Time-Randomized Processors for Secure and Reliable High-Performance Computing

David Trilla[†,‡], Carles Hernandez[†], Jaume Abella[†], Francisco J. Cazorla[⋆,†]
[†] Barcelona Supercomputing Center (BSC). Barcelona, Spain
[‡] Universitat Politècnica de Catalunya (UPC), Barcelona, Spain
[⋆] Spanish National Research Council (IIIA-CSIC). Barcelona, Spain.

***Abstract*—Time-randomized processor (TRP) architectures have been shown as one of the most promising approaches to deal with the overwhelming complexity of the timing analysis of high complex processor architectures for safety-related real-time systems. TRPs apply randomization techniques to the policies controlling hardware resources with variable latency to make the execution time of programs running on top of such processors to follow a probability distribution. Thus, with TRPs the timing analysis step mainly relies on collecting measurements of the task under analysis rather than on complex, and often inaccurate, timing models of the processor that are hard to build. Additionally, randomization techniques applied in TRPs provide increased reliability and security features when compared with time-deterministic architecture counterparts. In this paper, we elaborate on the reliability and security properties of TRPs and the suitability of extending this processor architecture design paradigm to the high-performance computing domain.**

I. Introduction

Nowadays processors are used almost everywhere, from data-centers and supercomputers to small personal devices and intelligent transportation systems. Processors in each domain have been subject to different requirements in terms of area, power consumption, temperature, performance, reliability and security. For instance, in the high-performance domain reliability has been a second order concern for many years in comparison with performance first and power/temperature later. This has led to high-performance processor designs where reliability is often addressed at late design stages and typical solutions resort to adding error detection and correction means in caches and memories, and hardening some registers and latches to improve resilience in front of soft errors. Analogously, security has received much less attention in high-performance processors where most effort has been put in adding specific cryptographic accelerators rather than in protecting them from malicious attacks. Conversely, in some other domains like safety-critical systems, reliability and security have been primary concerns already in early design stages for many years due to their potential threads affecting human lives and the integrity of the systems themselves.

However, the need to push technology scalability limits further every generation in the high-performance domain has made faults to be more frequent and thus, reliability has become also a primary concern in this domain. Additionally, high-performance processors used in data-centers and servers have to meet strong security requirements to avoid malicious attacks stealing data from users co-hosted in the system. Also, reliability is crucial to keep security levels high since the exploitation of system reliability vulnerabilities has been shown as a potential threat for security [11].

Several years ago probabilistic timing analysis (PTA) [5] – built upon randomized timing behavior – arised as a new timing analysis paradigm with the aim to facilitate the timing-verification of complex processors running safety-critical applications. Among other approaches, PTA proposes the utilization of time-randomized processors (TRPs) as a way to enable the derivation of timing bounds using probabilistic methods. Typically, complex commercial off-the-shelf (COTS) processor architectures are not amenable for timing-critical applications since they include hardware features such as caches, branch predictors, and multicore technology that severely complicate the derivation of execution time-bounds for the programs running on top of these processors. PTA applied on top of TRPs allows to reduce the complexity of the timing analysis process. To do so, TRPs employ hardware randomization techniques to make the latency of jittery resources to exhibit a probabilistic nature and thus, to allow the application of extreme value theory (EVT) [20] to bound program's execution time. TRPs simplify the timing verification process since the derivation of worst-case execution time estimates relies on the collection of execution time measurements and not on complex timing models of the processor that are often incomplete or inaccurate.

The properties of TRPs to enable the utilization of high-performance processors in time-critical applications at a reasonable cost have been deeply analyzed in the literature [19]. Conversely, in this paper the focus is to highlight the reliability and security properties TRPs can offer to both high-performance and safety-critical domains. On one hand, the utilization of randomization techniques matches very well with the random nature of faults. In this sense, several approaches have recently shown the good properties of randomization techniques for both reducing the biased utilization of resources that leads to worse aging conditions [30] and achieving a graceful degradation of performance in the presence of faults [27]. On the other hand, the random nature of the cache conflicts that TRPs exhibit are particularly interesting to increase security against side-channel attacks where randomization techniques have shown great potential.

