D3.5 Interim Report on Probabilistic Analysis for Mixed Criticality on Manycore Version 1.0

Document Information

Contract Number	611085
Project Website	www.proxima-project.eu
Contractual Deadline	m25, November-2015
Dissemination Level	PU
Nature	R
Authors	Jaume Abella (BSC), Enrico Mezzetti (UPD), Iain Bate (UoY), David Griffin (UoY), Benjamin Lesage (UoY) and Frank Soboczenski (UoY)
Contributors	BSC, INR, UPD, RPT, UoY
Reviewer	UPD
Keywords	Measurement-Based Probabilistic Timing Analysis, Manycore, Time-composable

Notices:
The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement n° 611085.

Change Log

Version	Description of change
v0.2	Initial Draft for internal review

Contents

Executive Summary **4**

1 Introduction **5**

2 Overview of the Platform and Simulator **6**
2.1 Architecture of the Manycore Platform 6
2.2 Tracing Capabilities 7
2.3 Architecture of the operating system ManyCOS 8

3 Impact of platform design on the timing analyses **10**
3.1 Intra-cluster resources 10
3.2 Inter-cluster resources 12

4 Inter-core Interference analysis **13**

5 Conclusions **15**

Acronyms and Abbreviations **16**

References **17**

Executive Summary

This deliverable forms part of milestone 2 of the PROXIMA project. The timing analysis of manycore systems is an evolving area of research. Existing literature [13, 19, 20] focuses with reason on the evaluation of the impact of the communication infrastructure and the resulting paradigms, e.g. Non-uniform Memory Architecture. Similarly, this deliverable is entirely concerned with the timing analysis of interferences that can be caused by effects external to the analysed task, i.e. tasks executing on other cores. It complements the deliverable D3.4 which provides both the timing analysis for tasks as if executing on a single core free of interferences and an interference computation model. This deliverable makes no assumptions about how the intra-core timing analysis is performed.
The techniques discussed in this deliverable comply to the PROXIMA approach to mixed-criticality systems (MCS) where tasks of different criticality execute concurrently. This approach relies on the use of hardware and software mechanisms to reduce the dependences on inter-core contention caused by resource sharing. The aim is to provide timing composability and performance isolation, asymmetric if need be to cater to the highest criticality tasks, without the cost of physical isolation or provisioning incurred by multi processor or statically partitioned platforms. The use of a single manycore system to host tasks of varying criticality further decreases the number of interconnected systems and alleviates the complexity of the system integration. At this stage of the project the specific focus is on analysing the MBPTA (Measurement-Based Probabilistic Timing Analysis) compliant PROXIMA manycore platform. The PROXIMA manycore platform has been designed with time-composability in mind, to ensure the estimates derived for a task in isolation, using the techniques for intra-core timing analysis presented in D3.4, are valid irrespective of the contention it can experience on the deployed system.
The principal purposes of this deliverable are to provide an update on the progress for manycore systems and to provide the reader with a sufficient introduction to understand the requirements of the analysis, the research challenges in meeting those requirements, and the applicability of the methods to the considered platform. It therefore introduces the following:

1. *Platforms* - One platform will be introduced: a Bespoke Manycore Platform (BMP) developed within PROXIMA.

2. *Manycore Simulator* - The simulator that provides inputs into the timing analysis based on the BMP.

3. *Timing Analyses* - This deliverable focuses on the impact of inter-core interferences on the BMP. Intra-core effects are captured as per the EVT-based timing analyses presented in D3.4. Other methods will be surveyed to understand how interferences on the platform affects timing analyses. We also discuss the application of the inter-core interference analysis presented in D3.4 to the BMP.

1 Introduction

The principal purposes of this deliverable are to provide an update on the progress for the PROXIMA manycore platform and to provide the reader with a sufficient introduction to understand the applicability of Measurement-Based Probabilistic Timing Analysis (MBPTA), i.e. the assumptions behind the analyses and the contribution of the BMP. In particular, we discuss the impact of the design choices at the platform level on the time-composability of the derived timing estimates. Composability is the favoured approach to mixed-criticality scheduling within PROXIMA. The PROXIMA platforms deploy hardware and software techniques to reduce the dependencies of the performances of high criticality tasks on their low criticality co-runners. The analyses by design of the platform derive for each task estimates valid irrespective of the contention suffered at deployment. This deliverable therefore introduces the following:

1. *Platforms* - One platform will be introduced: a Bespoke Manycore Platform (BMP) developed within PROXIMA. The platform, both architecture and supporting operating system (OS), is outlined further in Section 2 of this report. More information on both aspects can be found in their respective deliverables, D2.8 and D2.7 for the architecture and OS.

