HARPA
Harnessing Performance Variability

Project ref. FP7-612069
Call ref. FP7-ICT-2013-10
Activity ICT-10-3.4

Report on link between high-level reliability model and run-time engine

Simone Corbetta (imec)
Michalis Noltsis (ICCS)
Hans Cappelle (imec)
Francky Catthoor (imec)
Etienne Cappe (TCS)
Agnes Fritsch (TCS)
Eddy De Greef (imec)
Dimitrios Rodopoulos (ICCS)
Federico Terraneo (POLIMI)
Dimitrios Soudris (ICCS)

Report Number:	D4.5
Version:	v1.5
Date:	November 30, 2016

Revisions List

Date	Version	Author(s)	Description
November 11, 2016	v1.0	Simone Corbetta	First version: - Main document structure has been defined; - Text from introductory slides has been adapted; - Introduction to the objectives of the demo has been included as well.
November 20, 2016	v1.1	Michalis Noltsis	Major Changes - Sections 2 and 4 written
November 22, 2016	v1.2	Simone Corbetta	Minor changes: - Rephrasing and typo fixes - Figure 2 and Table 3 have been restructured Major changes: - Conclusions have been inserted
November 23, 2016	v1.3	Michalis Noltsis	Minor changes: - Address Simone's comments
November 25, 2016	v1.4	Simone Corbetta	Major changes: - Review after team meeting
November 30, 2016	v1.5	Simone Corbetta	- Added list of acronyms - Spelling review

Executive summary

This deliverable describes the results of the joint effort among imec, ICCS, TCS and (in minor part) POLIMI, to provide showcases to assess the methodologies designed by the partners.

The deliverable describes in detail the conceptual integration of the following technologies. The *HARPA-RTE* designed and developed by ICCS is a PID-based controller meant to mitigate performance variability induced by RAS mechanisms. The control is based on the dynamic selection of the most proper voltage/frequency operating point, thanks to the *DVFS driver support* developed by TCS targeting the reference ARM-based Freescale SoC. The need to support a larger number of operating points, imec and POLIMI have cooperated on the exploration of the voltage/frequency profile. Last, imec is able to provide a methodology to measure (instead of simulating) system-level failure rate (in terms of MTBF) as a function of the workload, at high stress conditions.

Table of contents

1 Introduction

HARPA project addresses the problem of *performance variability*; this is tied to *Reliability, Availability and Serviceability* (RAS) attributes of digital systems, due to the increasing susceptibility of silicon devices to variability and non-ideal manufacturing technology. Well-established guard-banding techniques are no longer applicable beyond the 10nm node, due to the unacceptable costs induced by the worst-case corner. Still, mitigation is required to reduce the costs associated to maintenance: lowering probability of field returns and intervention. The premises of HARPA are indeed toward this direction, as we read from the amended DoW [1] that the goal of HARPA is "*to enable next-generation embedded and high-performance heterogeneous many-cores*" in such a way that correct functionality and timing are guaranteed "*throughout the expected lifetime of a platform under thermal, power, and energy constraints.*"

In WP4, partners were called to define a cross-layer methodology that allows the system to adapt itself to dynamically changing operating conditions in order to guarantee reliability, while still meeting non-functional constraints such as power, energy and performance. The proposed solution comprehends technology developed from different partners, and including: a variability model of the impact of aging on digital circuits and systems, with particular stress on the workload characteristics; a control-based technique to respond to the overhead imposed by RAS mechanisms that intervene upon manifestation of a functional error; extended DVFS driver to support more operating points.

The deliverable is organized as follows. The central point lies in the PID controller, whose structure is reported in Section 2; two important contributions will be discussed in Section 3.1 and Section 3.2: measurement of failure rates from real hardware, and simulation of a general scenario in which the PID reacts to failures. A thorough discussion is provided in Section 4, and concluding remarks in Section 5.

1.1 List of acronyms

For convenience, Table 1 reports the list of technical acronyms used in the text, and their meaning. Acronyms related to the HARPA project (e.g., full name of partners) can be found in [1].

Table 1 List of acronyms

ACRONYM	MEANING
DVFS	Dynamic Voltage and Frequency Scaling
ECC	Error Correction Code
MA	Moving Average
MTBF	Mean Time Between Failures
PID	Proportional, Integral, Derivative
PMIC	Power-Management Integrated Circuit
RAS	Reliability, Availability and Serviceability
SoC	System-on-Chip
USB	Universal Serial Bus

2 Mitigating performance variability

Performance variability poses a serious threat to systems of aggressively downscaled technology nodes. In fact, time-zero and time-dependent variation phenomena deteriorate with the scaling of device dimensions, causing functional failures and dependability issues. To this end, ICCS has elaborated in detail [2] the use of RAS mechanisms that enable correct functionality: ECC, cache-line disabling or pipeline rollback. Although RAS interventions ensure reliable operation, they come at the cost of extra cycles and performance variability. Hence, the focus of our work in the current deliverable is to present a realistic case-study of the PID controller that constitutes part of the HARPA RTE and provides a widely-applicable solution to mitigate performance variability. We should stress the importance of another component of HARPA-Engine, the HARPA-OS, developed by POLIMI [3] that mitigates performance variability on the timing scale of a few seconds.

The theory of the PID controller has been presented in previous deliverables [4], [5]. The basic principle of the controller is its reactive response to performance fluctuations using DVFS to mitigate timing delays and meet constraints. In this section, we will present an overview of the theory of the PID controller with a brief summary of performance metrics and focus mostly on its implementation under a realistic case-study on the Freescale IMX6Q board. As a case-study, the controller has been developed on top of the Spectrum Sensing embedded application by TCS, a realistic application with hard deadline constraints, highlighting the use of the PID implementation.

