An Approach for Detecting Power Peaks during Testing and Breaking Systematic Pathological Behavior

David Trilla[†,‡], Carles Hernandez[†,⋆], Jaume Abella[†], Francisco J. Cazorla[†]
[†] Barcelona Supercomputing Center (BSC). Barcelona, Spain
[‡] Universitat Politècnica de Catalunya (UPC). Barcelona, Spain
[⋆] Universitat Politècnica de València (UPV). València, Spain.

***Abstract*—The verification and validation process of embedded critical systems requires providing evidence of their functional correctness and also that their non-functional behavior stays within limits. In this work, we focus on power peaks, which may cause voltage droops and thus, challenge performance to preserve correct operation upon droops. In this line, the use of complex software and hardware in critical embedded systems jeopardizes the confidence that can be placed on the tests carried out during the campaigns performed at analysis. This is so because it is unknown whether tests have triggered the highest power peaks that can occur during operation and whether any such peak can occur systematically. In this paper we propose the use of randomization, already used for timing analysis of real-time systems, as an enabler to guarantee that (1) tests expose those peaks that can arise during operation and (2) peaks cannot occur systematically inadvertently.**

I. Introduction

In embedded critical systems, the Verification & Validation (V&V) process builds not only on collecting evidence about their correct functional behavior, but also about their non-functional behavior including timing, power and temperature among other concerns. Due to economical and practical reasons, industry often relies on measurement-based approaches to derive such evidence [24].

The increasing performance needs in embedded critical systems are satisfied at a reasonable cost by using advanced (complex) hardware platforms. In those platforms, deriving test cases that trigger worst-case conditions becomes increasingly difficult for end users. For power verification, defining appropriate test cases and input vectors is critically important to identify whether (high) power peaks can occur and whether they can occur systematically [13]. Power peaks may lead to sporadic or frequent voltage droops that need lowering speed or stalling execution to preserve correctness [12], [28], [4], hence impacting timing of tasks in general, and real-time tasks in particular. For instance, power peaks may depend on the simultaneous occurrence of a number of events in cores, caches and on-chip interconnects, whose fine-grain control cannot be practically exercised. Thus, by analyzing power traces, end users are generally unable to tell whether higher power peaks can occur and, if so, whether they could occur systematically. The feasibility of triggering worst-case power scenarios determines whether real-time tasks, and especially those with some form of criticality (e.g. due to safety or security), can be successfully verified or not.

Recently, injecting randomization at hardware and software level has been proposed as a means to facilitate timing analysis of critical real-time tasks [18], [25] by (1) breaking systematic pathological timing behaviors, so that increasingly higher execution times have rapidly decreasing probabilities, and (2) making bad (long) execution times not to occur during test campaigns with probabilistically low bounds. The latter simplifies deriving the probability of occurrence of high execution times, i.e. those beyond the maximum observed execution time, with statistical means such as Extreme Value Theory, EVT [20].

However, to our knowledge the applicability and the specific application process of time-randomization solutions to mitigate power peaks and reduce the cost of power testing campaigns have not been explored. To cover this gap, we explore whether the randomization injected in time-randomized processors (TRP) [18] can be used in embedded systems to expose pathological worst-case power profiles and break their systematic occurrence, so that their impact is limited and can be properly accounted for. In particular, the contributions of this paper are as follows:

1) We identify pathological and systematic worst-case power dissipation profiles that may remain hidden during testing and occur during operation. We show how time-deterministic behavior of processors challenges, in general, identifying whether power measurements in the test campaigns expose relevant power peaks.
2) We make an extensive assessment of how time-randomized features in TRP contribute to the power dissipation variability by making power peaks manifest during testing and by mitigating their systematic occurrence.
3) Finally, the use of TRP, simplifies the use of statistical black-box techniques to bound probabilistically the frequency and magnitude of peak instantaneous power demand events.

The rest of the paper is organized as follows: In Section II we explain the background knowledge about validation, EVT and TRP. In Section III we present the challenges in power validation, and we explain its impact and how typically manufacturers deal with this issue. In Section IV we describe and explain the effects that TRP have on power dissipation and discuss how this helps in the detection and mitigation of systematic behaviors. We evaluate experimentally the impact of using TRP in power dissipation behavior in Section V. We discuss other related work in Section VI and finalize this paper with the conclusions in Section VII.

II. Background

In this work we aim at leveraging the randomization solutions devised for probabilistic timing analysis to deal with peak power concerns. In this section we introduce the relevant concepts of those timing analysis approaches.

A. Measurement-Based Probabilistic Timing Analysis (MBPTA)

MBPTA [6], [1], [25], [22] is a probabilistic framework to derive probabilistic timing bounds for real-time tasks. MBPTA builds upon two central elements: a platform with specific support to simplify reliable statistical analysis, and a statistical Worst-Case Execution Time (WCET) estimation tool. MBPTA requires that execution time sampling during the analysis phase of the system is representative (and a reliable bound) of the system behavior during operation [6], [1]. For that purpose, some components are time-randomized to relieve the end user from having to exercise any low-level control on the platform timing, and instead let randomization expose corner cases.