2. *Manycore Simulator* - The simulator that provides inputs into the timing analysis based on the BMP. The simulator is also covered in Section 2 of this report. It is presented in depth in D2.8.

3. *Timing Analyses* - To illustrate the drive towards the selected manycore platform, and the applicability of the EVT-based timing analyses defined in the context of multicore platforms (D3.4), the impact of the manycore components on the timing analyses is discussed in Section 3. Section 4 focuses on the inter-core interference analysis presented in D3.4 and its application on the BMP.

The contents of Section 2 are largely covered in MS2 deliverables D1.8 and D2.7. Therefore in this deliverable these are summarised with appropriate references provided to more details. The essence of this deliverable appears in Section 3.

2 Overview of the Platform and Simulator

The manycore platform is the one developed in the context of WP1 – first described in D1.2 and D1.5 (MS1), and then revised in D1.8 (MS2). For the sake of brevity, this section only provides an overview of the manycore platform, operating system and simulator. An extensive description of both the platform and the simulator are located in D1.8 (MS2). The operating system, ManyCOS, is developed in the context of WP2 and a more complete description is available in D2.7. This section also describes the tracing capabilities of the simulator, vital to the definition of timing analysis methods. Design choices from the perspective of the timing analyses are discussed in Section 3.

2.1 Architecture of the Manycore Platform

The simulator models a clustered manycore platform. Each cluster typically contains a local memory and 3 to 15 cores, although a different number of cores are possible. Each core features local first level data (DL1) and instruction (IL1) caches, with respective translation look-aside buffers (DTLB and ITLB). A L2 cache further supplements each cluster's local memory hierarchy. The L2 cache is shared among all cores in the cluster through an intra-cluster tree-based Network-on-Chip (NoC). Hence, the infrastructure of a cluster is akin to that of a multicore platform using a tree-based interconnect between the cores and the main memory system (including the L2).

A number of MBPTA-compliant policies have been implemented for the different layers of the local memory hierarchies. Caches, private or shared, and TLB use randomised placement and replacement policies. The shared L2 cache further supports partitions, exclusive to specific cores or shared among all of them. D1.8 demonstrates the superiority, for the NoC, of policies based on random permutations of the arbitration schedule [10], as opposed to bounded round-robin [14] or purely random (lottery) policies. Each L2 cache is connected to a memory controller which enforce the maximum arbitration latency during timing analyses, upper-bounding its deployed behaviour [14]. Other arbitration policies and designs will be investigated in the future.

The intra-cluster tree NoC can also prioritise specific cores, resulting in tighter timing estimates for the highest priority ones. There is however no guarantee on the bandwidth available to the lower priority requests. Heterogeneous bandwidth guarantees have been implemented as a trade-off between the tightness of timing estimates and the guarantees available to all cores.

Clusters are interconnected using a crossbar to allow for cluster-to-cluster communication from one local memory to the other. In practice, the platform features an inter-cluster, tree-based NoC per destination cluster. Each of those inter-cluster NoCs contains leaves for the other clusters. In a cluster, requests from the inter-cluster tree compete with the local core requests. This topology is illustrated in Figure 1. Additional leaves may be inserted to fit I/O devices. The same arbitration policies apply to both the inter- and the intra-cluster trees.

Overall, local memory requests, having to traverse only the intra-cluster NoC, are served much faster than remote ones. Requests to distant memories must, in order, go through (i) the intra-cluster NoC of the emitting cluster, (ii) the inter- and (iii)

