Lightweight and predictable memory virtualization on medium-size microcontrollers

Stefano Mercogliano, Daniele Ottaviano, Alessandro Cilardo, Marcello Cinque
Dept. of Electrical Engineering and Information Technologies
University of Naples Federico II, Naples, Italy
{stefano.mercogliano, daniele.ottaviano, acilardo, macinque}@unina.it

***Abstract*—Nowadays industry research is heading towards the consolidation of multiple real-time applications and execution environments on single microcontrollers, with the aim of optimizing area, power, and cost while keeping an eye on protection and flexibility. To this end, virtualization seems an attractive solution, but it must be redesigned according to the specific requirements of microcontroller tasks, different than traditional application processor workloads. This paper examines two possible hardware-based models to support virtual machines on medium-size microcontrollers providing an extensive and reproducible analysis over a RISC-V processor.**

***Index Terms*—Embedded Virtualization, Real-Time applications, low-power system design, MMU-less embedded systems**

I. Introduction and Motivations

Embedded system complexity is constantly increasing, pushed by emerging hardware technologies and the need for reliable and responsive services, particularly in the industrial sectors (e.g., Automotive, Aerospace, Nuclear, and Military). Predictability, safety, and certifiability are key features when machines interact with the environment, while guaranteeing performance and containing Size, Weight, Power, and Cost (SWaP-C). To achieve that, industry research focuses on the consolidation of different operating systems or execution environments as well as applications on a single device by using virtualization to prevent cross-interference and ensure that application requirements are met [1]. This is usually obtained by means of ad-hoc hypervisors running on processors offering hardware extensions for virtualization. However, application-class processors (e.g., Arm Cortex-A family) inherently introduce drawbacks in meeting real-time and power-saving requirements due complex underlying mechanisms (e.g., cache, page-based memory virtualization). In these scenarios, despite lower performance, microcontrollers are preferred due to their low power consumption and high predictability [1]. We specifically categorize *medium-size* microcontrollers as those having sufficient computing power for multiple applications, but only featuring memory protection mechanisms with no memory virtualization support. This places a limit on the consolidation of independent applications on such architectures, while ensuring the required protection. Microcontroller virtualization has been explored in several recent works [2], [3]. However, none of these integrates hardware mechanisms for address relocation, limiting the usability of the approach only to a set of applications statically linked together. This clearly points out the need for ad-hoc memory virtualization solutions that take into account power consumption and determinism as the main constraints. In the light of that, in this paper we (1) examine two hardware-based models to support lightweight and predictable memory virtualization for microcontrollers. Then, (2) we provide an open-source implementation of the two models on a RISC-V processor together with (3) an extensive and reproducible analysis of area and performance overhead on a number of benchmarks.

This work has been partially supported by Spoke 1 *Future HPC & Big Data* and Spoke 9 *Digital Society & Smart Cities* of ICSC - Centro Nazionale di Ricerca in High-Performance-Computing, Big Data and Quantum Computing, funded by European Union - NextGenerationEU. The work has also been partially supported by the EUROfusion Consortium, funded by the European Union via the Euratom Research and Training Programme (Grant Agreement No 101052200 - EUROfusion). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the European Commission can be held responsible for them.

II. Proposed Methods

Medium-size microcontrollers running multiple execution environments must enforce specific requirements such as efficiency, security, predictability, and flexibility to satisfy industry needs. We call a virtual memory scheme supporting these requirements a *lightweight and predictable virtual memory (LPVM)*. Due to relatively simple workloads, fully-fledged virtualization may be overkilled. In fact, unlike general-purpose tasks, microcontroller-targeted applications are unlikely to dynamically require more memory or implement disk swapping. Thus, segmentation is preferable to paging, offering finer memory organization without internal fragmentation. Additionally, typical applications require only a fixed translation upon deployment, eliminating the need for dynamic relocation and resulting in a lighter virtual memory mechanism compared to the classic one. The simplest approach to an LPVM consists in reusing a hardware table-based structure (e.g. ARM MPU, RISC-V PMP) configured by means of a private memory-mapped address space. Each memory transaction must be checked against all entries in parallel. If the access falls into the interval $[base, base + range]$, access permissions are checked and finally an *address relocation function* is applied. A tabular approach is quite simple, but as we show in the next section, it does not scale well in terms of area occupancy and power consumption. These issues can be addressed by adopting a trie-like structure as a possible generalization of ARM MPUs [4]. Unlike an MMU, an LPVM can take advantage of the sparsity of microcontroller memory to implement

979-8-3503-4859-0/DATE24/© 2024 EDAA

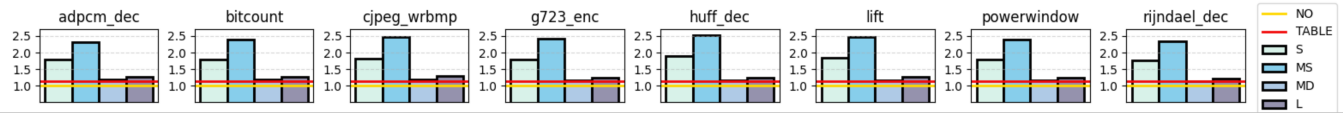

Fig. 1. Normalized execution times. For each benchmark, the GRT-based LPVM overhead is shown compared to the baseline and the table-based LPVM overhead. All the values are normalized to the baseline to provide a fair comparison between the benchmarks.

a hardware walker for guarded radix tries (GRTs), minimizing memory usage and latency. In fact, a GRT is based on the key idea that subparts of a virtual address prefix can be skipped during trie exploration because only one entry in that trie exists for it. As a consequence, multiple levels can be avoided, reducing overall performance degradation and improving flexibility and scalability.

