# X86_64 vs Aarch64 Performance Validation with COTSon

Marco Procaccini[1], Roberto Giorgi,

*Department of Information Engineering and Mathematics, University of Siena, Via Roma 56, 53100 Siena, Italy*

**ABSTRACT**

**In this study, we provide a set of architectural parameters for the HPLabs COTSon simulator that can be used to model existing processors, such as the Intel i7700 (X86_64 architecture) and the ARM A53 (Aarch64 architecture). We carry out an initial validation, by comparing the execution time while performing the weak scaling of the architecture, in the case of two common benchmarks. We use the Recursive Fibonacci and Matrix Multiplication benchmarks for simplicity. By using the simulator, we can then further study the sensitivity of the architecture and derive which features may matter most to evaluate the performance. Our goal here is to verify that the COTSon simulator can be used to model both the X86_64 and Aarch64 architectures. Based on this validation study, we have the possibility to analyze the bottlenecks and desirable microarchitectural features of modern architectures.**

KEYWORDS: Performance Evaluation, Computer Architecture, Computer Simulation

## 1 Architectures Simulation

Architecture simulation is an important tool to understand the relation between the architectural features and the performance of the running application. It enables a fast and cost-effective design space exploration, by allowing to simulate new architectural improvements without requiring RTL abstraction level. However, the use of a generic simulator may not help to reach the goal, since the simulation model may be too distant from the actual architecture [CAD09, ZYGP09].

We used the HP-Labs COTSon simulator, since it separates conveniently and neatly the functional emulation from the timing simulation: the *functional emulator* is based on the AMD SimNow virtual machine and it provides a complete list of events (i.e., instructions, interrupts, etc.) to be processed afterwards in the *timing simulator* [AFF+09]. The functional emulator part executes the instructions according to a particular Instruction Set Architecture (X86_64 for the SimNow) and provides all the information about an instruction like the instruction address, register, type, results, and memory address. The timing simulator simulates the timing behavior of the instruction for a given hardware implementation. This approach is called *functional-directed*, which gives us the possibility to have a cycle approximate accuracy of the behavior of the X86_64 processor. We also wanted to have a confir-

[1]E-mail: {procaccini,giorgi}@dii.unisi.it

mation of the findings of previous studies, which discovered that the performance of ARM and X86 does not depend on the specific ISA that is used, but rather it depends on the type and quantity of resources that are included in the platform (e.g., cache, TLB, branch predictor) [BMS13].

# 2 Evaluation and Results

In order to validate the COTSon simulation environment with the X86_64 architecture, we selected two benchmarks: Recursive Fibonacci (RFIB) and Matrix Multiplication (MM). RFIB is the recursive calculation of the Fibonacci number of a given $n$. To avoid too small threads, the recursion stops as $n$ becomes smaller than a certain threshold $t$. MM performs a matrix multiplication by a triple nested loop on square matrix, where the matrix is partitioned in sub-matrices, or blocks.

Table 1: Micro-Architecture Configurations

| Parameter | COTSon | X86_64 | Aarch64 |
|---|---|---|---|
| Core | 3 GHz, in-order super-scalar | Core Intel i7700 Kaby lake at 4.2 GHz out-of-order | Cortex A53 ARMv8-A at 1.5 GHz, in-order superscalar |
| Branch Predictor | two-level (history length=14bits, pattern-history table=16 KiB, 8-cycle miss prediction penalty) | two-level (history length=18bits, 17-cycles miss prediction penalty) | global (3072-entries pattern history table) 20-cycles miss prediction penalty |
| L1 Cache | Private I-cache 32KiB, private D-cache 32KiB, 8-ways, 4-cycles latency | Private I-cache 32KiB 8-ways, private D-cache 32KiB 8-ways 4-cycles latency | Private I-cache 32KiB 2-way, private D-cache 32KiB 4-ways 3-cycles latency |
| L2 Cache | Private D-Cache 256KiB, 4-ways, 10-cycles latency | Private D-cache 256KiB 4-ways 10-cycles latency | Shared D-cache 1MiB 16-ways 15-cycles latency |
| L3 Cache | Shared 8MiB, 16-ways, 35-cycles latency | Shared D-cache 8MiB 16-ways 35-cycles latency | no |
| I-L1-TLB | 4KiB pages, 64 entries, 4-ways 1-cycle latency | 4KiB pages, 64 entries, 4-ways 1-cycle latency | 4KiB pages, 10 entries fully-associative 2-cycles latency |
| D-L1-TLB | 4KiB pages, 64 entries, 8-ways, 1-cycle latency | 4KiB pages, 64 entries, 8-ways 1-cycle latency | 4KiB pages, 10 entries fully-associative 2-cycles latency |
| L2-TLB | 2MiB, 32 entries, 2-ways | 2MiB, 32 entries 4-ways | 4KiB, 512 entries, 4-ways |