2.1 PID controller premises

To control performance, the PID principle is to switch frequency and voltage of the processor based on a reactive response to monitored timing delays. To this end, we have introduced the definition of *slack* as the timing difference for specific code chunks with deterministic behaviour, named as scenarios [6], between their execution with nominal frequency in the absence of RAS interventions and their execution under fault manifestation. Therefore, negative slack is monitored when errors manifest and RAS events needed to ensure functional correctness are provoked. In this case, the PID controller should decide to boost frequency and voltage in order to ensure performance dependability and convergence of the slack to zero. On the contrary, positive slack is observed when frequency is over-boosted, translating into unnecessary energy losses. Then the PID should slow down the processor allowing it to operate on lower frequencies.

Another important aspect that should be noted is the intervention of RAS mechanisms and how they are perceived by the PID controller. Without delving in too much information, a distinction should be made between functional correctness and performance variability. After a functional error is detected the system's RAS mechanisms are responsible for correcting it and ensuring correct operation. The extra clock-cycles of such mechanisms are translated into timing delay or, in the context of the controller monitored delay and timing noise, forcing the system's slack to fluctuate. It is important to note that the controller cannot guarantee correct functional operation but rather focuses in performance variability.

To ensure slack convergence to zero, the controller monitors slack at specific time intervals and decides to switch frequency, choosing from a number of available operating points. In fact, in a spin-off experiment, we have verified that the number of DVFS points and therefore the granularity of the DVFS space is closely related to the settling time of slack, that is the time elapsed from the manifestation of a system disorder (in our case a RAS event) to the time at which the slack has entered and remained

within a specified error band. Hence, based on the monitored slack and from the combination of the proportional, integral and differential components, the new DVFS point is decided. The architecture of the PID controller has been elaborated in detail in [4] and is briefly shown in the next graph while the table summarizes what has been discussed so far.

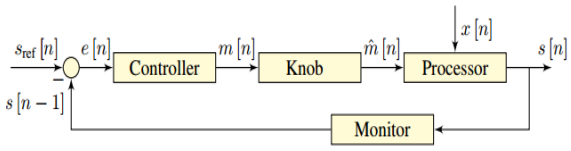

Figure 1 Block diagram of the PID controller

s_{ref}[n]	Target Slack specification	**x[n]**	Timing Noise – RAS event
s[n−1]	Slack of previous iteration	**s[n]**	Slack of current iteration
m[n]	Frequency multiplier	**e[n]**	Error signal

Table 2 Metrics used in our controller

2.2 PID controller implementation

After discussing the basis of the PID concept, we will now focus on its implementation on top of a realistic embedded application which is the Spectrum Sensing application by TCS. Having an error-detection mechanism has also been the work of imec [7] and was completed in previous deliverables. In our case, on the other hand, significant part of the work has been the development of a RAS mechanism that ensures correct systems functionality. Hence, we developed a rollback technique to correct errors. To elaborate in a more detailed manner, after an error has been detected (e.g., using the signature support as in [7]), the system is rolled back to a previous known state, where the operation is repeated. It is evident that this procedure introduces extra clock cycles and thus timing delay. With such a rollback event, the system can recover only from a transient (or recurring) errors; in the presence of permanent or semi-permanent (that is appearing for a long period of time) errors, correct system operation is not guaranteed.

The core of the PID implementation is the monitor used to measure timing delay and estimate slack and the knobs used to switch to other DVFS points. For monitoring purposes, we use the C `clock_gettime()` function that allows measuring time intervals between code chunks with a millisecond accuracy. To switch to different operating points, we used the DVFS driver developed by TCS that enables the use of as many as 50 different points in the frequency range of 396 to 1050 MHz. The matching voltage levels are estimated via experiments performed by POLIMI, whose results have shown a linear relation between frequency and voltage. It should be stressed that the DVFS timing overhead is measured approximately 750 μs. The main contribution to this is given by the time the protocol requires to set up a new voltage point (through I2C interface); this time can be reduced when using designs with on-chip regulators.

Despite being able to use up to 50 DVFS points and have a fine-grain granularity of the DVFS space, we selected only 14 points for our implementation. After performing a set of experiments we have concluded that after a certain amount of DVFS points, increasing their number does not reduce the settling time significantly. Figure 2 sums up our experimental results. As can be seen, increasing the DVFS points from 3 (which was the original version of unmodified DVFS driver) to 14, greatly reduces

settling time. On the contrary, a further increment reduces settling time less than 2 seconds or 1.1%. We should also note that in practise, embedded processors scarcely offer a pool of more than 14 DVFS points. Hence, our DVFS points configuration is shown in Table 3. Reference operating point for the PID controller is 792 MHz and 1.175 Volts.

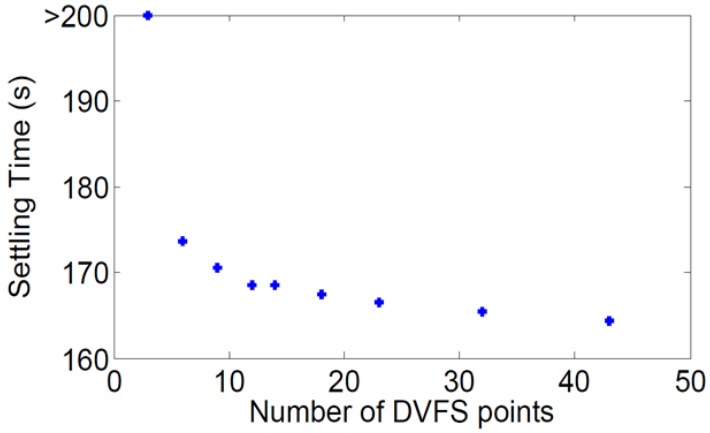

Figure 2 Graph showing the dependence of settling time on number of DVFS points

Table 3 Selected DVFS points in the PID implementation

Frequency [MHz]	Voltage [V]
396	0.975
444	1.0
492	1.025
540	1.05
588	1.075
636	1.1
696	1.125
744	1.15
792	1.175
840	1.2
888	1.225