The rest of the paper is organized as follows. In Section II introduces the properties of TRPs. Sections III and IV discuss the reliability and security properties of TRPs respectively. Finally, Section V elaborates on the potentials of TRPs for the high-performance domain and Section VI draws some conclusions.

II. Achieving predictability with non-reproducible timing behavior

The main difference between TRPs and conventional (time-deterministic) processor designs resides on the way hardware resources exhibiting jitter, i.e. what factors trigger different latencies for those resources that do not exhibit a constant latency. We distinguish between two main sorts of jittery

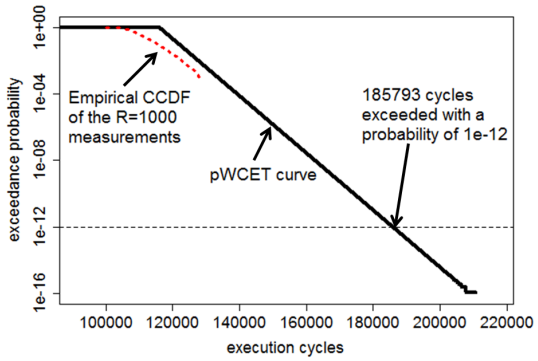

Fig. 1. Example pWCET obtained with a TRP.

resources namely intra-thread and inter-thread ones. Intra-thread jittery resources are, for example, cache memories in single-threaded cores whose latency depends on whether accesses resulted in a miss or a hit, and floating-point units (FPUs) whose latency often depends on the values operated. Inter-thread jittery resources are those exhibiting a latency that depends on the interactions of the different threads executing in the processor. For example, the bus arbiter and the memory controller are resources whose latency depends on the potential conflicts due to requests from threads running in the different cores. If simultaneous multi-threaded (SMT) cores are used, then those resources where contention could also occur across threads running in the same core should be also considered inter-thread jittery resources.

Regardless of its nature, TRPs apply specific policies to control the jitter. The two policies meeting the requirements of TRPs w.r.t. PTA consist in, either deterministically or probabilistically, matching or upperbounding the jitter during the analysis phase of the system w.r.t. what can occur during operation. For instance, when estimating the worst-case execution time (WCET) of a task (e.g., the airbag software of a car), execution conditions – including outputs – need to correspond to worst-case conditions during operation in terms of execution time. TRPs provide hardware support so that execution time measurements collected during analysis match or upperbound tightly those during operation. To upper-bound the jitter probabilistically, randomization (RND) techniques are required to make the jitter have a probabilistic nature during both analysis and operation phases. Deterministic jitter upperbounding (DUB) requires considering always the worst possible latency for those resources where it is used. RND policies are preferred because they do not lead systematically to the highest latency, as it is the case of DUB.

RND techniques have been applied satisfactorily to caches and shared resource arbitration. Regarding caches, random replacement and random placement policies have been proposed to match TRP requirements [13]. For the arbitration of shared resources the lottery bus [21] has been shown suitable for TRPs, while other new approaches like random permutations [16] have been shown be also suitable and offer improved performance. Regarding UB techniques, they have been used, for instance, for FP operations whose latency depends on the input values [19].

When all jittery resources are properly handled, EVT can be applied to the measurements collected for the programs executed in TRPs. However, TRPs will offer trustworthy WCET estimates amenable for the highest criticality levels when an appropriate methodology is followed. The timing analysis methodology for which TRPs were proposed is called Measurement-Based Probabilistic Timing Analysis (MBPTA). We refer the interested reader to [7], [6] to get the full picture of the MBPTA protocol. In short, time-measurements collected on top TRPs follow a probabilistic nature, are independent and identically distributed, and represent an upper-bound of the worst events due to the interactions of the different jittery resources that can occur in the processor. Once appropriate tests are passed, those measurements are used as input for EVT, which is a powerful statistical method to approximate the tail of a distribution. In the case of MBPTA, the tail of the distribution are high execution times. Then, the probabilistic WCET (pWCET) is the execution time value of the obtained distribution whose risk of being exceeded is upperbounded with a given arbitrarily low probability. The exceedance probability is chosen to be low enough so that safety standards allow neglecting it (i.e. residual risk [15]). For instance, if a program can be run up to 10 times per second (so up to 36,000 times per hour), we can use an exceedance threshold of 10^{-15} per run, which guarantees that failures per hour are below 10^{-9}. Figure 1 shows a pWCET curve in which a cutoff probability of 10^{-14} and the corresponding pWCET estimate are selected. Note that the pWCET curve is shown in the form of a complementary cumulative distribution function (CCDF) in logarithmic scale.

