In-the-Field Mitigation of Process Variability for Improved FPGA Performance

Konstantinos Maragos, *Student Member, IEEE*, George Lentaris, and Dimitrios Soudris, *Member, IEEE*

Abstract—The mitigation of process variability becomes paramount as chip fabrication advances deeper into the sub-micron regime. Conservative guard-bands result in considerable performance loss, while most low-level solutions impede dynamic customization at application level. This paper exploits the existing process variability of commercial off-the-shelf FPGAs to improve the operating frequency of a design, in-the-field, at anytime during the lifetime of a chip. We begin by measuring variability in prevalent FPGAs and assessing its impact on the performance of common DSP benchmarks. For the former, we develop a custom sensing network of Ring-Oscillators to generate detailed 2D maps per chip. For the latter, we perform intensive testing and statistical analysis to establish the relation between variability maps and benchmark frequencies. Accordingly, we propose a framework to automatically characterize the user's devices, place the design on the most efficient region, and scale its frequency based on user requirements and functional verification. Experimental results on 20 FPGAs of 28 nm Xilinx technology show up to 13 percent intra-die and 30 percent inter-die variability; with limited cost, our framework provides $10-14.7$ percent average gain by exploiting such variability, or up to $56-138$ percent by also customizing the guard-band.

Index Terms—FPGA, process variability, ring oscillators, frequency scaling, guard-band customization, performance improvement

1 Introduction

WHILE semiconductor technology strives to evolve according to Moore's law, the challenges in chip fabrication become even more pronounced. Controlling process variability is one of the most prominent problems, already present in current technology nodes and expected to become even more arduous in the future, deeper sub-micron regime [1], [2]. As a consequence, process variability decreases the yield and performance of the fabricated chips.

Process variability is classified in two main types: systematic and random [3], [4], [5]. On one hand, *systematic* refers to deterministic, spatially correlated variations induced into wafers and dies during fabrication due to operational shifts and equipment inaccuracies (e.g., photoresist development, etching). On the other hand, *random* refers to uncorrelated, unpredictable variations occurring in the atomic scale due to stochastic fluctuations in the process (e.g., random dopant fluctuation, gate oxide thickness variation). All these effects lead to deviations in the electrical properties of the transistors, such as the threshold voltage, Vth, and the effective channel length, L_{eff}. In turn, these deviations translate to variations in speed and power consumption between supposedly identical dies (inter-die variability), or even inside a single die (intra-die variability). Indicatively, in a set of 28 nm chips, inter- and intra-die variability measures up to 17 and 7.7 percent, respectively [6].

To overcome the resulting variability and eliminate malfunctions in their products, the vendors follow the conservative "one recipe fits all" strategy. They impose common global guard-bards to huge sets of fabricated chips. Hence, for nominal operation, they ensure functional integrity even for the worst-case scenario of their entire production line. However, this pessimistic approach prevents the majority of the chips from utilizing their full capability and leads to a significant performance loss [6], [7], [8], [9], e.g., 80 percent of potential increase in operation frequency.

Mitigating the process variability is of utmost importance for enhancing the efficiency of present-day and future processing systems. Classical techniques include statistical modeling at design time [5], post-fabrication tuning of parameters, either in the chip or in the equipment, speed-binning via performance testing [10], [11], etc. These techniques rely mostly on one-time variability analysis and are performed during the manufacturing cycle. In contrast, the current paper targets *in-the-field mitigation*, which can be applied multiple times during the lifetime of a chip, even at user level, without the vendor's support. Additionally, in-the-field mitigation enables dynamic adaptation to environmental and aging effects that degrade system performance erratically [12]. The industry has acknowledged the benefits of such mitigation and the need for relevant IPs [13].

Among the various processing platforms, particular interest should be paid to the Field-Programmable-Gate-Array (FPGA). Today, FPGAs are flooding the market due to their impressive performance-per-watt figures and acceleration capabilities. Their application varies from embedded system design to datacenter support, where the enormous number

• *The authors are with the School of Electrical and Computer Engineering, National Technical University of Athens (NTUA), Zografou 15780, Greece. E-mail: {komaragos, glentaris, dsoudris}@microlab.ntua.gr.*

of chips urges for a variability exploitation scheme. In contrast to conventional processor-based devices (multi-cores, GPUs, DSPs), the FPGAs offer increased opportunities for variability evaluation and in-the-field mitigation. They base on a uniform architecture of small identical resources [14], which allows the user to deploy multiple sensing structures across the entire fabric of the device to assess its variability [15], [16], [17]. Moreover, the FPGA's reconfigurability facilitates the remapping of a design to the most efficient region of the chip, even with respect to dynamic variability.

In the current paper, we focus on exploiting the process variability of present-day commercial FPGAs, in-the-field, as a means of improving their performance/throughput. We begin with a method to accurately assess intra- and inter-die variability, at user level, which we then quantify on a given set of 28 nm devices. For each device, we generate a corresponding *variability map*. Subsequently, we proceed by assessing the impact of this variability on the actual performance of various benchmarks. We establish that the measured variability can indeed be exploited in practice, even for the very challenging intra-die scenarios. Moreover, we correlate the performance of the examined benchmarks to the derived map-signature of each device, which we can thereafter use as a predictor of frequency boost. Accordingly, we propose a framework to increase the FPGA's speed by capitalizing on the generated maps and dynamic frequency scaling. The framework customizes each implementation to the underlying HW's variability and user-defined guard-band. Overall, the main contributions of the work are:

- To develop and explain a generic system that produces 2D maps for the fine-grain characterization of the actual performance capabilities of System-on-Chip (SoC) FPGAs and conventional FPGAs.
- To evaluate process variability in multiple, diverse, commercial 28 nm FPGAs by considering, besides nominal conditions, voltage-drop effects in the die.
- To assess the variability's impact on the performance of real benchmarks, for both inter- & intra-die cases.
- To propose and develop an enhanced framework exploiting inter- and intra-die variability, as well as the guard-bands, to improve the FPGA performance, in-the-field, without modifying the vendor's tools.

Compared to our previous publication [6], the main difference/extension of the current work lies in the exploitation of the intra-die variability. For this new and more challenging contribution, we employ our new statistical approach to analyze FPGA performance and we intensify the testing required. Furthermore, we automate this testing by developing a HW/SW infrastructure, we increase the diversity of the examined devices (in terms of number and size and type), as well as the variety of testing conditions (in terms of benchmarks and especially workload and voltage). Owing to these extensions, we consolidate our framework and add new features to it. As will be further explained in Section 2, the combination of the above ideas/contributions also differentiates our work from others in the literature. When specifically considering our intra-die results, to the best of our knowledge, this is the first work in the literature involving commercial devices and exploitation at user-level (hence, the presented method is central to this paper).

As a proof-of-concept, even though our methodology is technology- and vendor- agnostic, we perform tests on 20 Xilinx devices at 28 nm node. We note that 28 nm is the most widespread and profitable (cost per transistor) technology and is foreseen to hold a large share of the market also in the near future [18], [19]. Our tests show intra-die variability up to 13 percent and inter-die up to 30 percent, which increase with voltage drop. Our framework is evaluated with FIR and FFT benchmarks on Zynq XC7Z020T devices, for which the results show average performance improvement up to 5.9 and 9.9 percent exclusively due to intra- and inter-die variability exploitation, while their combined gain can increase to 14.7 percent, on average, or in the area of 20 percent for more extreme cases. Furthermore, significant improvement, e.g., twofold increase in speed, can be achieved via guard-band customization. Owing to its generic deployment and fast completion, e.g., less than a minute per chip, our framework is suitable for selecting the best region/device in embedded or high-performance applications, for re-evaluating the status of a chip anytime during its lifetime, or for examining hundreds of devices in an automatic fashion within a centralized facility, e.g., datacenters or HPC clusters.

The paper is organized as follows: Section 2 includes the related work. Section 3 presents our method and evaluation of conventional and SoC FPGAs using variability maps. Section 4 analyzes and establishes the correlation between the variability maps and the performance of common benchmarks. Section 5 presents the proposed framework. Section 6 evaluates the framework. Section 7 draws the conclusions.

2 Related Work

Evaluation of variability in FPGAs is being studied for more than a decade and several works on this topic exist in the literature. The authors in [15] developed a measurement circuitry based on an array of 884 ring oscillators (ROs) to analyze the within-die delay variability in 90 nm FPGAs. They tested 18 Cyclone II 2C35F672C6 devices and measured up to 4.47 percent stochastic variation (3-σ) and up to 3.66 percent systematic variation. Within-die delay variation was also analyzed in [16] for 65 nm Virtex-5 5VLX330T FPGAs. They collected data from 6,856 dies by deploying 6,480 ROs per die. The total within-die performance variation (both random and systematic) was measured at 22 percent (=max-min). Furthermore, they modeled and incorporated systematic variation in the static timing analysis (STA) tool to show 5.4 percent average frequency improvement for a set of benchmark designs. In [20], the authors used 112 ROs for online sensing of intra-die variations in delay, leakage, dynamic power and temperature; their system relied on Microblaze for monitoring the ROs on each of two 65 nm Virtex-5 XC5VLX110T FPGAs. The delay variation was measured at 2.3 percent (= σ/μ). In [17], an alternative technique is proposed for characterizing the delay variability in FPGAs. The key idea bases on measuring the delay of a combinatorial circuit under test (CUT) placed between a launch and a sampling register. A clock generator controls the registers and a stimuli generator provides inputs to the CUT. While stepping up the frequency, a custom circuit monitors the outputs of the CUT and the sampling register to detect the occurrence of timing errors. Consequently, the maximum error-free frequency is derived. The technique was applied to measure the delay variation of LUTs, carry-chain units and embedded multipliers in Cyclone II EP2C35 and Cyclone III EP3C25 FPGAs.

The existing mitigation approaches of process variability in FPGAs are applied on architecture/device and CAD levels. Regarding architecture/device level, the authors in [21] developed analytical models for leakage and timing variations to calculate the yield rate of a lot under process variability. By considering the Virtex-II Pro architecture as reference point, they showed that the tuning of various architectural (logic block and LUT size) and device (V_{dd}, V_t) parameters can increase the leakage-delay yield by 23 percent. In [22], an adaptive FPGA architecture is proposed to compensate for process induced threshold voltage variations. This architecture comprises an additional characterization circuit, which classifies the logic and routing blocks of the fabric into groups according to their measured performance. The classification information is stored in dedicated SRAM bits and is used by a body-bias circuit to configure accordingly the Vth value of each block. SPICE simulation results revealed significant decrease in delay and leakage variation (σ): up to 3.3x and 18x, respectively. In [23], the authors investigate the robustness of bidirectional and unidirectional routing architectures in the presence of Vth variation. They modified the existing VPR tool to account for delay variation effects. Experimental results on synthetic benchmarks showed that bidirectional routing is preferable only in case of short wire segments. As wire length increases, unidirectional routing presents smaller standard deviation in critical path delay, e.g., 36 percent. The authors in [24] evaluate the suitability of different routing architectures under process variation. They claim that the architectures including more short wire segments with more buffers present improved timing variation. Moreover, further improvements can be achieved by enhancing existing placement and routing algorithms to account for timing variations. Specifically, experiments based on VPR tool and synthetic benchmarks showed improvement up to 18.8 percent in timing variability (σ), which increases up to 22 percent when combining the variability-aware CAD optimizations. Moreover, improvements up to 28 and 68 percent were observed in delay ($\mu + 3\sigma$) and timing yield, respectively. In [25], the process variation tolerance of Transmission Gate (TG) and Pass Transistor (PT) based LUT structures at HK/MG 16 nm technology is studied. MonteCarlo simulations on SPICE showed that TG-based LUT can decrease the delay variation to 4.9 percent instead of the 23.3 percent in the case of PT-based LUT.

