T-CREST

TIME-PREDICTABLE MULTI-CORE ARCHITECTURE
FOR EMBEDDED SYSTEMS

Project Number 288008

D 2.5 Report on Architecture Evaluation and WCET Analysis

Version 1.0
25 September 2013
Final

Public Distribution

AbsInt Angewandte Informatik, Technical University of Denmark

Project Partners: **AbsInt Angewandte Informatik**, **Eindhoven University of Technology**, **GMVIS Skysoft**, **Intecs**, **Technical University of Denmark**, **The Open Group**, **University of York**, **Vienna University of Technology**

Project Partner Contact Information

AbsInt Angewandte Informatik Christian Ferdinand Science Park 1 66123 Saarbrücken, Germany Tel: +49 681 383600 Fax: +49 681 3836020 E-mail: ferdinand@absint.com	**Eindhoven University of Technology** Kees Goossens Potentiaal PT 9.34 Den Dolech 2 5612 AZ Eindhoven, The Netherlands E-mail: k.g.w.goossens@tue.nl
GMVIS Skysoft João Baptista Av. D. Joao II, Torre Fernao Magalhaes, 7 1998-025 Lisbon, Portugal Tel: +351 21 382 9366 E-mail: joao.baptista@gmv.com	**Intecs** Silvia Mazzini Via Forti trav. A5 Ospedaletto 56121 Pisa, Italy Tel: +39 050 965 7513 E-mail: silvia.mazzini@intecs.it
Technical University of Denmark Martin Schoeberl Richard Petersens Plads 2800 Lyngby, Denmark Tel: +45 45 25 37 43 Fax: +45 45 93 00 74 E-mail: masca@imm.dtu.dk	**The Open Group** Scott Hansen Avenue du Parc de Woluwe 56 1160 Brussels, Belgium Tel: +32 2 675 1136 Fax: +32 2 675 7721 E-mail: s.hansen@opengroup.org
University of York Neil Audsley Deramore Lane York YO10 5GH, United Kingdom Tel: +44 1904 325 500 E-mail: Neil.Audsley@cs.york.ac.uk	**Vienna University of Technology** Peter Puschner Treitlstrasse 3 1040 Vienna, Austria Tel: +43 1 58801 18227 Fax: +43 1 58801 918227 E-mail: peter@vmars.tuwien.ac.at

Contents

Document Control

Version	Status	Date
0.1	First draft	10 September 2013
0.5	Revised draft	23 September 2013
1.0	Final Version	25 September 2013

Executive Summary

This document describes the deliverable *D 2.5 Report on Architecture Evaluation and WCET Analysis* of work package 2 of the T-CREST project. In this deliverable we explore the impact of different cache architectures and hardware parameters on the guaranteed worst-case performance.

1 Introduction

Current processors are optimized for average case performance, often leading to a high worst-case execution time (WCET). Many architectural features that increase the average case performance are hard to be modeled for the WCET analysis. In this deliverable we consider Patmos, a processor optimized for low WCET bounds rather than high average case performance. Patmos is a dual-issue, statically scheduled RISC processor. The instruction cache is organized as a method cache and the data cache is organized as a split cache in order to simplify the cache WCET analysis. To fill the dual-issue pipeline with enough useful instructions, Patmos relies on a customized compiler, which is developed in the context of work package 5. The compiler also plays a central role in optimizing the application for the WCET instead of average case performance.

Real-time systems need a time-predictable execution platform so that the worst-case execution time (WCET) can be estimated statically. It has been argued that we have to rethink computer architecture for real-time systems instead of trying to catch up with new processors in the WCET analysis tools [9, 3, 11].

Patmos is a 32-bit, RISC-style microprocessor optimized for time-predictable execution of real-time applications [10]. In order to provide high performance for single-threaded code, a two-way parallel VLIW architecture was chosen. For multi-threaded code, the T-CREST chip-multiprocessor system, with statically scheduled accesses to shared resources, will be explored for time-predictable performance enhancements.

The main features of Patmos are:

- Dual-issue pipeline
- Full predication
- Support for a specialized method cache [6] and stack cache [1]
- Split caches with typed load/store instructions
- Split load

The processor and its software environment is intended as a platform to explore various time-predictable design trade-offs and their interaction with WCET analysis techniques as well as WCET-aware compilation. In this deliverable we investigate different hardware configurations of the Patmos processor to understand their influence on the computation of worst-case execution time bounds. Within T-CREST, the observation made are intended to steer the design of Patmos processor features.

1.1 Related Work

Thiele and Wilhelm argue that a new research discipline is needed for time-predictable embedded systems [11]. Berg et al. identify the following design principles for a time-predictable processor: "... recoverability from information loss in the analysis, minimal variation of the instruction timing, non-interference between processor components, deterministic processor behavior, and comprehensive documentation" [2]. The authors propose a processor architecture that meets these design principles. The processor is a classic five-stage RISC pipeline with minimal changes to the instruction set. Suggestions for future architectures of memory hierarchies are given in [13].

Heckmann et al. provide examples of problematic processor features in [5]. The most problematic features found are the replacement strategies for set-associative caches. In conclusion Heckmann et al. suggest the following restrictions for time-predictable processors: (1) separate data and instruction caches; (2) locally deterministic update strategies for caches; (3) static branch prediction; and (4) limited out-of-order execution. The authors argue for restriction of processor features. In contrast, we also provide additional features for a time-predictable processor.

Multi-Core Execution of Hard Real-Time Applications Supporting Analyzability (MERASA) [12] is a European Union project that aims for multi-core processor designs in hard real-time embedded systems. An in-order superscalar processor is adapted for chip multi-threading (CarCore) [7]. The resulting CarCore is a two-way, five-stage pipeline with separated address and data paths. This architecture allows issuing an address and an integer instruction within one cycle, even if they are data-dependent. CarCore supports a single hard real-time thread to be executed with several non-real-time threads running concurrently in the background.

2 Architectural Exploration

In this section we investigate two different hardware configurations and compare their impact on static worst-case execution time (WCET) bound computation. Instead of performing measurements we are interested in worst-case behavior. Thus we use the static WCET analyzer aiT to explore the benefits of each design. A detailed description of aiT can be found in deliverable *D6.1 Design of Adaptations and Extensions of Code-Level WCET Analysis*.

aiT for PATMOS is fully parameterizable for the following among others:

- Clock frequency
- Cache replacement policy (*e.g.,* , LRU, FIFO)
- Cache size
- Cache line size
- Cache associativity
- Cache-locked memory regions
- Memory access timing
- Memory burst length
- Memory port width

As discussed in D6.1 we have extended the static WCET analyzer aiT to support the novel hardware features of the Patmos architecture, *i.e.,* the various special caches (method cache and stack cache) and traditional caches.

To provide meaningful numbers we investigate the worst-case behavior of real-word application code from GMV. For this purpose we have selected several tasks from the *Airline Operational Centre* (AOC) control application, from the `Crew Alert System` (CAS), and from the `Control Application` (IOP). AOC is the on-board component of an air traffic and trajectory management system. The application was developed according to DO-178B. The source code comprises about 30 kloc and can thus be considered complex embedded application. CAS is a typical highly critical on-board application found on civil aircraft of all major manufacturers. The GMV implementation is

based on real industrial requirements and is used as a study prototype by GMV customers. IOP is a typical closed-loop control application that was developed for training and demonstration purposes.

From the AOC application we have selected the the following tasks:

- `AGP_ClientMainLoop`,
- `Airplane_PAOCMainTask`,
- `AOC_decoderLoop`,
- `AOC_feederLoop`,
- `AOC_replierLoop`, and
- `Pilot_MDCUMainLoop`.

From the CAS application we have chosen the `CAS_loop` tasks. From the IOP application we have chosen the `IOP_grbc_manager` task. In addition we a added the task `dry2_1_main` which is a common benchmark program.

By means of this architectural exploration we want to identify a promising hardware configuration that allows for a good worst-case performance. Section 2.1 discussed the hardware configuration used for architectural exploration. Section 2.3 shows the guaranteed performance comparing an I-Cache implementation with an M-Cache implementation for a single Patmos NOC. Section 2.4 shows the guaranteed performance comparing the stack cache against a traditional D-Cache for a single Patmos NOC. The full exploration results can be found in Section 4. Section 2.5 discussed the results.

2.1 Hardware Configuration

On the one hand we employ a traditional instruction cache (I-Cache) of 4 KB size with 4 ways updated according to the LRU replacement policy. The number of bytes per cache line are configured to either 16, 32, 64, or 128 bytes. On the other hand we use a method cache (M-Cache) of 4 KB size with either 4 or 8 ways.

As opposed to the traditional I-Cache design, the M-Cache caches a whole routine in a single cache line [6]. For the architecture exploration we investigate the impact on computed WCET bounds using either FIFO or LRU replacement. The principles of the used cache analyses are described in [4, 8]. More details can be found in deliverable *D6.1 Design of Adaptations and Extensions of Code-Level WCET Analysis*.

For the architectural exploration we use a stack cache of size of either 128, 256, 512, or 1024 bytes. In addition, we investigate the guaranteed performance of a Patmos implementation that does not support the stack cache and that uses a data cache only.

We have refined the analysis of the Patmos stack cache described in deliverable *D6.1* as follows.

Stack Cache Analysis Domain. Let $S_{sc} \in \mathbb{N}$ be the stack cache size. The abstract domain $\mathcal{D}_{sc}$ of the stack cache analysis is the set of tuples $\{ (a, b) \mid a, b \in \mathbb{N} \wedge a \leq b \wedge b \leq S_{sc} \}$. For $(a, b) \in \mathcal{D}_{sc}$ the value a denotes the *minimum* stack cache fill count and b denotes the *maximum* stack cache fill count.

Stack Cache Analysis Update and Join Functions. For the stack cache analysis we define the abstract update function $U_{sc} : \mathcal{D}_{sc} \times instruction \rightarrow \mathcal{D}_{sc}$ and join function $J_{sc} : \mathcal{D}_{sc} \times \mathcal{D}_{sc} \rightarrow \mathcal{D}_{sc}$ as follows:

$$U_{sc}((a,b), instruction) := \begin{cases} (min(a+d, S_{sc}), min(b+d, S_{sc})) \\ \quad \text{if } instruction == \text{sres } d \\ (a' = min(a,d), max(a',b)) \\ \quad \text{if } instruction == \text{sens } d \\ (max(0, a-d), max(0, b-d)) \\ \quad \text{if } instruction == \text{sfree } d \\ (a,b) \\ \quad \text{otherwise} \end{cases}$$

$$J_{sc}((a,b),(a',b')) = (min(a,a'), max(b,b'))$$

2.2 The Memory Subsystem

We assume a network of Patmos cores where a single access to external memory is costly, but subsequent access beats of a burst access are comparably cheap and are delivered in a back-to-back manner. Furthermore we assume that per access beat 32 bit of data are being delivered to a requesting core (*i.e.,* , the memory port width is 32 bits). Here, we consider a network comprising either 1, 16 or 64 processor cores.

Both NOC and each Patmos core is assumed to run at a frequency of 450 MHz. Additionally we assume a memory that runs at a frequency of 150 MHz. The NOC to memory clock ratio is 3:1.

In the plenary project meeting in Copenhagen from 16th to 17th of September the partners UoY, DTU and TU/e have identified the following preliminary access timing parameters. T_{rd} denotes the memory tree traversal time for read accesses, T_{wr} denotes the traversal time for write accesses. The number of processors is identified by N.

$$T_{rd} = 3 + 3 * log_2(N) \tag{1}$$

$$T_{wr} = 2 + 2 * log_2(N) \tag{2}$$

The number of wait states per read access t_{rd} and the number of wait states per write access t_{wr} for a single-beat DRAM accesses are determined by the following formulas according to deliverable *D4.4 Dynamic Memory Controller Design and Implementation*.

$$t_{rd} = T_{rd} + 3 * (24 + 14 * (N-1)) \tag{3}$$

$$t_{wr} = T_{wr} + 3 * (1 + 14 * (N - 1)) \quad (4)$$

Based in the given formulas, we assume the following memory access timing for single beat accesses:

	Processors		
Access Timing	1	16	64
t_{rd}	75	717	2739
t_{wr}	5	643	2663

Given a network of 16 Patmos cores, a cache line fill request demanding 32 bytes from main memory in 8 beats of 4 byte each then takes $717 + 24 = 741$ cycles for the whole request to complete. We have to add 3 cycles per access beat due to the 3:1 NOC to bus clock ratio.

