run time of the post-deployment placement is critical to ensure an acceptable performance for the applications running on the deployed system. The drastic improvement in the run time of our post-deployment placement is in fact in compliance with the mathematical analysis provided in 5.3. As the designs become larger and more complex, the improvement in run time becomes even more prominent.

We also estimated the leakage savings realized by voltage scaling. Since there is a significant number of SRAMs in the design, the variations in the leakage power due to process variations for the SRAMs fade away. The leakage savings are close to the nominal leakage saving as reported in Figure 9. We observe that more than 55% and 50% improvement in the leakage power is reached when the FPGA accommodates 10% and 30% more CLBs. The results indicate that voltage scaling can be used as a useful means to lower the increase of leakage power in new technologies for FPGA devices. While process variation is an inherent barrier against effective voltage scaling in SRAMs, due to the high levels of redundancy in FPGA, the extra resources can compensate for the higher probabilities of errors incurred by voltage scaling. Our proposed placement framework will realize fault-free designs in the presence of process variation with voltage scaling.

Table 1. Comparison of different placement techniques, relative to VPR

		Pre-Deployment		**Post-Deployment**		**VPR (faulty CLBs excluded)**	
		10%	**30%**	**10%**	**30%**	**10%**	**30%**
D	**μ(%)**	8.7	10.1	11.5	11.3	10.2	11.7
	σ/μ	N/A	N/A	1.2	1.1	1.9	1.7
WR	**μ(%)**	5.6	8.9	7.8	10.4	8.5	10.3
	σ/μ	N/A	N/A	1.4	1.3	1.6	1.8
Run Time (%)		23.2	32.3	-65.3	-76.2	12.7	15.3

7. Conclusion

In FPGA devices, the configuration memory is contributing a significant amount of leakage power consumption. In this paper, we propose to exploit the abundance of homogenous resources on FPGA, in order to realize voltage scaling in the presence of process variation. We introduce a novel 2-phase placement algorithm that maximizes the reliability of implemented design. In the first phase, we maximize the reliability of the designs implemented on the design considering the a priori distribution of SRAM failures due to process variation and voltage scaling. The second phase is performed once the device is fabricated. The second phase determines a fault-free placement of the design for the FPGA device. Our results indicate significant leakage power reduction (more than 50%) in the configuration memory when voltage selection is combined with our placement technique, while the integrity of the design is maintained, with little delay degradation.

References

[1] International Technology Roadmap for Semiconductors, www.itrs.net. 2008.

[2] T. Tuan, et al., "A 90-nm Low-Power FPGA for Batery-Powered Applications," *IEEE TCAD*, vol. 26, no. 2, Feb. 2007.

[3] Q. Chen, et al., "Modeling and testing of SRAM for new failure mechanisms due to process variations in nanoscale CMOS," in *IEEE VTS*, 2005.

[4] S. Mukhopadhyay, H. Mahmoodi, and K. Roy, "Modeling of failure probability and statistical design of SRAM array for yield enhancement in nanoscaled CMOS," *IEEE TCAD*, vol. 24, no. 3, 2005.

[5] Fei Li, et al., "FPGA Power Reduction Using Configurable DualVdd," in *DAC*, 2004.

[6] Altera, White Paper, "Stratix III programmable Power".

[7] A. Gayasen, et al., "Reducing leakage energy in FPGAs using regionconstrained," in *Int. Symp. FPGA*, 2004.

[8] S. Garg, et al., "System-level throughput analysis for process variation aware multiple voltage-frequency island designs," *ACM TODAES*, vol. 13, no. 4, 2008.

[9] A. A. M. Bsoul, et al., "Reliability- and Process Variation-Aware Placement for FPGAs," in *DATE*, 2010.

[10] G. Locus, et al., "Variation-Aware Placement for FPGAs with Multi-cycle Statistical Timing Analysis," in *ACM FPGA*, 2010.

[11] Khajeh, A., et al., "Cross Layer Error Exploitation for Aggressive Voltage Scaling," *8th International Symposium on Quality Electronic Design, ISQED*, pp. 192-197, Mar. 2007.

[12] A. Mathur, C. L. Liu, "Timing-Driven Placement Reconfiguration for Fault Tolerance and Yield Enhancementin FPGAs," in *DATE*, 1996.

[13] Amit Agarwal, et al., "Fault Tolerant Placement and Defect Reconfiguration for nano-FPGAs," in *ICCAD*, 2008.

[14] XAPP151, Xilinx Application Note, "Virtex Configuration Architecture Advanced Users' Guide".

[15] XAPP093, Xilinx Application Note, "Dynamic Reconfiguration," 1997.

[16] E. Bareisa, et al., "Testing of FPGA Logic Cells," in *Elektronika IR Elektrotechnica*, 2004.

[17] S. Lu,et al., "Fault Detection and Fault Diagnosis Techniques for LookupTable FPGAs," in *VLSI Design*, 2002.

[18] C. Wu, et al., "Fault Detection and Location of Dynamic Reconfigurable FPGAs," in *International Symposium on VLSI Technology, Systems, and Applications*, 1999.

[19] Kulkarni, S.H.; Sylvester, D.; Blaauw, D., "A Statistical Framework for Post-Silicon Tuning through Body Bias Clustering," *Computer-Aided Design, ICCAD*, pp. 39-46, 2006.

[20] Y. Taur and T. H. Ning, *Fundamentals of Modern VLSI Devices*. Cambridge University Press, 1998.

[21] H. S. Wong, et al., "Discrete random dopant distribution effects in nanometer-scale MOSFETs," *Microelectronics and Reliability*, vol. 38, no. 9, pp. 1447-1456, Sep. 1998.

[22] http://www.eas.asu.edu/~ptm. [Online]. Predictive Technology Model (PTM)

[23] Vaughn Betz, et al., *Architecture and CAD for Deep-Submicron FPGAs*. KLUWER Academic Publishers, 1999.