Regarding mitigation at CAD level, the works in [26] and [27] propose a *chip-wise* placement method to leverage correlated process variation and improve the performance of FPGA applications. Given the variability map of the underlying FPGA as input to the proposed placement algorithm, the most efficient solution can be retrieved. Experiments with synthetic benchmarks present performance improvement of $4-26$ percent compared to the conventional placer of VPR tool (T-Vplace). In [28], the authors study the performance improvement through path reconfiguration within a LUT cluster. Specifically, for a given logic network, they investigate the performance impact of possible LUT mapping and placement scenarios in the presence of process variation. The efficient combination of LUT mapping and placement results in over 10 percent critical path delay improvement according to MonteCarlo experiments. In this direction, a variation-aware chip-wise routing method is presented in [29]. On top of VPR router, they enhance the related cost function and timing analysis to include information extracted from variability maps of actual FPGAs. Experiments with maps of 100 Cyclone III FPGAs and synthetic benchmarks revealed up to 6.4 percent improvement in critical path delay. In [30] an NBTI/PV-aware placement technique is presented to improve the performance and reliability of FPGA designs. The technique is integrated in VPR's T-VPlace to consider the joint PV/NBTI effect in regional delay estimations and move/swap cost functions. The process variability data were retrieved from 5 Virtex-II Pro FPGAs, whereas the switching probabilities for nets and LUT inputs were obtained from the PowerModel tool. The experiments carried-out on synthetic benchmarks showed more than 60 percent PV/NBTI reduction for 60 percent of the chips.

Finally, a limited amount of recent works in the literature show the performance gained due to guard-band customization. The most widely adopted approach bases on frequency scaling and online measurement of timing slack in critical paths. Representatively, [9] and [8] modify the design netlist by employing additional shadow registers or in-situ detectors in the most critical paths, while scaling the frequency at the maximum level until no error is detected. Their experimental results on a 60 nm Cyclone IV and a 28 nm Zynq ZC7020 showed performance improvement up to 39 and 98 percent, respectively. However, the analysis of the critical paths is performed off-line, according to STA, without taking into account the existing process variability and aging of the underlying chips. In practice, this could mislead the selection of the actual critical paths. The authors in [7] propose an approach to isolate and measure the actual timing slacks of critical paths replicas under various voltage/temperature conditions. Accordingly, they build a calibration table such that, when the target design is mapped on the FPGA, its operation frequency can be adjusted based on the pre-stored values of the calibration table. This method requires developing an augmented CAD tool and analysis of the critical paths with STA. Experiments based on a 65 nm Cyclone IV FPGA and an FIR benchmark showed 50 percent increase of the operating frequency over STA.

In comparison to the aforementioned works, we develop a framework for in-the-field mitigation of variability at user level, where no modification is required in device parameters, vendor's CAD tool, or design netlist. In effect, like many of these works, our framework considers inter- and intra-die process variability, guard-band customization, or even adaptation to environmental and aging effects (without relying extensively on the time-consuming processes of Synthesis, Placement & Routing, and bitstream generation of user IPs).

3 Process Variability Evaluation in FPGA

The current section presents in detail the design and deployment of our custom sensing network for automatic in-the-field measurement of process variability. The key idea is to analyze the actual performance capabilities of present-day

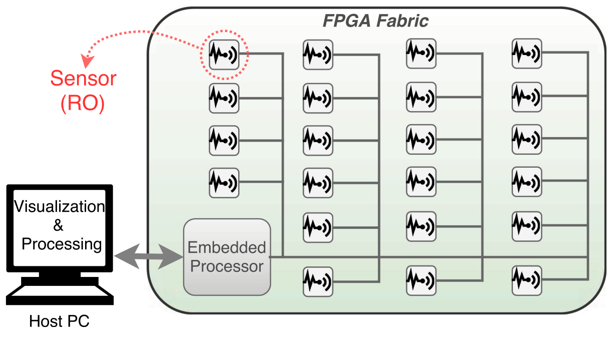


Fig. 1. Proposed system for variability evaluation.

conventional and SoC FPGAs without relying on the estimation of the vendor's STA tool. By comparing the maximum achieved speed among multiple identical chips, we determine the inter-die variability. Similarly, by comparing the maximum local speed of distinct regions inside a chip, we determine the intra-die variability.

For the purposes of the evaluation, we developed the system depicted in Fig. 1. It consists of an embedded processor (EP) and a network of uniformly distributed sensors programmed in the FPGA. In essence, each sensor measures the local speed of a distinct region in the chip. The EP supervises the operation of the sensors, collects their local data and communicates with any external CPU (via serial or Ethernet port) to forward the results for further processing. Our method is generic and can be applied to most FPGAs, with any hard/soft-core processor (e.g., ARM, Microblaze), and at any granularity (number of sensors). The following sections provide details regarding our sensor, the system's operation, and results derived for 28 nm Xilinx chips.

3.1 RO Sensor Design for Robust Replication

The fundamental building block of our evaluation system is our custom sensor utilizing the well-established Ring Oscillator (RO) circuit [6]. In general, an RO consists of a chain of inverters forming an asynchronous loop as shown in the middle box of Fig. 2. When the number of gates is odd and the RO is fed with a constant signal, its output oscillates to provide a square wave. The frequency of this wave depends only on the delay of the RO's asynchronous path, i.e., on the number and electrical characteristics of the transistors in the loop. If we guarantee that the gates and routing in this loop (the total path) remain relatively intact when we move the RO within the FPGA, then its frequency will change depending only on the electrical characteristics of the specific local region (its overall variability).

The complete structure of our sensor is shown in Fig. 2. It comprises a three-stage RO augmented with a 16-bit up-counter and I/O registers. The goal is to measure the frequency of the signal generated by the RO. To do so, we use this signal as a clock to drive the up-counter. Upon initialization, the counter is reset and the 1-bit input register holds an activation signal for a predefined time period T. During T, the RO oscillates and the counter increments continuously as if it was measuring the number of positive edges of the generated signal. At the end of T, the 16-bit output register (synchronous to the RO signal) holds the final outcome c_{ro} of the counter. The RO's frequency is calculated by EP as $f_{ro} = c_{ro}/T$ after accessing the register.

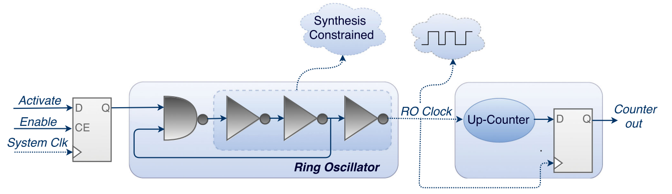


Fig. 2. Ring oscillator sensor structure.

To obtain reliable results, as mentioned above, we must ensure that the f_{ro} depends only on the location of the RO in the fabric and is not affected by the synthesis/implementation process of the vendor's SW tool. Therefore, we pay particular attention in terms of VHDL, coding attributes, and tool constraints to produce a robust RO structure, which can be replicated with the exact same resources and routing –almost– anywhere within the FPGA. Several in-house experiments with our sensor and the devices/tools verified that the operation of the RO is highly sensitive to any environmental stimuli or design alteration. Overall, according to our experiments, we summarize that:

- The RO should be isolated from any surrounding logic to eliminate power noise and avoid local temperature increase (noticeable shifts occur otherwise).
- All ROs must be implemented with identical routing resources (relative to each other). Tests verified that each wire connection has considerable impact on the RO timing and, hence, attention should be given to ensure that all connections are constrained/fixed.
- All ROs must utilize identical logic resources (LUTs, Flip Flops and Carry Chains) with fixed placement on the fabric. Unconstrained placement allows routing uncertainty and, hence, leads to different timings (for Xilinx, even the SLICEs must be of the same type, i.e., *SLICEM* or *SLICEL*, due to individual routing connections and construction of LUTs).

In respect of the above, we developed a custom built-in macro block describing a robust RO sensor, with various constraints/attributes that prevent all unwanted phenomena. As noted in Fig. 2, we use appropriate synthesis constraints to prevent the SW tool from optimizing the chain of inverters, i.e., from removing the functionally redundant gates. Moreover, we consider the individual architecture of the underlying FPGA to map the RO sensor to specific logic resources: each inverter is mapped to a distinct LUT followed by a pass-through latch, whereas the counter is mapped on four specific carry chain units together with 16 flip-flops. The four pass-through latches add extra delay to the signal transition between the stages of the RO chain (as in [20]). Fig. 3a illustrates the resulting mapping of resources with a floorplan view of the FPGA fabric (the orange highlighted small blocks represent DFF/latches, mid-sized blocks represent LUTs, large blocks represent carry-chains). In addition, we use routing constraints to guarantee that the connections among the above resources will remain identical, relative to each other, regardless of where the RO sensor is placed on the FPGA. Fig. 3b shows the routing of our sensor with nets (green highlight) connecting the fixed resources shown in Fig. 3a. Notice that the implemented sensor is quite compact (only four CLBs, or eight slices, occupying 0.06 percent of a mid-range

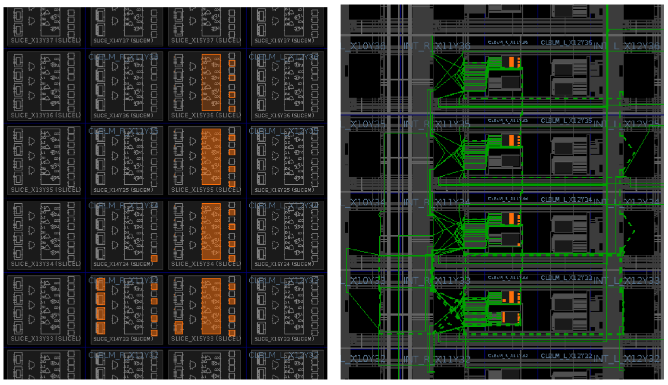

(a) Constrained logic resources (orange color). (b) Constrained routing resources (green color).

Fig. 3. Floorplan of ring oscillator implementation on XC7Z020T FPGA.

XC7Z020 chip) and allows the deployment of an entire RO network alongside real designs by paying a minor resource overhead.

3.2 Deployment and Operation of Sensing Network

The robust sensor macro-block described in the previous section is deployed in several copies and placed uniformly across the FPGA to sufficiently cover the entire fabric. We automated the deployment via parametric VHDL code and a device-specific Python script, which generates all of the necessary constraints to drive the implementation process (after synthesis). In practice, the script includes a template for the sensor and information regarding the architectural details of the FPGA (e.g., the size of the fabric at slice level). The script inputs the necessary net names output from the synthesis process. For each net of each sensor, it generates the necessary constraints by replicating the sensor's template and increasing the X-Y location on the fabric. Specifically, for each sensor, the script defines its eight slices (using X-Y) and, subsequently, defines the routing and logic resources within and among the eight slices (using fixed identifiers and the sensor's template). The script outputs a big constraint file imported in Xilinx Vivado to be used in the implementation process (placement and routing) of the network. The entire flow is depicted in Fig. 4.

