A Design Space Exploration Tool Set for Future 1K-core High-Performance Computers

Roberto Giorgi
Department of Information Engineering and Mathematics, University of Siena
giorgi@diism.unisi.it

Marco Procaccini
Department of Information Engineering and Mathematics, University of Siena
procaccini@diism.unisi.it

Farnam Khalili
Department of Information Engineering and Mathematics, University of Siena
Department of Information Engineering, University of Florence
khalili@diism.unisi.it

ABSTRACT

Given the constantly growing complexity of multi-core architectures, Design Space Exploration (DSE) tools play an important role to evaluate different design options. In this paper, we present a DSE toolset targeting massively parallelized HW/SW architectures with a high degree of flexibility in order to successfully simulate multi-core-multi-node platforms. Our DSE tools provide a rapid and simple-to-use work-flow to easily retrieve and analyze the key metrics and eventually evaluate the design. We examine the DSE toolset and methodology while performing several simulations of a general purpose 1K-core architecture and evaluate not only standard metrics like the L2 cache miss rates, but also operating system activity and its impact. We leverage the knowledge gained in our methodology to develop and evaluate a novel dataflow execution model named "DataFlow-Threads" (DF-Threads). We validated the outcomes of the simulator against an equivalent FPGA-based design.

Keywords

Design Space Exploration; Simulation; Performance Analysis and Design; Multi-Core

1. INTRODUCTION

Recently, in order to match the performance request with the design requirement, researchers more than ever rely on the heterogeneous and domain-specific architectures [24]. Future architectures may be composed of thousands of tightly coupled cores (CPUs and GPUs), residing nearby accelerators, and become more complex than current ones [3]. Moreover, modern embedded systems are increasingly based on heterogeneous Multi-Processor SoC (MP-SoC) architectures. To cope with the design complexities of such architectures, Design Space Exploration (DSE) is an important portion of the design flow. DSE and its automation is a significant part of modern performance evaluation and estimation methodologies in order to reduce the design complexity, time-to-market, and find the best compromise among design constraints in respect to the application. Computer designers, therefore, rely on simulations as part of the DSE work-flow to perform early-stage design assessments in order to save time and costs. A simulator not only ensures the functional correctness but also may provide an accurate timing information.

We started to develop DSE tools within the TERAFLUX project [7] in order to evaluate a complex architecture with e.g., 1000 general purpose cores and running a full OS like Linux. In this paper, we present a set of DSE tools and experiments that we made by the COTSon simulator during the AXIOM project [1, 8, 9, 12, 29, 30] and the TERAFLUX project. Our DSE tools permit us to model features of architectures which are not yet available on the market in a rapid way, we were able to easily evaluate several execution models such as Cilk, OpenMPI, and a novel execution model like DataFlow-Threads (DF-Threads) [14, 18, 31] either for embedded [11] or for multi-core [10] systems.

The remainder of this paper is organized as follows: in Section, 2 we discuss the important challenges and issues regarding the exploration of an architecture; in Section, 3 we outline our DSE toolset and present our simulation platform; in Section, 4 we evaluate our DSE through different types of test case and discuss the evaluation results; in Section, 5 we highlight some related DSEs and scholar evaluation platforms, and finally, we conclude the paper in Section 6.

2. PROBLEM STATEMENT

Evaluation of a multi-core architecture even at the prototype stage is quite challenging, time-consuming. Moreover, it is not always possible to get the "perfect" setup, and hardware prototyping possibly imposes several limitations. In this section, we briefly highlight these limitations and show the importance of simulator (like COTSon [2]) when it comes to assess and retrieve key metrics of a high-performance computer, e.g., 1000 general purpose cores.

We report in Table 1 different approaches like using a physical Cluster, FPGA, and Simulator to evaluate and do research related to 1000-core computing system (Information revised from the RAMP project [6]).

Given a cluster at a scale of, e.g., 1000 general purpose

RAPIDO '19, January 21–23, 2019, Valencia, Spain

ISBN 978-1-4503-6260-3/19/01…$15.00

DOI: https://doi.org/10.1145/3300189.3300195

Table 1: comparison different approaches for evaluating large computing system. the grade points are scaled between 0 and 5 (grade of 5+ implies the superiority); GPA: Grade Point Average

	Cluster	FPGA	Simulator
Scalability	5	5)	5
Cost	3	4(€0.1-0.2M)	5+(€0.01M)
Observability	3	5+	5+
Reproducibility	2	5+	5+
Reconfigurability	3	5+	5+
Credibility	5+	3.5 to 4.5	3
Development time	4	3	5+
Performance (clock)	5(3GHz)	1(GHz)	3
Modifiable	0	4	5
GPA	3.38	3.2 to 3.7	4.8

cores, the best possible solution to connect the nodes (each node may consist of several cores) could be through InfiniBand interconnect. The main disadvantage of a physical cluster is its high cost and its inflexibility towards modification of architecture as well as poor extensibility in order to reconfigure the Instruction Set Architecture (ISA).