The maximum number of subsequent access beats following the first one is determined by the *burst length*. In combination with the port width, the burst length limits the maximum number of bytes that can be transferred per request. For our experiments we configured the burst length to match the instruction (if existent) and data cache line sizes. To fill a 128 byte cache line, the a burst length of 32 is required.

We consider a write-through data cache of 4 KB size with 4 ways updated using the LRU replacement policy. In the experiments we investigated data cache line sizes of either 16, 32, 64 or 128 bytes.

In total we examined 90 different hardware configurations per task to investigate the possible guaranteed worst-case behavior.

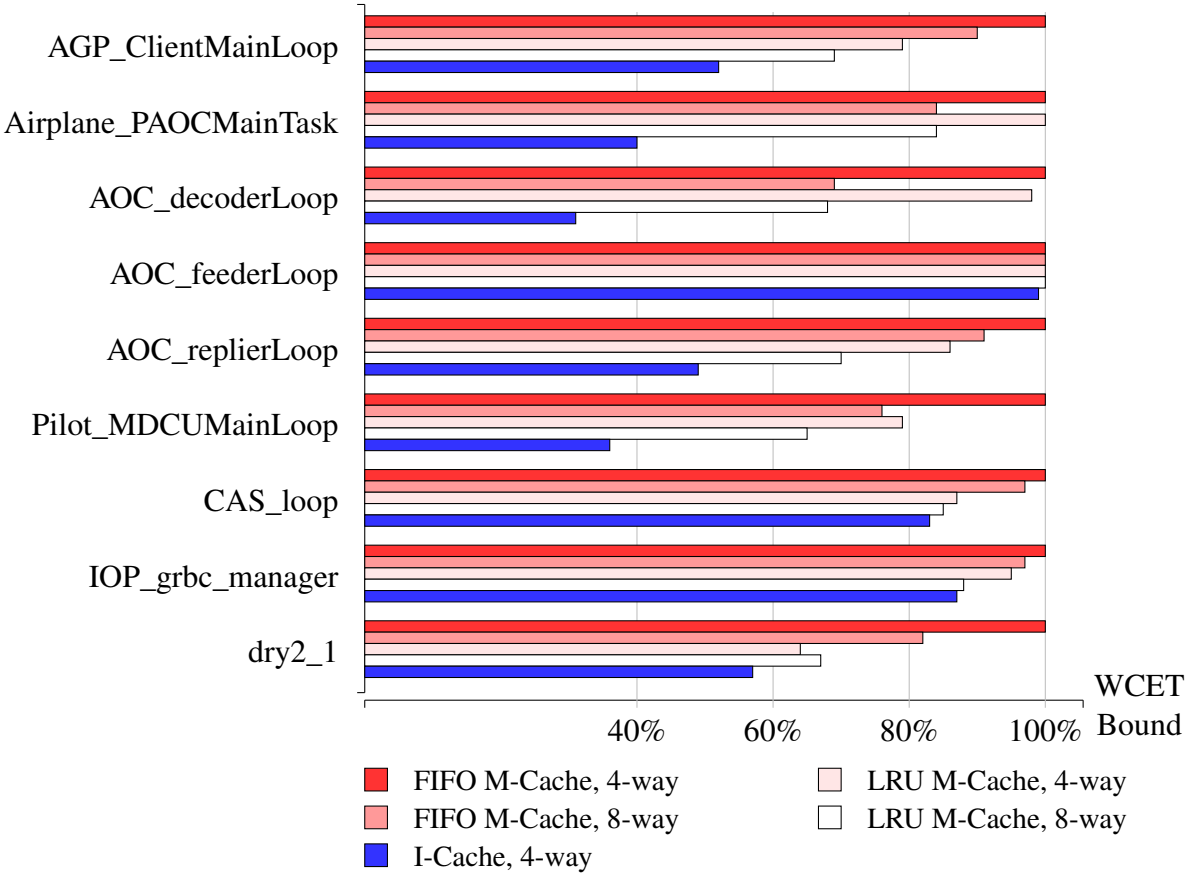
2.3 I-Cache vs. M-Cache Result Excerpt

2.3.1 Nodes: 1, Burst Length: 4, Line-Size: 16 bytes

The table shows the computed WCET bound in processor cycles per test case and configuration.

Task	FIFO M-Cache		LRU M-Cache		I-Cache
	4-way	*8-way*	*4-way*	*8-way*	*4-way*
AGP_ClientMainLoop	399133	359584	316930	277381	210416
Airplane_PAOCMainTask	500606	424486	500606	424486	202469
AOC_decoderLoop	358873536	249198810	353448825	246777611	113170696
AOC_feederLoop	28764	28764	28764	28764	28744
AOC_replierLoop	405811	369995	350040	285877	199508
Pilot_MDCUMainLoop	415855071	316878230	332254763	271126051	149934985
CAS_loop	115041	111602	100101	98845	95914
IOP_grbc_manager	37555548	36687737	35698715	33257004	32785036
dry2_1	29755	24498	19098	20036	17103

The figure shows depicts the above table as bar chart. The results are normalized per test case to the hardware configuration that caused the largest WCET bound.

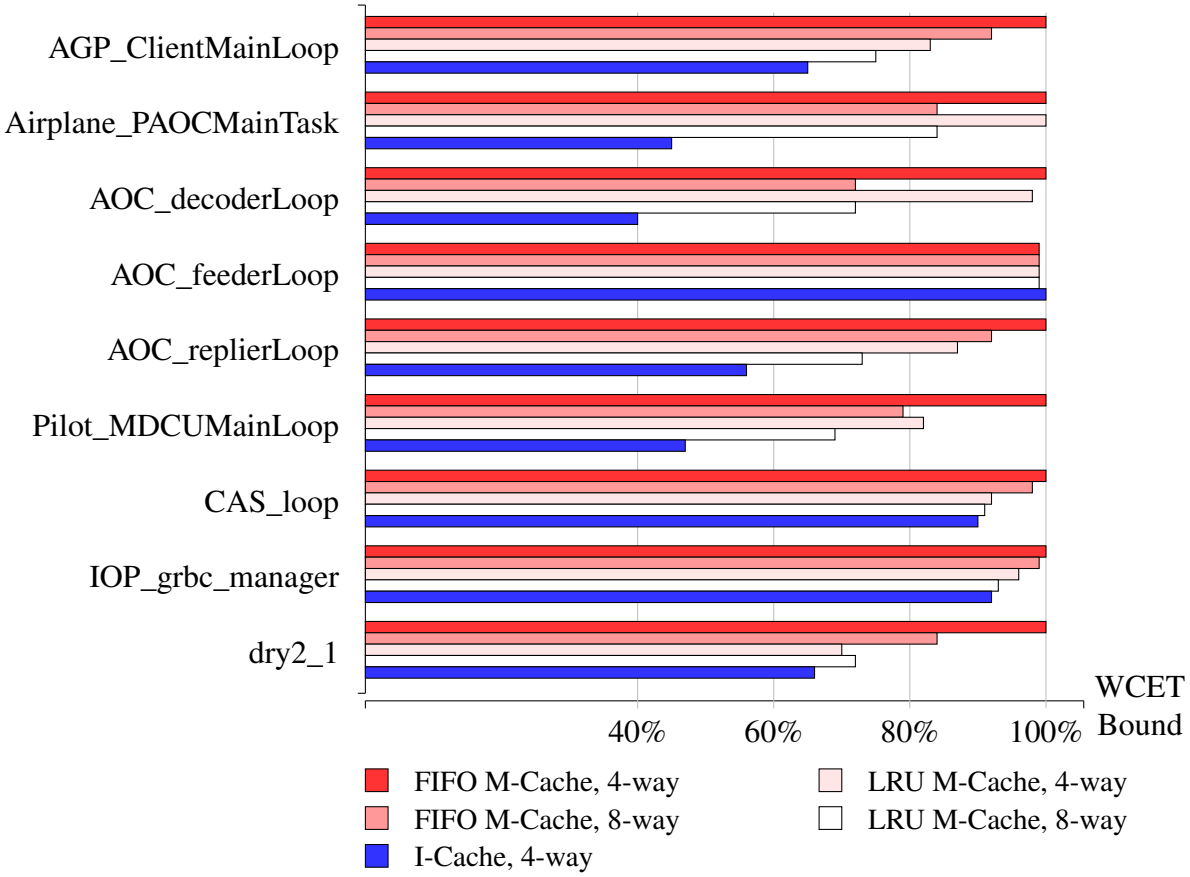


2.3.2 Nodes: 1, Burst Length: 8, Line-Size: 32 bytes

The table shows the computed WCET bound in processor cycles per test case and configuration.

Task	FIFO M-Cache		LRU M-Cache		I-Cache
	4-way	*8-way*	*4-way*	*8-way*	*4-way*
AGP_ClientMainLoop	282827	260298	234812	212283	184073
Airplane_PAOCMainTask	307756	260940	307756	260940	139886
AOC_decoderLoop	226308768	164515840	223178057	163123537	90668275
AOC_feederLoop	29982	29982	29982	29982	29991
AOC_replierLoop	260043	240507	227138	191465	145648
Pilot_MDCUMainLoop	275381589	218291930	227400293	192170175	130792176
CAS_loop	111585	109478	102935	102197	101090
IOP_grbc_manager	34440570	34123147	33354817	32062302	31962636
dry2_1	20911	17764	14842	15226	13949

The figure shows depicts the above table as bar chart. The results are normalized per test case to the hardware configuration that caused the largest WCET bound.

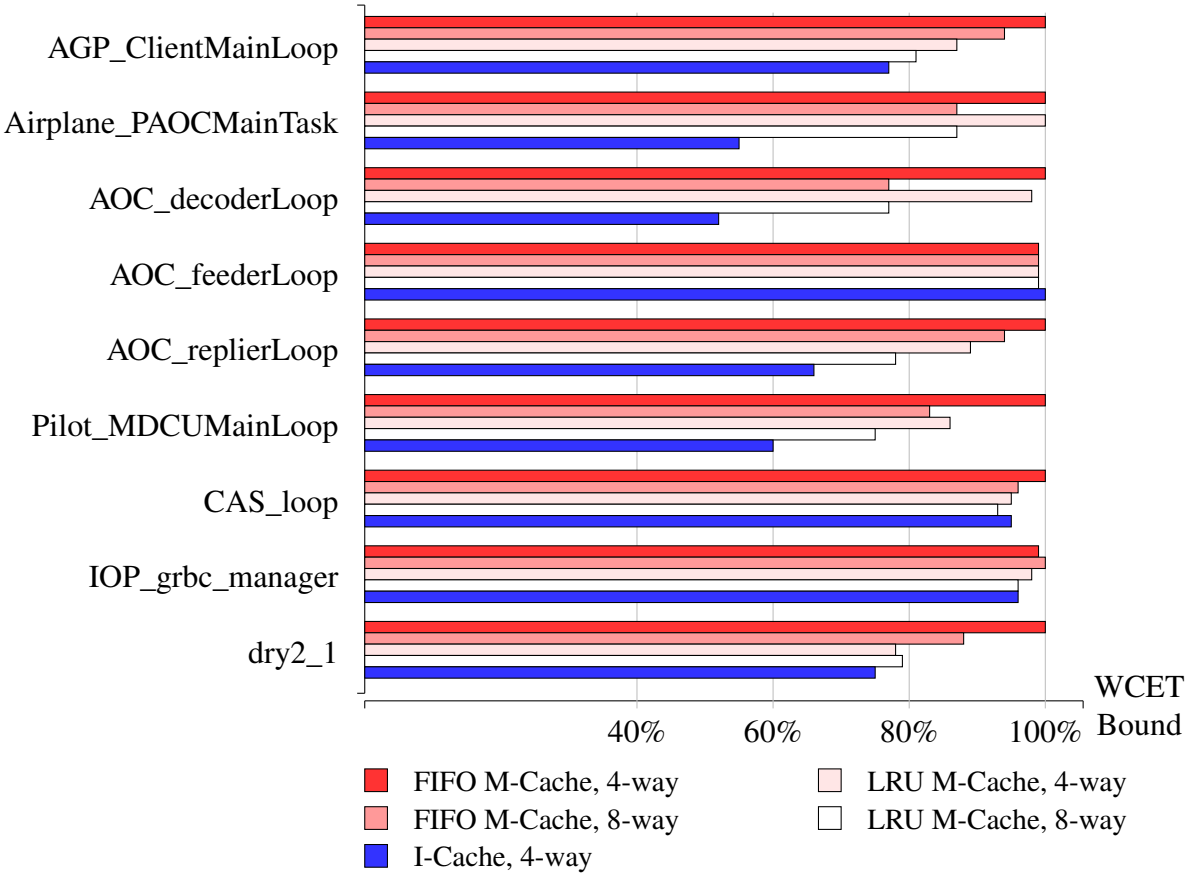


2.3.3 Nodes: 1, Burst Length: 16, Line-Size: 64 bytes

The table shows the computed WCET bound in processor cycles per test case and configuration.