A. Randomized cache designs

Randomized caches are required to remove the dependence of execution time on memory placement, thus releasing end users from controlling where objects are placed in memory when different software modules are integrated. This makes cache jitter follow a probabilistic behavior and be independent of the memory location of objects. Random cache designs use random replacement and random placement to make pathological cache behaviors to occur with a given probability and thus, to allow applying EVT to the execution timing measurements to derive probabilistic estimates. To achieve this, the random placement function takes a random seed and the upperbits of the accessed address to determine the set that is accessed. Random placement implementations suitable for MBPTA are either based on a parametric hash function or on a random permutation of the index bits. The latter, a.k.a random modulo, uses a Benes interconnection network to perform a permutation of the cache set index bits that is driven by a combination of a random bits and the upperbits of the address. Figure 2 shows a schematic of random modulo. As shown in the figure, given a fixed seed the set were a given data is placed in the cache also depends on the upper bits of the address. This makes possible to make estimates not to depend on the actual layout that is finally deployed in the system since the spectrum of layout conflicts can get covered by simply varying the random seed. To ensure data consistency the same seed has to be kept for a given run and changing the seed requires flushing cache contents across runs.

B. Randomized arbitration

With a random arbiter the contention that requests experience when accessing shared resources – under worst contention conditions – can be captured probabilistically and thus, the particular way requests from different threads align becomes irrelevant. Random arbitration can be implemented

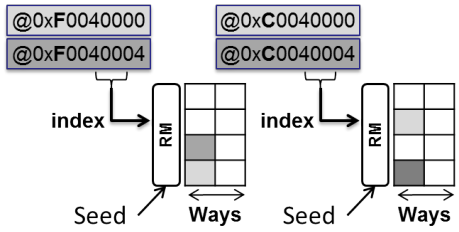

Fig. 2. Random modulo.

by simply selecting on a random manner the request that is granted access to a shared resource as proposed in the Lottery bus [21]. However, with the lottery bus the amount of time a given request waits until it is granted access is unbounded and the performance of the arbitration decreases. On the contrary, as shown in [16], performance guarantees can be improved by implementing an arbiter based on random permutations. This random arbiter generates a permutation window consisting on a random permutation of all possible contenders and serves requests according to the order determined by the permutation. Once a permutation is completed a new permutation has to be generated. With this particular randomization technique, we ensure that the maximum amount of time a request is waiting to access a shared resource is bounded. For instance if we have to arbitrate amongst 4 contenders, the worst possible contention is given by $2 \cdot (4-1) = 6$ arbitration slots, which represents that the request arrived just after the expiration of its assigned slot (the first one) and in the next random permutation it is assigned the last slot.

C. A Leon-based 4-core time-randomized processor

For illustrative purposes in this section we show what are the modifications needed to build a TRP starting from a baseline RTL processor description of the SparcV8 Leon processor [1]. Table I lists the processor features that have been modified in the baseline processor. First, caches were modified to make replacement policy in all caches random. This includes all L1 caches and TLB's as well as the L2 replacement policy. Regarding placement, TLBs do not need it since they are fully associative. For the L1 and L2, we use the most suitable random placement implementations that are random modulo (RM) for L1 caches [13] and hash-based random placement (hRP) for the L2 cache [18]. Since cores are simple in-order pipelined cores with always-taken branch prediction, we only regard FPU operations with input-dependent latency as non-MBPTA-compliant [19], [6]. We have modified the FPU so that each operation takes its worst latency regardless of the input data.

Arbitration policies in the bus and the memory controller have been modified to support random permutations as described in [16]. Regarding the L2 cache, the activity of the other cores is also irrelevant since caches are partitioned in the context of safety critical systems to avoid interferences that could invalidate WCET estimates. All in all, although the evaluation is performed using one core of the multicore, results remain valid for multicore workloads on fully-MBPTA-compliant processors. Finally, for randomizing hardware features, linear feedback shift registers based hardware pseudo-random number generators (PRNGs) [2] have been integrated in the processor.