The communication of the EP with the RO network is realized with an AXI-lite port and a large multiplexing structure connecting all the RO sensors. The EP sends all of the appropriate 32-bit commands for the initialization, triggering, and data collection from the sensors. Successively, the EP resets all sensors, activates them for a predefined period T and, when the period T is over, it deactivates their operation. Also, the EP sequentially retrieves the registered results: for each sensor, it sends a distinct word over AXI-lite representing an address of the multiplexer and allowing the corresponding sensor to forward its result over AXI. To determine the measurement period T as accurately as possible, in case of a SoC FPGA, we use the *private* timer of the ARM processor, or in case of conventional FPGA with a soft-core Microblaze, we use the AXI Timer IP; the EP sets its timer before issuing the activation command to the sensors, waits for 10,000 timer cycles to receive an interrupt, and issues the deactivation command. The timing error of this procedure tends to be systematic and less than 0.7 percent. The entire

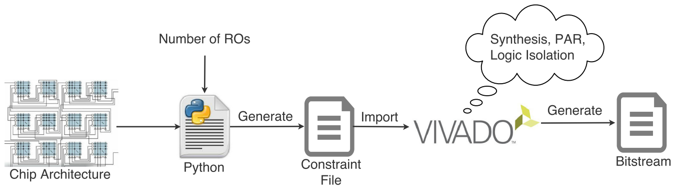


Fig. 4. Flow of automatic deployment of the proposed sensing network.

process is executed in a bare-metal environment, whereas EP communicates with a host PC via serial port to forward the *variability map*.

3.3 Evaluation Results: Variability Maps

By using the aforementioned sensing network, we evaluated the process variability in 20 conventional and SoC FPGAs in our lab. The examined FPGAs are listed in the first two columns of Table 1. They are all fabricated at 28 nm and the speed-grade varies from "-1" to the faster "-2". The "Zynq" category indicates SoC FPGAs integrating a Processing System (PS) with a dual-core ARM Cortex A9 processor and a variety of peripherals, as well as the Programmable Logic (PL) with the actual FPGA resources under investigation. The "Virtex-7" are conventional FPGAs including only Programmable Logic. In Zynq FPGAs, we use the ARM processor as the EP described in previous sections, while in Virtex-7 devices we deploy a lightweight version of the Microblaze soft-core processor. For each distinct FPGA we measure the intra-die variability. Furthermore, we measure inter-die variability in 13 identical Zynq XC7Z020T FPGAs (all hosted on Zedboards), in 2 extra XC7Z020T FPGAs (1b and 2b, hosted on ZC702 evaluation boards), and in 2 identical Virtex-7 XC7VX485T FPGAs (hosted on VC707 evaluation boards). In all cases, the measurements were conducted under same environmental conditions (e.g., 26°C ambient temperature).

Let us begin by focusing on the XC7Z020T devices playing the key role in our experiments. Indicatively, Fig. 5 illustrates the results for the first nine variability maps (one per FPGA). Each map represents the area of the chip, which is divided in 408 small cells (top-left is the location of PS, not considered here), with each cell being colored according to the frequency of its corresponding RO (one RO per cell). The frequency of each RO is averaged over 10 runs with the standard deviation being negligible, i.e., 0.15 MHz. The color scale is common to all maps and ranges from 380 to 440 MHz (chip 4 reaches down to 381 MHz, while chip 8 reaches up to 476 MHz). Considerable intra-die variation is shown in all maps; e.g., chip 3 presents differences up to 30.7 MHz indicating that the top right corner is faster than the central area of the chip. Overall, the frequencies vary rather smoothly in each map and reveal homogeneous regions subjected to systematic sources of variability. Moreover, considerable difference is observed among the chips, both in mean values (RO frequencies) and in the direction/distribution of their intra-die maps. For instance, the fastest chip 8 outperforms the slowest chip 4 by 68,7 MHz (mean RO frequency).

Similar maps are shown in Fig. 6 for the Virtex-7 XC7VX485T FPGAs. Notice that, in this case, the construction of Microblaze requires the utilization of programmable logic and a part of the chip's fabric is reserved for this purpose, i.e., it cannot be evaluated. To overcome this problem, we

TABLE 1
Variability Results for Multiple Conventional and SoC FPGAs from Xilinx at 28 nm Technology Node

FPGA Devices		# Sensors	Mean Freq. (MHz)	Intra-Die Variability (MHz)			Inter-die Variability
Type (id)	Speed Grade			*range = max-min*	*range/min (%)*	σ	*range′/min′ (%)*
Zynq XC7Z020T (1)	-1	408	396.71	24.22	6.31	4.13	30.32
Zynq XC7Z020T (2)	-1	408	403.04	29.97	7.7	6.29	
Zynq XC7Z020T (3)	-1	408	414.84	30.73	7.64	5.35	
Zynq XC7Z020T (4)	-1	408	393.91	26.26	6.89	4.55	
Zynq XC7Z020T (5)	-1	408	406.86	23.54	5.94	4.21	
Zynq XC7Z020T (6)	-1	408	419.52	29.97	7.36	5.58	
Zynq XC7Z020T (7)	-1	408	410.6	30.25	7.6	5.94	
Zynq XC7Z020T (8)	-1	408	462.56	23.61	5.24	3.91	
Zynq XC7Z020T (9)	-1	408	435.67	32.7	7.73	6.87	
Zynq XC7Z020T (10)	-1	408	354.95	28.19	8.22	5.46	
Zynq XC7Z020T (11)	-1	408	371.6	22.35	6.18	4.15	
Zynq XC7Z020T (12)	-1	408	457.04	35.7	8.05	6.23	
Zynq XC7Z020T (13)	-1	408	401.4	25.13	6.45	4.5	
Virtex-7 XC7VX485T (1)	-2	216	546.94	66.61	12.85	13.26	8.55
Virtex-7 XC7VX485T (2)	-2	216	593.69	28.39	4.89	5.08	
Zynq XC7Z020T (1b)	-1	408	385.16	27.53	6.87	4.16	10.5
Zynq XC7Z020T (2b)	-1	408	425.59	25.69	6.66	4.74	
Zynq XC7Z045	-2	1060	513.48	26.52	5.28	4.25	-
Zynq XC7Z100	-2	1310	584.78	42.71	7.55	5.2	-
Virtex-7 XC7VX690T	-2	1296	533.13	29.09	5.59	4.81	-

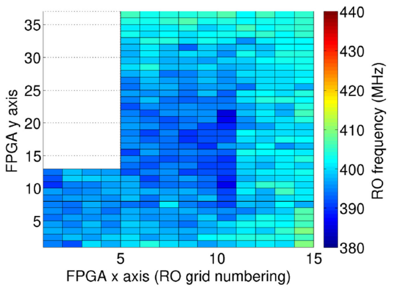
construct two distinct variability maps per FPGA. In each of the two partial maps, Microblaze is located in a different corner of the chip. By superimposing the two partial maps we provide a full coverage variability map of the chip. We stress that the measured frequency of the common cells between the two partial maps is almost identical (0.16 percent difference on average), which verifies the robustness of our RO sensors. The derived maps show noticeable intra-die variation in the first chip (Figs. 6a, 6b), i.e., up to 66.6 MHz across its bottom left to top right corner, while the second chip (Figs. 6c, 6d) has a more homogeneous distribution with no significant differences among the cells, i.e., up to 28.4 MHz. The second chip is considerable faster than the first chip with the mean RO frequency difference being 46.6 MHz.

In summary, Table 1 provides statistics regarding the measured intra-die variability for all examined FPGAs. It lists the number of ROs employed in each distinct FPGA, the mean frequency value of each map, as well as results regarding intra- and inter-die variability. For intra-die, we calculate the *range* between the *maximum* and *minimum* frequency of RO in the chip, i.e., their difference, and we compare it to the *minimum* frequency to report the chip's variability as a percentage. For inter-die, similarly, we calculate the percentage by considering the mean frequency per chip and comparing the fastest to the slowest chip (*range′/min′*). We note that the frequencies of the RO sensors in each chip tend to follow a Gaussian distribution. As expected, the mean frequencies show that higher speed grade indicates faster silicon fabric. When considering our *range/min* metric, the average intra-die variability measures at 7 percent for the 15 smaller XC7Z020T devices, while it reaches up to 12.8 percent for the bigger devices. Similarly, the inter-die variability measures at 30.3 percent for the 13 Zynq Zedboards and 8.6 percent for the 2 Virtex-7 VC707 boards. Finally, when we compare the fastest RO of the fastest chip to the slowest RO of the slowest chip, for same families, we get a variability of 39.7 and 17.5 percent for XC7Z020T and XC7VX485T, respectively.

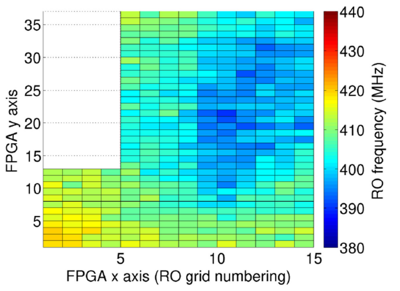
3.4 Evaluation Results: Variation Due to Voltage Drop

One step further, we evaluate the change of intra-die variability under the influence of voltage drop effects. In this direction, we perform two distinct tests: i) we introduce parasitic workload on the chip to force IR-drop, ii) we decrease the supply voltage directly via a power controller hosted on the ZC702 evaluation board.

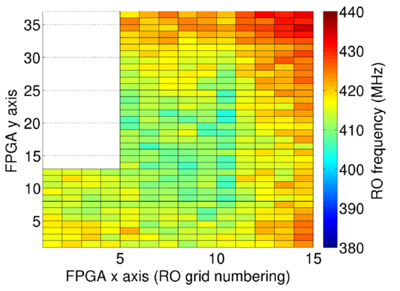
For the first test, we construct a baseline sensing network, e.g., with 186 RO sensors, to retrieve an initial variability map to be used as reference. On top of this network, we gradually add parasitic workload and retrieve corresponding variability maps. To quantify the impact of IR-drop, we compare these maps to the reference and report differences for various workloads. Representatively, we use the Zynq XC7Z020T chip 1 and we introduce additional workload by adding numerous unconstrained ROs (randomly placed and routed in the fabric of the FPGA by the Vivado tool). Fig. 7 depicts four maps of XC7Z020T for various workloads: 0, 400, 800 and 1,600 extra ROs. As shown for bigger workloads, the performance of the chip degrades and intra-die variability increases. Specifically, intra-die variability increases from 4 percent in the reference map to 7 percent when 1,600 extra ROs are employed, while the overall performance decreases by 10 percent (mean frequency value of fixed RO sensors). We note that, when 2,000 extra ROs operate simultaneously, the FPGA stops functioning after a few seconds. The same test was repeated with a larger Zynq FPGA, i.e, XC7Z045, with 1,060 RO sensors in the network and 2, 4 and 8 K extra ROs as workload. The variability increased from 5,3 percent (baseline) to 8 percent (8 K extra ROs) and the performance decreased by almost 13 percent. The shape of the map undergoes a minor change due to the



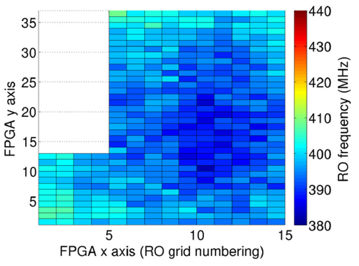
(a) Chip 1.