Task	FIFO M-Cache		LRU M-Cache		I-Cache
	4-way	*8-way*	*4-way*	*8-way*	*4-way*
AGP_ClientMainLoop	239777	226572	208856	195651	185486
Airplane_PAOCMainTask	219108	191828	219108	191828	122292
AOC_decoderLoop	160220956	124314116	158264107	123416725	83415523
AOC_feederLoop	35420	35420	35420	35420	35457
AOC_replierLoop	195235	183839	174170	152335	130161
Pilot_MDCUMainLoop	215009765	179335706	184929957	162920901	130130423
CAS_loop	124605	120628	119137	116011	118610
IOP_grbc_manager	37099774	37149539	36435155	35768706	35692342
dry2_1	17683	15722	13834	14072	13311

The figure shows depicts the above table as bar chart. The results are normalized per test case to the hardware configuration that caused the largest WCET bound.

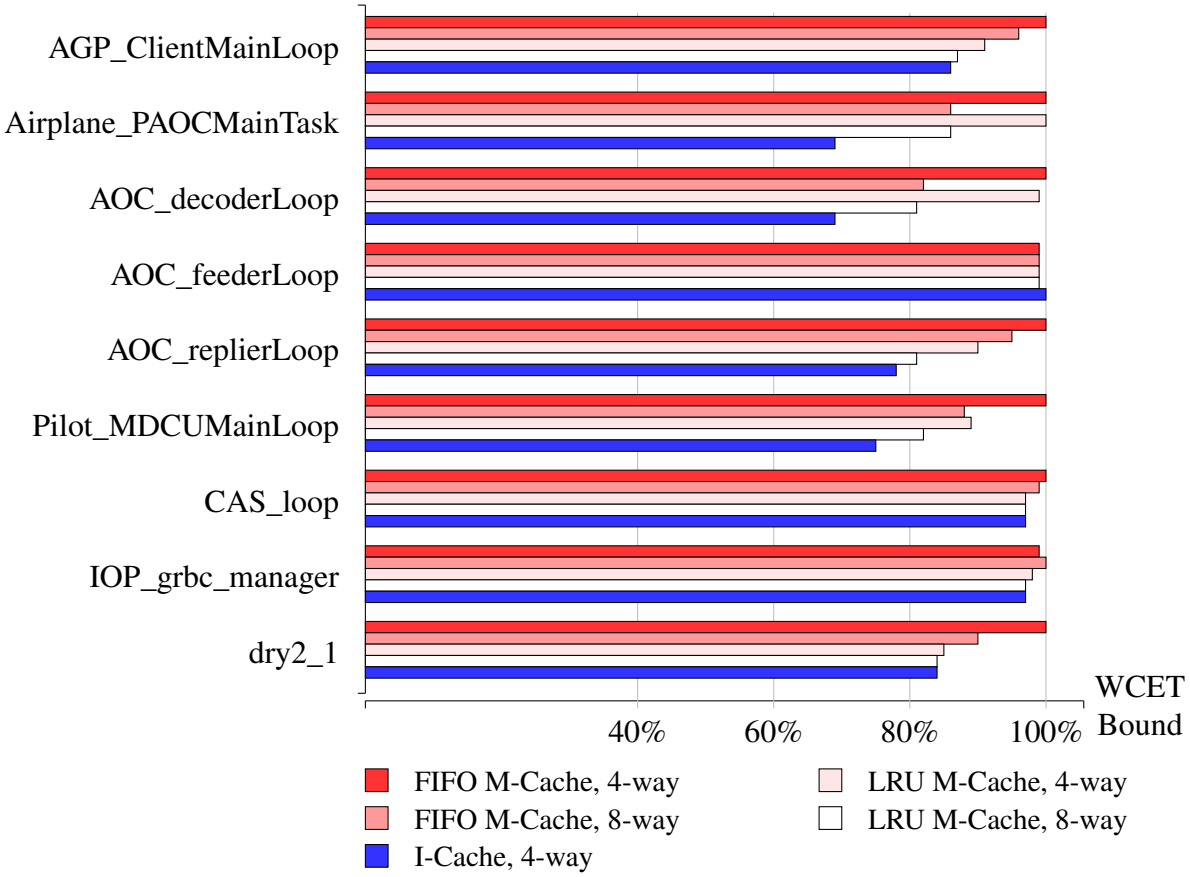


2.3.4 Nodes: 1, Burst Length: 32, Line-Size: 128 bytes

The table shows the computed WCET bound in processor cycles per test case and configuration.

Task	FIFO M-Cache		LRU M-Cache		I-Cache
	4-way	*8-way*	*4-way*	*8-way*	*4-way*
AGP_ClientMainLoop	249221	239864	228068	218711	214672
Airplane_PAOCMainTask	183892	159868	183892	159868	128345
AOC_decoderLoop	142928568	117563842	141549659	116945653	99029636
AOC_feederLoop	46814	46814	46814	46814	47024
AOC_replierLoop	164033	155893	149258	133935	128237
Pilot_MDCUMainLoop	206372541	182827600	185355809	170814907	156206926
CAS_loop	155135	154360	151295	151001	151194
IOP_grbc_manager	47238306	47330283	46784513	46374505	46058747
dry2_1	18461	16778	15722	15572	15570

The figure shows depicts the above table as bar chart. The results are normalized per test case to the hardware configuration that caused the largest WCET bound.



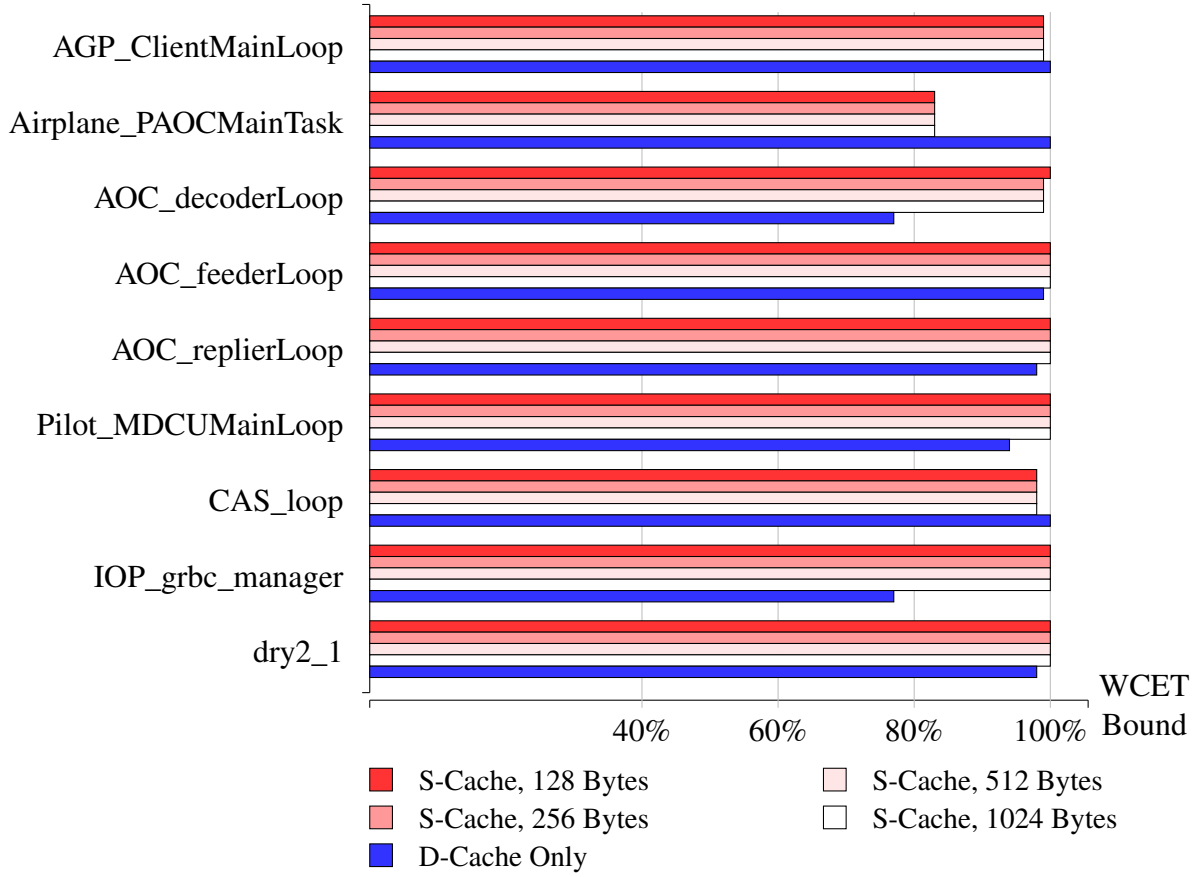
2.4 D-Cache vs. S-Cache Results Excerpt

2.4.1 Nodes: 1, Burst Length: 32, Line-Size: 128 bytes, I-Cache

The table shows the computed WCET bound in processor cycles per test case and configuration.

Task	Stack Cache Size (Bytes)				D-Cache Only
	128	256	512	1024	
AGP_ClientMainLoop	214672	214672	214672	214672	216492
Airplane_PAOCMainTask	128345	128345	128345	128345	154158
AOC_decoderLoop	99031616	99029636	99029636	99029636	76256099
AOC_feederLoop	47024	47024	47024	47024	47015
AOC_replierLoop	128237	128237	128237	128237	126351
Pilot_MDCUMainLoop	156206926	156206926	156206926	156206926	147521286
CAS_loop	151194	151194	151194	151194	154169
IOP_grbc_manager	46058747	46058747	46058747	46058747	35634672
dry2_1	15570	15570	15570	15570	15264

The figure shows depicts the above table as bar chart. The results are normalized per test case to the hardware configuration that caused the largest WCET bound.