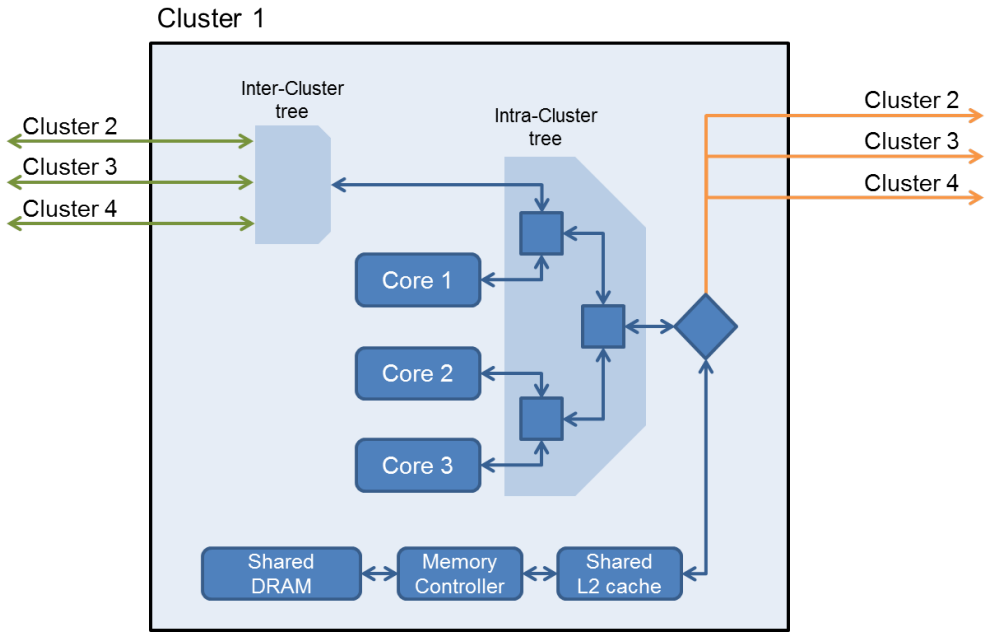


Figure 1: Example of a 3-core 4-cluster manycore connected with intra- and inter-cluster tree NoCs.

intra-cluster NoCs of the receiving cluster. If inter-cluster communication occurs at a high enough frequency, then the NoCs should be rearranged.

2.2 Tracing Capabilities

The manycore simulator interfaces with Rapita Verification Suite (RVS) to produce traces in the appropriate format; the simulator captures instrumentation points in the application and reports the cycles in which they have been observed. Neither the instrumentation nor the tracing interferes with the timing simulation. The binary is unchanged whether instrumentation is active or not, while tracing occurs at emulation level. RVS support is provided without affecting the timing of the simulated application.
In support of potential future Extended Path Coverage (EPC) methods (presented in D3.4 and [22]) to provide a single core timing analysis, the tracing process has been adapted, as described in D1.8, to report additional information for each instrumentation point in the form of the sequence of accessed memory addresses. The tracing is highly flexible and the numerous end-of-simulation statistics could be reported at a finer granularity, such as that of the iPoint. The availability of such information is of prime importance in the application and evaluation of MBPTA methods suited to the BMP. A non-exhaustive list of those statistics includes execution cycles, hits and misses in each level of the memory hierarchy, accesses to the NoC, stall cycles due to contention, etc..
Therefore, the simulator enables many ways of collecting a wide spectrum of information so that manycore-specific timing analyses, like the interference analysis proposed in D3.4, can be effectively investigated and released. The malleability of the simulator also allows for the exploration of alternate sources information, in line with the instrumentation facilities provided by concrete hardware platforms,

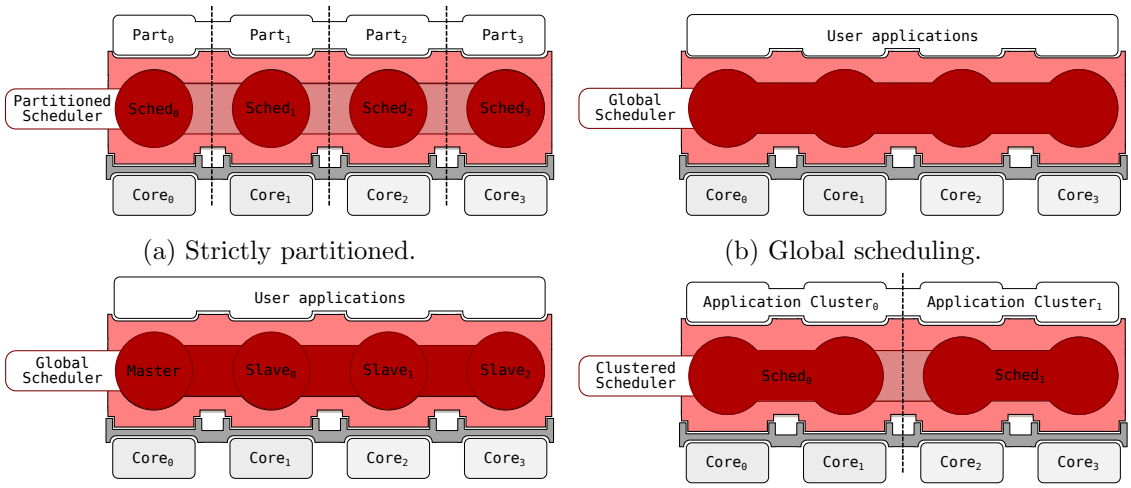
for the proposed analyses.