TABLE I
A LEON-BASED TIME-RANDOMIZED PROCESSOR AND ITS DETERMINISTIC COUNTERPART.

	Time-deterministic	Time-randomized
Core	32 bit Sparc ISA 7-stage pipeline FPU	32 bit Sparc ISA 7-stage pipeline Fixed latency FPU
3*Caches	L1private 4-way 16KB Instruction 4-way 16KB Data LRU replacement Modulo placement	L1 private 4-way 16KB Instructions 4-way 16KB Data Random replacement RM and hRP placement
	L2 shared Unified 128K 4-way LRU Modulo placement	L2 shared Unified 128K 4-way Random replacement RM and hRP placement
	TLBs 1-Instruction, 1-Data 64 entries 4K pages Fully associative LRU	TLBs 1-Instruction, 1-Data 64 entries 4K pages Fully associative Random replacement

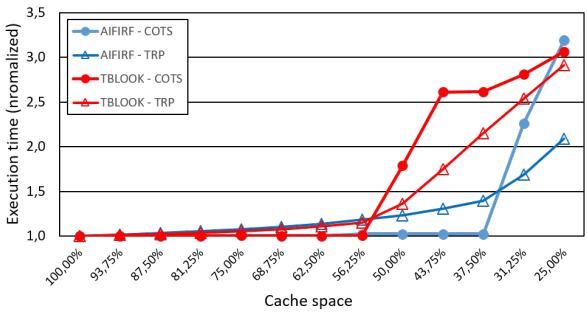

Fig. 3. Execution time (normalized) as cache size decreases for COTS and TRP designs.

III. RELIABILITY PROPERTIES OF TIME-RANDOMIZED PROCESSORS

The majority of current processors, regardless of the application domain, are provided with some form of fault tolerance. The most common protection mechanisms are the error correction codes included in cache structures but other forms of protection mechanism such as redundancy can be found as well in some domain-specific designs (e.g., for real-time or automotive domains [3], [14]). Hereafter, we analyze how TRPs help maximizing the effectiveness of existing protection mechanisms and also provide enhanced robustness capabilities.

A. Improving graceful performance degradation

Fault-tolerant mechanisms are able to keep system functioning despite the presence of faults but sometimes this comes at the expense of a reduction in performance. For example, when one or several cache lines are permanently damaged, protection mechanisms disabling faulty lines allow the processor operating correctly enlarging its lifetime. However, depending on the particular location of those faulty lines in cache, the performance provided by the processor can vary significantly [8]. Random cache policies included in TRPs make this degradation to occur more gracefully [28].

For instance, Figure 3 shows the execution time (normalized w.r.t. the fault-free case) for two EEMBC benchmarks [25], which is a benchmark suite representative of some automotive functionalities, as we decrease the cache space available in fully-associative DL1 caches. As shown, in the case of TRPs performance decreases gracefully as cache space decreases once the working set does not fit in cache anymore. Instead,

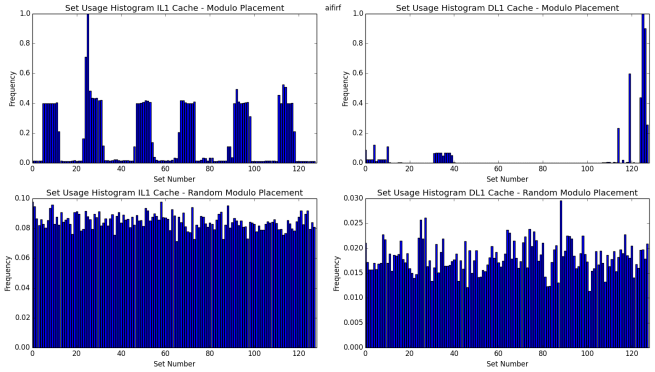

Fig. 4. Cache Utilization with and without random placement.

execution time in COTS processors remains almost unchanged until its working set does not fit in cache. Then, execution time grows suddenly, as it is the case of `aifirf` when cache space decreases from 37.5% to 31.25% and `tblook` from 56.25% to 50.0%. In the case of set-associative caches, the very same effect occurs in each individual set as shown in [8].