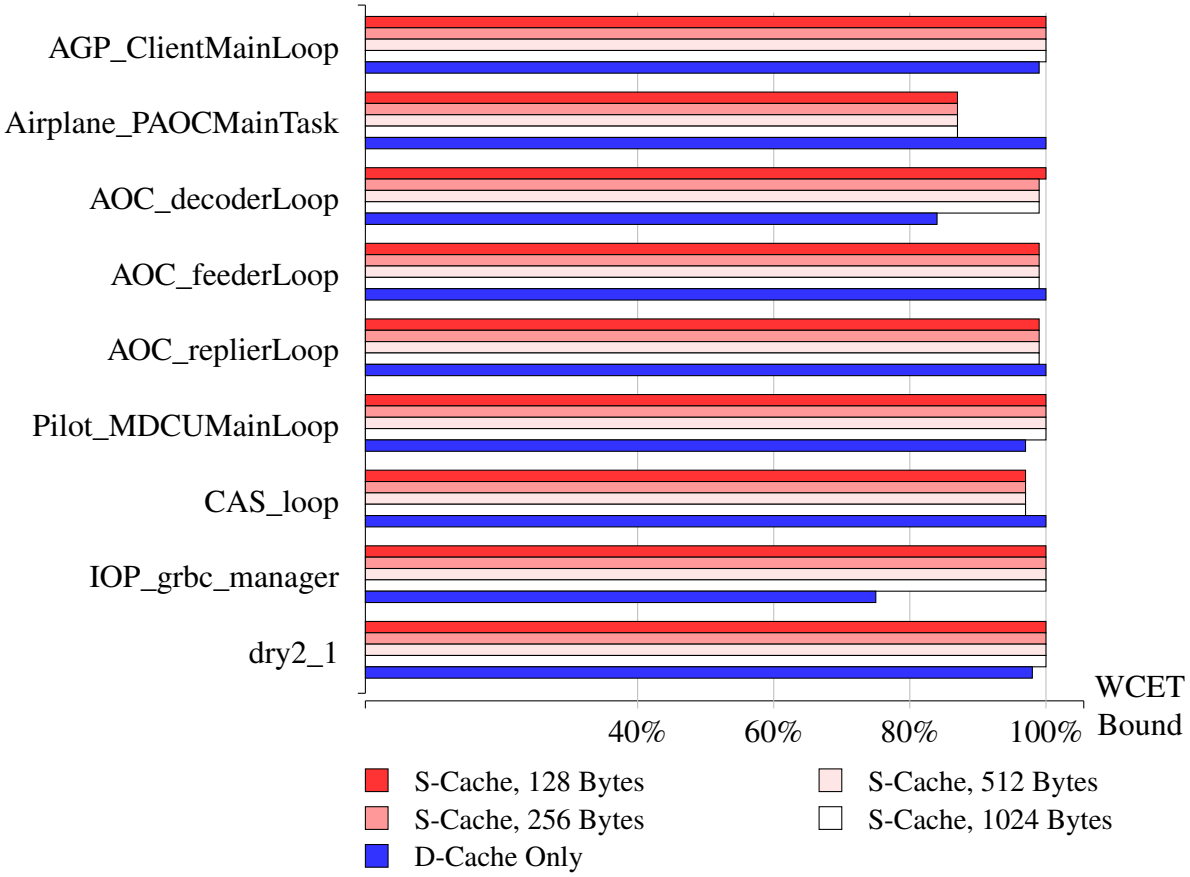


2.4.2 Nodes: 1, Burst Length: 32, Line-Size: 128 bytes, FIFO M-Cache

The table shows the computed WCET bound in processor cycles per test case and configuration.

Task	Stack Cache Size (Bytes)				D-Cache Only
	128	256	512	1024	
AGP_ClientMainLoop	249221	249221	249221	249221	247887
Airplane_PAOCMainTask	183892	183892	183892	183892	209635
AOC_decoderLoop	142930548	142928568	142928568	142928568	120768478
AOC_feederLoop	46814	46814	46814	46814	46821
AOC_replierLoop	164033	164033	164033	164033	164994
Pilot_MDCUMainLoop	206372541	206372541	206372541	206372541	200220523
CAS_loop	155135	155135	155135	155135	158607
IOP_grbc_manager	47238306	47238306	47238306	47238306	35623557
dry2_1	18461	18461	18461	18461	18265

The figure shows depicts the above table as bar chart. The results are normalized per test case to the hardware configuration that caused the largest WCET bound.



2.5 Observations

Comparing a Patmos hardware configuration that supports a data cache only with a Patmos implementation that features a stack cache, we observe better WCET guarantees for the stack cache implementation with an increasing number of Patmos cores in the NOC. The WCET bounds for tasks `AOC_decoderLoop`, `Pilot_MDCUMainLoop`, and `IOP_grbc_manager` do not benefit from using the stack cache. This is caused by unresolved call targets inside system routines that are called in a loop. In this situation the WCET analyzer has to assume that all stack cached data needs to be reloaded after returning to the loop. With all (system) call targets being resolved, we are confident that a Patmos implementation featuring a stack cache allows for better WCET bounds.

In general we observe that an LRU M-Cache performs better than a FIFO M-Cache for all tasks. This is due to the bad static analyzability of the FIFO cache replacement policy. In most cases a higher M-Cache associativity allows for smaller WCET bounds, except for instances of `dry2_1` when a cache line size of less than 128 bytes is being used. Here the costs for filling the M-Cache lines increases for an 8-way M-Cache because the routines are split up into multiple M-Cache blocks.

Furthermore we observe that the traditional I-Cache implementation allows for better WCET guarantees than the M-Cache implementation. The only exception to this observation is the task `AOC_feederLoop`, where both I-Cache and M-Cache perform more or less equally well. This is caused by the structure of the analyzed task that comprises a single routine only.

Finally, we notice that a larger cache line size allows for better WCET guarantees if the number of Patmos cores increases. In the following we only consider a Patmos NOC using a traditional I-Cache implementation:

- The following table shows the best hardware configuration per task assuming a NOC comprising a single Patmos core.

Burst Length	Task
4	AOC_feederLoop
	CAS_loop
8	AGP_ClientMainLoop
	Pilot_MDCUMainLoop
	IOP_grbc_manager
16	Airplane_PAOCMainTask
	AOC_decoderLoop
	AOC_replierLoop
32	dry2_1

- The following table shows the best hardware configuration per task assuming a NOC comprising 16 Patmos cores. Here, a burst length of 4 (*i.e.,* 16 byte cache line size) does not allow for better WCET guarantees than for other burst lengths. Most tasks benefit from a burst length of 32 words.

Burst Length	Task
8	AOC_feederLoop
	CAS_loop
16	IOP_grbc_manager
32	AGP_ClientMainLoop
	Airplane_PAOCMainTask
	AOC_decoderLoop
	AOC_replierLoop
	Pilot_MDCUMainLoop
	dry2_1

- The following table shows the best hardware configuration per task assuming a NOC comprising 64 Patmos cores. For all tasks but `AOC_feederLoop` WCET guarantees are best when using burst lengths of 32. This observation is not unexpected, because the first access beat is very costly (see Section 2.1) but subsequent access beats are comparably cheap.

Burst Length	Task
16	AOC_feederLoop
32	AGP_ClientMainLoop
	Airplane_PAOCMainTask
	AOC_decoderLoop
	AOC_replierLoop
	CAS_loop
	Pilot_MDCUMainLoop
	IOP_grbc_manager
	dry2_1

3 Conclusion

In this document we have explored several hardware configurations to identify their impact on the guaranteed worst-case performance for a representative set of tasks from the application planned for the project demonstration.

The experiments are made with the help of the static WCET analyzer aiT for Patmos. The explored design space includes both architectural features that are direct parameter to aiT, like line sizes or burst lengths, and features that required to enhance aiT's Patmos timing model, like the inclusion of the various caches.

The experiments show a significant large influence of the considered parameters on the worst-case execution time bounds and there is a large performance benefit when these parameters are specially optimized for one application.

In contrast to the classical optimization approach based on observed/simulated execution times, our approach has a set of distinctive advantages:

- No "worst-case" input is required and hence there is no need to spend time to design test cases. The aiT analyzer finds automatically the worst-case paths without test inputs.

- The results are reliable worst-case results. With simulation, finding "worst-case" inputs for each configuration is practically close to impossible.
- The method is fast. aiT's automatic batch mode allows for an automated execution of all experiments in a few hours. Large parameter spaces can be efficiently explored and ideal configuration can be efficiently identified.

References

[1] Sahar Abbaspour, Florian Brandner, and Martin Schoeberl. A time-predictable stack cache. In *Proceedings of the 9th Workshop on Software Technologies for Embedded and Ubiquitous Systems*, 2013.

[2] Christoph Berg, Jakob Engblom, and Reinhard Wilhelm. Requirements for and design of a processor with predictable timing. In Lothar Thiele and Reinhard Wilhelm, editors, *Perspectives Workshop: Design of Systems with Predictable Behaviour*, number 03471 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2004. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.

[3] Stephen A. Edwards and Edward A. Lee. The case for the precision timed (PRET) machine. In *DAC '07: Proceedings of the 44th annual conference on Design automation*, pages 264–265, New York, NY, USA, 2007. ACM.

[4] Christian Ferdinand. Cache Behavior Prediction for Real-Time Systems. PhD Thesis, Universität des Saarlandes, 1997.

[5] Reinhold Heckmann, Marc Langenbach, Stephan Thesing, and Reinhard Wilhelm. The influence of processor architecture on the design and results of WCET tools. *Proceedings of the IEEE*, 91(7):1038–1054, Jul. 2003.

[6] Benedikt Huber, Wolfgang Puffitsch, and Martin Schoeberl. Worst-case execution time analysis driven object cache design. *Concurrency and Computation: Practice and Experience*, 24(8):753–771, 2012.

[7] Jörg Mische, Irakli Guliashvili, Sascha Uhrig, and Theo Ungerer. How to enhance a superscalar processor to provide hard real-time capable in-order smt. In *23rd International Conference on Architecture of Computing Systems (ARCS 2010)*, pages 2–14, University of Augsburg, Germany, February 2010. Springer.

[8] Jan Reineke. *Caches in WCET Analysis*. PhD thesis, Universität des Saarlandes, November 2008.

[9] Martin Schoeberl. Time-predictable computer architecture. *EURASIP Journal on Embedded Systems*, vol. 2009, Article ID 758480:17 pages, 2009.

[10] Martin Schoeberl, Pascal Schleuniger, Wolfgang Puffitsch, Florian Brandner, Christian W. Probst, Sven Karlsson, and Tommy Thorn. Towards a time-predictable dual-issue microprocessor: The Patmos approach. In *First Workshop on Bringing Theory to Practice: Predictability and Performance in Embedded Systems (PPES 2011)*, pages 11–20, Grenoble, France, March 2011.

[11] Lothar Thiele and Reinhard Wilhelm. Design for timing predictability. *Real-Time Systems*, 28(2-3):157–177, 2004.

[12] T. Ungerer, F. Cazorla, P. Sainrat, G. Bernat, Z. Petrov, C. Rochange, E. Quiñones, M. Gerdes, M. Paolieri, and J. Wolf. Merasa: Multi-core execution of hard real-time applications supporting analysability. *Micro, IEEE*, 30(5):66–75, 2010.

[13] Reinhard Wilhelm, Daniel Grund, Jan Reineke, Marc Schlickling, Markus Pister, and Christian Ferdinand. Memory hierarchies, pipelines, and buses for future architectures in time-critical embedded systems. *IEEE Transactions on CAD of Integrated Circuits and Systems*, 28(7):966–978, 2009.

4 Appendix - Exploration Results

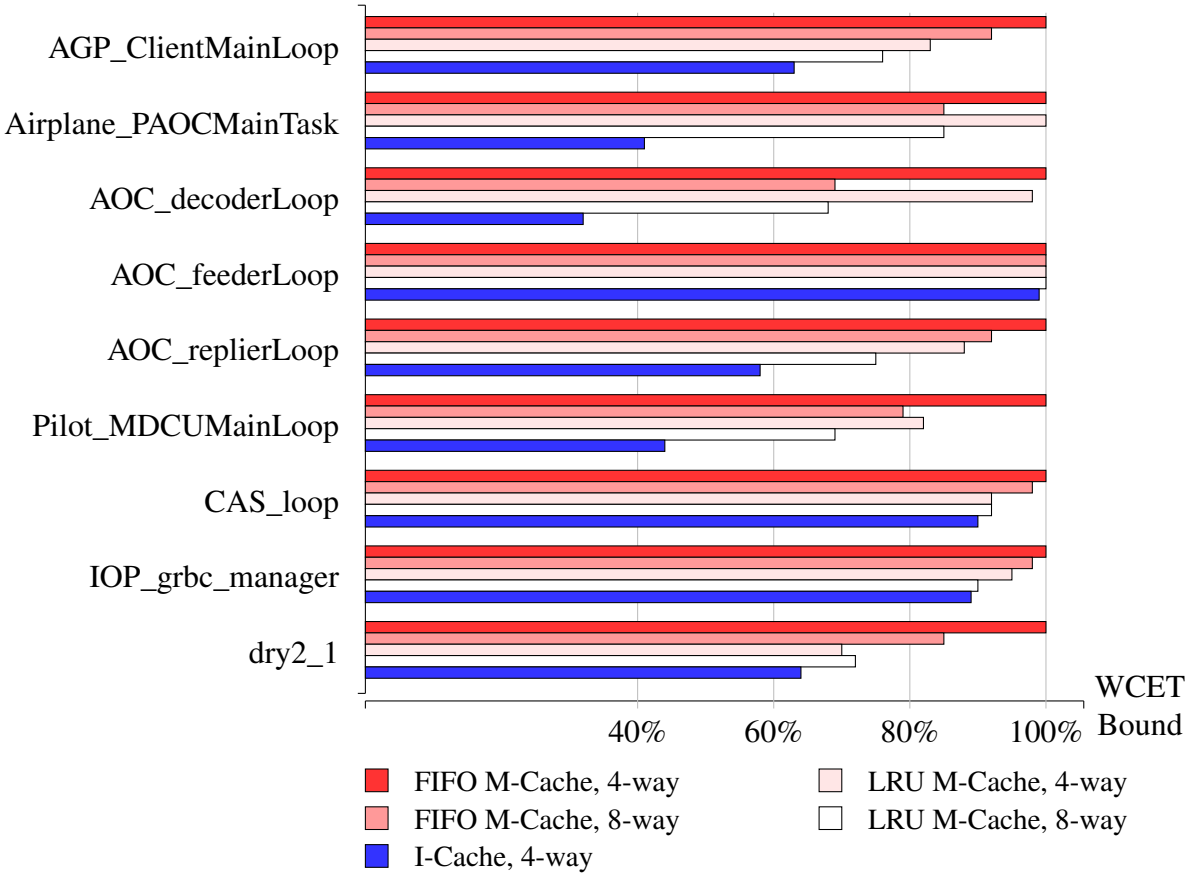
4.1 I-Cache vs. M-Cache Results

4.1.1 Nodes: 16, Burst Length: 4, Line-Size: 16 bytes

The table shows the computed WCET bound in processor cycles per test case and configuration.