2.3 Architecture of the operating system ManyCOS

A custom RTOS, first described in D2.7, has been developed to support the execution of mixed-criticality applications on top of the PROXIMA manycore simulator. The definition of a PTA-compliant hardware architecture releases the software layer and RTOS from the need to guarantee that the whole system is amenable to PTA. The features implemented within the manycore RTOS aim at guaranteeing a better applicability of timing analyses in general and are intended to be equally convenient and useful in probabilistic and deterministic platforms.
The architectural design of the RTOS has been steered by two main principles: (i) support to mixed-criticality systems and (ii) time composability, as a fundamental enabler to incremental development and qualification. A mixed-criticality RTOS is required to support the execution of applications or functions characterized by different levels of criticality, as defined by the specific certification standards applied in the domain.
Hypervisor-based solutions do not scale smoothly to manycore platforms where support to mixed-criticality applications is provided with a combination of hardware partitioning (e.g., resorting to a clustered architecture) and bounded resource usage. With this respect the RTOS is expected to exploit the functionalities of the underlying hardware platform. Leveraging the clustered model of the manycore platform, various scheduling models with different degrees of resource sharing can be implemented, ranging from partitioned approaches to more flexible cluster-based schedulers. Following this observation, ManyCOS, the PROXIMA manycore RTOS, has been designed as a generic RTOS framework, equipped with a plug-in mechanism that allows ManyCOS to assume different personalities. Scheduler plug-ins can be defined to realize a score of different configurations. In the scope of PROXIMA we plan to implement a scheduler plug-in for the following configurations:

- *Strictly partitioned scheduling*: where the RTOS (and its scheduling structures) is replicated on each core;
- *Global scheduling*: where the RTOS is responsible for scheduling the application globally over the physical cluster;
- *Global scheduling with master core*: similar to the previous configuration but with a core acting as a master and detaining the scheduler logic;
- *Cluster scheduling*: where scheduling is organized around exploiting configurable logical clusters within the physical one.

The same configurations are depicted in Figure 2. The combination of highly configurable scheduling framework and HW-level mechanisms in the PROXIMA manycore platform allows users to choose among a wide range of configurations, depending on the applications to be deployed and their criticality level. As an example, a restrictive configuration would consist of deploying a high criticality application on a specific core within a cluster with strict partitioned scheduling and leave low criticality applications (soft real-time) to be scheduled globally on a separate cluster.



(a) Strictly partitioned.

(b) Global scheduling.

(c) Global scheduling with master core.

(d) Cluster scheduling.

Figure 2: Scheduling plug-in configurations.

Time Composability, as defined in deliverable D6.3, is a property that applies to individual system components, both hardware and software: it guarantees that the bound on the timing behaviour of a component, in response to an execution request, can be determined independently of its composition with other components. A time-composable RTOS is notably expected to transparently support the execution of the user applications without incurring additional sources of history or data dependences [3, 4]. The usefulness of a time-composable RTOS layer has been already evaluated in the scope of the PROARTIS project [15], in a single core setting with an ARINC-653 compliant RTOS [4]. Time composability in the RTOS layer was primarily realized by implementing constant-time kernel primitives, whose timing behaviour is independent from the SW state, and avoiding (or minimizing) any interference on the HW state that may affect the execution of the user applications. The same principles could be used in ManyCOS by leveraging on the various degree of segregation that can be guaranteed by the different instantiation of the RTOS framework, from completely partitioned to clustered systems.