The main time-randomized processor components are cache placement and replacement policies [16], [17], [14], as well as arbitration policies for shared resources (e.g. NoCs and memory controller) [15], [27]. For instance, by randomizing cache placement, whether two addresses are placed in the same or different sets is a probabilistic event and hence, the impact of cache conflicts is probabilistically captured by sampling different (random) placements. In the case of caches, hardware support includes a hash module to randomize placement whose input are the address accessed and a random seed. Since each random seed leads to a fixed-but-random placement, placement is deterministic as long as the seed is not changed, thus allowing to access cache contents normally, but changes randomly by changing the random seed. Thus, the seed is changed across runs so that end-to-end execution times are random and independent. Software support can also emulate the same behavior by either placing objects in random memory locations at binary load time [17] or by randomizing their placement in the binary itself [19].

B. Statistical Analysis

MBPTA builds upon Extreme Value Theory [20] to predict the probabilities of high execution times (probabilistic WCET, pWCET, estimation). Based on an execution time sample, an appropriate use of EVT [6], [1], [25] allows delivering reliable exceedance probability bounds for high execution times, including values above those observed. MBPTA-CV implementation [1] in particular, (1) tests whether measurements in the execution time sample can be regarded as independent and identically distributed (i.i.d). This holds probabilistically for any MBPTA-compliant platform and hence, samples converge statistically to these properties. Then, MBPTA-CV tests (2) whether the pWCET can be reliably upper-bounded with an exponential distribution. As for i.i.d. properties, this property holds for the distribution sampled, so eventually the sample will also meet the statistical property. MBPTA-CV (3) imposes the use of a sample sufficiently large so that the number of high values is sufficient for a very tight pWCET estimation (typically between few hundreds and few thousands of measurements). Finally, MBPTA-CV (4) delivers a pWCET distribution fitting an EVT distribution with $\xi = 0$, thus with exponential slope.

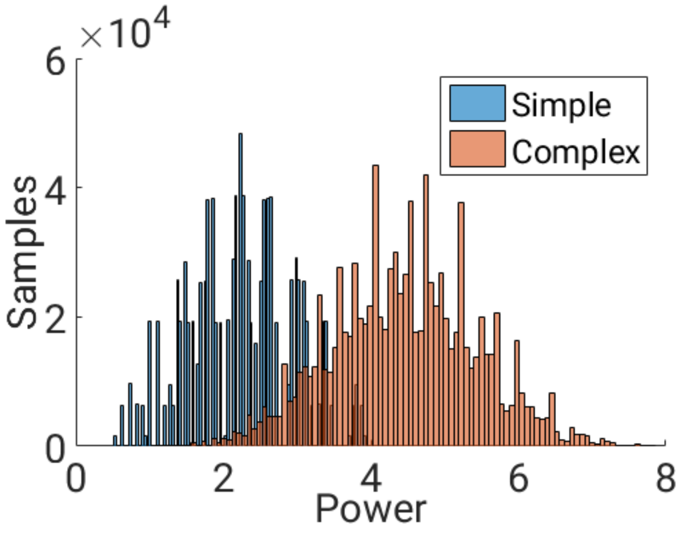

Fig. 1. Experiment showing the increase in power variability as processor complexity increases.

The key advantage of the use of EVT, as part of MBPTA, is that it is a black-box method that can be applied on any sample. However, the obtained distribution is relevant only for the system sampled. In the case of embedded systems, this implies that execution conditions used during analysis match (or upper-bound) those during operation, which is a too demanding constraint in the general case, especially for increasingly complex hardware and software. The use of time-randomization (in the case of timing) allows guaranteeing representativeness of analysis conditions w.r.t. operation ones, and thus, simplifies obtaining reliable pWCET estimates.

III. Challenges of Power Verification in Complex Processors

A. Power Delivery Network Sizing

Power Delivery Networks (PDN) in processors are typically designed to serve enough power ”in most cases”, but due to efficiency reasons, they are not designed to meet the power requirements in the absolute worst case, since it may occur only occasionally. For instance, Figure 1 shows the power profile of an arbitrary benchmark running in a simple and a more complex processors (see details in Section V). We observe that power variation is significant and the relative difference between the absolute worst case observed and the typical case is large, and it increases in absolute terms for increasingly complex designs. Thus, sizing PDNs for the absolute worst case would result in a waste of resources.

Overall, instantaneous power demand may surpass the capacity of the PDN, thus leading to a scenario where circuits become underpowered during relatively short time intervals, until the power demand decreases. In such scenario, voltage decreases to levels where correct operation cannot be preserved – often referred to as voltage droops – and actions such as decreasing operating frequency must be taken to decrease power demand and preserve correct operation [12], [28], [4]. While the effect of droops is relatively small in high-performance systems, in critical systems their impact on metrics like worst-case timing and power budgeting can be high.

B. Critical Real-Time Systems Verification

Processor verification is typically performed using power viruses [7] to characterize the corner power cases of the processor, size its PDN and accommodate mechanisms able to detect overly high power consumption and throttle (or even stop) operation to preserve processor physical integrity. For embedded critical (real-time) systems verification, processor integrity is not a concern, since appropriate means have already been set by the chip manufacturer. However, voltage droops as well as overly high sustained power dissipation may lead to performance issues due to, for instance, performance throttling. Authors in [28] show that a usual solution would be decreasing operating frequency down to its minimum (e.g. 1/32 of its maximum value) and increase it progressively as long as power demand does not exceed affordable limits. Hence, assessing during system analysis phases whether power peaks can occur, their magnitude and their frequency is critically important to evaluate whether timing bounds will be respected. However, end users often lack the knowledge of how power peaks arise in a specific processor, and lack the means to assess whether applications can trigger them. This may jeopardize the complete timing verification of the system if the impact of voltage droops is not properly accounted for during testing.