B. Reducing the bias in resource utilization

With conventional placement algorithms such as modulo, cache sets access distribution is completely program dependent. On the contrary, with random placement algorithms [13], [18] accesses to the cache sets are randomly distributed since random placement algorithms employ a combination of address tags with random bits from a random seed to generate the cache index so, for every run (or every time the seed is changed), a different set is accessed for any given address. Having a highly biased cache set utilization is expected to lead to higher degradation since the the most used sets are more exposed to hot carrier injection (HCI) [10] among other sources of transistor degradation, since HCI effects are directly proportional to the activity produced, which in turn depends on the access distribution across cache sets. In this context, having set access distributions as uniform as possible is very convenient to mitigate those aging effects [12]. This can be achieved easily with random caches when convenient random placement functions are employed [30]. Figure 4 shows that almost perfect set access distributions can be achieved with TRPs. The figure shows access distributions for IL1 (left) and DL1 (right), without randomization (top) and with randomization (bottom). Vertical axes are normalized w.r.t. the modulo placement case (without randomization).

C. Power stability and voltage noise resilience

TRPs performance is more stable than that of time-deterministic processors. The reason is that randomization removes systematic pathological scenarios that can lead to corner situations with significantly bad performance. Pathological cases occur, for example, due to systematic cache conflicts associated with repetitive access patterns resulting in conflict misses. Pathological situations can also occur in the access to the shared resources when requests in a loop systematically experience the worst possible alignment w.r.t. the arbitration window in a round-robin or FIFO arbitration scheme. On the contrary, in a randomly arbitrated resource the particular order in which resources are served can be associated to a given probability thus, reducing the possibility of having such extreme situations.

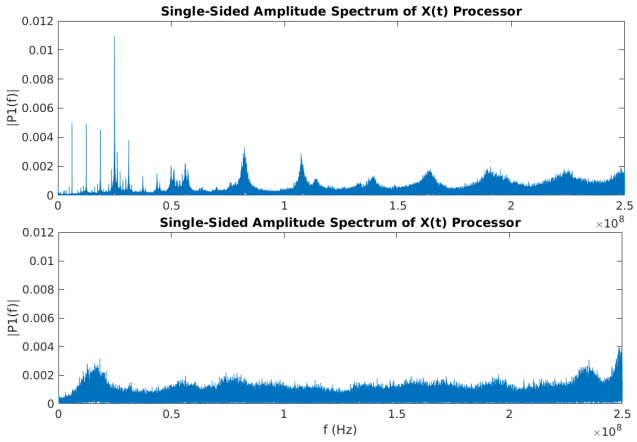

Fig. 5. Fast Fourier transform of the power consumption of pathological patterns in a TRP (a) and in an time-deterministic one (b).

Controlling the occurrence of pathological scenarios is not only important to achieve performance stability but also to increase the robustness of the system. On one hand, pathological scenarios negatively affect thermal and power profile of the system since they can potentially lead to the systematic occurrence of recurrent high power demanding events. Exposing the systems to high voltage and temperature conditions has been shown to significantly shorten the lifetime of processors [17]. On the other hand, the synchronization of power demanding events and the frequency at which those events occur has been shown the factors with major contribution to the voltage noise in the power distribution networks [9]. Voltage noise is created by power fluctuations and this effect is amplified when such fluctuations are repetitively caused by the synchronization of high power consuming events.

TRPs break systematic alignments of the events for which randomization techniques are applied. Thus, TRPs can diminish the impact of voltage noise effects. Figure 5 shows the fast Fourier transform of the power consumption resulting from the execution of a LRU pathological case in a time-deterministic processor (a) and in a TRP (b). As shown in the plot with a time-deterministic (conventional) processor with LRU high demanding power events can occur at a given frequency, whereas in a TRP such behavior is avoided by means of randomization. For deterministic designs, such events can get aligned with the resonance frequency of the power network distribution which will significantly amplify the impact of voltage noise fluctuations [9]. TRPs help decreasing the probability of occurrence of these events.

IV. Security Properties of Time-Randomized Processors

Side-channel attacks extract secret key data by exploiting the information leakage resulting from the physical implementation of the system. The most common side channel attacks are based on exploiting the information leakage of cache conflicts. Additionally, power analysis attacks have also been shown effective to threat system's security.