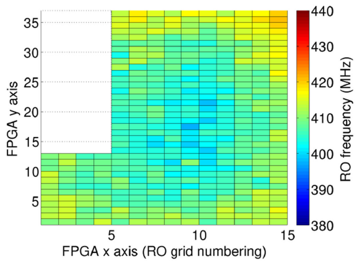
(b) Chip 2.



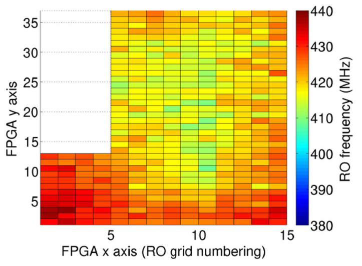
(c) Chip 3.



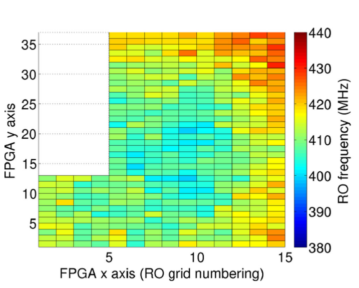
(d) Chip 4.



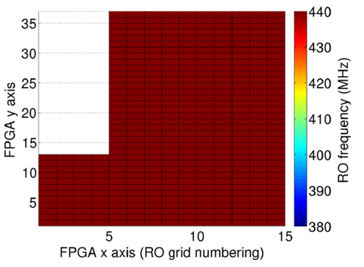
(e) Chip 5.



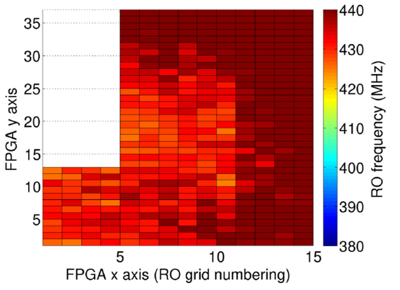
(f) Chip 6.



(g) Chip 7.



(h) Chip 8.



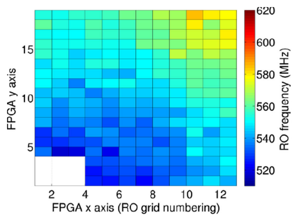
(i) Chip 9.

Fig. 5. Variability maps of 9 distinct Zynq XC7Z020T-1CSG324 FPGAs (common scale, the color represents maximum RO frequency per cell).

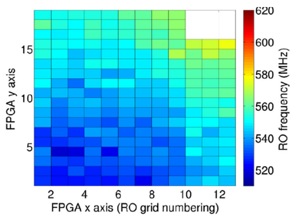
non-uniform placement of the extra workload by the tool (more congested regions lead to higher IR-drop).

In our second test, we use the two Xilinx ZC702 boards each hosting a XC7Z020T FPGA (id's 1b and 2b) and a power controller chip (UCD9248PFC). To perform voltage scaling, we use the ARM processor and the I2C interface of PS to control the voltage rail driving the Zynq's PL subsystem. The communication with the controller bases on the PMBus protocol and a custom developed SW program, which implements all the necessary functions to set the user defined V_{PL}. The baseline design is a sensing network with 408 RO sensors operating at the nominal supply voltage of 1 V. We gradually decrease this supply voltage from 1 to 0.86 V with a granularity of 0.01 V (the FPGA becomes non-functional below 0.86 V). Fig. 8 shows that intra-die variability increases in an almost uniform fashion, linearly to the supply voltage reduction. In contrast to the first test, the variability map retains its shape during this procedure (correlation ≥ 0.92). By comparing the reference map to that of 0.86 V, we measure 0.74−1.1 percent increase of intra-die variability and 32.1−35.4 percent performance degradation.

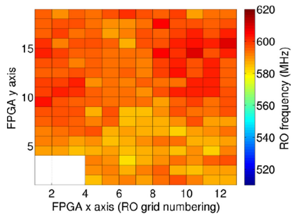
Overall, both tests show that voltage decrease has a considerable effect on intra-die performance variation: up to 3 percent increase when highly active designs are operating on the FPGA, or up to 1.1 percent exclusively due to the 14 percent voltage scaling. When added to the initial process variability of the chip, the intra-die differences increase to more than 9 percent for XC7Z020, or more than 14 percent for bigger devices, and are expected to increase even further with aging effects. Thus, as shown throughout this section, the significant intra- and inter-die variability motivates us to



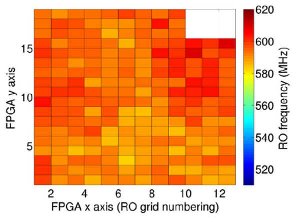
(a) Chip 1



(b) Chip 1

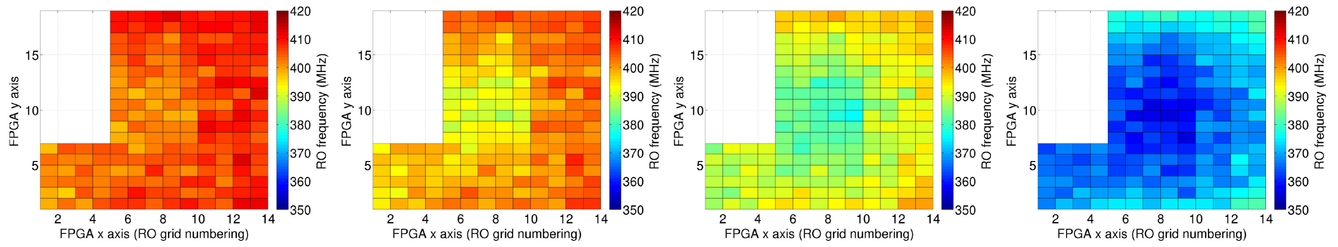


(c) Chip 2



(d) Chip 2

Fig. 6. Variability maps of 2 distinct Virtex-7 XC7VX485T-2FFG1761C FPGAs (common scale, represents maximum frequency per cell).



(a) Reference map (4.03% var.). (b) 400 extra ROs (5.82% var.). (c) 800 extra ROs (6.29% var.). (d) 1600 extra ROs (7.09% var.).

Fig. 7. Variability maps of XC7Z020T chip 1 with 186 fixed ROs and varying additional workload.

study its impact on the performance of actual benchmarks in the FPGA and investigate the possibility of exploiting local differences for the sake of throughput improvement.

4 Impact of Inter- and Intra-Die Variability on Actual Benchmark Performance

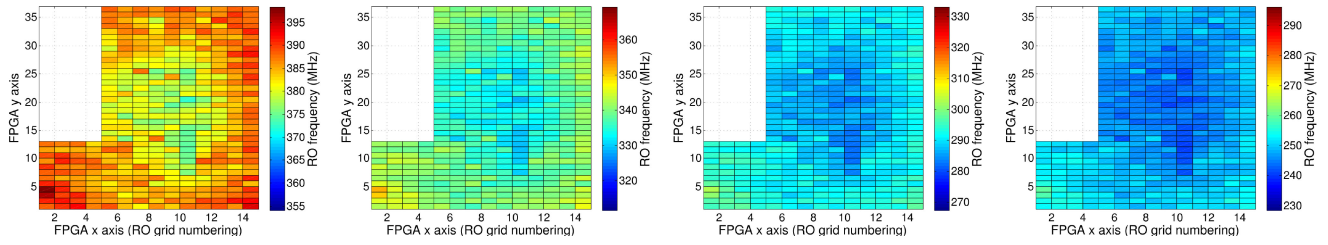
In the current section, we study whether and how the aforementioned process variability of 28 nm FPGAs affects the performance of bigger benchmarks, in practice. We evaluate both how inter-die differences increase the throughput of fast devices and how much the maximum speed depends on local intra-die variations. Specifically for the latter, due to challenges involved in the procedure, we follow a custom approach based on repetitive testing and correlation analysis. In both cases, the goal is to relate the variability maps derived in Section 3 to the actual benchmark performance, such that we can use the maps as predictors within the framework proposed in Section 5.

4.1 Experimental Setup

Hereafter, we focus on the first nine Zynq XC7Z020T FPGAs used in Section 3 (id's 1 to 9). We implement VHDL code to extract maximum benchmark frequency with respect to, i) mapping on distinct FPGAs, for assessing inter-die variations, and ii) local mapping on the fabric of individual FPGAs, for assessing intra-die variations. To extract the maximum frequency on HW (which tends to be higher than the rated of STA), we developed a support architecture enabling the automatic frequency scaling and the verification of functional correctness of each benchmark. The architecture is depicted in Fig. 9. It includes a Phase-Locked-Loop unit (PLL, bottom) driving the clock of the benchmark (User IP, top right), which is placed within a dedicated clock domain. To realize frequency scaling, the PLL is reconfigured at runtime by the ARM processor (PS, left) through the AXI-lite port. The reconfiguration of the clock requires 30 μsec. Moreover, 2 dual-clock FIFOs serve as synchronizers for clock domain crossing and correct transferring of the I/O data between the IP and the DMA controller (DMA, center). The DMA implements AXI-stream PS-PL communication for faster data transferring. The DMA and PLL operate in their own clock domain at 100 MHz, whereas the ARM processor operates at 667 MHz. The PS is connected to the external DDR memory storing I/O data of the user IP (test vectors, as explained below). The resource utilization of this support architecture is 4.7 K LUTs and 6.5 K DFFs (8.9 percent of the XC7Z020, but less than 2 percent for bigger devices, e.g., XC7Z100). The architecture is generic enough to be deployed either in SoC or in conventional FPGAs (e.g., with Microblaze) and forms a basis for even more complex systems.

When a benchmark (User IP) is integrated in the support architecture, the embedded processor extracts the maximum error-free operating frequency by executing a custom SW script as shown in Algorithm 1. In particular, the embedded processor initializes the frequency of the user design, f_{IP}, with the value reported by STA, f_{STA}. Subsequently, the design is executed using as inputs a set of test vectors (2 mega-samples generated pseudo-randomly) to produce the correct results, D_g (golden data), which are stored in the external DDR memory. Then, iteratively, the f_{IP} is increased in steps of 1 MHz, the application is re-executed and the fetched results, D_n, are compared to the golden data, D_g. In case of data mismatches/errors, the process is terminated and the highest error-free frequency is reported.