C. An Illustrative Example

Let us consider a simple example with two programs running simultaneously in different cores of a multicore processor. Figure 2 shows their joint power profile, with power measured every 43ns (see Section V) and the x-axis showing each of these observations over time. The two programs iteratively spend some time performing local (in core) computations, followed by a period of sustained memory write operations. As shown, frequent power peaks due to memory accesses interleave accesses of both programs and stay below 1.4W.

In a second experiment, we modified one of the benchmarks introducing few delays in between their memory accesses, thus effectively decreasing its average power dissipation and without impacting its individual maximum power dissipation. As shown in Figure 3, the time alignment of the power peaks changes slightly and, despite the overall average power dissipation decreases, the power peaks increase in magnitude, being above 1.4W sustainedly. If the PDN of this processor could only afford up to 1.4W of power, we would move from a scenario with no voltage droops to a scenario with systematic droops. And potentially, the latter scenario could not occur during testing, which would lead to the risk of missing deadlines systematically due to frequent unforeseen voltage droops.

In this particular example, we first created the two programs and run them without any specific synchronization. Then, since the platform used is a performance simulator, we had access to the internals of the architecture and could debug why some events were not occurring simultaneously and applied reverse engineering to cause a pathological systematic behavior where events align perfectly and lead to higher power peaks. However, in the general case this is not doable. In fact, we repeated the same experiment modelling a more complex processor with 4 cores instead of 2, allowing multiple memory requests in flight and increasing store buffers and, despite having full access to the architecture in the simulator, we were unable to exercise the control needed to synchronize events. Figure 4 shows the power profile of the execution of four benchmarks in the 4 cores and, as shown, some peaks occur from time to time, but it is unclear whether higher peaks can occur and whether they can occur systematically.

In summary, in complex hardware with time-deterministic behavior it cannot be assessed whether tests trigger the highest power peaks and whether those can occur systematically. This jeopardizes the confidence that can be obtained from test campaigns with uncertainty on the risk of deadline violations due to voltage droops since they cannot be bound reliably.

IV. Time-Randomization for Power Analysis

Power variation is highly correlated with the same events that create timing variation, which include cache hits/misses, arbitration delays in shared resources, variable delays in queues, etc. Time-randomization, either implemented by hardware or software means [18], allows exploring, for timing analysis purposes, the different outcomes of those events enforcing probability distributions that hold during analysis and operation. In this section, we analyze how time-randomization serves also the purpose of exploring power peaks, either in frequency or in magnitude, as well as the limits of time-randomization.

A. Event Alignment

Power peaks emanate from the simultaneous occurrence of multiple high-power events. Next we review how events relate to each other and the influence that time-randomization may have on them:

Potentially aligned events. Some events may align under certain conditions, such as those shown in Figures 2 and 3. By introducing time-randomization at a fine granularity (by making arbitration delays vary by few cycles, and making some cache hits become misses and vice versa) the power-hungry events that might concur are enforced to concur with some probability. This contrasts with the scenario drawn for

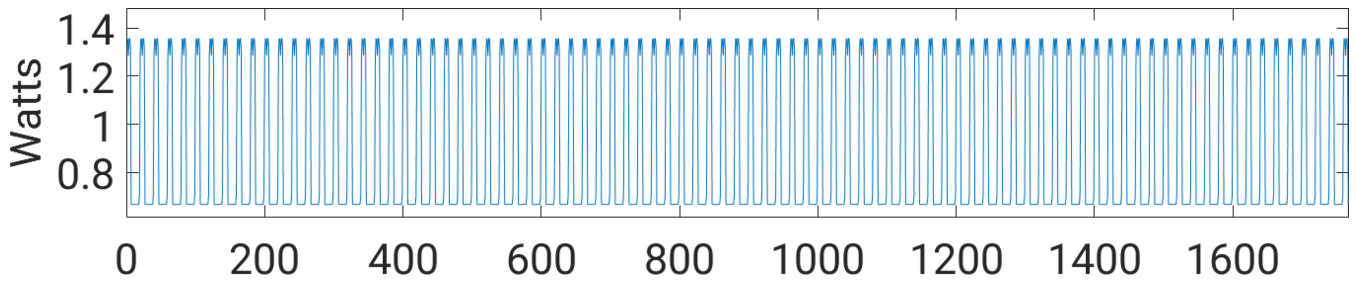

Fig. 2. Power profile on a conventional (simple) architecture when running two unsynchronized benchmarks.