A. Cache side-channel attacks

Cache side channel attacks can be classified into contention-based attacks and reuse-based attacks [22]. In contention-based attacks the attacker may contend for the same cache set with the victim process and the contention results in eviction of

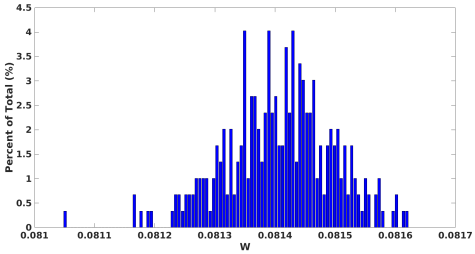

Fig. 6. Power consumption variability in a TRP.

one's cache line by the other. If the contention and eviction is deterministic, the attacker can infer the memory address of the victim based on the cache set it is mapped. Examples of this type of attack are the prime-probe [24] and the Evict-time [23]. Reuse-based attacks like the one in [4] exploit the fact that previously accessed data are loaded in the cache making execution time of the attacker to decrease if it accesses the same addresses the victim accessed.

Layout randomization has been shown an effective mechanism to protect against contention-based attacks [31]. Since TRPs are based on the utilization of random cache designs [13], these processor designs are inherently protected from contention-based attacks. On the contrary, as pointed out in [22], layout randomization is not enough to protect against reuse-based attacks since they do not care about the location of cache memory lines. Thus, additional randomization mechanisms such the one proposed in [22] are required. Similar mechanisms can be employed on top of TRPs to achieve enhanced security. However, as we explain later, we believe that controlling the sources of jitter will naturally enhance the security properties TRPs provide. Or in other words, the more features are time-randomized the more protected the processor will be.

B. Power analysis side channels

Power dissipation can also leak cryptographic information. When instructions are executed with fixed-time repetitive executions, they provide similar power profiles. Therefore, since cryptographic algorithms use multiple iterations for a given secret key, attackers can match the similar power profiles obtained to infer the cryptographic data. Randomizing the execution time delay to achieve protection against power analysis attacks was proposed in [26], where authors introduce random noise by means of randomly interleaving dummy instructions with the actual code when the execution of encryption algorithms is detected. Conversely, TRPs provide time randomization by default by randomizing the existing processor jitter and thus, avoid both inserting dummy instructions and having to detect the execution of cryptographic algorithms. Figure 6 shows the power variability resulting from running 1000 times an encryption algorithm in the TRP described in Table I.

V. Towards Secure and Reliable High Performance Time Randomised Processors

While TRPs proposed so far have focused on relatively simple processors targeting safety critical systems, we see potential for extending this paradigm to other domains with the aim of reconciling predictability, performance, security, and reliability aspects in a one-fits-all processor design.

A. Randomization granularity

In TRPs randomization is applied to those elements introducing jitter in the application's execution time. Therefore, the granularity at which randomization is applied completely depends on the particular processor architecture. Recalling the processor configuration shown in Table I, we observe that randomization in this relatively simple processor is only applied to caches and shared resources arbitration limiting the security and reliability capabilities of the randomization approach. For instance, the TRP presented in Table I is not sufficiently protected against reuse-based attacks [4]. To overcome this issue, or in other words, to increase reliability and security properties of TRPs, randomization should occur at a finer granularity. Luckily, high-performance processors incorporate a higher number of features that have to be randomized to make such processors compatible with PTA as explained later. The randomization of such hardware features will contribute to improve security and reliability properties of TRPs and reduce the need for introducing fine grain randomization techniques that only serve to enhance security or reliability properties of the processor. For instance, randomization approaches such as random pipeline stalls insertion [29] can enhance processor security but are not required for the application of probabilistic timing analysis techniques and cost performance.

B. Further Hardware Randomization

Applying EVT on top of more complex processor architectures would require controlling additional sources of jitter. Hereafter, we show two examples of hardware features where randomization is required to make the processor probabilistically analyzable.