Furthermore, to automate the experimental procedure and involve multiple FPGAs, we developed the custom infrastructure depicted in Fig. 10. Our infrastructure hosts all FPGAs, together with an Ethernet switch for communicating with each board. Initially, each FPGA loads Linux OS (petalinux) on the ARM processor, with a fixed IP address, while the bitstreams of the benchmarks and the test vectors are preloaded to the SD card of its board. By using a host-PC in the same network, we broadcast specific commands



(a) 1V (6.88% var.). (b) 0.95V (7.25% var.). (c) 0.90V (7.61% var.). (d) 0.86V (8.01% var.).

Fig. 8. Variability maps of XC7Z020T chip 1b with 406 fixed ROs and varying supply voltage.

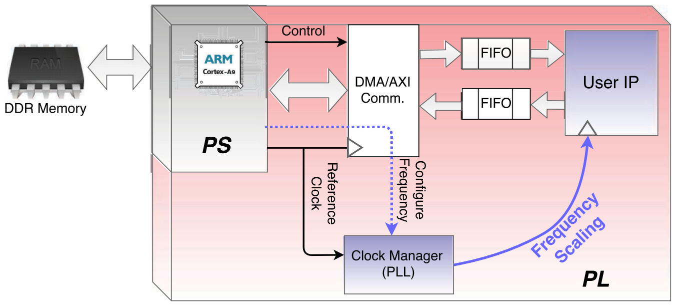


Fig. 9. Support architecture (frequency scaling, functional verification).

to the known IP addresses of the FPGAs to establish multiple remote connections and execute, effectively in parallel, a bash script on each ARM CPU. For each FPGA/test, the script initializes the DDR memory, invokes the frequency extraction process (Algorithm 1), and stores the results. This HW/SW setup allows for simultaneous control of all FPGAs and execution of the 1,000's tests required in an automatic fashion. Also, it facilitates the generation of all variability maps. We stress that the idea can be easily extended in large scale, e.g., in data centers, where the central node performing resource management can also broadcast the appropriate commands to all FPGAs for concurrent testing.

Algorithm 1. Maximum Frequency Extraction Process

```
Input: Reported frequency by STA, $f_{STA}$
Output: Achieved performance, $f_{IP}$
  $f_{IP} = f_{STA}$, $f_{step} = 1$ MHz
  $D_g = execute(IP, f_{STA})$ # retrieve golden results $D_g$
  $f_{IP} = f_{IP} + f_{step}$
  while true do
    $D_n = execute(IP, f_{IP})$ # retrieve new results $D_n$
    if ($D_g == D_n$) then
      $f_{IP} = f_{IP} + f_{step}$
    else # last error-free frequency
      return $f_{IP} = f_{IP} - f_{step}$
    end if
  end while
```

Regarding the selected benchmarks, our goal is to assess considerably larger areas of the FPGA compared to that utilized by each RO sensor. We select regions of 1.6 K LUTs and 3.2 K DFFs, which are sufficient for hosting realistic benchmarks. We divide the FPGA fabric into 28 such regions (almost full coverage of the chip, Fig. 11) to evaluate the performance variation across the entire fabric. The benchmarks mapped in these regions are two very popular DSP algorithms, which are widely adopted in a plethora of real-world applications: a 16-tap FIR filter (1.17 K LUTs, 1.32 DFFs) and a 16-size FFT (1.1 K LUTs, 1.86 K DFFs), with $10-13$ I/O bits accuracy. They are implemented with Xilinx Vivado 2017.1 and, according to the STA, the former operates nominally at 140 MHz, while the latter at 240 MHz.

4.2 Challenges of Impact Assessment

To achieve the level of detail required for the analysis in this task, we face two main challenges when relying on a user-level test setup. First, we face the *performance measurement* variation. When a benchmark/bitstream is tested multiple

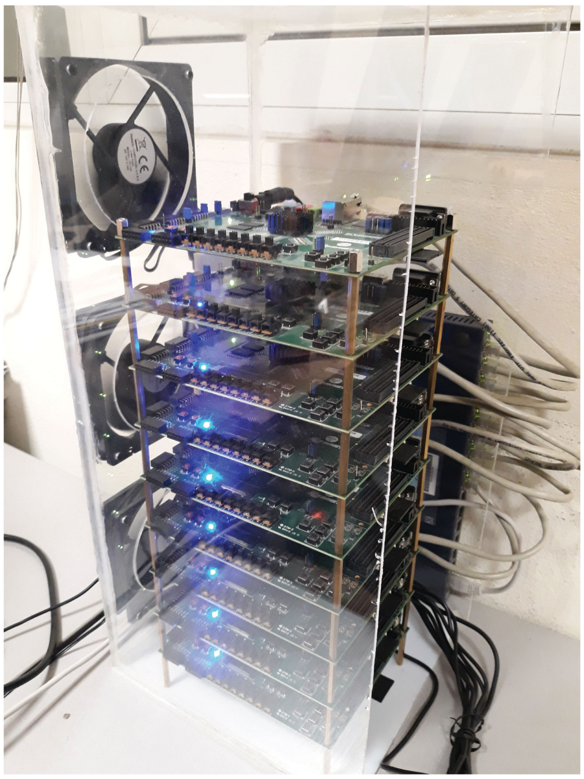


Fig. 10. Parallel testing of 9 Zedboards (XC7Z020T) in a custom infrastructure with an Ethernet switch for communication with the host PC.

times to find its maximum frequency, we observe a small variation, i.e., $\sim$1% or up to $\pm$3 MHz. This phenomenon appears when the transistors operate at their very psychical limits, when operation is susceptible to the slightest fluctuation that might occur e.g., in power supply or local temperature. Such variation becomes important in our tests, especially when we examine chip regions with performance difference less than 1 percent. To overcome this issue, for each region of the chip, we calculate the average maximum frequency over 100 runs of each bitstream.

Second, we face the small architectural variation of the XC7Z020T FPGA fabric. The latest Xilinx FPGAs are template based (with respect to CLBs, RAMBs, DSPs, etc.) but the resources are not distributed ideally uniformly across the fabric. Hence, when moving our already placed design from region to region in the chip, each time, we increase the probability that the tool utilizes slightly different relative resources. As a result, we are not comparing exactly the same designs/paths. On top of that, when implementing our benchmark in a new region, we suffer from the randomness introduced by the vendor's place & route heuristic algorithms. These effects, which become evident already at SW level, impose slight differences even on supposedly identical bitstreams. Hereafter, we refer to such differences as *SW variability*. Experimentally, by considering all of our generated bitstreams, we evaluated the SW variability to 6.5 percent in terms of timing (according to STA). This number is comparable to the amount of underlying intra-die process variability (Section 3). It can be viewed as a form of noise that prevents us from establishing the use of variability maps as predictors of benchmark performance. To tackle this problem, we rely on the idea of averaging filters: for each benchmark, we generate and test multiple

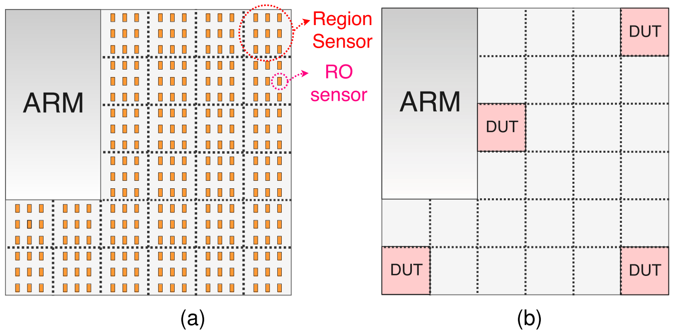


Fig. 11. (a) Benchmarking granularity, with the PL fabric partitioned in 28 regions. (b) Most prominent regions for intra-die variation analysis.

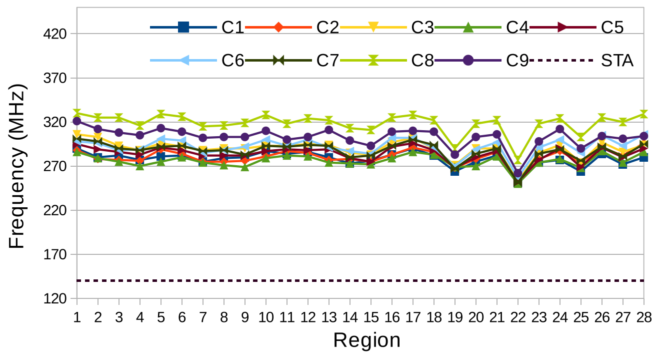


Fig. 12. Inter-die performance variation of FIR on 28 FPGA regions.

distinct bitstreams per region to retrieve their mean max frequency. This average value represents the region in the remaining of the analysis. More specifically, for Xilinx, we begin with a single synthesized netlist of the benchmark and we proceed to the PAR stage, where we perform multiple distinct implementations by tweaking small details via the floorplan utilities of Vivado (we avoid modifying "strategies" or "constraints" to avoid major changes in the netlist). For example, we alter the position of a single register or a LUT by moving to an adjacent location/slice. This minor change forces a distinct implementation result when repeating the PAR process, similarly to changing the seed of the heuristic algorithm. Thereby, we produce 20 distinct bitstreams per chip region, which however represent the same benchmark. We note that this multiplicity also captures the real-world behavior, where supposedly identical benchmarks could vary among users in terms of bitstream due to the smallest coding/tool detail. Also notice that SW variability does not affect the inter-die comparisons in our lab, because it is valid to use the exact same bitstream file in all XC7Z020T FPGAs.

4.3 Proposed Assessment Approach

By combining the aforementioned SW and HW infrastructure and proposed problem solutions, we now outline our approach for assessing the impact of process variability on actual benchmarks. First, to extract maximum frequency, we developed on Linux OS two scripts to handle the cases of inter- and intra-die variability testing. For inter-die testing, which is not affected by SW variability, we map each benchmark on 28 adjacent regions in each chip (e.g., as in Fig. 11) by implementing one bitstream per region. This granularity allows us to examine the entire fabric and, moreover, instead of performing a single inter-die measurement on only 1 set of chips, we repeat our test by assuming 28 distinct sets of chips (smaller chips, but practically distinguished from one another due to the local variation in each XC7Z020T map). For the intra-die case, due to SW variability, we implement 4×20 bitstreams per benchmark on 4 remote regions (e.g., as in Fig. 11b), which are selected arbitrarily and present noticeable change in the variability maps (Fig. 5). Since we repeat this intra-die test for multiple big chips, individually, the above number of bitstreams and regions suffices to establish the performance relations. In total, for this purpose, we perform almost $2{\cdot}10^5$ executions on the FPGAs: 50.4 K for the inter-die case (2 benchmarks $\times$ 28 regions $\times$ 100 runs $\times$ 9 FPGAs), and 144 K for the intra-die case (2 benchmarks $\times$ 4 regions $\times$ 20 bitstreams $\times$ 100 runs $\times$ 9 FPGAs). In an automatic fashion, in parallel for all FPGAs, the host PC broadcasts via Ethernet specific commands to them to complete the script as follows:

1) Download & program each FPGA with the first FIR bitstream, i.e., the one referring to the first region.
2) Execute 100 runs of the frequency extraction process (Algorithm 1) and store the mean value in SD card.
3) In case of intra-die testing, download the next bitstream (referring to the same region) and execute steps 2 and 3 for all 20 bitstreams of that region.
4) Reconfigure the PL to place the FIR in the next region and repeat steps 2-4. When all regions under examination are tested (4 or 28), proceed to step 5.
5) Reconfigure the PL to implement the FFT bitstream, starting from the first region, repeat steps 2-4 until the entire FFT benchmarking is also completed.
6) Fetch sequentially to the host PC all data stored in SD cards, from all FPGAs, to perform our analysis.