936	1.25
996	1.275
1050	1.35

Lastly, after estimating the slack at a specific time interval (s[n]), the decision for the DVFS switch is based on the proportional, integral and differential parts of the PID controller. The proportional component K_p produces an output value that is proportional to the current error value. The integral term K_i is proportional to both the magnitude of the error and its duration therefore the integral part represents the memory of the controller. Lastly, the derivative part K_p is depending on the slope of the error. The gains of the three components are manually tuned during the experimentation process so as to achieve the minimum settling time. Tuning methods such as the Zieger-Nichols can also be used but lay outside the scope of our implementation.

3 Live demo

The purpose of the live demo is to visually demonstrate the use of technologies developed by partners in a real use-case. The adopted scenario is the commercial application from TCS. The audience will be able to understand and appreciate the following contributions:

- How to measure failure rates from a real hardware, running a reference application;
- How to simulate the impact of faults on performance variability of a system, and the intervention of the PID controller to mitigate such overhead.

3.1 Measuring failure rates

Simulation for reliability assessment is not scalable, due to the complexity of the reference design and the mutual intervention of reliability phenomena [7]. For this reason, the live demo is run throughout the experimental setup that has been previously discussed in [8]. This is reported for clarity in Figure 3.

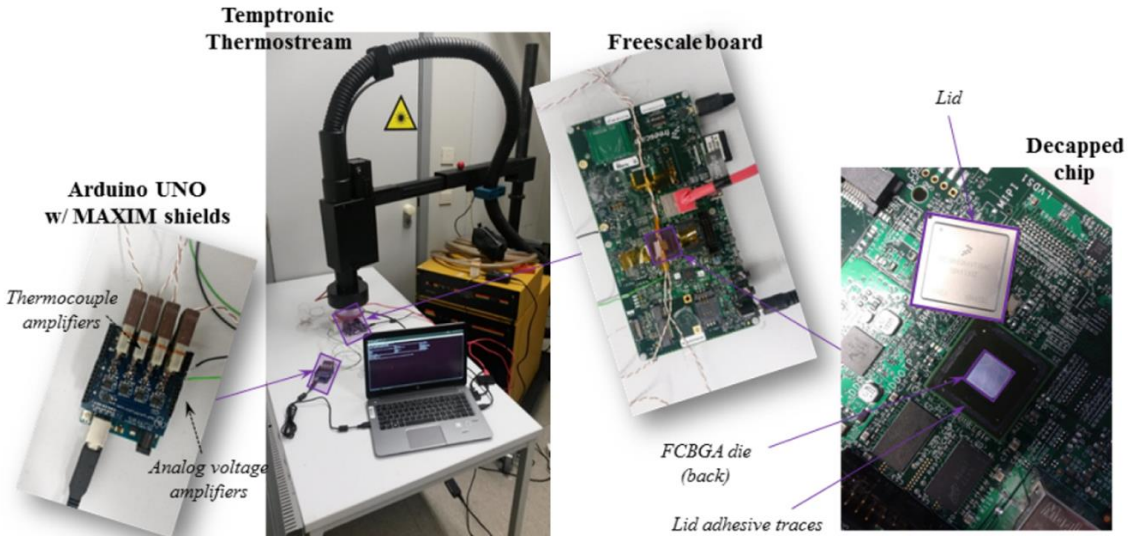

Figure 3 AMSIMEC laboratory experimental setup

The output of an experimental campaign using the aforementioned setup is an understanding of the time-varying failure rate, expressed in terms of MTBF. An example is given in Figure 4: data relates to an accelerated run at 130°C, 1.45V and 996MHz. The vertical axis shows the time between successive failures, and the evolution over time (an indication of aging) is highlighted through the MA filter (10 samples window) super imposed to the raw samples.