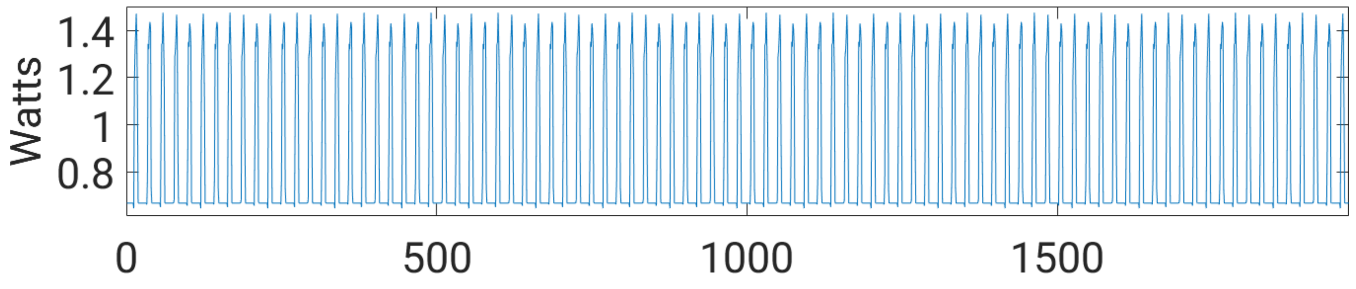

Fig. 3. Power profile on a conventional (simple) architecture when running two synchronized benchmarks.

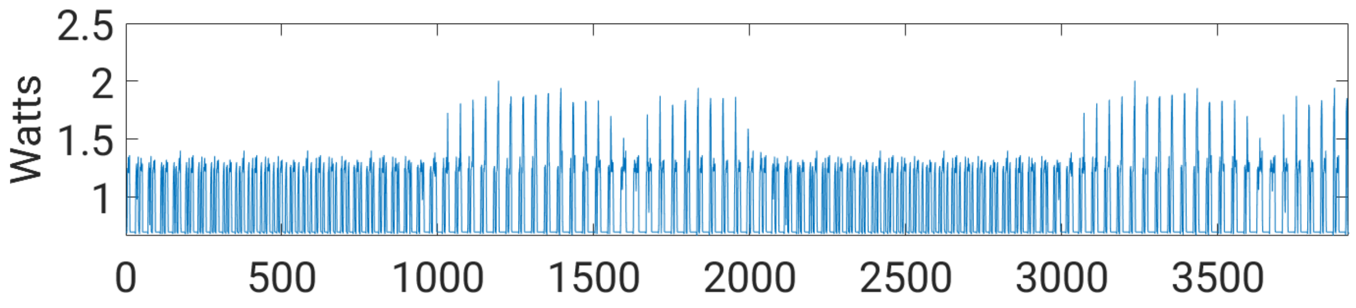

Fig. 4. Power profile on a conventional (complex) architecture when running four benchmarks.

time-deterministic platforms, in which events may never (or frequently) align with specific tests, and whose behavior can change completely during operation simply because the initial state of the processor or memory varies subtly. Overall, time-randomization allows making a probabilistic argument on the appearance of such type of events, and more importantly, make them not occur systematically.

Figures 5 and 6 show the same experiments done for Figures 2 and 3, but carried out on a time-randomized platform. In particular, random placement and replacement caches as well as random bus and memory controller arbitration are implemented, as detailed later in Section V [18]. As shown, both power profiles show those peaks occurring when power-hungry events align, but they do not occur systematically. Moreover, power profiles are probabilistically almost identical among them since event alignment occurs with similar probabilities across experiments. Therefore, power peaks are naturally exposed and can be accounted for conveniently.

Never aligned (or nonexistent) events. Some events may never align in time-deterministic systems because, for instance, the initial conditions that trigger their alignment never occur during operation. In this case, the difficulties emanate from the fact that, upon not observing their alignment, end users lack information on whether they can never align, whether tests simply failed to align them (as in Figure 2), or even whether higher peaks exist. Time randomization, instead, will make those events align with a probability, so they are observed and accounted for (perhaps pessimistically). Instead, when time randomization does not make them align, then confidence is gained that they cannot align with high probability.

It may also occur that time randomization causes some high-power events that would not occur without time randomization (e.g. causing few additional cache misses). While this effect is known to be very low since time-randomization degrades performance negligibly in the average case [18], it may lead to some pessimism due to triggering peaks that would not exist ever without time randomization.

Systematically aligned events. Some events may be highly aligned leading to systematic power peaks. If randomization may unalign them, it will allow reducing their impact due to

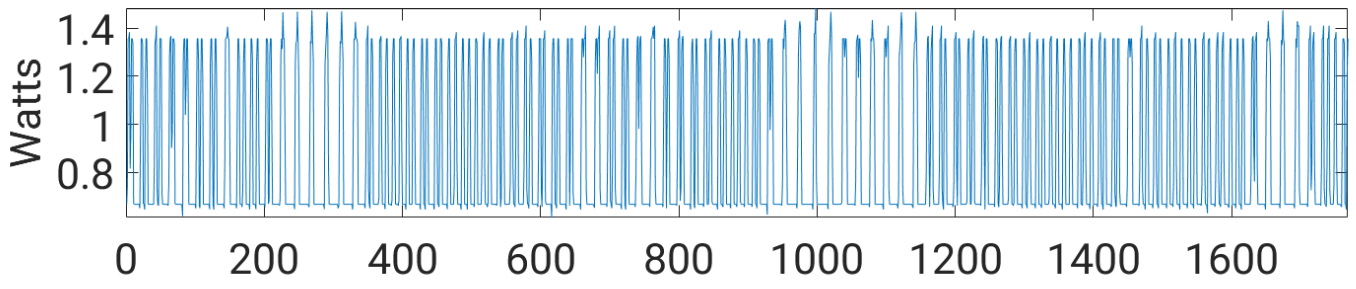

Fig. 5. Power profile on a time-randomized (simple) architecture when running two unsynchronized benchmarks.