3 Impact of platform design on the timing analyses

Current research on MBPTA has been focused on the multicore platform, and as such it is necessary to consider how it may be extended to the manycore platform. As discussed in D1.8, the primary difference between multicore and manycore is the presence of multiple inter-connected clusters. A simple argument can be constructed for the applicability of MBPTA to the BMP. Specifically, any delay caused by accesses to shared resources, be it interference or an arbitration delay, will have a probability distribution associated with when it occurs and for how long. Hence, any such latency is by definition, a random variable, and hence can be modelled more easily by MBPTA. Furthermore, the BMP has been designed to produce time-composable estimates, valid irrespective of the contention experienced by the analysed task, either through composable policies or the use of analysis-specific execution modes.
One important consideration is the manycore platform will suffer greater levels of interference from other tasks than the multicore case. Therefore additional data and effort will be required to provide a similar level of confidence in the validity of the composability features than when compared with multicore. However currently the effects of inter-task interference are not being considered. This will be addressed later in the project.
By introducing related work in the domain of timing analyses, this section supports the choices in the design of the BMP and discusses the applicability of MBPTA. This section follows the taxonomy established in D3.1 to distinguish between intra-cluster (§ 3.1) and inter-cluster resources (§ 3.2).

3.1 Intra-cluster resources

Shared resources on concurrent architectures are subject to inter-task interferences as a result of contention or shared state. Contention occurs as a single resource cannot accommodate for simultaneous requests from different cores. Shared state interferences are the result of concurrent modifications of the resource state, resulting in latencies unexpected when the task was running in isolation. Different analyses have been proposed to take concurrency into account in timing estimates [1]. These approaches focus on either deriving composable but pessimistic estimates independent of the behaviour of concurrent tasks, as is the case within the project, or taking into account requests emanating from contenders, i.e. tasks running simultaneously on other cores. Composable estimates are naturally more suited to mixed criticality systems (MCS) where tasks with different criticality requirements execute concurrently; composable estimates embed the required isolation of the performance of high criticality tasks from the behaviour low criticality ones.
The arbitration policy orders the requests to a shared resource, resulting in additional latencies for the delayed ones. This delay on time-triggered policies, such as the time-division multiple access policy (TDMA), depends on the moment a request arrives. Under event-triggered policies, e.g. round robin, the latency suffered by a request depends on the number of concurrent requests. As they are independent of pending requests, time-triggered schemes are deemed more predictable, whereas event-driven scheme offer a work conserving alternative suited for average case performance [8].

In either case, the analysis of a composable architecture can assume or enforce that each request suffers the maximum possible arbitration latency [14], i.e. all cores have a higher priority request or the access just missed its slot. The analysis of the offset in which a request arrives with regards to the arbitration slot allows for the derivation of less pessimistic estimates [2, 6] using time-driven policies. More proactive approaches [2] optimise the slot arrangement within the arbitration slot to reduce the overall latency suffered by the most critical tasks. Such approaches however require costly micro-architectural models to accurately capture the offsets of a request.

The BMP relies on a randomised event-driven policy for the arbitration of accesses between nodes of intra-cluster tree to the shared L2. The Random Permutation arbiter randomises the schedule of every arbitration slot resulting in a randomised arbitration latency. This produces an increased absolute worst-case latency, when a request is allocated the first window in the slot but misses it, and is scheduled last in the next slot, but this scenario only occurs with a low probability. The arbitration latency suffered by requests in the program is therefore randomised and fit for estimation using probabilistic analyses. To allow observations independent of the contenders in the system, the BMP supports a WCET analysis mode where all contenders besides the analysed one are assumed to always have pending requests. This shifts the observed arbitration latency towards its worst-case profile. The arbiter further allows the allocation of bandwidth to a high criticality task, in the form of additional dedicated windows in the arbitration slot, to reduce the probabilistic arbitration latency suffered by its requests. This does not impede the composability of the platform provided tasks are analysed with the same bandwidth guarantee as they are deployed with.

The L2 cache and the memory controller in the BMP can also be subject to shared state interferences. Modifications of the resource state by rival tasks can result in unexpected latencies for the analysed task. As an example, a rival task might evict from the cache a block useful to the analysed task. Similarly, a rival task can close the memory row currently used by the analysed task. Assumptions or observations made while the analysed task is isolated cannot in general be considered a representative of the final system. Different analyses techniques, mostly for shared caches [6, 9, 12], have been proposed to take into account the interference effects of contenders on the shared resource state. The more precise of approaches to timing analysis suffer from a high complexity, trying to compute possible interleavings of accesses of different tasks [12]. Abstractions on the ordering and quantity of cache accesses performed by other tasks, i.e. assuming all rivals' useful blocks are loaded in cache, result in more pessimistic estimates [6, 9] and call for supporting techniques such as bypass to reduce the pressure on the shared caches.