## 2.1 Experiments Setup

In order to facilitate the experiments and speed up the Design Space Exploration, we developed a set of tools (named "MYDSE"), through which is possible to easily configure the COTSon framework, manage experiments and collect results by reducing the experimentation time from days/week to hours/minutes [GKP19]. Thanks to the MYDSE toolset it is possible to define the complete micro-architecture of the processor by using a higher level description and also indicate the desired values of the architectural parameters to be explored. The tools automatically generates the experiment points for the COTSon simulator, distributes and manages the parallel simulation of the points on as many simulation

hosts as possible (e.g., in a cluster system). The MYDSE configuration file follows a simple <*key*>=<*value*> syntax: if multiple values are specified, then the design space covers all the indicated values. Moreover, it is possible to define other higher-level parameters of each experiments (e.g.,OS image, applications and their inputs, the standard library to be used).

Initially, we validated the execution of the COTSon simulator against the Intel i7700 Core by using the architecture parameters reported in the table 1. About the Aarch64 architecture, we relied on the AXIOM-Board hardware specification, which is a single computer board developed during the AXIOM-Project at the beginning of the 2017 [T+17,Gio,A+15,GBG+16, Gio17]. To analyze the sensitivity to the input, we consider the weak scaling of the execution time: the input size is chosen in such a way that any successive value generates a double number of operations. For example, in the MM benchmark, the number of operations vary as O($N^3$), where $n$ is the size of the square matrix. Therefore, we have increased the size of the matrix by a factor of $\sqrt[3]{2}$.