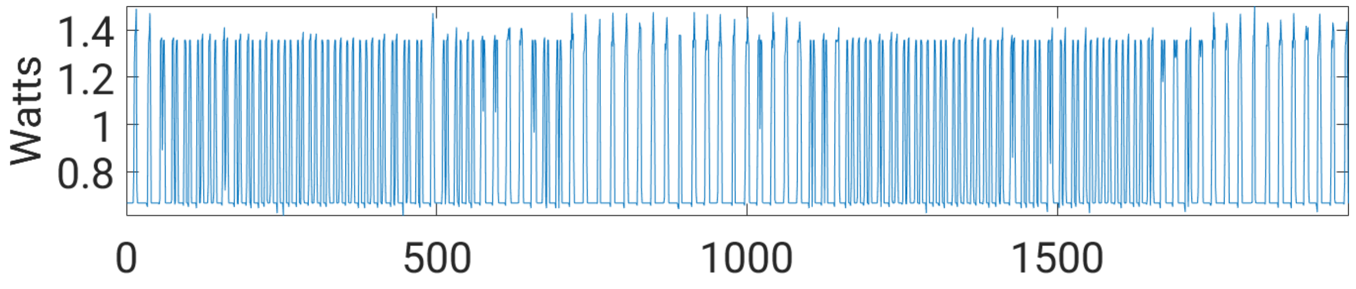

Fig. 6. Power profile on a time-randomized (simple) architecture when running two synchronized benchmarks.

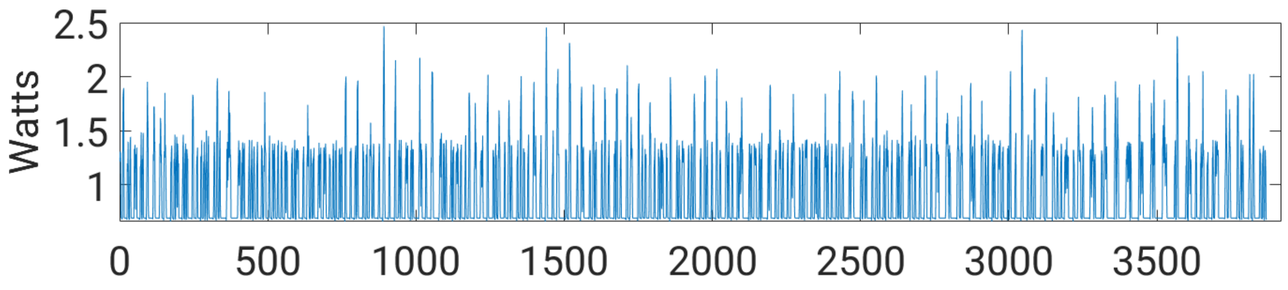

Fig. 7. Power profile on a time-randomized (complex) architecture when running four benchmarks.

voltage droops. Instead, if they cannot be unaligned because their occurrence is caused by events with no practical variability (e.g. sustained floating point operations), then randomization brings no quantitative difference. Yet, randomization brings confidence on the fact that high power peaks are observed during testing, so that their worst impact can be reliably predicted.

Overall, while time-randomization will have little influence in the average power dissipation and average number of power peaks across programs, it has two key advantages:

1) It guarantees probabilistically that peaks that can occur during operation are observed during testing.
2) If systematic behavior can be broken, it is effectively broken, thus allowing to account for peaks probabilistically without having to resort to overly pessimistic assumptions.

B. An Illustrative Example

For the sake of completeness, we have also repeated the experiment on the complex processor with time-randomization. As shown in Figure 7, power peaks are naturally exposed. In fact, some peaks are clearly higher than those observed in the time-deterministic setup. Thus, time-randomization allows accounting for their occurrence. Instead, in the case of time-deterministic platforms, it is unknown whether they can occur in practice and, if so, whether they can do it systematically, thus defeating any confidence had on the test campaign.

C. On Predicting Power Peaks

With time-randomized platforms we can use power measurements to predict both (1) peaks magnitude and (2) frequency. For that purpose, we build on the MBPTA-CV method, given that the properties needed for its input data are preserved:

- i.i.d.: MBPTA-CV inherits from the use of EVT the need for i.i.d. input data. While power measurements are not fully independent in practice at any time granularity, they quickly become independent since processor events last typically up to some tens of nanoseconds, which is the same order of magnitude of peaks duration to cause voltage droops. Hence, measurements at short distance are already independent. Moreover, EVT, in practice, does not need i.i.d. measurements but i.i.d. maxima which means

that dependencies across those values not belonging to the upper-tail of the distribution are irrelevant [22]. In any case, input samples passed to MBPTA-CV need to be tested against i.i.d. statistical properties for a reliable use of MBPTA-CV.
- Exponentiality. MBPTA-CV fits exponential tails, thus discarding heavy tails. This is only a reliable choice for distributions that have a maximum value, despite such maximum can be unknown. In the case of power, either due to temperature limitations or due to power supply limitations, a maximum power is known to exist and hence, the premise for the use of MBPTA-CV holds.