To achieve the computation of composable estimates, independent of the analysed or observed contenders, partitioning approaches instead rely on the allocation of a dedicated space to the benefit of the analysed task [7, 16, 21]. Analysis or observations can then be performed assuming the analysed task only accesses its interference-free segment of the resource. The trade-off is a higher worst-case execution time resulting from the reduced available space, but a tighter worst-case estimate. On the BMP, ways, columns of the L2 cache be dedicated to the sole benefit of a task. Representative observations can be made provided the partition allocation [18] is the same during analysis and on the deployed system. Arbitration

of requests to the same bank, a line of the L2 cache, is not required. The intra-cluster tree serialises requests to the fully pipelined L2 cache which, combined with the use of a dedicated port to fill the cache on memory reads, prevents access conflicts on the cache banks.
The order and type of memory requests previously served by the memory controller also impacts the execution latency of a memory request [5]. Much like caches, successive accesses to close memory locations can benefit from the locality in the shared row buffer. As discussed previously, inter-core interferences however result in unscheduled evictions in the row-buffer for the analysed task. The BMP memory controller in each cluster implements a *closed page* policy where the row buffer is emptied after a request has been served. This is a predictable alternative to the *open page* policy, and a cost-efficient solution compared to solutions duplicating private banks for each core [17]. Furthermore, enforcing at analysis time a fixed memory latency, an upper-bound of the latency experienced on the deployed system is trivial.

3.2 Inter-cluster resources

The inter-cluster NoC on the BMP relies upon the same mechanisms as the intra-cluster one. Requests to a specific cluster from the outside have to go through its inter-cluster tree. Arbitration of requests within the tree, from the bottom to the top, uses the same principles and offers the same benefits as the intra-cluster NoC. This results however in a non-uniformed memory architecture, as requests to the local memory and requests to distant memories have different execution time profiles. Observations should therefore be made using a similar configuration of task-to-cluster mapping. This is far less restrictive than networked many-core architectures [20], where the end-to-end communication delays depend on many factors such the mapping of tasks to core, the routing, interfering communication channels, etc..

4 Inter-core Interference analysis

The manycore platform has been designed to provide for the computation of time-composable estimates through a combination of isolation mechanisms and analysis modes tailored to enforce the worst inter-core interference scenarios. Isolation of the state space in resources shared by concurrent tasks, even partial [11], is a strong requirement to the provision of guarantees on the performance of high criticality tasks. Enforcing the worst-case occurrences of conflicts however results in pessimistic estimates, especially regarding the load of inter-cluster requests on the system.
The inter-core interference analysis, presented in depth in D3.4, builds a multi-variate interference model that relates observed interferences to their impact on the execution time of the Unit of Analysis. The model effectively derives a safety margin specific to the contention observed on the inter-core shared resources. This margin is applied as a multiplicative factor on top of the execution time derived by an interference-free analysis, such as the PROXIMA timing analysis presented in D3.4. The inter-core interference analysis is a multi-step process hence summarised:

- Step 0. Build an instrumentation framework to capture an overview of the system's behaviour at runtime, through performance metrics beyond timings.
- Step 1. Select using Principal Components Analysis the factors most relevant to the impact of interferences on the Unit of Analysis.
- Step 2. Build an ensemble of interference models, gathered in an ensemble to forecast the impact of observed interference on the Unit of Analysis.
- Step 3. Derive a safety margin for an observed inter-core interference scenario.

Step 0 and step 2, the construction of the instrumentation framework and inter-core interference model, are tied to the platform under analysis. The instrumentation relies on the platform monitoring infrastructure, e.g. performance monitoring counters. As discussed in Section 2, tracing on the simulator is extremely flexible and offers the collection of a large variety of statistics without incurring any additional cost on the Unit of Analysis. The interference model on the other hand should capture the setvs specific to the platform, deploying families of forecasting models appropriate to the setvs' properties.
The inter-core interference analysis can be applied at different levels on the BMP, either encompassing both intra- and inter-cluster resources in a single model, or deriving distinct models for each. The latter scenario allows the use of different forecasting models per family of resources. The safety margins computed for the intra- and inter-cluster interferences should be orthogonal and could be applied on top of each other.
The results of the inter-core interference analysis depend on the contention observed during Step 3, the final step of the analysis. The model can be fed a worst-case inducing interference scenario, but that would lead to the same pessimism as the use of the worst-case analysis mode. Changes in the contention suffered by the Unit of Analysis, unless they can be demonstrated to result in less interferences, require the derivation of a new safety margin. Only the final step of the analysis should