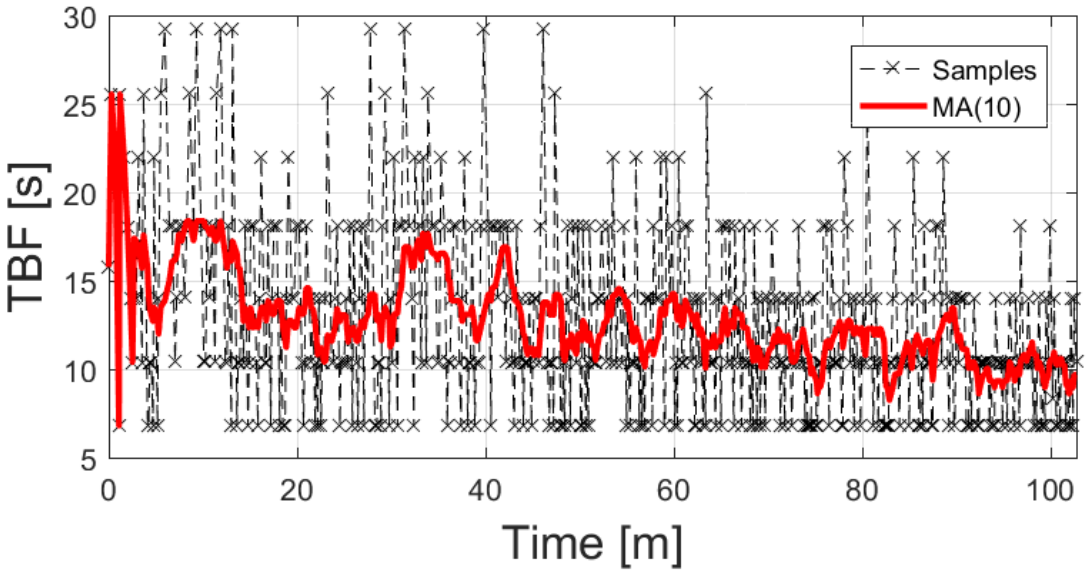

Figure 4 Example of TBF evolution over time

From multiple experiments at the same stress conditions or from multiple experimental campaigns at different voltages, we can derive the required failure rate. Notice that these rates are valid *at the accelerated test conditions* only. The extrapolation toward nominal conditions (namely, nominal VDD or temperature or both) is possible after an extensive campaign, as explained in details in [7]. However, only a minimal of these experiments have been carried out, such that the extrapolation can be performed; although not absolutely accurate this is sufficient (and necessary) to show the impact of faults on the PID behaviour (as explained in Section 3.2).

There exist three main stress knobs to control the generation of faults [9]: workload (i.e., instruction and data stream), temperature and operating voltage. The experimental setup [7] allows to increase the temperature and the operating voltage[1], while running a reference workload on the desired platform. The setup comprises:

- *Arduino UNO w/ MAXIM shields* – this is an Arduino-based platform that has been enhanced with MAXIM shields to develop a complete measurement platform. The platform reads analog voltages from the thermocouples and from on-board resistor, performs digital to analog conversion and sends digital traces to the laptop via standard USB interface;
- *Temptronic Thermostream* – aka “The Elephant”, this is a thermostream to force the operating (ambient) temperature of the chip to a desired value;
- *Freescale board* – this is the reference development platform where the application from TCS is running. Thermocouples are taped on top of the silicon die. The Ethernet and USB cables are used as interface mediums with the laptop;
- *Decapped chip* – it is evident the chip with no lid, to expose the silicon die to more reliable measurements;

[1] Frequency is important as well, and we map maximum operating frequency to every voltage step.

- *DVFS driver* – an important software component that allows us to modify the operating voltage and frequency of the platform at will.

The reference application is enhanced with the signatures concept, which allows to detect functional faults [7]. The comparator resides on the external laptop/setup controller to avoid common-mode noise.

3.2 Simulating PID intervention

After measurements, we can employ the failure rates found so far to mimic a real-case scenario in which faults trigger RAS mechanisms whence the experienced performance variability, that will be mitigated by the intervention of the PID controller. It is important to notice that the failure rates measurement and the simulation of the impact of faults are two different aspects of the same problem, and for robustness and honesty of results, we decided to decouple them. Although it would be possible to show at once the intervention of the PID controller using the experimental setup, this is not the appropriate way for different reasons:

- The objective of the measurements is to show the failure rates at stress conditions (required in order to be able to generate faults in a reasonable amount of time), while the objective of the PID controller (as for the demo) is purely functional: to show the intervention of DVFS knob as a mean to mitigate performance variability at nominal conditions (i.e., realistic scenarios);
- For demo purposes, the running workload (running application + support software) has been modified, modifying *de facto* the digital-level waveforms on the silicon. This impacts the failure rates [7]. It has been found, indeed, that the most sensitive part of the SoC lies in the I/O or in the memory interfaces. However, finding a solution to mitigate these parts was not part of the original objectives, and they are left as future works.

Please notice that the conclusions do not change with such limitations.

To show how the PID controller intervenes, the application has been enhanced and the laptop is now interfaced with the board using an intuitive user interface. The interface consists of two main windows: “Summary” and “Detailed” information is reported in Figure 5 and Figure 6 respectively.