III. Experimental Results

We implemented both the table- and GRT-based schemes on a 32-bit open-source RISC-V processor, cv32e41s [5]. The results presented are collected from a purposely developed cycle-accurate simulator [6] and the experiments are easily reproducible using scripts available at [7]. Results from Table I show various synthesis results for multiple configurations of cv32e41s on a Nexys A7-50T board in terms of occupancy, power usage, and working frequency. A key observation is that, in the table-based solution, the maximum working frequency drops by 16% compared to the PMP baseline, regardless of the entry count. Increasing entries only raises resource usage and, consequently, power consumption. While eight entries have minimal impact on area and power, scaling to 64 entries becomes burdensome for medium-size microcontrollers, resulting in a 16% worsening factor for area and 9% for power.

Table I highlights LPVM-GRT advantages over the table-based approach, but this analysis does not completely reveal the impact on processor execution time and predictability. For deeper insights, we tested our solutions with benchmarks from the TACLEbench suite [8]. The results in Figure 1 show the worsening factor for the execution time when the trie-based relocation mechanism is enabled compared to execution times without relocation (yellow line) and with table-based relocation mechanisms (red line) by varying the complexity of the platform, namely *Small* (S), *Medium Static* (MS), *Medium Dynamic* (MD), and *Large* (L). MD and L platforms use off-chip dynamic memory, requiring multiple clock cycles for CPU requests, while S and MS utilize on-chip static memory with a one-clock cycle response. Furthermore, for each run, in order to retrieve the WCET, we impose each memory access to walk the tree up to the last level. Due to space constraints, only a portion of the normalized results is displayed, while full results are available at [7]. Results indicate a considerable reduction in execution time degradation for platforms with off-chip dynamic memory (MD: 11.1%-19.7%, L: 16.6%-29.8%) compared to those with static on-chip memory only (S: 65.8%-94.3%, MS: 113.3%-155.0%) due to the processor's pipelined nature. Given the significant degradation in the first two platforms, we recommend a table-based approach for small platforms with on-chip static memory and few memory regions. On the other hand, the GRT-based approach appears viable for enabling memory virtualization in medium- to large-size microcontrollers.

TABLE I
Area, Power, and Frequency results for cv32e41s-based SoC baseline, with different PMP, LPVM-TAB, and LPVM-GRT configurations. Data refer to the Nexys A7-50T board.

	Look Up Tables	**Flip Flops**	**Power Usage**	**Working Frequency**
Baseline (no PMP)	7223	5853	201 mW	$\sim 48Mhz$
8 PMP entries	9176	6232	214 mW	$\sim 40Mhz$
32 PMP entries	15323	7136	228 mW	$\sim 40Mhz$
64 PMP entries	27651	8343	273 mW	$\sim 40Mhz$
8 LPVM-TAB entries	10071	6488	220 mW	$\sim 35Mhz$
32 LPVM-TAB entries	19517	8127	251 mW	$\sim 35Mhz$
64 LPVM-TAB entries	32072	10285	298 mW	$\sim 35Mhz$
LPVM-GRT	10193	6592	211 mW	$\sim 40Mhz$

IV. Conclusions

This paper proposed a twofold solution to support lightweight and predictable memory virtualization in medium-size microcontrollers. Through an extensive analysis of performance overhead, energy consumption, and space occupation, we have shown that an ad-hoc hardware-based implementation can enable virtual addresses on microcontrollers without losses in predictability and with a reasonable overhead. We have also demonstrated that different approaches can achieve better results on different classes of microcontrollers.

References

[1] D. Oliveira, M. Costa, S. Pinto, and T. Gomes, "The future of low-end motes in the Internet of Things: A prospective paper," *Electronics*, vol. 9, no. 1, p. 111, 2020.

[2] N. Klingensmith and S. Banerjee, "Using virtualized task isolation to improve responsiveness in mobile and IoT software," in *Proceedings of the International Conference on Internet of Things Design and Implementation*, 2019, pp. 160–171.

[3] D. Oliveira, T. Gomes, and S. Pinto, "uTango: an open-source TEE for IoT devices," *IEEE Access*, vol. 10, pp. 23 913–23 930, 2022.

[4] R. Pan, G. Peach, Y. Ren, and G. Parmer, "Predictable virtualization on memory protection unit-based microcontrollers," in *2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 2018, pp. 62–74.

[5] S. Mercogliano and D. Ottaviano, "cv32e41s," https://github.com/Granp4sso/cv32e41s, 2023.

[6] ——, "cv32e41s SoC Environment," https://github.com/Granp4sso/cv32e41s_SoC_env, 2023.

[7] D. Ottaviano and S. Mercogliano, "XPMP experiments repository," https://github.com/DanieleOttaviano/XPMP_experiments, 2023.

[8] H. Falk, S. Altmeyer, P. Hellinckx, B. Lisper, W. Puffitsch, C. Rochange, M. Schoeberl, R. B. Sørensen, P. Wägemann, and S. Wegener, "TACLeBench: A benchmark collection to support worst-case execution time research," in *16th International Workshop on Worst-Case Execution Time Analysis*, 2016.