be reapplied; the interference model remains the same as long as the analysed task (or relevant parameters such as allocated partition in the cache) does not change. This is an additional argument for the separation of the intra- and inter-cluster interference models to enable the independent evaluation of the impact of changes on contention at different levels.
Development of the inter-core interference analysis, as discussed in D3.4, focused on the analysis of the Aurix/ERIKA COTS multi-core platform where mitigation strategies for inter-core interferences are limited. The BMP on the other hand allows by construction the computation of composable timing estimates. The deployment of the interference analysis on the BMP, subject to refinements of the instrumentation framework, still offers numerous benefits. The analysis indeed allows an evaluation and comparison of the composability, impact on interferences, and compliance to analysis of different architectural configurations. This also includes building an understanding of how timings within a cluster are impacted when inter-cluster interferences are considered. The evaluation of the manycore will provide guidelines for the design and analysis of manycore architectures. Such guidelines are an important requirement to establish with industrial partners a roadmap for the deployment of manycore platforms.

5 Conclusions

In this deliverable, the platform and associated simulator, the timing analysis and controlled experiments to establish validity, and a usage guide is presented. The current approach uses the EVT analysis provided by INRIA. However , in future work, the EPC work from UPD may be extended to deal with the BFP many-core platform. The following is a review of how well this deliverable has met the requirements from deliverable D6.3. The text in *italics* is quoted text from D6.3.

1. Support for WP1 - *Limitations in the capabilities of the hardware platforms of interest, in terms of tracing, debug and performance monitoring support, I/O mechanisms, PTA-compliance and interfacing* - Achieved

2. WP3 Requirements - *Analysis prototypes and tools* - Achieved

3. Support for WP4 - *Abstract model of the case studies characterising the applications, interactions, complexity and time behavior. Tool-chain requirements on instrumentation and timing analysis* - Achieved except for the abstract model of the use case. The capability has been created and is presented here, however the actual use case has not yet been analysed.

4. MS2 Objectives - *The experience gained until MS2 will also be used to build preliminary certification arguments and a description of safety techniques/measures applicable to some of the industrial domains of interest* - Partly achieved. The controlled experiment helps establish evidence for some of the important claims and assumptions that are necessary for a PTA compliant approach. However the current safety argument from D2.8 does not as yet provide detailed claims and assumptions for the specific timing analyses that will be used. As this argument is developed further, then the controlled experiments may need to be extended to provide the necessary evidence. It should also be shown that the results are equally applicable to the PROXIMA platforms, however there is no reason to believe they would not be.

5. From Manycore Simulator Success Criteria *at m18 the simulator will be ready to start the porting of the timing analysis tool, the RTOS and the associated tool chain* - Achieved.

Acronyms and Abbreviations

- BMP: Bespoke Simulator Platform.
- COTS: Commercial Off-The-Shelf.
- EPC: Extended Path Coverage
- EVT: Extreme Value Theory.
- ETP: Execution time profile.
- HWM: High WaterMark
- MBPTA: Measurement-Based Probabilistic Timing Analysis.
- MCS: Mixed Criticality System
- NoC: Network on Chip.
- OS: Operating System
- PTA: Probabilistic Timing Analysis.
- pWCET: Probabilistic Worst-Case Execution Time.
- RVS: Rapita Verification Suite.
- setv: Sources of Execution Time Variability.
- VICI: Variable Inter-Core Interferences.
- WCET: Worst-Case Execution Time.

References

[1] Andreas Abel, Florian Benz, Johannes Doerfert, Barbara Drr, Sebastian Hahn, Florian Haupenthal, Michael Jacobs, AmirH. Moin, Jan Reineke, Bernhard Schommer, and Reinhard Wilhelm. Impact of resource sharing on performance and performance prediction: A survey. In *CONCUR 2013 Concurrency Theory*, volume 8052 of *Lecture Notes in Computer Science*, pages 25–43. 2013.

[2] Alexandru Andrei, Petru Eles, Zebo Peng, and Jakob Rosen. Predictable implementation of real-time applications on multiprocessor systems-on-chip. In *VLSID*, USA, 2008.

[3] A Baldovin, E Mezzetti, and T Vardanega. A Time-composable Operating System. In *Proceedings of the 12th International Workshop on Worst-Case Execution Time Analysis*, 2012.