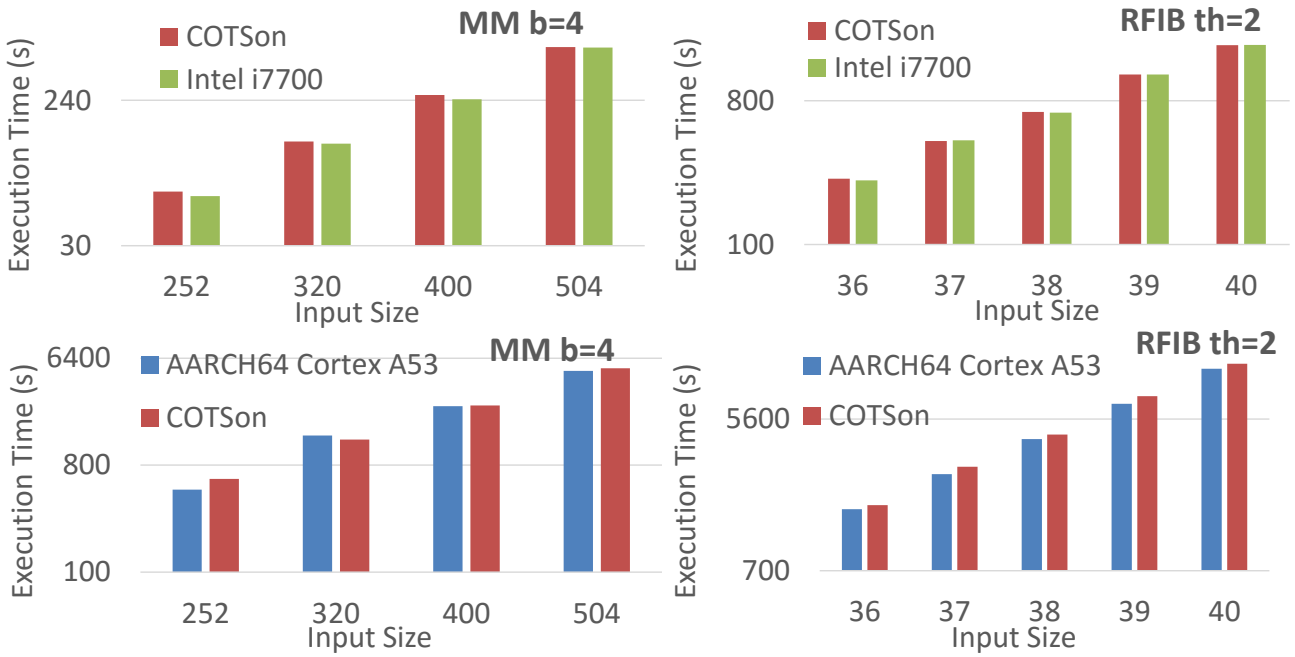


Figure 1: Performance validation of the benchmarks Recursive Fibonacci (RFIB) with 2 as threshold (th) and Matrix Multiply with 4 as block size (b) between the COTSon simulator, the Intel i7700 (top) and the ARM Cortex A53 (bottom) architectures.

## 2.2 Results

The validation of the results is based on the comparison of the execution time of the sequential execution of the MM and RFIB benchmarks on the COTSon simulator with the two target architectures: Intel i7700 for the X86_64 architecture and ARM Cortex A53 for the Aarch64. As depicted in Figure 1, the COTSon simulation results follows quite closely the weak scaling that is performed on the Intel i7700 and on the ARM Cortex A53 architectures, for both the MM and RFIB benchmarks and for almost all the input range, once a "effective frequency" correction is done.

# 3 Conclusion

In this paper, we illustrated how the COTSon simulation framework is capable to simulate both X86_64 and Aarch64 architectures and obtain results very close to the real hardware, by using the MM and RFIB benchmarks. Thanks to this validation, the COTSon simulator gives us the possibility to analyze in depth the behavior of such architectures during the execution of applications along with the related operating system activity Currently, we are working on expanding the benchmark set (e.g, Cholesky factorization, radix sort, FFT, BFS). with the aim to have a more consolidated validation of our initial findings.

# References


[A+15] C. Alvarez et al. The AXIOM software layers. In *IEEE Proc. 18th EUROMICRO-DSD*, pages 117–124, Aug. 2015.

[AFF+09] Eduardo Argollo, Ayose Falcón, Paolo Faraboschi, Matteo Monchiero, and Daniel Ortega. COTSon: infrastructure for full system simulation. *SIGOPS Oper. Syst. Rev.*, 43(1):52–61, 2009.

[BMS13] Emily Blem, Jaikrishnan Menon, and Karthikeyan Sankaralingam. Power struggles: Revisiting the risc vs. cisc debate on contemporary arm and x86 architectures. In *High Performance Computer Architecture (HPCA2013), 2013 IEEE 19th International Symposium on*, pages 1–12. IEEE, 2013.

[CAD09] Jianwei Chen, Murali Annavaram, and Michel Dubois. Slacksim: a platform for parallel simulations of cmps on cmps. *ACM SIGARCH Computer Architecture News*, 37(2):20–29, 2009.

[GBG+16] R. Giorgi, N. Bettin, P. Gai, X. Martorell, and A. Rizzo. *AXIOM: A Flexible Platform for the Smart Home*, pages 57–74. Springer, 2016.

[Gio] R. Giorgi. Scalable embedded systems: Towards the convergence of high-performance and embedded computing. In *EUC 2015*.

[Gio17] R. Giorgi. AXIOM: A 64-bit reconfigurable hardware/software platform for scalable embedded computing. In *IEEE 6th Mediterranean Conf. on Embedded Computing (MECO)*, pages 113–116, June 2017.

[GKP19] R. Giorgi, F. Khalili, and M. Procaccini. A design space exploration tool set for future 1k-core high-performance computers. In *ACM RAPIDO Workshop*, pages 1–6, 2019.

[T+17] Theodoropoulos et al. The axiom platform for next-generation cyber physical systems. *ELSEVIER Microprocessors and Microsystems*, pages 540–555, 2017.

[ZYGP09] Hui Zeng, Matt Yourst, Kanad Ghose, and Dmitry Ponomarev. Mptlsim: a simulator for x86 multicore processors. In *2009 46th ACM/IEEE Design Automation Conference*, pages 226–231. IEEE, 2009.