We identify two different ways of using MBPTA-CV in the context of power verification:

- High power peaks determination to retrieve either the highest power value that could occur with a meaningful probability (e.g. that could only be exceeded with a probability below 10^{-12} per time unit). Also whether a particular power value could be exceeded with a probability higher than a given threshold (e.g. whether a peak causing a voltage droop occurs with a probability above 10^{-12} per time unit). Note that the time unit relates to the granularity at which voltage droops may occur (e.g. a peak of few picoseconds would be irrelevant).
- Estimating the number of times that a given threshold is exceeded. By measuring the number of times the threshold is exceeded in each run of the program or the workload, we can estimate the highest number of peaks we can expect whose exceedance probability is below a given threshold (e.g. how many power peaks we can expect so that a higher number of peaks is expected less than once every 10^{12} runs). In this case, by using an upper-bound of the time to recover from a voltage droop (e.g. 100ns), we can increase the WCET estimate accounting for the maximum number of peaks expected (e.g. 50 peaks) multiplied by the recovery time for any such peak.

V. Quantitative Assessment

In this section we show a practical application of time-randomized platforms together with MBPTA-CV for power verification.

A. Experimental Setup

For controllability purposes, as needed for the examples in Sections III and IV, we evaluate our proposal using two connected simulators. We use an enhanced version of the microarchitectural simulator SoCLib [23], a cycle accurate performance simulator. We connected SoCLib to McPAT [21], a power simulator that uses access counts and processor configuration parameters to estimate power dissipation. We have configured McPAT to simulate a processor with usual characteristics for critical real-time systems: 700 MHz operating frequency, 90nm process technology, and 0.9V. However, our methodology is orthogonal to the specific parameters used. The

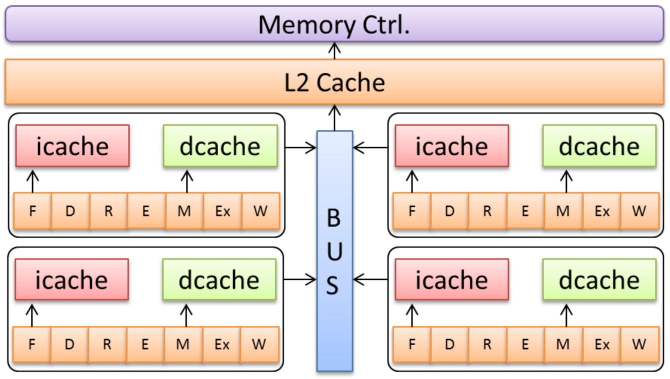

Fig. 8. Simplified block diagram of the NGMP Architecture

power sample rate is set to 30 cycles, effectively meaning every 43 ns. Finer sample rates have not been considered due to the intrinsic limits of the power model, which fails to spread power dissipation of an event across multiple cycles, thus creating anomalies at too fine rates (e.g. every cycle).

As processor model we use the Cobham Gaisler NGMP [2], a multicore processor for future space missions of the European Space Agency (ESA). The multicore, shown in Figure 8, is composed of 4 cores, with 16kB 4-way L1 Caches and a shared 256KB 4-way L2 Cache (it can be partitioned across cores), all with 32B lines, connected through a bus interconnect. The cores have 7-stage in-order pipelines. For the examples in Sections III and IV, we use a simple version with only 2 cores, 2-entry store buffers and up to one outstanding core request (L1 cache miss). For the complex version, we use the full 4-core setup with 8-entry store buffers and up to 6 outstanding core requests (typically non-blocking store operations).

The time-randomized setup uses random modulo placement L1 caches, random hash placement L2 cache, random replacement in all caches, and random permutation arbitration in the bus and memory controller [14]. The time-deterministic setup, instead, uses modulo placement and LRU replacement caches, and round-robin bus and memory controller arbitration.

Apart from the benchmarks used for the previous examples, we use the EEMBC Automotive reference benchmark suite [26], which includes a number of representative applications for critical real-time systems.

B. Power Verification

First, we evaluate the highest power peak expected. Whether this analysis needs to be done at chip level (so for the full workload) or at core level (so for each benchmark individually) relates to the organization of the PDN, and so the region where voltage droops can occur. For instance, in the case of a multi-core with an independent PDN for each core, it might be more appropriate for the methodology to require individual per core analysis of peaks, while with a shared workload or PDN, whole-system peak analysis might be more suitable. However, this is irrelevant for the application of the methodology. For instance,