[4] Andrea Baldovin, Enrico Mezzetti, and Tullio Vardanega. Towards a time-composable operating system. In *Proceedings of the 18th International Conference on Reliable Software Technologies - Ada-Europe 2013*, pages 143–160, 2013.

[5] Roman Bourgade, Clément Ballabriga, Hugues Cassé, Christine Rochange, and Pascal Sainrat. Accurate analysis of memory latencies for WCET estimation. In *16th International Conference on Real-Time and Network Systems (RTNS)*, October 2008.

[6] Sudipta Chattopadhyay, Abhik Roychoudhury, and Tulika Mitra. Modeling shared cache and bus in multi-cores for timing analysis. In *Proceedings of the 13th International Workshop on Software & Compilers for Embedded Systems*, SCOPES '10, pages 6:1–6:10, New York, NY, USA, 2010. ACM.

[7] D. Chiou, P. Jain, S. Devadas, and L. Rudolph. Dynamic cache partitioning via columnization. In *DAC*, Los Angeles, CA, USA, 2000.

[8] Christoph Cullmann, Christian Ferdinand1, Gernot Gebhard, Daniel Grund, Claire Maiza, Jan Reineke, Beno?t Triquet, and Reinhard Wilhelm. Predictability considerations in the design of multi-core embedded systems. In *ERTS*, 2010.

[9] Damien Hardy, Thomas Piquet, and Isabelle Puaut. Using bypass to tighten wcet estimates for multi-core processors with shared instruction caches. In *Proceedings of the 2009 30th IEEE Real-Time Systems Symposium*, RTSS '09, pages 68–77, 2009.

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

[11] Benjamin Lesage, Isabelle Puaut, and André Seznec. PRETI: Partitioned Real-time Shared Cache for Mixed-criticality Real-time Systems. In *Proceedings of the 20th International Conference on Real-Time and Network Systems*, RTNS '12, pages 171–180, New York, NY, USA, 2012. ACM.

[12] Mingsong Lv, Wang Yi, Nan Guan, and Ge Yu. Combining abstract interpretation with model checking for timing analysis of multicore software. In *Real-Time Systems Symposium (RTSS), 2010 IEEE 31st*, pages 339–349, Nov 2010.

[13] B. Nikolic, P. Meumeu Yomsi, and S.M. Petters. Worst-case memory traffic analysis for many-cores using a limited migrative model. In *19th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2013.

[14] M. Paolieri, E. Quinones, F.J. Cazorla, G. Bernat, and M. Valero. Hardware support for WCET analysis of hard real-time multicore systems. In *ISCA*, 2009.

[15] PROARTIS. Probabilistically analyzable real-time systems. feb 2010. http://www.proartis-project.eu/.

[16] Jan Reineke, Isaac Liu, Hiren D. Patel, Sungjun Kim, and Edward A. Lee. Pret dram controller: Bank privatization for predictability and temporal isolation. In *Proceedings of the Seventh IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, CODES+ISSS '11, pages 99–108, 2011.

[17] Jan Reineke, Isaac Liu, Hiren D Patel, Sungjun Kim, and Edward A Lee. PRET DRAM controller: Bank privatization for predictability and temporal isolation. In *Proceedings of the seventh IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, pages 99–108. ACM, 2011.

[18] J. E. Sasinowski and J. K. Strosnider. A dynamic programming algorithm for cache memory partitioning for real-time systems. *IEEE Trans. Comput.*, 42(8):997–1001, August 1993.

[19] M. Schoeberl, F. Brandner, J. Sparso, and E. Kasapaki. A statically scheduled time-division-multiplexed network-on-chip for real-time systems. In *NOCS*, 2012.

[20] Zheng Shi and A. Burns. Real-time communication analysis for on-chip networks with wormhole switching. In *Networks-on-Chip, 2008. NoCS 2008. Second ACM/IEEE International Symposium on*, pages 161–170, April 2008.

[21] Vivy Suhendra and Tulika Mitra. Exploring locking & partitioning for predictable shared caches on multi-cores. In *Proceedings of the 45th Annual Design Automation Conference*, DAC '08, pages 300–303. ACM, 2008.

[22] M. Ziccardi, E. Mezzetti, T. Vardanega, J. Abella, and F. Cazorla. Epc: Extended path coverage for measurement-based probabilistic timing analysis. In *RTSS (to appear)*, 2015.