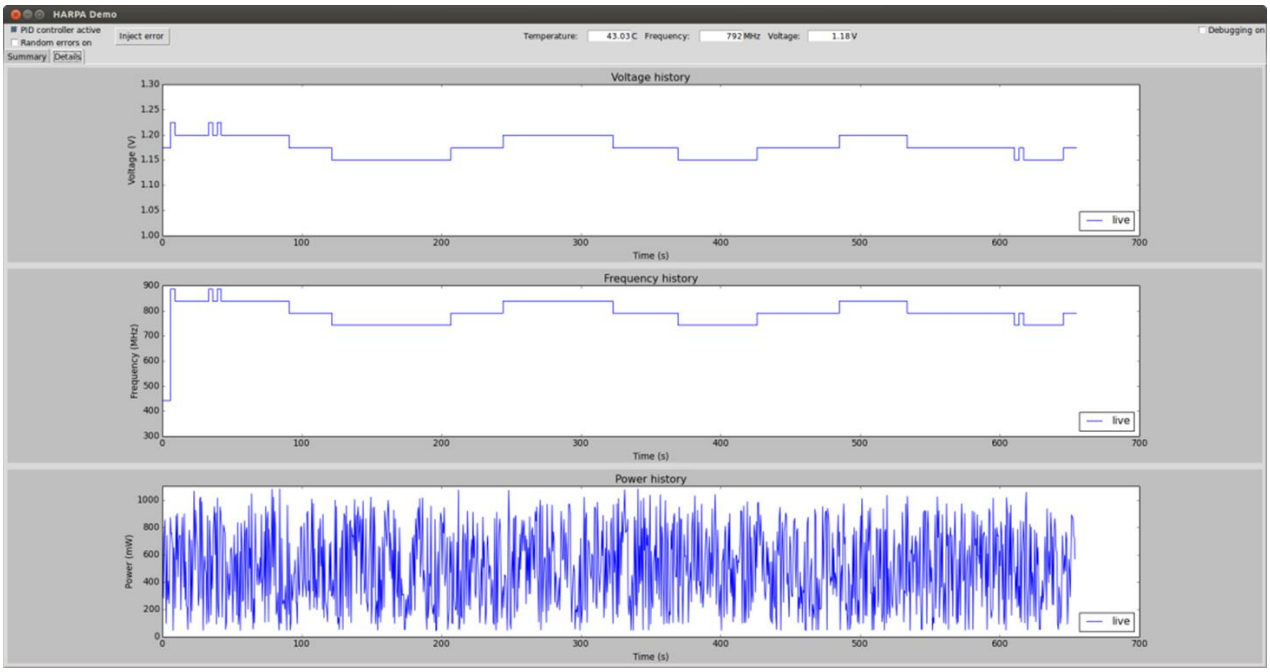

Figure 5 Detailed view window

The detailed window reports the following information:

- Voltage [V] and frequency [MHz] timed trace. Voltage and frequency are determined by the PID controller that chooses the most appropriate DVFS configuration from the set of available ones;
- Power [mW] timed trace reports the time varying power consumption by the SoC containing the ARM cores. Notice that it is not possible to measure the power consumed by the sole ARM subsystem.

The PID controller reacts to performance slack by modifying the operating point of the platform (refer to Section 2), with the intent to make the slack converging to a neighbourhood of 0. The slack will increase as a result of fault mitigation mechanisms, after a fault has been detected. For instance, once the on-line signature comparison process informs about the presence of a wrong datum (binary correctness), rollback mechanism will re-run the application from the latest checkpoint. This additional re-run has an overhead that degrades the performance margins from the real-time constraint. In the demo, faults are generated in either one of two ways:

- Physically, by means of high temperature and voltage;
- Or by injecting a fault directly in the running application, according to the fault rate for the specific platform/application pair [7].

For reasons discussed above, we have decided not to use the first option, but the second one as described in Section 3.1.

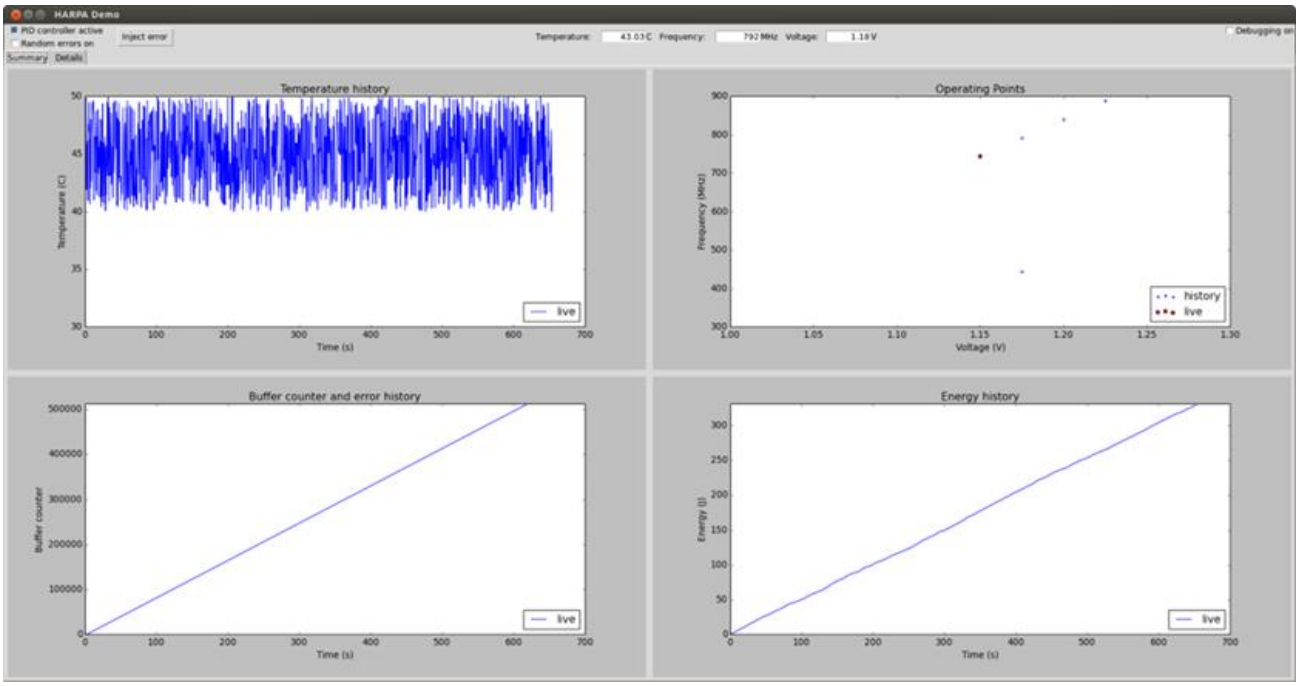

Figure 6 Summary view window

The summary window reports the following information:

- Temperature [°C] timed-trace. Values refer to Core#0, onto which the application is residing. Values are computed by the Arduino, starting from analog voltages in the thermocouples tip;
- Voltage/frequency plane. This plane shows the current operating condition and the points (DVFS steps) that have been used so far;
- Buffer counter. A timed trace representing the application advancing;
- Energy [J] timed-trace. Power integrated over time to show the long-term impact of the PID controller decisions.

4 Discussion

This important section discusses advantages and drawbacks of the employed technologies, as well as important aspects related to the PID control. In particular, the PID responsiveness lies in its ability to guide the slack to 0 convergence. In the following sections, the *time to converge* is the time required for the slack to reach such value from the fault detection time.