Data prefetchers. The latency of cache misses can be partially hidden by using a data prefetcher. Thus, software execution time is significantly impacted by whether the data it requires were previously fetched by the prefetcher or not. Like in the case of caches, the effectiveness of the data prefetcher depends on the particular addresses the software uses and the policy employed to determine the data that are prefetched into the cache. For instance, one of the most common (and basic) prefetching policies is known as "one block lookahead", which prefetches a cache line **c+1** when the cache line **c** is accessed in the cache. To allow TRPs incorporating data prefetchers, new prefetching policies able to avoid systematic effects introduced by data prefetching mechanism are required. Again, randomization techniques appear to be the most suitable approach to deal with the systematic execution time effects introduced by prefetchers. Additionally, randomizing prefetching policies can also enhance TRPs protection against reuse-based attacks in a similar way [22] does.

Out-of-order execution. Out-of-order processors severely complicate the derivation of timing bounds due to the presence of timing anomalies. Timing anomalies are expected when a local execution time increase can lead to a global execution time decrease due to different resource allocations in an architecture. Since the timing anomalies cannot be effectively avoided in out-of-order processors, randomization appears as an effective means to deal with them. The randomization of resource allocation decisions would lead to a system where the impact of timing anomalies is randomized and in turn, probabilistically quantifiable.

C. Other applications of EVT

By construction, timing events occurring in TRPs can be modeled as a random variable, which allows applying EVT to reason about extreme effects occurring due to the particular alignment of those events. The main goal of TRPs is to allow attaching a (very low) probability to the alignment of events leading to the highest execution times. However, the same approach can be used to determine other extreme behavior that is of interest in processor design.

Voltage Noise. As explained before, power distribution is subject to voltage noise conditions that can lead to operational failures. The magnitude of voltage noise directly depends on the power fluctuations occurring in the processor as a consequence of the switching activity and how the activity of the different cores and within-core hardware resources aligns. To make processors resilient to such voltage fluctuations, extreme and very lowly probable alignment conditions have to be considered [9]. TRPs can help reducing overdimensioning safety margins by allowing an effective and probabilistic quantification of such extreme events. If the worst possible alignment of events is below a given failure probability, applying safety margins to these improbable events is unnecessary.

Maximum Power Density. With the current transistor integration levels, power density limits have become a significant performance limitator of current high-performance processor designs. If processors operate above their maximum power density allowed, this can have consequences on the integrity of the processors and can cause unrecoverable damage. Generally, processors are designed to make average workloads to operate below the threshold. However, the potential occurrence of systematic pathological behavior that is hard to avoid and predict can cause the system to operate above the threshold. When the threshold is trespassed, the system enters in a safe state where performance is decreased due to a voltage and/or frequency reduction to protect the processor. EVT applied on top of TRPs can be a useful tool to determine the likelihood of reaching such operating conditions and thus, help to appropriately designing system margins.

VI. Conclusions

In this paper we review and analyze the reliability and security properties of TRPs. As shown, TRPs provide unique properties that make them an ideal baseline to combat some of the reliability and security challenges that high-performance processors have to face these days.

Finally, we have also elaborated on the suitability of utilizing TRPs in markets other than the safety-critical systems one. Our conclusions are that, while some research and development is still needed to enable an efficient utilization of randomization techniques in some of the jittery hardware features included in most of the high-performance processors available today, the approach is promising since it will bring reliability and security properties needed in future processor designs by construction.

Acknowledgments

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under the PROXIMA Project (www.proxima-project.eu), grant agreement no 611085. This work has also been partially supported by the Spanish Ministry of Science and Innovation under grant TIN2015-65316-P and the HiPEAC Network of Excellence. Jaume Abella has been partially supported by the Ministry of Economy and Competitiveness under Ramon y Cajal postdoctoral fellowship number RYC-2013-14717. Carles Hernández is jointly funded by the Spanish Ministry of Economy and Competitiveness and FEDER funds through grant TIN2014-60404-JIN.

References

[1] Aeroflex Gaisler. *LEON 3 Processor*. http://www.gaisler.com/.

[2] P. Alfke. *Efficient Shift Registers, LFSR Counters, and Long Pseudo-Random Sequence Generators*. Xilinx, 1996.

[3] ARM. ARM Cortex-R Series Processors Specification. http://infocenter.arm.com/help/topic/com.arm.doc.set.cortexr/index.html.