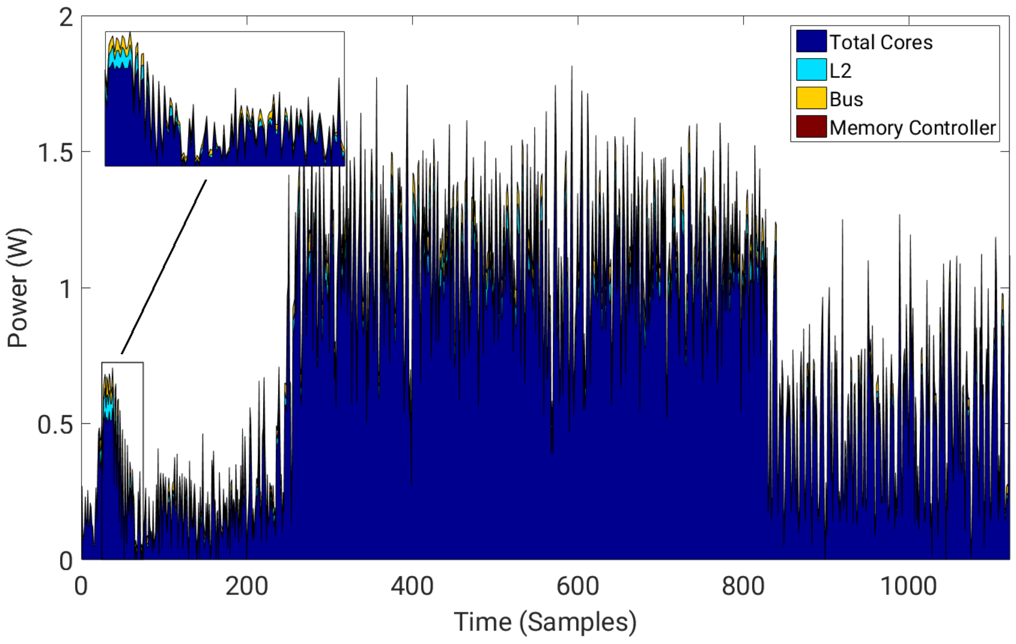

Fig. 9. Power dissipation over time of 4 different eembc in a randomized hardware

Figure 9 shows the power profile of the whole chip for one run of a 4-benchmark workload on the time-randomized complex setup. As shown, the randomized behavior of the power peaks can be noticed regardless of the integration level.

For simplicity and illustration purposes, the rest of the discussion is done for individual benchmarks executed in a single-core. Figure 10 shows the probabilistic power distribution for `aifirf` benchmark in μW, in the form of the complementary cumulative distribution function (CCDF). The red dashed line corresponds to the actual measurements, the black thick line to the estimated high power distribution, and the blue thin lines to the 95% confidence interval. As shown, by having the full distribution, we can obtain the power value for any exceedance probability or the exceedance probability for any power value.

While estimating the highest power peak for a given program may have several applications, in the context of critical real-time system we regard as more relevant estimating the number of peaks causing a voltage droop, so we focus on the latter due to limited space. For the sake of illustration, we set the threshold to determine a high power peak for samples above 95% of the maximum observed power. Table I shows how many peaks we observe in one run (execution) on the deterministic setup, the highest number of peaks per run observed across 1000 runs in the time-randomized setup and the number of peaks that could only be exceeded up to once every 10^{12} program runs[1]. The latter is derived using the number of peaks per run in the time-randomized setup as input for MBPTA-CV. As shown, the use of time-randomized setups allows us estimating the highest number of peaks expected, which ranges between few tens and few thousands of peaks. Then, by multiplying those peaks by the cost to recover from a voltage droop, the pWCET estimate can be padded conveniently to account for the cost of those voltage droops. Instead, the number of peaks for the time-deterministic setup comes without any guidance on how to determine whether a higher number of peaks is possible. In fact, the deterministic nature of such a setup could lead to arbitrarily higher power peaks due to events aligning systematically.

[1]Other values, e.g. 10^{-9}, deliver similar conclusions.

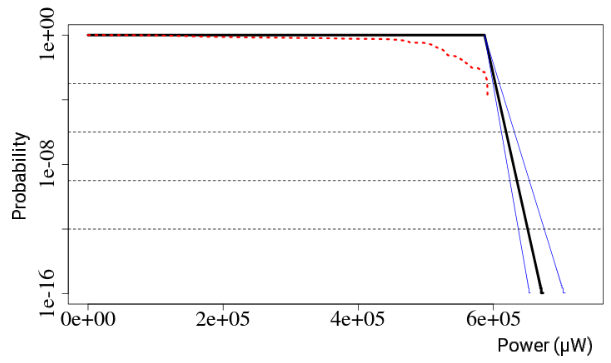

Fig. 10. Probabilistic curve and empirical sample of power dissipation values (in μW) for benchmark `aifirf`.

VI. Related Work

Power simulators have been used to provide power estimates despite the inaccuracies of their estimates, since they have been proven useful to evaluate the practicality of new techniques and perform comparisons [10], [5]. In our case, we rely on a particular simulator as a research vehicle to illustrate the applicability of our approach. However, our proposal is orthogonal to the source of the power measurements.

TABLE I
MAXIMUM PEAK COUNT FOR THE DETERMINISTIC AND RANDOMIZED ARCHITECTURES, AND PROBABILISTICALLY ESTIMATED NUMBER OF POWER PEAKS

EEMBC	**MAX Det**	**MAX Rand**	**Worst Case Number of Peaks** (10^{-12})
a2time	105	113	130
aifftr	148	186	292
aifirf	34	46	77
aiifft	142	181	281
basefp	135	145	160
bitmnp	56	60	87
cacheb	2850	3273	7794
canrdr	69	74	104
idctrn	10	27	93
iirflt	5	84	346
matrix	848	849	1230
pntrch	262	72	124
puwmod	483	489	509
rspeed	99	102	103
tblook	81	94	131
ttsprk	320	374	583

The use of EVT for power analysis has also being explored in [11]. In particular, this work targets maximum circuit power, for which worst-case scenarios can be created with appropriate power viruses. However, such a solution is not enough to estimate the highest power peak of a task since there is no way to relate testing data with operation behavior, and thus cannot be used for the problem considered in our work.