Task	FIFO M-Cache		LRU M-Cache		I-Cache
	4-way	*8-way*	*4-way*	*8-way*	*4-way*
AGP_ClientMainLoop	4350345	4015476	3653748	3318879	2756932
Airplane_PAOCMainTask	4283280	3642200	4283280	3642200	1768959
AOC_decoderLoop	3064584714	2134532022	3018586887	2113954421	983405578
AOC_feederLoop	373742	373742	373742	373742	373722
AOC_replierLoop	4153841	3849669	3680770	3136391	2411874
Pilot_MDCUMainLoop	4074168245	3236408620	3366103525	2848728525	1825062651
CAS_loop	1770553	1741434	1643905	1633019	1608902
IOP_grbc_manager	368857924	361581621	353053641	332334840	328864784
dry2_1	304119	259066	213854	221220	197101

The figure shows depicts the above table as bar chart. The results are normalized per test case to the hardware configuration that caused the largest WCET bound.

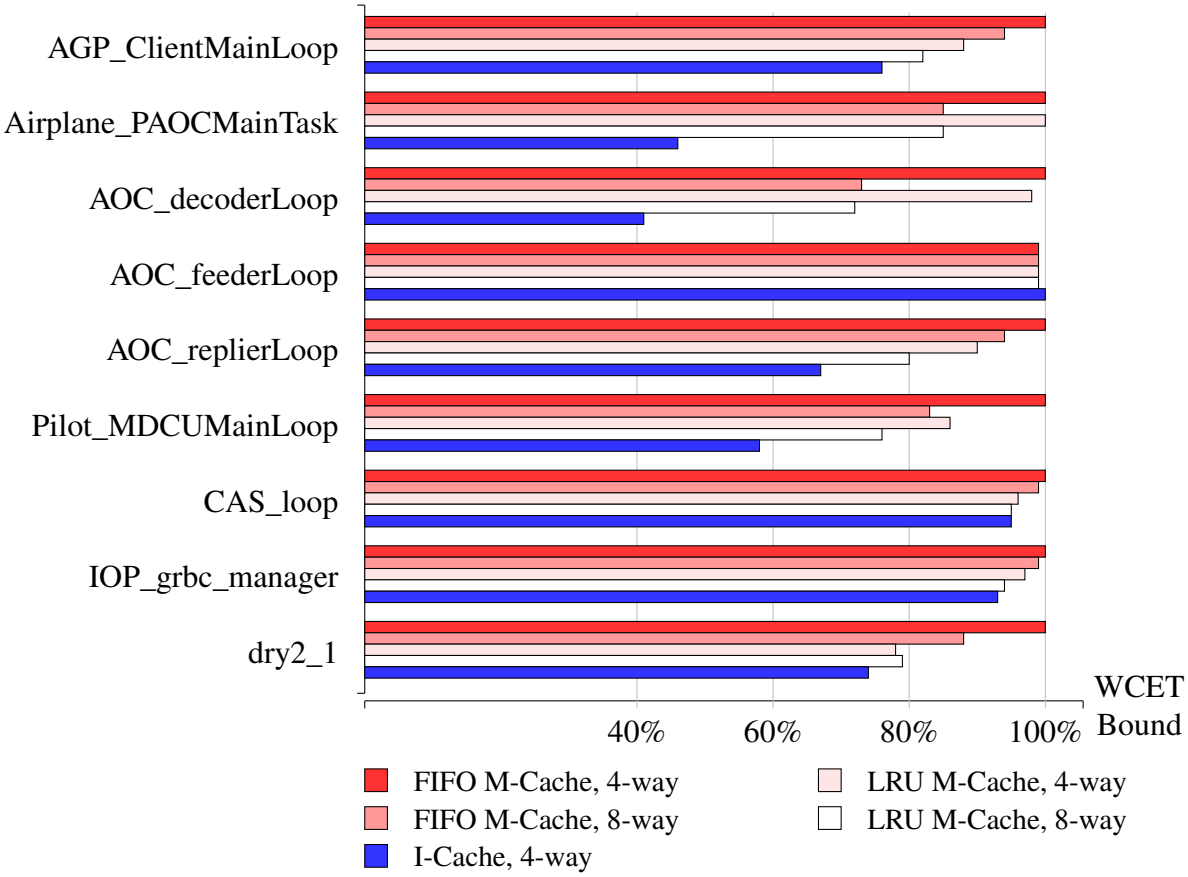


4.1.2 Nodes: 16, Burst Length: 8, Line-Size: 32 bytes

The table shows the computed WCET bound in processor cycles per test case and configuration.

Task	FIFO M-Cache		LRU M-Cache		I-Cache
	4-way	*8-way*	*4-way*	*8-way*	*4-way*
AGP_ClientMainLoop	3136869	2966680	2771064	2600875	2385201
Airplane_PAOCMainTask	2389780	2032236	2389780	2032236	1109966
AOC_decoderLoop	1748890660	1281912218	1725088833	1271289881	717455309
AOC_feederLoop	361478	361478	361478	361478	361487
AOC_replierLoop	2710603	2563951	2458776	2190057	1839290
Pilot_MDCUMainLoop	2653889907	2221417070	2290463195	2023674591	1558684032
CAS_loop	1657327	1641096	1591539	1585665	1577496
IOP_grbc_manager	313537448	311593197	305193885	295594540	294721306
dry2_1	211177	186852	165304	167622	157999

The figure shows depicts the above table as bar chart. The results are normalized per test case to the hardware configuration that caused the largest WCET bound.

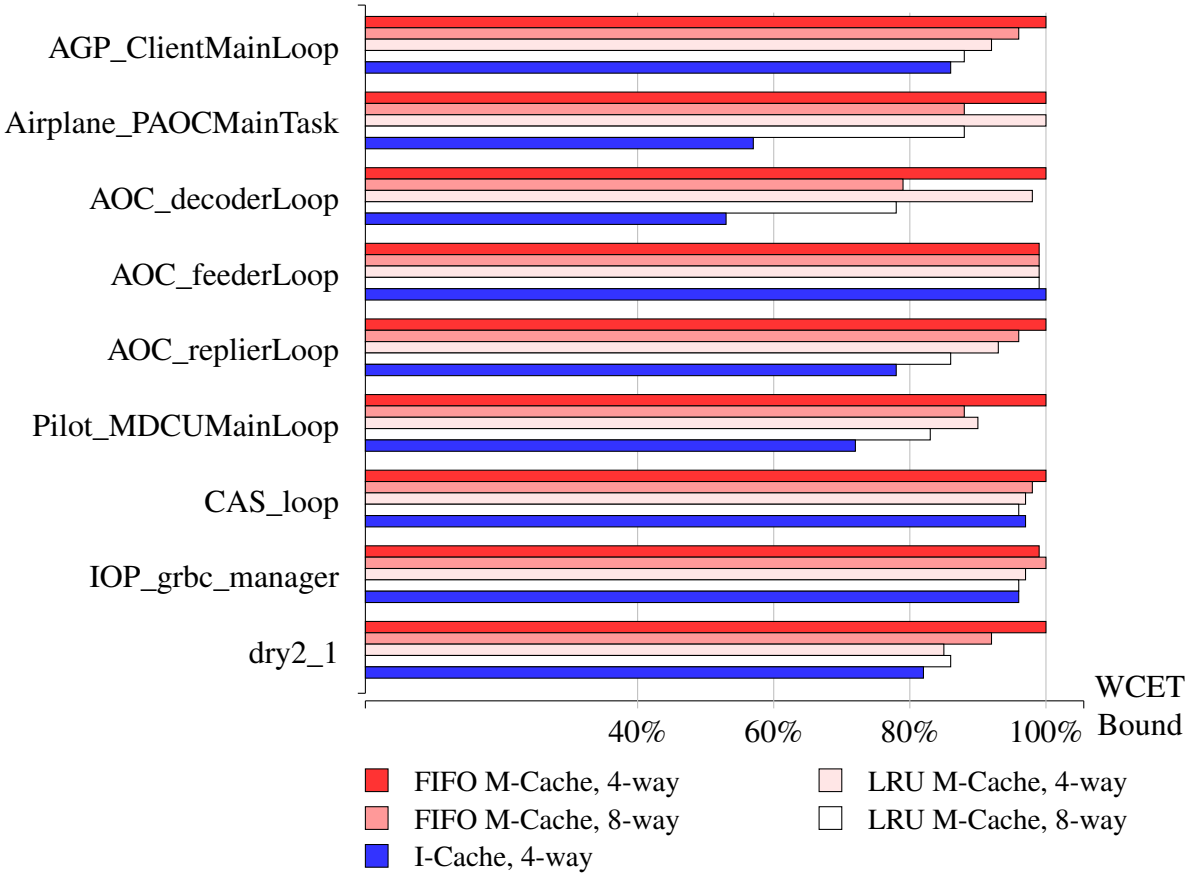


4.1.3 Nodes: 16, Burst Length: 16, Line-Size: 64 bytes

The table shows the computed WCET bound in processor cycles per test case and configuration.

Task	FIFO M-Cache		LRU M-Cache		I-Cache
	4-way	*8-way*	*4-way*	*8-way*	*4-way*
AGP_ClientMainLoop	2548761	2468788	2348352	2268379	2195940
Airplane_PAOCMainTask	1479372	1310852	1479372	1310852	844560
AOC_decoderLoop	1046310626	829797576	1033866707	823963847	557819507
AOC_feederLoop	363706	363706	363706	363706	363743
AOC_replierLoop	2019203	1951311	1881936	1747109	1590757
Pilot_MDCUMainLoop	1947638059	1722375622	1757420339	1618554443	1405656643
CAS_loop	1621563	1596400	1586563	1566745	1582826
IOP_grbc_manager	289004108	290612649	284735301	281194543	279947892
dry2_1	166223	153998	141830	143360	137463

The figure shows depicts the above table as bar chart. The results are normalized per test case to the hardware configuration that caused the largest WCET bound.