For FPGA, the hardware and software must be configured and set up, which invokes considerable time and also effort.

The simulators might not show satisfying credibility, but as they evolve, their credibility also improves. The main problem of simulators is their less performance in comparison with a physical cluster. However, we consider the simulators very useful for approaching a reasonable level of accuracy, scalability and simulation speed. Importantly, COTSon [2] offers a flexible simulation environment which made possible to design our DSE, and add new instructions [19, 25, 26] in order to evaluate a multi-core architecture, with a dataflow execution model like DF-Threads.

We use COTSon to offer a flexible DSE toolset that easily can adopt new hardware/software platforms, and support scalability for a multi-node architecture. For instance, in order to address the challenges of a 1k-core architecture, we should be able to have a full-system simulation including Operating System (OS), application benchmarks, a memory hierarchy and all peripherals as well.

3. SIMULATION FRAMEWORK

Our proposed framework allows us to modify system parameters such as the number of cores and number of simulated instances (nodes), which are running in parallel on completely independent hosts. This framework is also able to run Shared Memory application like OpenMP as well.

The proposed simulation framework relies on HP-Labs COTSon simulation environment and on a set of customized tools that are intended to easily setup the experimental environment, run experiments, extract and analyze the results.

3.1 The COTSon simulator

COTSon simulator [2] is based on the so-called "functional-directed" approach, where the functional execution is decoupled from the timing feedback. COTSon simulator uses the AMD SimNow virtualizer tool, which is proposed by AMD in order to test and develop their processors and platforms. COTSon executes its functional model into the SimNow virtual machine, running and testing the execution of the functional model. A custom interface is provided, in order to facilitate the exchange of the data between COTSon and the internal state of SimNow. As can be seen in Figure 1, COTSon architecture is made of three main components:

1) FUNCTIONAL MODELS: it contains the instances of the SimNow virtualizer, which executes the functional model based on a configurable x86-64 dynamically translating instruction-level platform simulator.
In fact, we were able to customize the x86-64 instruction set of SimNow in order to introduce new instructions for the implementation of the DF-Threads execution model [19].

2) TIMING MODELS: it implements simulation acceleration techniques, such as dynamic sampling, the tracing, profiling and statistics collection. Through the specification of a timing model for a given component (i.e., L1 cache, networking), we can model different behaviors. The timing models are decoupled from the functional execution of SimNow, allowing us the flexibility to model different types of architectural feature.

3) SCRIPTING GLUE: the final part is related to the scripts used to boot/resume/stop each virtual machine, the setup of the parallel simulation instances of SimNow and the time synchronization among all the virtual machines.

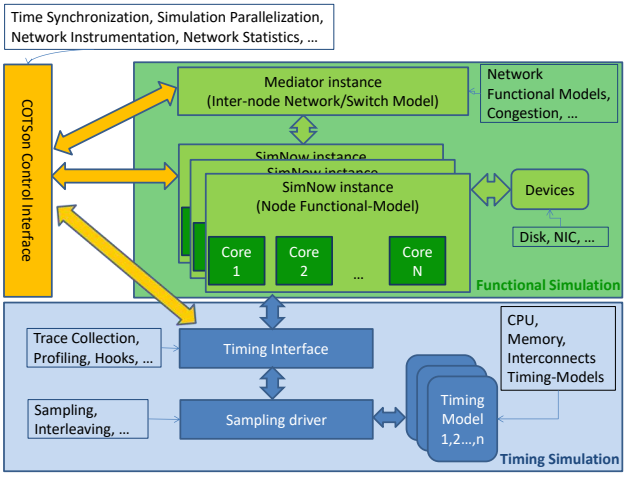

Figure 1: The COTSon simulation framework architecture.

3.2 Design Space Exploration Tools

3.2.1 *The tool set*

In order to guarantee a proper scientific methodology for experimentation, we developed the Design Space and Exploration Tools (DSE Tools) through which is possible to easily set up the COTSon simulation environment, extract and analyze the results of the experiments.
The normal toolflow is to follow the next eight steps.

i) GENIMAGE: the SimNow virtual machine needs a hard-drive image, which contains the Operative System to run. The GENIMAGE tool has the goal to create a customized version of a Linux distribution by other tools like VMBuilder and Debootstrap [5].

ii) ADD-IMAGE: this tool is preparatory to the GENIMAGE tool and it serves to load a given hard-drive image and the related virtual machine snapshot.

iii) BOOTSTRAP: it is a preparatory tool to prepare a user-based COTSon installation. This tool aims at solving the dependencies needed by the toolset in the host machine, avoiding the manual installation of them. It requires root permission once per machine. Moreover, the tool tunes some kernel parameters such as the number of host memory pages needed by SimNow.

iv) CONFIGURE: it enables multiple users on the host machine to run a configuration of their own simulation setup without the need of system administrator intervention. In fact, the tool runs completely in user-space, without the need of root permissions.