4.1 Employed monitors and knobs

An important aspect of the overall methodology is the use of *monitors* to gather information about the status of the system and the *knobs* used to perform the desired task. Monitors and knobs are discussed in [10], and it is out of the scope of this deliverable to go into the details, but the limitations of used monitors and knobs are hereby discussed.

4.1.1 Monitors

There are two important monitors in our setup: signatures and timing. A *signature* is computed as a function of the data computed by the application [8], and it is used to decide whether the platform is experiencing failures. In principle, a detected wrong signature (by means of comparison[2] with a golden one stored in persistent and safe memory) is an indication that something went wrong, somewhere. Without profiling it is not possible to detect the true source of the variability, but the system effect is visible. At this point, RAS mechanisms should intervene. The signature concept is very handy and powerful if application data is known in advance, since we can compute at once and store the golden reference signatures trace. And this is indeed matching the premises of the HARPA project. Nevertheless, a more robust signature definition should be found for the general case.

The PID controller process is based on the knowledge of a reference time: this is the wall-clock time that represents the constraint of the real-time application. Reliable and accurate timing measurements are required. The current software implementation allows to read time through an API to hardware-level timers. For faster processing, in the hardware implementation of the PID controller the timer could easily share the value through the system bus, avoiding expensive software call stacks.

4.1.2 Knobs

The PID process controls the frequency and voltage of the platform dynamically through the DVFS driver. The voltage/frequency actuator is scattered between software and hardware: the driver will initiate the request and the off-chip PMIC will regulate the operating voltage accordingly. This knob is generally widely used for power and performance management, but can be also proactively triggered to keep the temperature of the chip under control [10], [11]. The off-chip location of the control circuit is one of the reasons of the additional overhead incurred after a DVFS reprogramming (see details in Section 4.5), but on-chip regulators represent a suitable solution to this problem [12].

[2] Comparison is done on the laptop while computing the failure rates, while it is moved on-chip, along with the application, while simulating the effects of a fault.

4.2 Single fault reaction

An example of timed evolution of the slack versus frequency is shown in Figure 7. A fault is detected around t=90s, and this is visible by the fact that the slack drops. The 4s slack is due to the RAS mechanism (in this case a rollback). The PID boosts immediately frequency to the highest one, awaiting for the slack to decrease. After few moments beyond the specs (slack is positive), frequency is reduced and the closed-loop eventually brings the slack to 0. Here, the time to converge is around 170s.

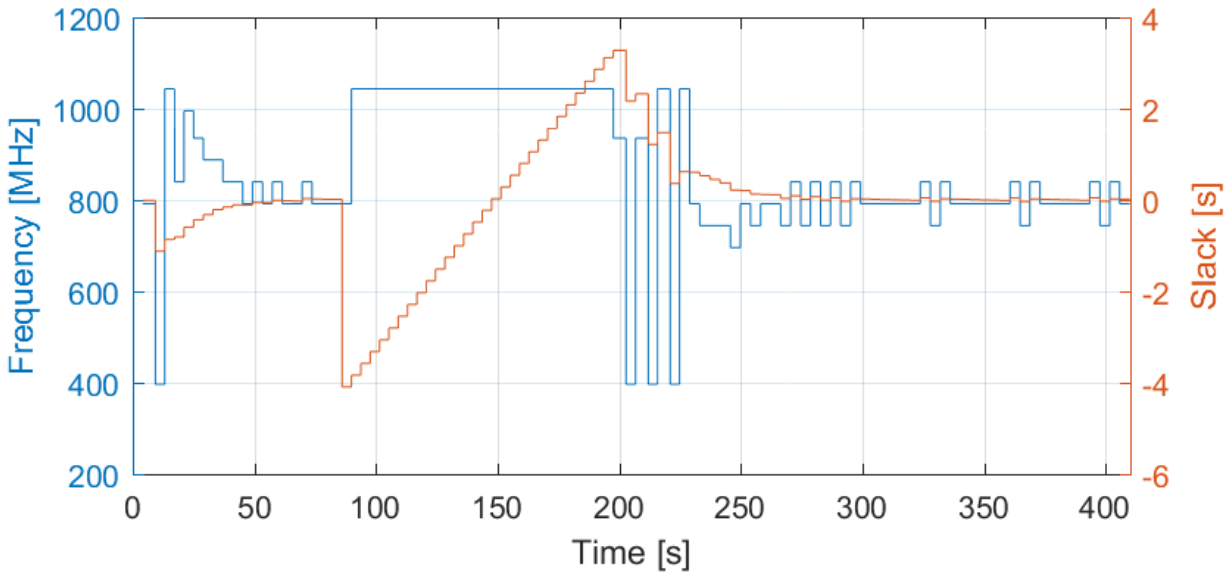

Figure 7 Slack and frequency over time. Rollback and overshooting region as well are visible. Slack eventually converges to 0, with frequency oscillating between close points.

The overshoot is appearing because of the aggressiveness of the PID controller; this can be reduced by fine-tuning the controller's gains. Nevertheless, in our case this overshoot remains under accepted margins and no further tuning is necessary. The same process is repeated for the next fault.

4.3 Multiple faults reaction

In the presence of multiple faults, two scenarios are to be considered: failure rate with a pace lower than the PID intervention, and with higher pace. In the former case the PID controller will react (by design) by keeping the operating frequency at its maximum, while time to converge will increase with the number of faults. In the latter case, on the other hand, the single-fault evolution shown above applies. Realistically, the failure rate at nominal stress conditions is expected to be lower than the one derived from accelerated experiments (see also comments in Section 4.6), so we envision to likely fall in the second scenario. The extreme case is given by looking at Figure 7: here we understand that after a

rollback event, zero-slack axis is crossed after approximately 60 seconds. Assuming that the PID controller is able (ideally) to recognize this in 0 time, error rates as high as 1 per minute will force the operating frequency to its maximum for the entire lifetime of the application. Therefore, there exists a theoretical limit that depends on the error rates and the speed of the implementation.