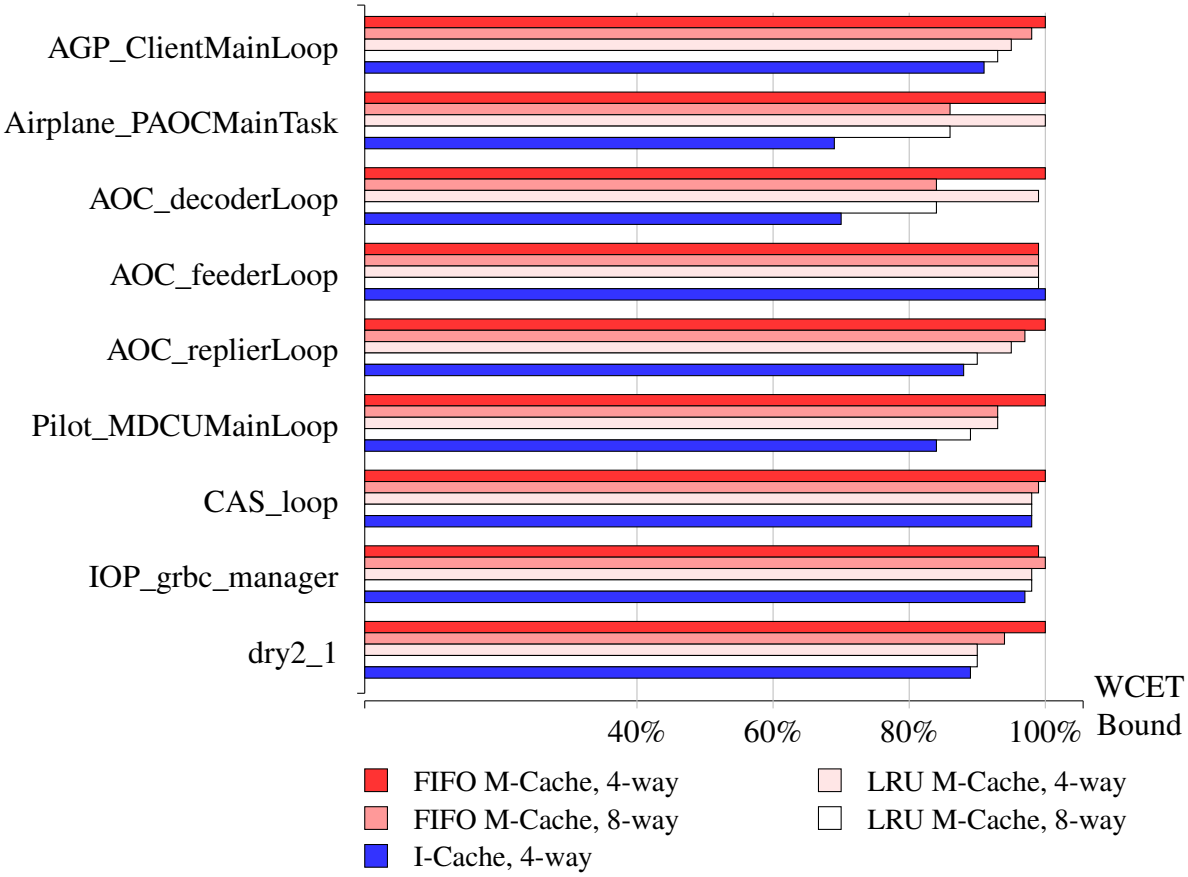


4.1.4 Nodes: 16, Burst Length: 32, Line-Size: 128 bytes

The table shows the computed WCET bound in processor cycles per test case and configuration.

Task	FIFO M-Cache		LRU M-Cache		I-Cache
	4-way	*8-way*	*4-way*	*8-way*	*4-way*
AGP_ClientMainLoop	2292417	2249676	2186520	2143779	2105714
Airplane_PAOCMainTask	1033276	896260	1033276	896260	722855
AOC_decoderLoop	753453946	638942972	746601987	635810711	528514224
AOC_feederLoop	373174	373174	373174	373174	374026
AOC_replierLoop	1625271	1588883	1548864	1477045	1432827
Pilot_MDCUMainLoop	1619852363	1511947338	1517326027	1450719567	1366137438
CAS_loop	1614215	1610872	1594967	1593389	1593582
IOP_grbc_manager	286576778	288603743	284347855	283299057	280423649
dry2_1	147099	139646	133446	133304	132010

The figure shows depicts the above table as bar chart. The results are normalized per test case to the hardware configuration that caused the largest WCET bound.

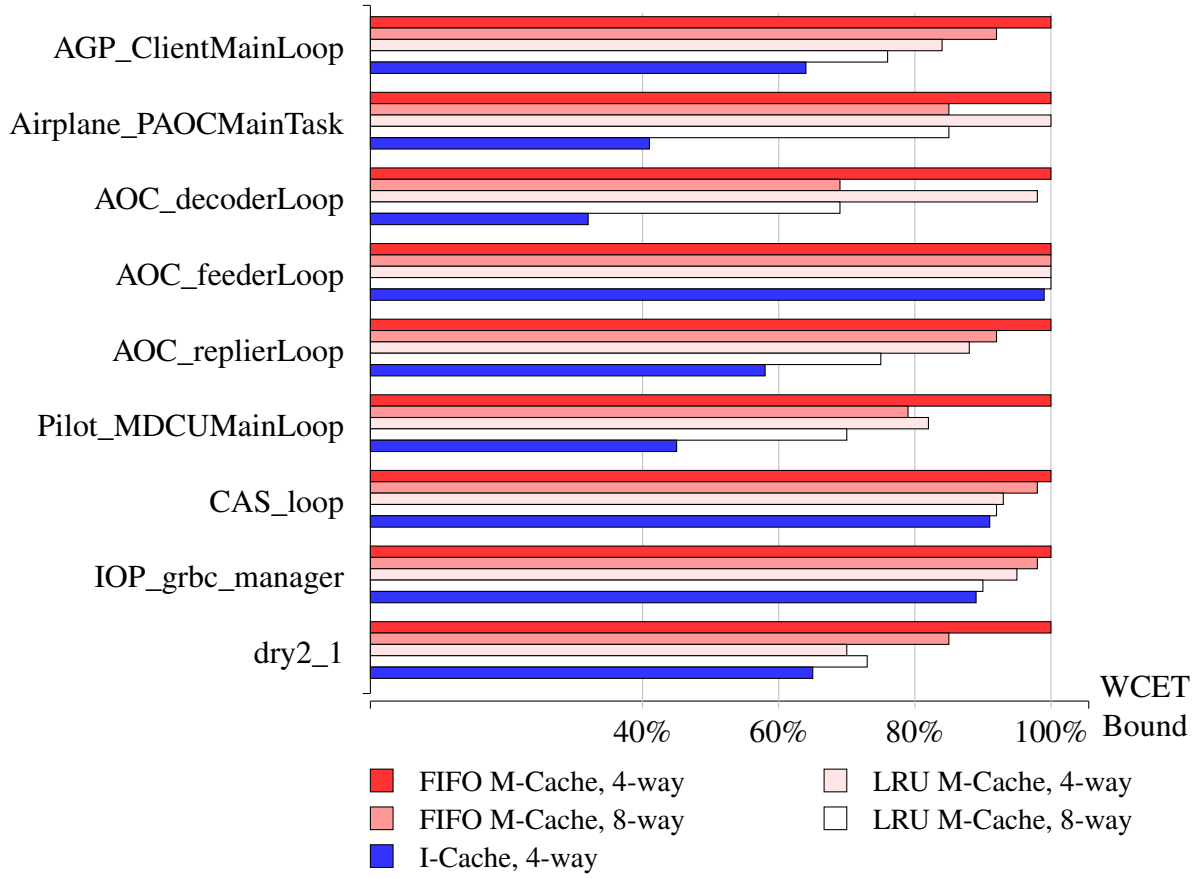


4.1.5 Nodes: 64, Burst Length: 4, Line-Size: 16 bytes

The table shows the computed WCET bound in processor cycles per test case and configuration.

Task	FIFO M-Cache		LRU M-Cache		I-Cache
	4-way	*8-way*	*4-way*	*8-way*	*4-way*
AGP_ClientMainLoop	16814317	15549328	14182666	12917677	10796768
Airplane_PAOCMainTask	16198610	13778170	16198610	13778170	6704345
AOC_decoderLoop	11587161294	8073338364	11413377111	7995576581	3725056906
AOC_feederLoop	1463070	1463070	1463070	1463070	1463050
AOC_replierLoop	15973603	14824235	14186232	12129397	9395012
Pilot_MDCUMainLoop	15607150509	12442570340	12932314697	10977977089	7111937287
CAS_loop	7000497	6890498	6522021	6480805	6389962
IOP_grbc_manager	1413446430	1385988545	1353714197	1275431082	1262518312
dry2_1	1169329	998916	828336	855926	765081

The figure shows depicts the above table as bar chart. The results are normalized per test case to the hardware configuration that caused the largest WCET bound.

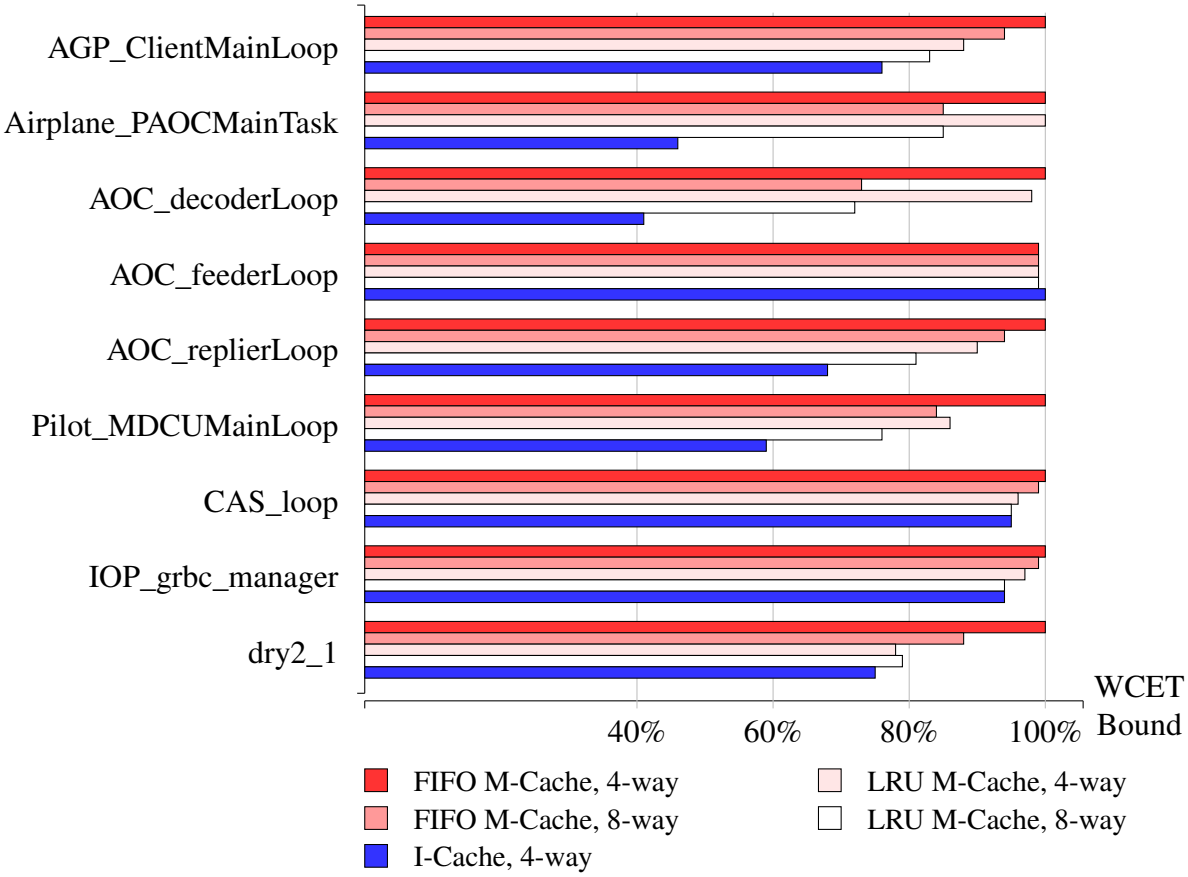


4.1.6 Nodes: 64, Burst Length: 8, Line-Size: 32 bytes

The table shows the computed WCET bound in processor cycles per test case and configuration.