Second, we analyze the data fetched, statistically. The goal is to associate the benchmark's performance to the RO sensors, per region and per chip. As variables, we use the FIR & FFT frequencies retrieved by our script. Also, we calculate the spatial mean RO frequency of each region to emulate a bigger sensor to match the size of the benchmark (e.g., in the case of Fig. 11a, we average 9 ROs to denote a *region sensor*). The relationship of the resulting variables is not 100 percent deterministic/exact, both due to their benchmark-dependent magnitude and the limited randomness involved in the procedure. Therefore, we propose to base the analysis on the concept of *correlation* which, besides establishing the aforementioned relationship, allows us to study the covariance of the measured frequencies and make predictions when given a reference point. In particular, we rely on the *Pearson* product-moment coefficient, which is the most commonly used correlation in the literature, and facilitates our study as shown below.

4.4 Inter-Die Performance Analysis, Correlation to Variability Maps

The maximum frequencies of the two benchmarks measured on 28 regions and 9 chips are reported in Figs. 12 and 13. Their actual performance is significantly higher than that reported by STA (black dashed line), i.e., 124 percent higher on average. Furthermore, considerable performance variation is observed among the FPGAs: 19.6−16.6 percent for FIR and FFT, respectively. More concisely, Fig. 14 depicts the

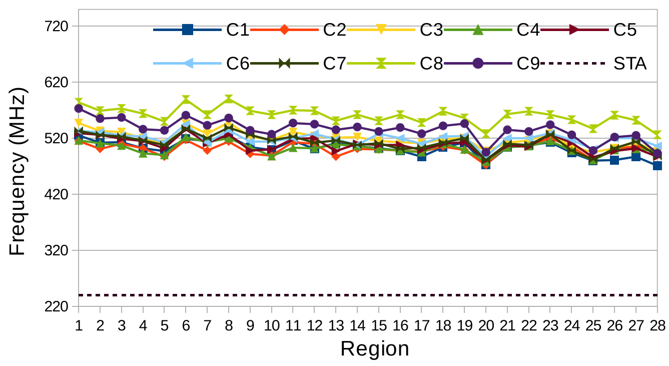


Fig. 13. Inter-die performance variation of FFT on 28 FPGA regions.

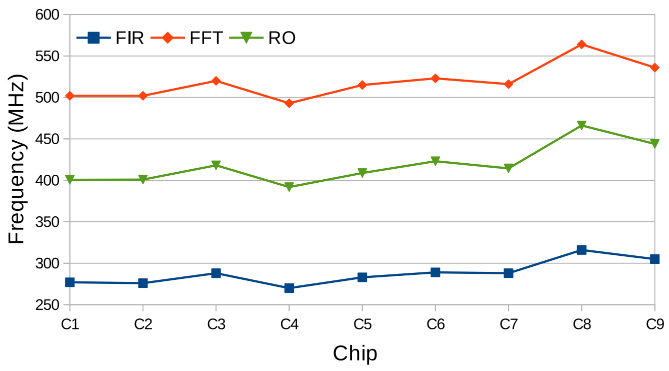


Fig. 15. Inter-die performance variation of RO, FIR, and FFT on region 4.

correlation between the frequency of each benchmark and the ROs of each chip, for 28 distinct chip regions. Notice that correlation=1 implies exact linear relationship. As shown, we measure extremely high correlation with the vast majority of the coefficients (91 percent) lying in the range of 0.96–1. Hence, we deduce that both FIR and FFT follow closely the behavior of the RO sensors. Indicatively, Fig. 15 focuses on region 4 to show how predictably the maximum frequency of FFT and FIR changes when we know a priori the behavior of the ROs in each chip.

Overall, the derived results prove the existence of an exploitable inter-die variability, up to 19.6 percent, and the importance of the variability maps, i.e., the high correlation. It is worth-mentioning that, in principle, this conclusion also applies to the intra-die scenario when no SW variability exists. That is, if we assume a *pseudo-chip* consisting of 9 identical regions allowing the bitstream to be copied seamlessly among them (in our case, such a chip is constructed hypothetically by stitching 9 co-located regions originating from 9 real chips), then Fig. 14 would show the intra-die correlation to ROs as measured for 28 distinct pseudo-chips. In all of our 28 pseudo-chips, the correlation is above 0.88. However, in practice, the analysis of intra-die correlation is more complicated for the reasons explained in Section 4.2 and will be tackled separately in the following section.

4.5 Intra-Die Performance Analysis, Isolation and Correlation to Variability Maps

Following the approach described in Section 4.3 and the maps of Fig. 5, we select the four regions shown in Fig. 11b (kept common for all chips). We measure up to 4.9 percent frequency variation between region sensors in each quadruplet (to illustrate this, Fig. 16 shows each region's frequency increase versus the slowest region sensor of its chip). Then, we place our benchmarks on these regions and we isolate their performance by averaging the frequency of 20 bitstreams per region (to filter out SW variability, Section 4.2).

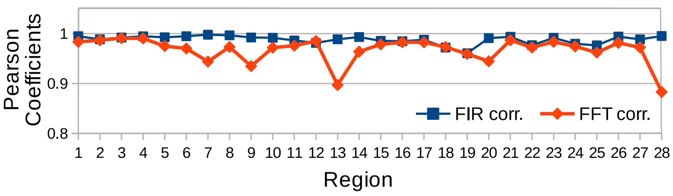


Fig. 14. Inter-die correlation of variability maps to FIR/FFT performance.

The intra-die performance varies among chips between 0.6–4.7 percent for FIR and 2.7–6.8 percent for FFT. Fig. 17 depicts the correlation between the FIR/FFT performance and the region sensors for all 9 chips. The coefficients range in 0.4–0.96 for FIR and 0.69–0.99 for FFT. As an outlier, a 0.4 coefficient appears for FIR in chip 8. This is attributed to the low intra-die process variability of the regions of chip 8, i.e., 0.1–1.6 percent (Fig. 16), which is not a sufficient bias given the 6 percent SW variability in each region. For the majority of the chips, the correlation ranges in 0.74–0.99 and implies that the benchmarks indeed follow closely the RO behavior. Similarly to the inter-die case, the above results indicate exploitable variability and predictable benchmark behavior. The confidence level increases for chips having areas with considerable performance differences, e.g., 2.5 percent, and is further analyzed in Section 6 in more practical scenarios.

In summary, this section showed considerable benchmark performance variation in 28 nm FPGAs, both for the inter- and intra-die cases. For the former, the variation ranges in the area of 16.6-19.6 percent whereas, for the latter, in the area of 0.6-6.8 percent. Moreover, the analysis showed that these differences are predictable, i.e., we can estimate the benchmark's frequency by considering the variability maps derived beforehand, without the need for an exhaustive search/test. Essentially, the results were the same for both benchmarks (the thoroughly examined FIR and FFT), and thus, can be considered conclusive for our purposes without the need for extra benchmarking. In total, when also customizing the guard-band to bypass the pessimistic estimations of STA, we measured significant room for

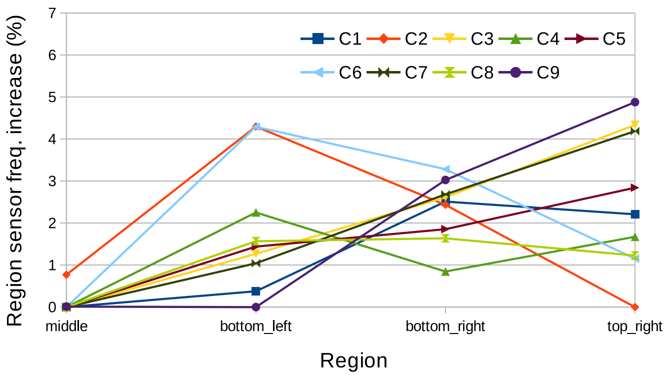


Fig. 16. Process variability effect on 4 selected regions for 9 tested chips.

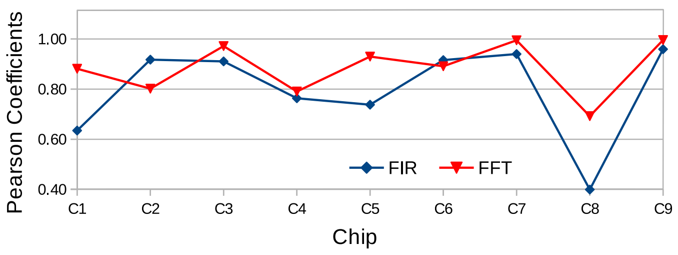


Fig. 17. Intra-die correlation of RO frequency to benchmark frequency.

performance improvement ranging in 95–138 percent (for zero margins). All the above numbers quantify the potential gain of an FPGA user and demonstrate clearly the need for a framework to exploit process variability and guard-bands, in-the-field.

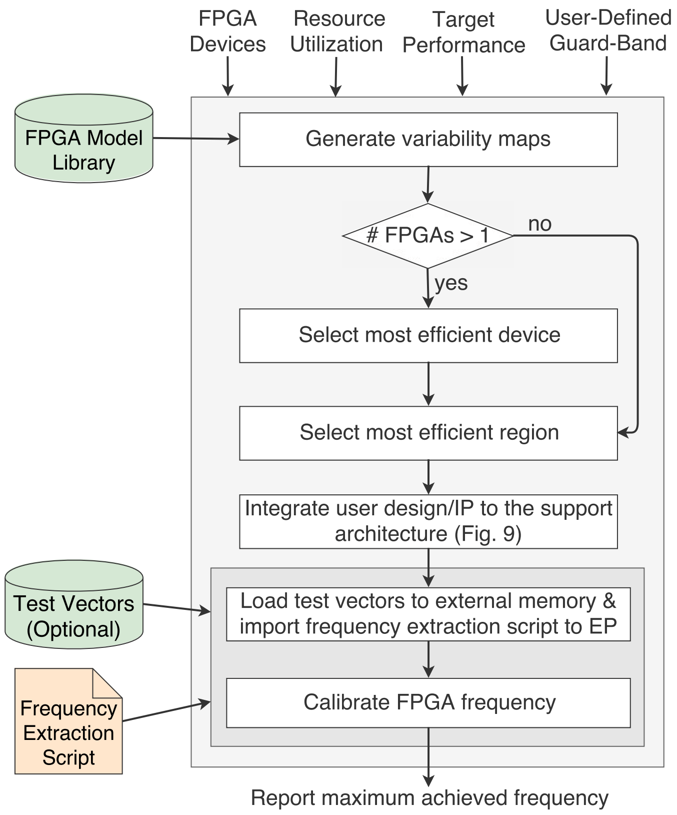
5 Proposed Framework for Exploiting Process Variability

Motivated by the results of Section 4, we developed a framework exploiting the existing variability in FPGAs towards improving the performance of realistic applications. Our framework considers both inter- and intra-die variability, as well as guard-band customization, to reap the full—hidden—potential of the underlying FPGA(s).