4.4 DVFS points selection

Figure 8 shows the DVFS points used during our experiment. The reference point at 792 MHz and 1175 mV is reported for comparison, and the maximum point in 1044 MHz and 1375 mV indicates the aggressiveness of the PID choices. In the experiment, the PID controller switches among eight different DVFS points, although 14 of them are available. Notice that the selection of such points is a function of the error rate, the controller gains and the granularity of the DVFS points (see Section 2.2). As a matter of fact, rates of errors can activate different groups of DVFS points and different tuning could activate different as well. The presented results refer to an instance of the possible scenarios.

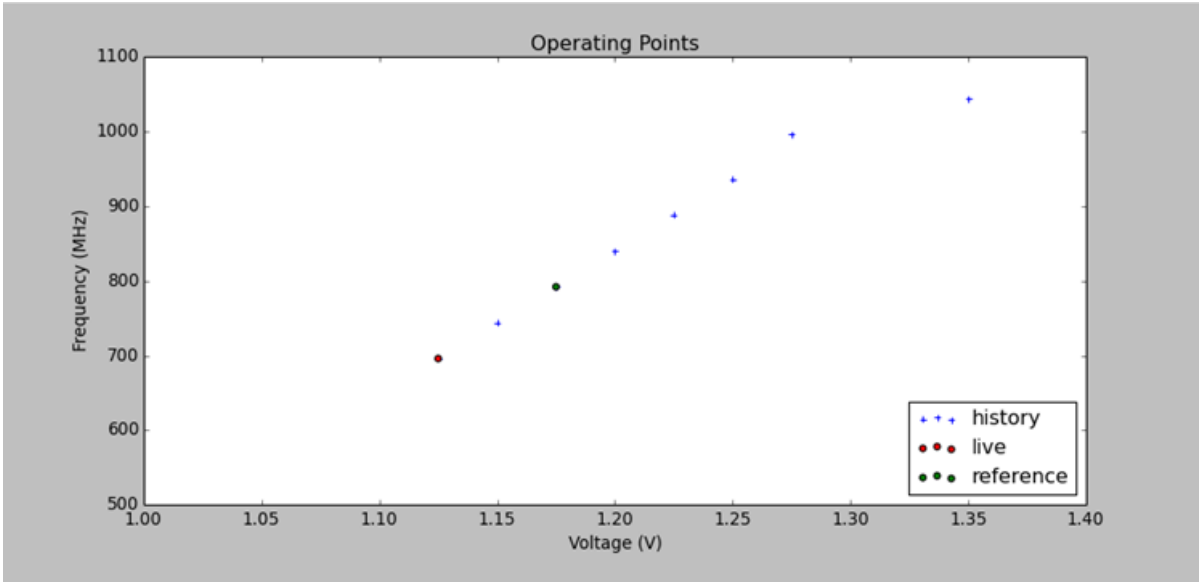

Figure 8 Selected operating points. Snapshot from the GUI demo

4.5 Performance of the PID intervention

It has been shown that the time-to-converge is approximately 170 seconds (refer to Figure 7). The time-to-converge depends on both the PID performance and the actuators performance. The PID performance can be maximized through a hardware implementation, where the

controller is closer to the source of variability, and software stack has not to be traversed. The actuators performance depends on two additional elements: the software required to setup the new operating point and the hardware required to perform frequency synthesis and voltage regulation. In the latter part, the voltage regulation is the worst case, and it can be highly optimized using on-chip regulators [12]. The Freescale platform has on-board (off-chip) voltage regulators that require accessing the PMIC through slow protocol-oriented I2C bus. The software required to initiate the actuation lies in the OS (device driver) and it can be optimized accordingly. However, the current implementation has been measured to cause almost 800µs of delay. In addition, it has been shown that DVFS switches cannot be performed often: one DVFS switch every 3 seconds makes the system freeze. This aspect is under investigation still, but it is clear that the ~4 seconds of delay are the major contributions to the high (unacceptable in real use-cases) time-to-converge. It is belief of the authors, however, that this time can be smashed down to the ms region using on-chip (silicon integrated) voltage regulators and optimized DVFS drivers.

4.6 Shaving margins

From the results of the experimental campaign, we learnt several aspects related to the performance and reliability of the platform, and regarding also margins taken by vendors. Indeed, we have found that these margins are sometimes too high under specific circumstances. An example is evident when looking at stress experiments [8]: the maximum operating temperature of the Freescale chip is reported to be 125°C [13], and the maximum voltage/frequency setting is 1.225V/996MHz. However, in general margins are taken in a conservative way, such that they apply for specific (worst-case) workloads. For the application of interest in HARPA (spectrum sensing) we performed several experiments at different temperature and voltage configuration [14]; these margins have been proved to be conservative in two extreme cases: at 160°C (i.e., +28% higher than the qualified point) the application still runs with no error at nominal DVFS point and, at low temperature, the platform is able to run at 1.7V (i.e., +39% than the nominal one) and nominal frequency. Hence, for some applications, the margins are just too conservative, and the gained margin can be redistributed to perform other operations. If we then consider that we can tackle reliability and performance variability through developed mechanisms (e.g., check-pointing + PID) it is realistic to claim that more risks can be taken during the normal operation of the chip (where the fault rates are much lower, and stress conditions are lower as well). This aspect can be tackled by future research, in which margins can be actually used to boost platforms in a reliable way.