Resonant supply noise has also been deeply studied. Authors in [3] evaluate the events producing dangerous power peaks in a multicore, thus allowing to improve chip-wide strategies to power-up/use cores. Some authors solve the resonant supply noise problem that these power peaks cause by using a staggered core activation [8], whereas other works suppress such supply noise by using active damping circuits [9]. In any case, those works cannot be used to forecast neither the frequency nor the magnitude of power peaks caused by user tasks, as our proposal does.

VII. CONCLUSIONS

Power verification of embedded critical (real-time) systems is a mandatory step to assess their correct operation. Voltage droops caused by power peaks may lead to performance losses to allow recovering from those droops. Unfortunately, to the best of our knowledge, there is no practical way to estimate reliably how many such power peaks can occur in complex processors. In this paper we have presented an approach that, based on the use of time-randomized platforms, allows exposing power peaks during testing, breaking systematic behavior and estimating reliably the number of power peaks occurring during operation, so that the cost of recovery can be accounted for to prove that critical real-time tasks can execute timely.

ACKNOWLEDGEMENTS

This work has been partially supported by the Spanish Ministry of Economy and Competitiveness (MINECO) under grant TIN2015-65316-P, the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 772773), and the HiPEAC Network of Excellence. MINECO partially supported Jaume Abella under Ramon y Cajal postdoctoral fellowship (RYC-2013-14717).

REFERENCES

[1] J. Abella et al. Measurement-based worst-case execution time estimation using the coefficient of variation. New York, NY, USA, 2017. ACM.
[2] Aeroflex Gaisler. *Quad Core LEON4 SPARC V8 Processor - LEON4-NGMP-DRAFT - Data Sheet and Users Manual*, 2011.
[3] R. Bertran et al. Voltage noise in multi-core processors: Empirical characterization and optimization opportunities. In *MICRO*, 2014.
[4] K. A. Bowman et al. A 22 nm all-digital dynamically adaptive clock distribution for supply voltage droop tolerance. *IEEE Journal of Solid-State Circuits*, 48(4):907–916, 2013.
[5] D. Brooks et al. Wattch: a framework for architectural-level power analysis and optimizations. In *ISCA*, 2000.
[6] L. Cucu-Grosjean et al. Measurement-based probabilistic timing analysis for multi-path programs. In *ECRTS*, 2012.
[7] A. Joshi et al. Automated microprocessor stressmark generation. 2008.
[8] A. Paul et al. Staggered core activation: A circuit/architectural approach for mitigating resonant supply noise issues in multi-core multi-power domain processors. In *IEEE CICC*, 2012.
[9] J. Xu et al. On-die supply resonance suppression using band-limited active damping. In *ISSCC*, 2007.
[10] S.L. Xi et al. Quantifying sources of error in mcpat and potential impacts on architectural studies. In *HPCA*, 2015.
[11] N.E. Evmorfopoulos et al. A monte carlo approach for maximum power estimation based on extreme value theory. *TCAD*, 2002.
[12] M. S. Floyd et al. Adaptive clocking in the POWER9 processor for voltage droop protection. In *(ISSCC)*, 2017.
[13] K. Ganesan et al. System-level max power (sympo) - a systematic approach for escalating system-level power consumption using synthetic benchmarks. In *PACT*, 2010.
[14] C. Hernandez et al. Random modulo: a new processor cache design for real-time critical systems. In *DAC*, 2016.
[15] J. Jalle et al. Bus designs for time-probabilistic multicore processors. In *DATE*, 2014.
[16] L. Kosmidis et al. A cache design for probabilistically analysable real-time systems. In *DATE*, 2013.
[17] L. Kosmidis et al. Probabilistic timing analysis on conventional cache designs. In *DATE*, 2013.
[18] L. Kosmidis et al. Fitting processor architectures for measurement-based probabilistic timing analysis. *Microprocessors and Microsystems*, 47:287 – 302, 2016.
[19] L. Kosmidis et al. TASA: Toolchain Agnostic Software Randomisation for Critical Real-Time Systems. In *ICCAD*, 2016.
[20] S. Kotz et al. *Extreme value distributions: theory and applications*. World Scientific, 2000.
[21] S. Li et al. Mcpat: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In *MICRO*, 2009.
[22] G. Lima et al. Extreme value theory for estimating task execution time bounds: A careful look. In *ECRTS*, 2016.
[23] LiP6. *SoCLib*, 2011. http://www.soclib.fr/trac/dev.
[24] Magneti Marelli. System validation, 2018.
[25] K. Palma et al. On using GEV or gumbel models when applying EVT for probabilistic WCET estimation. In *RTSS*, 2017.
[26] J. Poovey. *Characterization of the EEMBC Benchmark Suite*. NCSU, 2007.
[27] M. Slijepcevic et al. pTNoC: Probabilistically time-analyzable tree-based noc for mixed-criticality systems. In *DSD*, 2016.
[28] C. Takahashi et al. A 16nm FinFET heterogeneous nona-core SoC complying with ISO26262 ASIL-B: Achieving 10^{-7} random hardware failures per hour reliability. In *(ISSCC)*, 2016.