The overview of the proposed framework is illustrated in Fig. 18. It consists of five complementary stages: i) generation of variability maps, ii) device selection based on inter-die variability, iii) region selection based on intra-die variability, iv) integration of user design/IP to the support architecture, and v) performance improvement of the design via frequency scaling. The inputs of the framework are:

- *FPGAs*: number and model of the employed FPGAs.
- *Resource Utilization*: the cost of the user's design/IP in terms of FPGA resources (LUT, DFF, RAMB, DSP).
- *Target performance*: user requirement for performance in terms of operating frequency f_{IP} of the design. If no specific value is given, the framework will automatically target the maximum error-free frequency.
- *User-defined guard-band*: imposed as a safety margin (e.g., for aging, environmental effects, or low confidence level of a test vector procedure), it is expressed as a percentage of the f_{IP}. For instance, 30 percent guard-band implies that the design should operate correctly even as high as $f_g = 1.3 \cdot f_{IP}$ during calibration.

In more detail, the first stage of our framework involves the generation of variability maps for the available FPGA(s). Depending on the FPGA model, a custom library provides the corresponding bitstream for the deployment of the RO network, along with the SW program loaded to the embedded processor. The second stage refers to the selection of the fastest FPGA with respect to the generated variability maps (when more than one FPGA are available). The selection bases on the average value of the RO sensors employed in each chip. The third stage is related to the mapping of the user design on the most efficient region of the selected FPGA. The location and the size of the candidate region depend on the variability map of the FPGA and the resource



FPGA Devices
Resource Utilization
Target Performance
User-Defined Guard-Band
FPGA Model Library
Generate variability maps
# FPGAs > 1
no
yes
Select most efficient device
Select most efficient region
Integrate user design/IP to the support architecture (Fig. 9)
Test Vectors (Optional)
Load test vectors to external memory & import frequency extraction script to EP
Frequency Extraction Script
Calibrate FPGA frequency
Report maximum achieved frequency

Fig. 18. Proposed framework.

utilization of the user design. Notice that the spatial resolution of the RO map is relatively high to facilitate searching for this optimal location. The search procedure is automated via a custom Python script, which implements a 2D moving average calculation across the variability map to detect the region aggregating the highest score (coded via *box filters* and *integral images*). The size of the moving average window reflects the size of the design (according to resource utilization), plus additional margins to alleviate routing congestion for the successful PAR (e.g., 25 percent in LUTs and DFFs). The outcome of the script is a placement constraint file, which is imported to the Vivado tool to drive the placement of the user design to the most efficient region. The fourth stage involves the integration of the user design to the support architecture described in Section 4.1.

Finally, the *calibration* stage loads an exploration script to the embedded processor, as well as test vectors to the external memory (generated randomly when not user-defined). The exploration script calibrates the FPGA frequency with respect to user requirements. First, the design is tested with the user's target frequency and including the desired guard-band ratio. In case of functional errors, the script executes the frequency extraction process of Algorithm 1 to find the highest error-free frequency (also with guard-band).

6 Framework Evaluation Results

6.1 Runtime Analysis

The evaluation begins with the time cost of our framework. Representatively, we analyze the case of XC7Z020T FPGAs and FIR and FFT benchmarks. Time is devoted to two main parts: i) the generation of variability maps, and ii) the

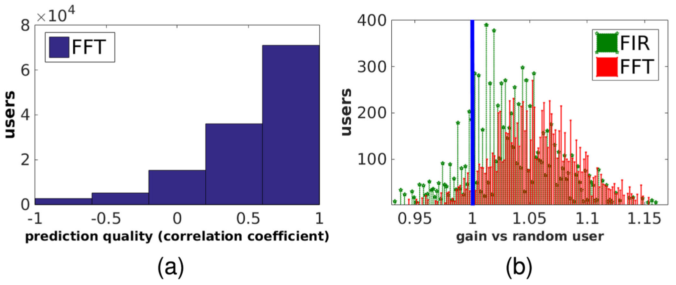


Fig. 19. Histograms quantifying the success of the proposed framework. (a) Correlation of the FFT performance with the performance predicted by the tool on 4 locations of chip 2. (b) Distribution of gain factors achieved when employing our framework on chip 9 (1x = no gain).

execution of the frequency extraction script. The generation of variability maps requires less than 2 sec per chip when 10 consecutive runs are being averaged. We stress that the execution time of a Python script for detecting the most performance-efficient region is negligible, i.e., ~1 ms on an Intel i7-5500U CPU for a typical 14×36 variability map. The execution time of the frequency extraction process depends on multiple factors: the throughput rate of the design, the PS-PL communication method, the size of the test vectors, the clock frequency of the processor, and the iterations needed to find the highest error-free frequency in FPGA. Indicatively, for the FIR and FFT benchmarks using 2 Msample test vectors, the time required to perform the DMA communication, retrieve the results, and compare to the golden data stored in DDR memory, was measured at 0.6−1.8 sec. Hence, when the design meets the target performance immediately and only one iteration is performed, the cost is only 1.8 sec. Otherwise, time depends on the number of iterations searching for the highest error-free frequency (Algorithm 1) and could increase at most to 138−351 sec on Zynq's ARM A9. We note that this time can be significantly decreased, i.e., to less than one minute, by pruning the search space with adaptable frequency steps of 10−1 MHz (to perform a coarse-grain before the fine-grain search) [6].

6.2 FPGA Performance Improvement

To assess the effectiveness of our framework on improving the performance of FPGAs, we assume numerous ordinary users performing a single trial each. We perform a statistical analysis to quantify their gains, as a whole, while we scrutinize the framework's output to distinguish benefits due to guard-band customization, intra-die, and inter-die variability exploitation. In particular, we compare each output to a default implementation of the vendor's tool, on a random chip/location, as done in lieu of an exploitation framework. Similarly to Section 4, we assume classical DSP benchmarks, such as 16− to 1024−point FFTs and 16− to 128−tap FIRs with 7− to 27−bit accuracy, implemented with CLBs on Xilinx Zynq-7020 FPGAs. This experimental setup includes 9 devices, 5 benchmarks/designs, and 10^4-10^6 hypothetical users combining more than 1,400 distinct implementations/netlists.

We begin by evaluating the frequency increase exclusively due to the guard-band customization. For the sake of this analysis, we apply our framework using zero guard-band and we consider evenly the entire set of results, regardless of netlist, location, and chip. In such configuration, the minimum speed gain per benchmark varies from 1.56× to 2.09× (with overall average at 1.9×). Therefore, for our given set of devices and CLB-only benchmarks, we estimate the *observable* guard-band introduced by the STA tool at around 56-109 percent. We note that this *observable* gap is a mere indication of the actual/sophisticated guard-band calculated by the vendor's tool (Xilinx) and it only serves as reference for the remainder of our analysis.

To continue the evaluation with respect to intra-die variability, we rely on our set of zero guard-band implementations, which facilitate decoupling from SW margins and focusing on actual HW capabilities. First, we verify the importance of the correlations shown in Section 4. We assume that a user tests the prediction quality of our framework by implementing the same benchmark on 4 arbitrary locations on a single device. Ideally, without SW variability, the 4 results should be in accordance to the gains predicted by the framework, i.e, by the RO map of the device. In practice, for verification, it suffices to acquire high correlation between the 4 actual FPGA results and the internal predictions of our framework. Thus, we assume 2 million users (distinct combinations of 18 netlists, on 4 locations, on 9 devices, for 2 benchmarks) and we calculate a coarse, 5-bin histogram of 2 million Pearson coefficients. As depicted in Fig. 19a, the concentration of coefficients implies a strong bias due to intra-die process variability. That is, on a random device, when testing FIR−FFT benchmarks, the user gets 69-82 percent possibility of deriving *good-to-excellent* correlation, 25-15 percent for *indifferent-to-worse*, and only 6-3 percent possibility of anticorrelation (Fig. 19a, left bar). We stress that devices of substantial intra-die variability, e.g., of 7.7 percent, lead to 84-93 percent occurrence of *good-to-excellent* predictions.

To further quantify the success rate of our framework, we calculate the possibility that a user will benefit by performing a single execution of our proposed tool (compared to the default output of the vendor's tool). For fairness, we assume 10^4 distinct users, each one comparing his/her own framework result to one random implementation (as would be acquired in the default approach). On a random device, the user will improve the performance with 76 percent possibility and achieve an average gain of 4 percent. On devices of substantial intra-die variability, this possibility increases to 85-94 percent for FIR and FFT, respectively, whereas the average gain improves to 4.5-5.9 percent; Fig. 19b depicts such a distribution where the gains reach beyond 15 percent (also due to SW variability). Furthermore on such a device, if we enhance the framework to implement the IP twice and keep the best result (similarly to exploiting SW variability), the possibility of improvement becomes 93-97 percent with an average gain of 5.2-6.9 percent.

When assuming multiple devices per user, i.e., when seeking to exploit the inter-die variability, the framework achieves almost 100 percent success rate in improving the frequency with an average gain of 8.2-9.9 percent for FFT and FIR, respectively. We stress that this gain is derived by comparing to the random choice made among 9 devices, in lieu of a relevant framework, without relocating the design between chips. When the available devices are limited in number and highly diverse in performance, the gain increases up to 13-16 percent, or even up to 20 percent for specific areas of the chip.

In total, when we combine the exclusive gains of intra- and inter-die variability exploitation reported above, the proposed framework leads to 10−14.7 percent average improvement compared to the random/default choice (regardless of benchmark). The 14.7 percent is estimated according to the aforementioned percentages by assuming that the fast device of the user also has substantial intra-die variability, whereas the 10 percent is measured with our example set of 9 devices. The best case scenarios (versus the worst possible choice, including SW variability) involve gains up to 25.1 percent or 28.9 percent. When we also include the gain due to guard-band customization, the framework improves the performance of our example FIR and FFT benchmarks by up to 138 percent (e.g., up to 577 MHz versus the 242 MHz estimate of STA for the 16-point FFT).

6.3 Comparison to Similar Methods

Considering recent works in the literature that introduce additional registers in the critical paths of the design [8], [9], the relevant methods achieve improvements in the area of 39 and 98 percent for 60 nm Cyclone IV and 28 nm Zynq ZC7020 FPGAs, respectively. Our method also provides gain in this range, partly due to our guard-band customization, which can take place regularly during the chip's lifetime for adaptation purposes (without re-executing the time-consuming processes of Synthesis, Placement, Routing, and bitstream generation), and partly due to inter-/intra-die variability exploitation reaching up to the area of 20 percent by itself. However, our method has the advantage of being applied at user-level without requiring modification of the IP netlist or the CAD tool. Moreover, we avoid binding the resource overhead to the complexity of the IP netlist (our support architecture has constant cost). That is, the above published methods might suffer from increasing overhead, e.g., 5 percent additional FPGA slices [8] to monitor 100 critical paths, which could render their approach impractical for large-scale IPs with numerous paths.