Another aspect is the one of maximum operating frequency: characterization of VDD/f pairs has shown that above the platform is unstable (even at ambient temperature) above 1050MHz, whereas the 996MHz is limit imposed by the system integrator providing the DVFS driver.

Although Freescale allows (in some cases) to boost to 1.2GHz [13], we found out this node being really unreliable.

HARPA does not deal with exploiting such room, but with the proof that shaving margins is possible. The authors believe that this has been achieved by means of the technology developed during the project.

5 Conclusions

This deliverable describes the results of the joint effort among imec, ICCS, TCS and POLIMI. The main purpose of the deliverable is to highlight the contributions that each partner has given to a common perspective on performance variability aspects.

The joint demonstrator has covered several aspects of paramount importance:

- The standard commercial DVFS drivers are not flexible enough to address performance variability issues; a larger number of operating points is needed, to allow fine-grain and efficient utilization of the available resources;
- Performance variability can be tackled in a closed-loop control way, such that violation of real-time constraints remains (time-wise) locally placed: slack reclaiming is the concept driving this technology;
- At non-stressed conditions (i.e., ambient temperature), the voltage and frequency points belong to a linear relationship, and measurements are needed to perform characterization to spot the margins decided by silicon integrators;
- Failure rates models can be retrieved from ad-hoc measurements campaigns, even though nothing can be claimed (with the tools in our hands) about the real source of such variability.

In addition, as the outcome of this 3 year HARPA project and as an important part of the learning, some very relevant aspects are left for future research following up on HARPA:

- It is not clear the effects of the PID controller on the fault rate at short and long terms. Particular emphasis should be taken in future measurements on this;
- A hardware implementation of the PID controller (e.g., embedded in the power-management unit) is believed to be the most promising solution to ensure fast adaptation at run-time;
- The current error mitigation technique relies on check pointing. That is appropriate to mitigate errors in the memory organization inside the multi-core processor platform. When the errors (also) reside in the external memory buffers between the multi-core platform and the outside environment, other error mitigation techniques have to be applied, belonging to the class of “resending the data”. The specific way to implement this for the demonstrator platform of Freescale has not been tackled yet, and also that needs to be addressed as future work;
- This demonstrator belongs to the embedded application domain. We believe a similar approach, at least at the conceptual level, can be suitable also for the HPC domain where current and future peta- and exa-scale servers are very prone to transient wearout and degradation related faults [15]. Working this out and demonstrating that in detail is also an important future research goal;
- A link between the BTI analysis flow [9] and the concepts discussed in this deliverable is possible. Dynamic (on-line) reliability analysis can be conducted if the system scenarios methodology [6] is integrated with simplified analytical models of the target fabric (from extensive off-line simulation): the system will then be able to react to the reliability prediction as a function of the current workload, such that RAS mechanisms and intervention are foreseen while PID controller still mitigates performance variability in a proactive way.

6 Works Cited

[1] AA.VV., *HARPA - Harnessing Performance Variability. Annex I - "Description of Work",* 2016.

[2] F. C. D. S. Dimitrios Rodopoulos, "Tackling Performance Variability due to RASMechanisms with PID-Controlled DVFS," *IEEE COMPUTER ARCHITECTURE LETTERS,* vol. 14, no. 2, pp. 156-159, 2015.

[3] S. L. W. F. Giuseppe Massari, "Definition of amulti-objectivecontrol and optimizationstrategy," HARPA Harnessing Performance Variability, 2016.

[4] D. S. Dimitrios Rodopoulos, "System Scenarios HARPA RTE: Instantiation," HARPA Harnessing Performance Variability, 2015.

[5] D. S. Dimitrios Rodopoulos, "System Scenarios HARPA RTE: Assessment," HARPA Harnessing Performance Variability, 2016.

[6] N. Z. a. D. S. Dimitrios Rodopoulos, "System Scenarios HARPA RTE: Methodology," HARPA Harnessing Performance Variability, 2015.

[7] S. Corbetta, W. Meeus, D. Rodopoulos, E. Cappe, F. Catthoor and A. Fritsch, "System-Wide Reliability Analysis on Real Processor and Application under Vdd and T stress," in *GLSVLSI (to appear)*, 2016.

[8] S. Corbetta, F. Catthoor and W. Meeus, "D4.2 - Report on temperature measurement/estimation methodology," 2016.

[9] D. Stamoulis, S. Corbetta, D. Rodopoulos, P. Weckz, P. Debacker, B. H. Meyer, B. Kaczer, P. Raghavan, D. Soudris, F. Catthoor and Z. Zilic, "Capturing True Workload Dependency of BTI-induced Degradation in CPU Components," in *GLSVLSI'16 - Great Lakes Symposium on VLSI*, 2016.

[10] P. Englezakis, Z. Hadjilambrou, L. Ndreu, N. Panagiota, C. Nicopoulos and Y. Sazeides, "D3.3 - Final Report on Novel Monitors and Knobs," 2016.

[11] G. Massari, S. Libutti and W. Fornaciari, "Definition of a multi-objective control and optimization strategy," 2016.

[12] W. Kim, M. S. Gupta, G.-Y. Wei and D. Brooks, "System Level Analysis of Fast, Per-Core DVFS using On-Chip Switching Regulators," in *HPCA 2008*, 2008.

[13] Freescale Semiconductor, "AN4724 - i.MX 6Dual/6Quad Product Usage Likeftime Estimates," 2013.

[14] S. Corbetta, W. Meeus and F. Catthoor, "D4.3 - Proof-of-concept HARPA modelling framework," 2015.

[15] N. Raju, G. Raju, Y. Liu, C. B. Leangsuksun, R. Nassar and S. Scott, "Reliability Analysis of HPC clusters".