Task	FIFO M-Cache		LRU M-Cache		I-Cache
	4-way	*8-way*	*4-way*	*8-way*	*4-way*
AGP_ClientMainLoop	12145247	11509998	10778552	10143303	9337205
Airplane_PAOCMainTask	8948836	7612644	8948836	7612644	4166918
AOC_decoderLoop	6545162808	4802045572	6456256625	4762352941	2692473319
AOC_feederLoop	1408344	1408344	1408344	1408344	1408353
AOC_replierLoop	10443909	9896901	9502580	8499875	7188658
Pilot_MDCUMainLoop	10156084077	8541329642	8799151709	7803062079	6066880560
CAS_loop	6541515	6480800	6295769	6273719	6243308
IOP_grbc_manager	1193702988	1186634989	1162500715	1126738716	1123428994
dry2_1	811507	720460	640270	648658	612749

The figure shows depicts the above table as bar chart. The results are normalized per test case to the hardware configuration that caused the largest WCET bound.

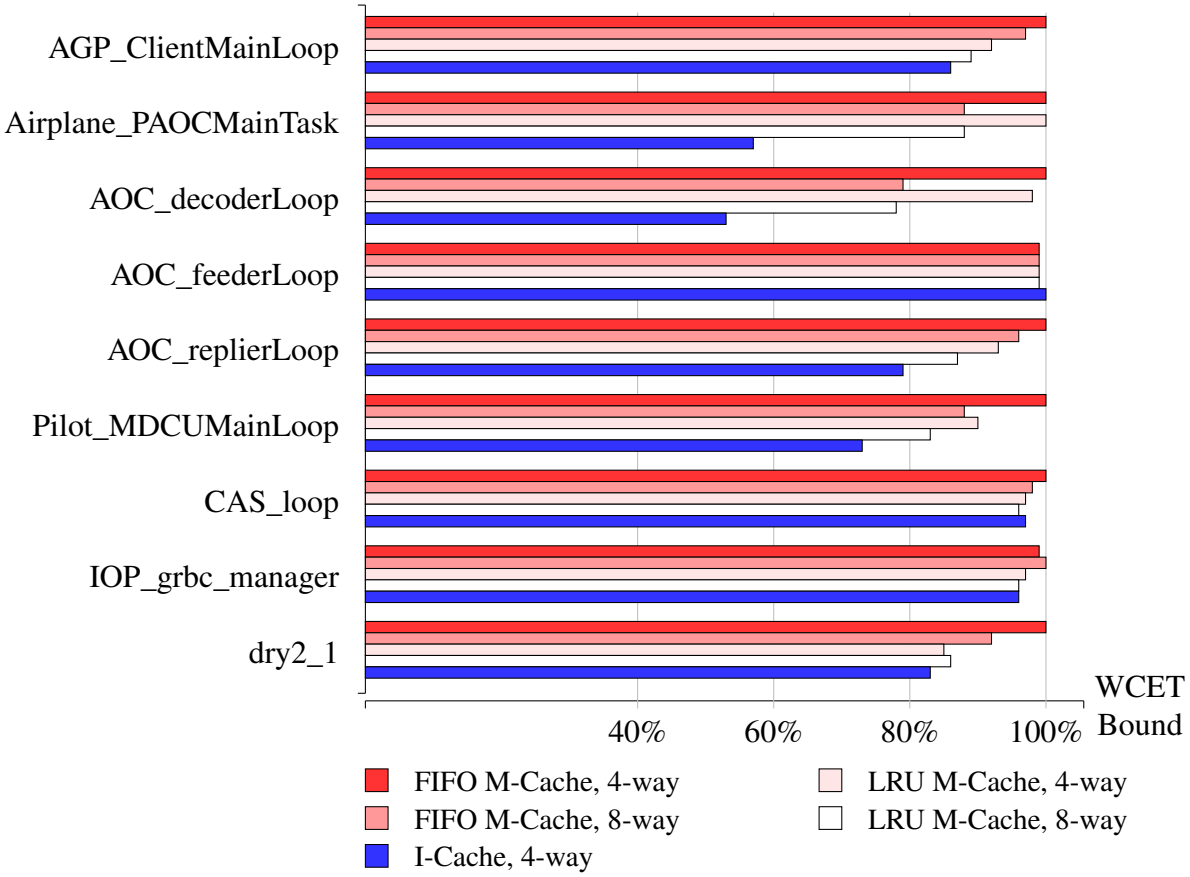


4.1.7 Nodes: 64, Burst Length: 16, Line-Size: 64 bytes

The table shows the computed WCET bound in processor cycles per test case and configuration.

Task	FIFO M-Cache		LRU M-Cache		I-Cache
	4-way	*8-way*	*4-way*	*8-way*	*4-way*
AGP_ClientMainLoop	9840461	9550200	9106244	8815983	8547410
Airplane_PAOCMainTask	5450268	4836908	5450268	4836908	3121020
AOC_decoderLoop	3837937930	3052605950	3792464641	3031225063	2052930301
AOC_feederLoop	1400462	1400462	1400462	1400462	1400499
AOC_replierLoop	7779037	7533209	7275788	6785089	6206139
Pilot_MDCUMainLoop	7415611373	6593234138	6721034061	6214123725	5433972023
CAS_loop	6352083	6260194	6224071	6151681	6210224
IOP_grbc_manager	1083525820	1090043777	1067905505	1055312253	1050379498
dry2_1	635125	590552	546028	551606	529533

The figure shows depicts the above table as bar chart. The results are normalized per test case to the hardware configuration that caused the largest WCET bound.

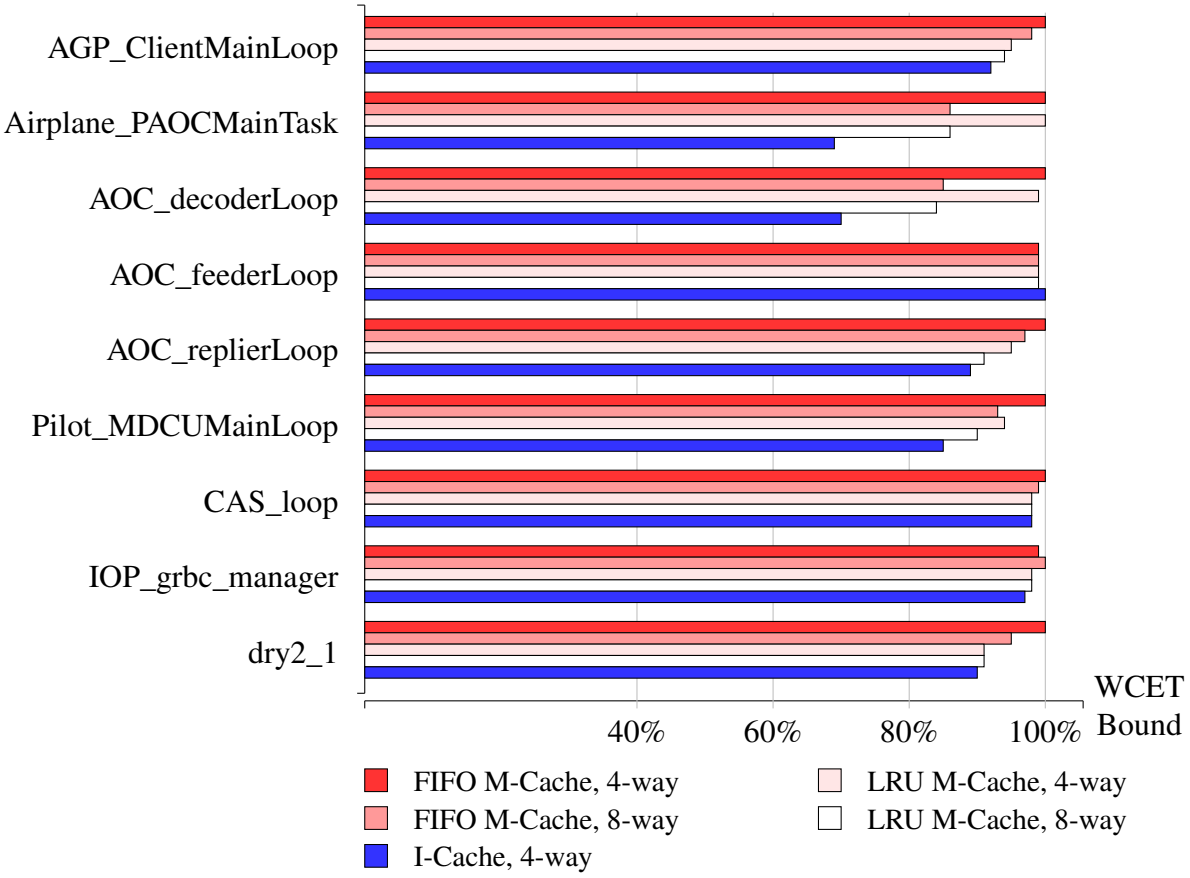


4.1.8 Nodes: 64, Burst Length: 32, Line-Size: 128 bytes

The table shows the computed WCET bound in processor cycles per test case and configuration.

Task	FIFO M-Cache		LRU M-Cache		I-Cache
	4-way	*8-way*	*4-way*	*8-way*	*4-way*
AGP_ClientMainLoop	8747009	8599124	8374208	8226323	8081092
Airplane_PAOCMainTask	3710092	3217204	3710092	3217204	2596937
AOC_decoderLoop	2677202454	2281928692	2653112945	2270878279	1882068842
AOC_feederLoop	1403864	1403864	1403864	1403864	1406738
AOC_replierLoop	6242675	6117319	5972156	5722401	5556863
Pilot_MDCUMainLoop	6082657125	5709057256	5723413625	5492824987	5187856990
CAS_loop	6225437	6214006	6157661	6152039	6152232
IOP_grbc_manager	1041521886	1049643159	1033702133	1030641445	1019704325
dry2_1	553319	527672	505292	505154	499812

The figure shows depicts the above table as bar chart. The results are normalized per test case to the hardware configuration that caused the largest WCET bound.



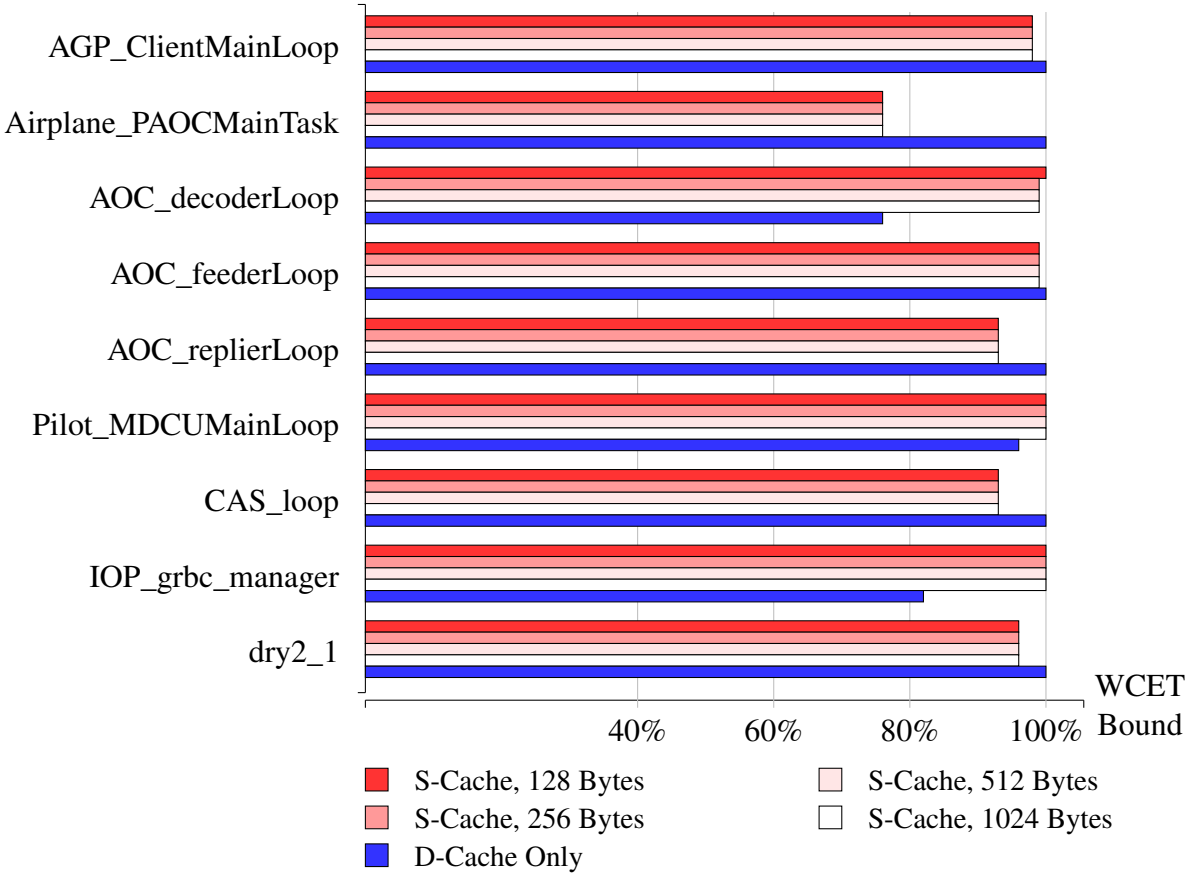
4.2 D-Cache vs. S-Cache Results

4.2.1 Nodes: 16, Burst Length: 32, Line-Size: 128 bytes, I-Cache

The table shows the computed WCET bound in processor cycles per test case and configuration.