Considering recent works proposing off-line evaluation of critical paths replicas [7], the publications present performance improvement of 50 percent for an FIR filter implemented on a 65 nm Cyclone IV FPGA. Such gain is also within the range of our own framework. However, similarly to the register-based methods and in contrast to our framework, this method relies on augmented CAD tools and STA analysis. Also, their calibration for in-the-field adaptation (e.g., to aging effects) consumes considerably more time than us, because it requires the off-line construction of a calibration LUT by performing extensive lab experiments.

Overall, the aforementioned comparisons are performed against the most representative works of the field and highlight the advantages of our framework. Additional comparisons, e.g., to older papers, lead to similar conclusions. The advantages are summarized in the higher level of modification, the faster calibration, and the limited resource overhead, which is independent of the size of the user IP: 8.9 percent LUT utilization in ZC7020 or less than 2 percent in ZC7100. On top of that, compared to the aforementioned works, the proposed is the only method to explicitly exploit the inter- and intra-die variability of the underlying FPGA. Therefore, the performance gain of our framework matches that of state-of-the-art works due to guard-band customization, and prevails by 10−14.7 percent when we also consider variability maps to place the design on the most efficient chip regions.

7 Conclusions

The current work evaluated process variability on 28 nm commercial FPGAs and proposed a user-level framework to exploit local differences for improving the FPGA performance, in-the-field. Based on a custom, programmable sensing network, we generated detailed variability maps for 20 randomly selected conventional and SoC FPGAs from Xilinx. The maps revealed up to 13 percent intra-die and 30 percent inter-die variation, which further increases by 1−3 percent due to voltage drop effects. Thorough performance analysis via intensive benchmarking and correlation techniques established that process variability has a direct impact on actual designs in classic user applications. For an example set of Zynq-7000 FPGAs and FIR/FFT benchmarks, we measured considerable speed differences ranging up to 7 percent for intra-die and 20 percent for inter-die comparisons. However, these differences were shown to be predictable and in accordance to the variability map of each device. Hence, we developed a generic framework to automatically test the user's FPGA(s) and IP(s) to provide the most suitable placement and frequency scaling for each application. Statistical results for relatively diverse chips show that users derive improved performance with a possibility of 85−100 percent and average gain around 10−14.7 percent, i.e., 5 percent exclusively due to intra-die and 9 percent due to inter-die exploitation, which increases to the area of 20 percent in extreme cases and to 56−138 percent when also applying guard-band customization. Compared to similar works, our framework has the advantages of avoiding modifications at fabrication or CAD or netlist level, while it has small constant resource overhead and limited calibration time allowing for in-the-field mitigation and dynamic adaptation.

Acknowledgments

This paper has been partially funded by the European Union's Horizon 2020 research and innovation programme, under project SDK4ED, grant agreement No 780572. The authors would like to thank Xilinx for the Zedboard donation, HiPEAC for the collaboration grant with the University of Manchester, and the National Technical University of Athens for the research fellowship supporting this work.

References

[1] ITRS, "International technology roadmap for semiconductors," 2015. [Online]. Available: http://www.itrs2.net/itrs-reports.html

[2] S. Mittal, "A survey of architectural techniques for managing process variation," *ACM Comput. Surveys*, vol. 48, no. 4, 2016, Art. no. 54.

[3] K. Bernstein, D. J. Frank, A. E. Gattiker, W. Haensch, B. L. Ji, S. R. Nassif, E. J. Nowak, D. J. Pearson, and N. J. Rohrer, "High-performance CMOS variability in the 65-nm regime and beyond," *IBM J. Res. Develop.*, vol. 50, no. 4.5, pp. 433–449, 2006.

[4] L.-T. Pang, K. Qian, C. J. Spanos, and B. Nikolic, "Measurement and analysis of variability in 45 nm strained-Si CMOS technology," *IEEE J. Solid-State Circuits*, vol. 44, no. 8, pp. 2233–2243, Aug. 2009.

[5] S. K. Saha, "Modeling process variability in scaled CMOS technology," *IEEE Des. Test Comput.*, vol. 27, no. 2, pp. 8–16, Mar./Apr. 2010.

[6] K. Maragos, G. Lentaris, D. Soudris, K. Siozios, and V. F. Pavlidis, "Application performance improvement by exploiting process variability on FPGA devices," in *Proc. Des. Autom. Test Eur. Conf. Exhib.*, 2017, pp. 452–457.

[7] S. Zhao, I. Ahmed, C. Lamoureux, A. Lotfi, V. Betz, and O. Trescases, "A universal self-calibrating dynamic voltage and frequency scaling (DVFS) scheme with thermal compensation for energy savings in FPGAs," in *Proc. IEEE Appl. Power Electron. Conf. Expo.*, 2016, pp. 1882–1887.

[8] J. L. Nunez-Yanez, M. Hosseinabady, and A. Beldachi, "Energy optimization in commercial FPGAs with voltage, frequency and logic scaling," *IEEE Trans. Comput.*, vol. 65, no. 5, pp. 1484–1493, May 2016.

[9] J. M. Levine, E. Stott, and P. Y. Cheung, "Dynamic voltage & frequency scaling with online slack measurement," in *Proc. ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays*, 2014, pp. 65–74.

[10] L. Y.-Z. Lin and C. H.-P. Wen, "Speed binning with high-quality structural patterns from functional timing analysis (FTA)," in *Proc. 21st Asia South Pacific Des. Autom. Conf.*, 2016, pp. 238–243.

[11] A. M. Islam, J. Shiomi, T. Ishihara, and H. Onodera, "Wide-supply-range all-digital leakage variation sensor for on-chip process and temperature monitoring," *IEEE J. Solid-State Circuits*, vol. 50, no. 11, pp. 2475–2490, Nov. 2015.

[12] Z. Ghaderi, M. Ebrahimi, Z. Navabi, E. Bozorgzadeh, and N. Bagherzadeh, "SENSIBle: A highly scalable sensor design for path-based age monitoring in FPGAs," *IEEE Trans. Comput.*, vol. 66, no. 5, pp. 919–926, May 2017.

[13] Moortec, "The implementation of embedded PVT monitoring subsystems in todays cutting edge technologies," 2017. [Online]. Available: http://www.moortec.com/download

[14] V. Betz, J. Rose, and A. Marquardt, *Architecture and CAD for Deep-Submicron FPGAs*. Berlin, Germany: Springer, 2012.

[15] P. Sedcole and P. Y. Cheung, "Within-die delay variability in 90nm FPGAs and beyond," in *Proc. IEEE Int. Conf. Field Programmable Technol.*, 2006, pp. 97–104.

[16] T. Tuan, A. Lesea, C. Kingsley, and S. Trimberger, "Analysis of within-die process variation in 65nm FPGAs," in *Proc. 12th Int. Symp. Quality Electron. Des.*, 2011, pp. 1–5.

[17] J. S. Wong, P. Sedcole, and P. Y. Cheung, "Self-measurement of combinatorial circuit delays in FPGAs," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 2, no. 2, 2009, Art. no. 10.

[18] H. Jones, "Semiconductor industry from 2015 to 2025," *SEMI*, 2015, http://www1.semi.org/en/node/57416

[19] S. Alam and G. Douglass, "Moore or less?" Accenture, 2017, https://www.accenture.com/t20180511T035547Z_w_/us-en/_acnmedia/PDF-61/Accenture-Moore-Less-POV.pdf

[20] K. M. Zick and J. P. Hayes, "On-line sensing for healthier FPGA systems," in *Proc. ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays*, 2010, pp. 239–248.

[21] H.-Y. Wong, L. Cheng, Y. Lin, and L. He, "FPGA device and architecture evaluation considering process variations," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, 2005, pp. 19–24.

[22] G. Nabaa, N. Azizi, and F. N. Najm, "An adaptive FPGA architecture with process variation compensation and reduced leakage," in *Proc. 43rd ACM/IEEE Des. Autom. Conf.*, 2006, pp. 624–629.

[23] F. S. Pourhashemi and M. Saheb Zamani, "Evaluation of FPGA routing architectures under process variation," in *Proc. ACM Great Lakes Symp. VLSI*, 2011, pp. 351–354.

[24] A. Kumar and M. Anis, "FPGA design for timing yield under process variations," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 18, no. 3, pp. 423–435, Mar. 2010.

[25] A. Alzahrani and R. F. DeMara, "Process variation immunity of alternative 16nm HK/MG-based FPGA logic blocks," in *Proc. IEEE 58th Int. Midwest Symp. Circuits Syst.*, 2015, pp. 1–4.

[26] L. Cheng, J. Xiong, L. He, and M. Hutton, "FPGA performance optimization via chipwise placement considering process variations," in *Proc. Int. Conf. Field Programmable Logic Appl.*, 2006, pp. 1–6.

[27] Z. Guan, J. S. Wong, S. Chaudhuri, G. Constantinides, and P. Y. Cheung, "A two-stage variation-aware placement method for FPGAs exploiting variation maps classification," in *Proc. Int. Conf. Field Programmable Logic Appl.*, 2012, pp. 519–522.

[28] P. Sedcole, E. Stott, and P. Y. Cheung, "Compensating for variability in FPGAs by re-mapping and re-placement," in *Proc. Int. Conf. Field Programmable Logic Appl.*, 2009, pp. 613–616.

[29] Z. Guan, J. S. Wong, S. Chaudhuri, G. Constantinides, and P. Y. Cheung, "Exploiting stochastic delay variability on FPGAs with adaptive partial rerouting," in *Proc. IEEE Int. Conf. Field Programmable Technol.*, 2013, pp. 254–261.

[30] A. A. Bsoul, N. Manjikian, and L. Shang, "Reliability-and process variation-aware placement for FPGAs," in *Proc. Des. Autom. Test Eur. Conf. Exhib.*, 2010, pp. 1809–1814.

Kostantinos Maragos received the BS degree from the School of ECE, Technical University of Crete (TUC), in 2013. He is currently working toward the PhD degree at the National Technical University of Athens (NTUA), School of ECE. His research interests focus on process variability and reliability of FPGAs, including radiation testing and verification of FPGA programming tools, as well as on design of parallel and reconfigurable architectures. He has participated in multiple relative projects funded by the European Space Agency. He is a student member of the IEEE.

George Lentaris received the BSc degree in physics, the two MSc degrees in "logic, algorithms, and computation" and in "electronic automation", and the PhD degree in computing from the National & Kapodistrian University of Athens (NKUA), Greece. His research interests include parallel architectures and algorithms for DSP, digital circuit design, image processing, computer vision, H.264/AVC and HEVC encoding, digital signal filtering, etc. He is currently a research associate with the National Technical University of Athens (NTUA), working on HW/SW co-design with single-, multi-, and SoC- FPGA platforms for space applications, as well as on process variability and reliability of FPGAs, including radiation testing and tool verification.

Dimitrios Soudris received the PhD degree in electrical engineering from the University of Patras, Greece. He is a professor with the School of Electrical and Computer Engineering, National Technical University of Athens, Greece. His research interests include embedded systems design, methodologies and design of hardware accelerators, HPC systems, reconfigurable architectures, low power VLSI design, as well as dependability and reliability of digital systems at application level and/or circuit level. He has published more than 400 papers in international journals and conferences with more than 2,400 citations. He is leader and principal investigator in numerous research projects funded from the Greek Government and Industry, the European Space Agency, and the European Commission. He is a member of the IEEE.