[4] Daniel J. Bernstein. Cache-timing attacks on aes. Technical report, 2005.

[5] F.J. Cazorla et al. PROARTIS: Probabilistically analysable real-time systems. *ACM TECS*, 2012.

[6] F.J. Cazorla et al. Upper-bounding program execution time with extreme value theory. In *WCET Workshop*, 2013.

[7] L. Cucu-Grosjean et al. Measurement-based probabilistic timing analysis for multi-path programs. In *ECRTS*, 2012.

[8] J. Abella et. al. Low vccmin fault-tolerant cache with highly predictable performance. In *MICRO*, 2009.

[9] R. Bertran et. al. Voltage noise in multi-core processors: Empirical characterization and optimization opportunities. In *MICRO '14*, pages 368–380, Cambridge, UK, December 2014. IEEE Computer Society.

[10] V. Huard et al. Managing sram reliability from bitcell to library level. In *Reliability Physics Symposium 2010*, May.

[11] Y. Kim et. al. Rowhammer: Reliability analysis and security implications. *CoRR*, abs/1603.00747, 2016.

[12] E. Gunadi, A. A. Sinkar, N. S. Kim, and M. H. Lipasti. Combating aging with the colt duty cycle equalizer. In *MICRO 2010*.

[13] C. Hernandez et al. Random modulo: a new processor cache design for real-time critical systems. In *DAC*, 2016.

[14] Infineon. AURIX - TriCore datasheet. highly integrated and performance optimized 32-bit microcontrollers for automotive and industrial applications, 2012.

[15] International Organization for Standardization. *ISO/DIS 26262. Road Vehicles – Functional Safety*, 2009.

[16] J. Jalle, L. Kosmidis, J. Abella, E. Quinones, and F.J. Cazorla. Bus designs for time-probabilistic multicore processors. In *DATE*, 2014.

[17] Joonho Kong, Sung Woo Chung, and Kevin Skadron. Recent thermal management techniques for microprocessors. *ACM Comput. Surv.*, 44(3):13:1–13:42, June 2012.

[18] L. Kosmidis et al. A cache design for probabilistically analysable real-time systems. In *DATE*, 2013.

[19] L. Kosmidis et al. Probabilistic timing analysis and its impact on processor architecture. In *DSD*, 2014.

[20] Samuel Kotz and Saralees Nadarajah. *Extreme value distributions: theory and applications*. World Scientific, 2000.

[21] K. Lahiri, A. Raghunathan, and G. Lakshminarayana. LOTTERYBUS: a new high-performance communication architecture for system-on-chip designs. DAC '01, pages 15–20, 2001.

[22] Fangfei Liu and Ruby B. Lee. Random fill cache architecture. In *MICRO 2014*.

[23] Dag Arne Osvik, Adi Shamir, and Eran Tromer. Cache attacks and countermeasures: The case of aes. In *RSA Conference on Topics in Cryptology*, 2006.

[24] Colin Percival. Cache missing for fun and profit. In *Proc. of BSDCan 2005*, 2005.

[25] Jason Poovey. *Characterization of the EEMBC Benchmark Suite*. North Carolina State University, 2007.

[26] H. Qu, J. Xu, and Y. Yan. A random delay design of processor against power analysis attacks. In *2010 10th IEEE International Conference on Solid-State and Integrated Circuit Technology*, Nov 2010.

[27] M. Slijepcevic et al. DTM: Degraded test mode for fault-aware probabilistic timing analysis. In *ECRTS*, 2013.

[28] M. Slijepcevic et al. Timing verification of fault-tolerant chips for safety-critical applications in harsh environments. *IEEE Micro*, 34(6), 2014.

[29] Imagination Technologies. Mips32 m5100 processor core family datasheet. Technical report, 2031.

[30] D. Trilla, C. Hernandez, J. Abella, and F.J. Cazorla. Resilient random modulo cache memories for probabilistically-analyzable real-time systems. In *IOLTS*, 2016.

[31] Zhenghong Wang and R. B. Lee. A novel cache architecture with enhanced performance and security. In *2008 41st IEEE/ACM International Symposium on Microarchitecture*, pages 83–93, Nov 2008.