Task	Stack Cache Size (Bytes)				D-Cache Only
	128	256	512	1024	
AGP_ClientMainLoop	2105714	2105714	2105714	2105714	2138114
Airplane_PAOCMainTask	722855	722855	722855	722855	941652
AOC_decoderLoop	528530328	528514224	528514224	528514224	404534953
AOC_feederLoop	374026	374026	374026	374026	375931
AOC_replierLoop	1432827	1432827	1432827	1432827	1537989
Pilot_MDCUMainLoop	1366137438	1366137438	1366137438	1366137438	1316419916
CAS_loop	1593582	1593582	1593582	1593582	1699961
IOP_grbc_manager	280423649	280423649	280423649	280423649	231608659
dry2_1	132010	132010	132010	132010	136800

The figure shows depicts the above table as bar chart. The results are normalized per test case to the hardware configuration that caused the largest WCET bound.

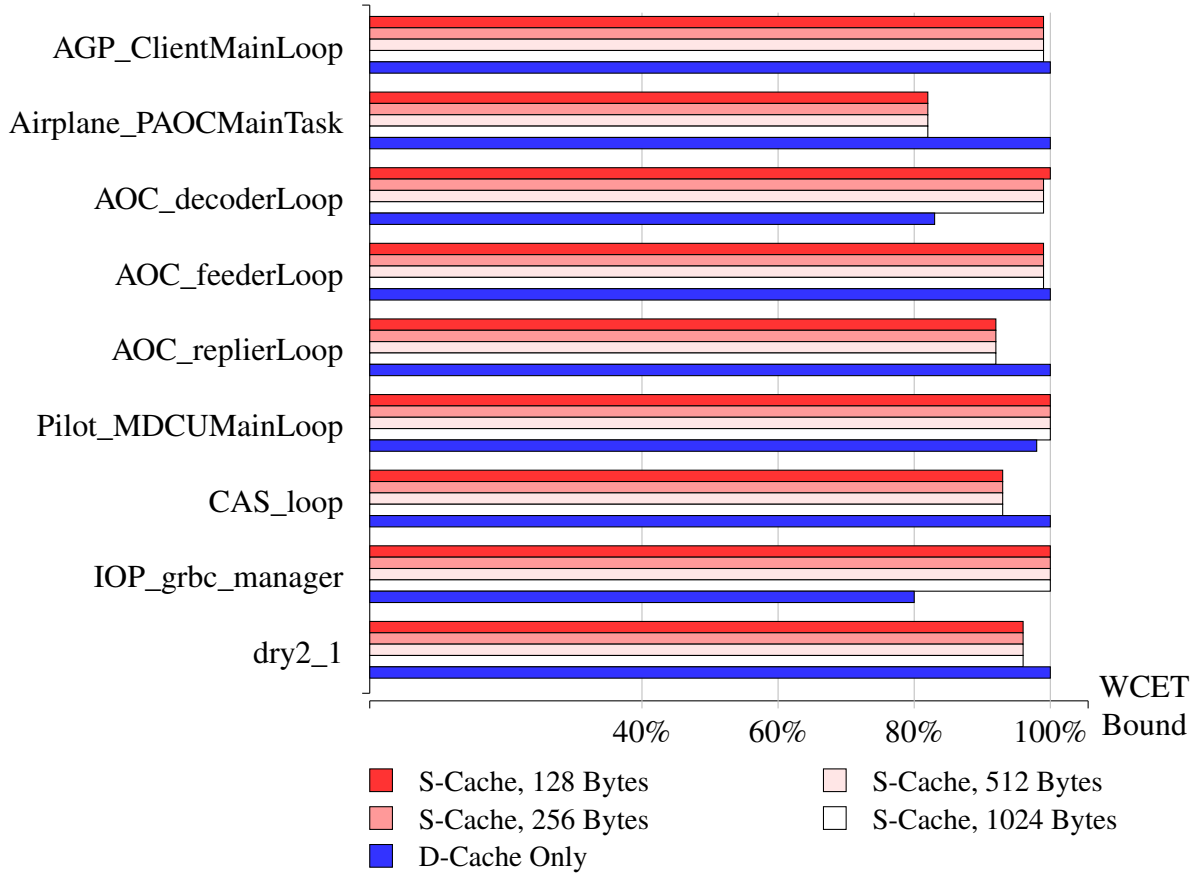


4.2.2 Nodes: 16, Burst Length: 32, Line-Size: 128 bytes, FIFO M-Cache

The table shows the computed WCET bound in processor cycles per test case and configuration.

Task	Stack Cache Size (Bytes)				D-Cache Only
	128	256	512	1024	
AGP_ClientMainLoop	2292417	2292417	2292417	2292417	2308181
Airplane_PAOCMainTask	1033276	1033276	1033276	1033276	1252003
AOC_decoderLoop	753470050	753453946	753453946	753453946	629715732
AOC_feederLoop	373174	373174	373174	373174	375095
AOC_replierLoop	1625271	1625271	1625271	1625271	1748688
Pilot_MDCUMainLoop	1619852363	1619852363	1619852363	1619852363	1591886091
CAS_loop	1614215	1614215	1614215	1614215	1723659
IOP_grbc_manager	286576778	286576778	286576778	286576778	232066273
dry2_1	147099	147099	147099	147099	152641

The figure shows depicts the above table as bar chart. The results are normalized per test case to the hardware configuration that caused the largest WCET bound.

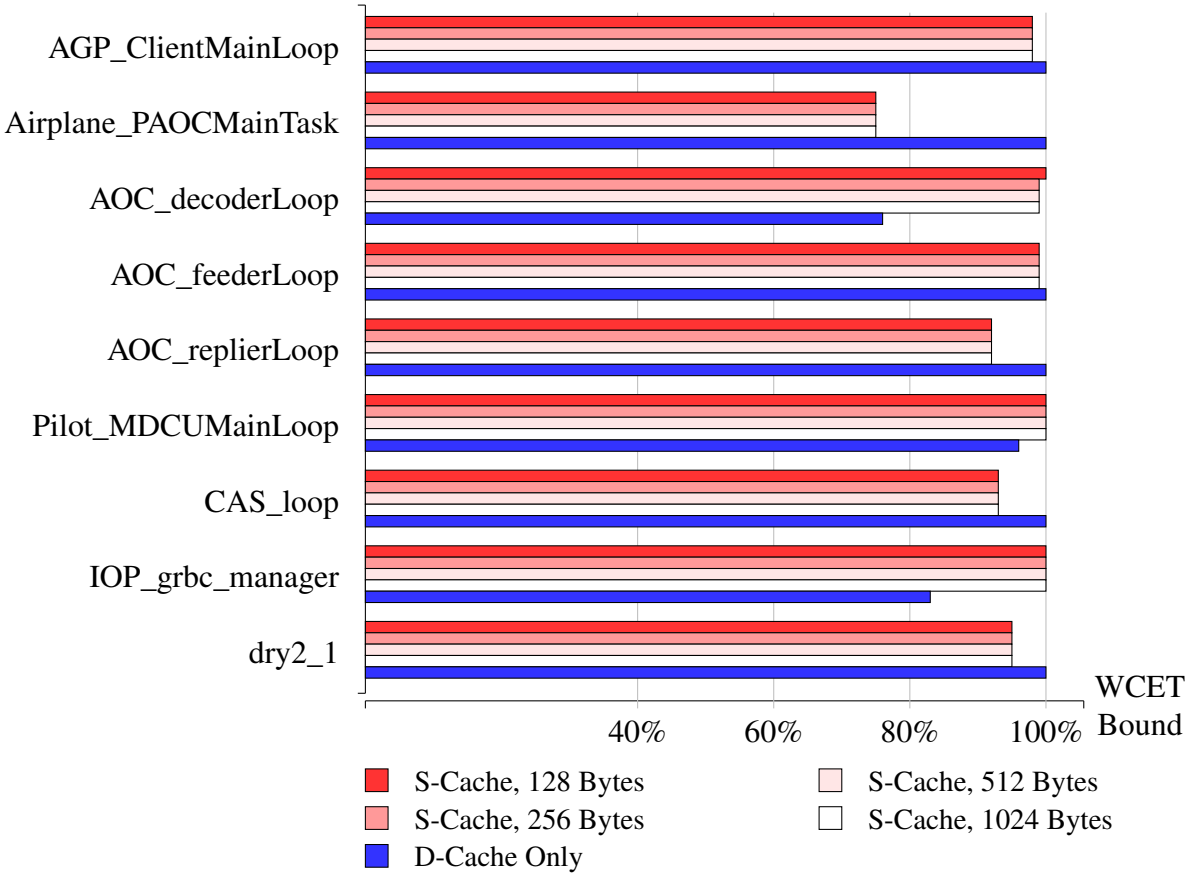


4.2.3 Nodes: 64, Burst Length: 32, Line-Size: 128 bytes, I-Cache

The table shows the computed WCET bound in processor cycles per test case and configuration.

Task	Stack Cache Size (Bytes)				D-Cache Only
	128	256	512	1024	
AGP_ClientMainLoop	8081092	8081092	8081092	8081092	8210430
Airplane_PAOCMainTask	2596937	2596937	2596937	2596937	3425928
AOC_decoderLoop	1882129430	1882068842	1882068842	1882068842	1439503673
AOC_feederLoop	1406738	1406738	1406738	1406738	1414703
AOC_replierLoop	5556863	5556863	5556863	5556863	6001317
Pilot_MDCUMainLoop	5187856990	5187856990	5187856990	5187856990	5009019780
CAS_loop	6152232	6152232	6152232	6152232	6585875
IOP_grbc_manager	1019704325	1019704325	1019704325	1019704325	850092639
dry2_1	499812	499812	499812	499812	520758

The figure shows depicts the above table as bar chart. The results are normalized per test case to the hardware configuration that caused the largest WCET bound.



4.2.4 Nodes: 64, Burst Length: 32, Line-Size: 128 bytes, FIFO M-Cache

The table shows the computed WCET bound in processor cycles per test case and configuration.

Task	Stack Cache Size (Bytes)				D-Cache Only
	128	256	512	1024	
AGP_ClientMainLoop	8747009	8747009	8747009	8747009	8817249
Airplane_PAOCMainTask	3710092	3710092	3710092	3710092	4539013
AOC_decoderLoop	2677263042	2677202454	2677202454	2677202454	2233634812
AOC_feederLoop	1403864	1403864	1403864	1403864	1411845
AOC_replierLoop	6242675	6242675	6242675	6242675	6753912
Pilot_MDCUMainLoop	6082657125	6082657125	6082657125	6082657125	5986097713
CAS_loop	6225437	6225437	6225437	6225437	6670233
IOP_grbc_manager	1041521886	1041521886	1041521886	1041521886	851933301
dry2_1	553319	553319	553319	553319	577039

The figure shows depicts the above table as bar chart. The results are normalized per test case to the hardware configuration that caused the largest WCET bound.