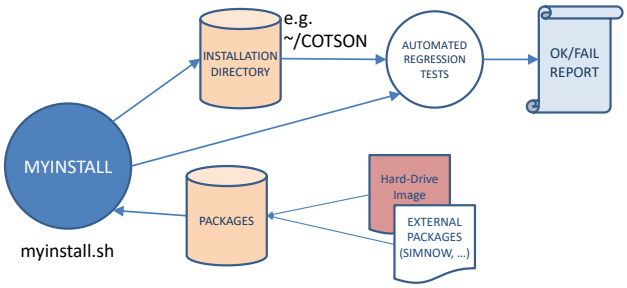

Figure 2: TOOLFLOW for the MYINSTALL tool. MYINSTALL prepares the whole environment for simulation-based Design Space Exploration with a single command. The Configuration File specifies which additional tool or tool-options should be used (e.g., non-public tools, or tools under NDA).

v) MYINSTALL: the purpose of this tool is to facilitate the installation process of the simulator and the hard-disk image which contains the pre-selected Operative System that will runs into the SimNow machine (see Figure 2). Moreover, MYINSTALL allows the choice of the simulation software version, in order to enable more versions of the simulator to co-exist for regression test. Finally, the tool performs several regression tests at the end of the installation phase, in order to verify the software is correctly patched, compiled and installed. The entire process is completely automatic and it can be easily repeated on multiple and parallel simulation hosts.

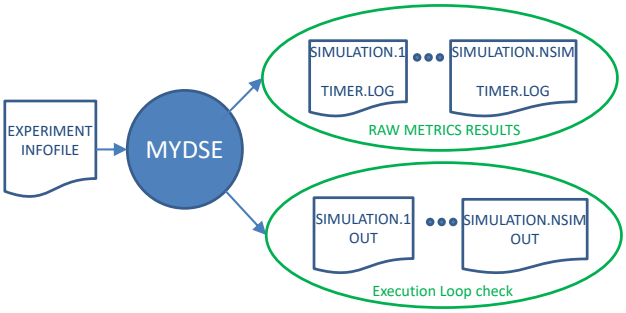

Figure 3: TOOLFLOW for the MYDSE tool. The experiment INFOFILE defines the simulation points and output files generated during each simulation are organized in order to facilitate their accessing and parsing by the other tools.

vi) MYDSE: we found a substantial need to implement a specific tool, which is able to easily catch possible failures or errors and, mostly, the automatic management of experiments in case of a large number of design points to be explored. As depicted in Figure 3, MYDSE relies on a small configuration file, named "INFOFILE", which is described in more details in the subsection 3.2.2.
Also, the tool is able to spread the simulation among multiple hosts and,if necessary, it can use the same binary with different GLIBC library version across the hosts. This allows us to use different operating systems, with a different version of the GLIBC library, in different guests. During the experiment, MYDSE controls the simulation loop, collecting in an ordered way the several files from each simulation point. Statistics based on user formulas are printed out at the end of each simulation, in order to provide an overall evaluation of the results. In case of failures, the tool kills the failed simulation and the related processes after a certain time, trying the re-execution of the failed simulation automatically. The timeout is derived by a simulation estimation model (i.e., proportional to the number of nodes and cores of the system).

vii) GTCOLLECT: once a campaign of experiments has been concluded, we need to collect, analyze and plot results in a simple way. In this perspective, we can extract data from experiments with the GTCOLLECT tool (GT stand for Graphic Table Collect), which prints out the collected data, based on the "INFOFILE" information and a "LAYOUT" text files where the user can specify the relevant output metrics to select. Furthermore, some additional calculations are performed on the data, such as the Coefficient of Variation, in order to analyze the correctness of the results.
With the GTCOLLECT tool, we can perform a complete analysis of the raw data produced by the MYDSE.

viii) GTGRAPH: once the results are collected in the GTCOLLECT format, the GTGRAPH tool can produce a graphical view of the data, like Figure 7,8,9.

Additionally, COTSon permits a connection with McPAT [2] to analyze the power consumption and the temperature of an experiment.

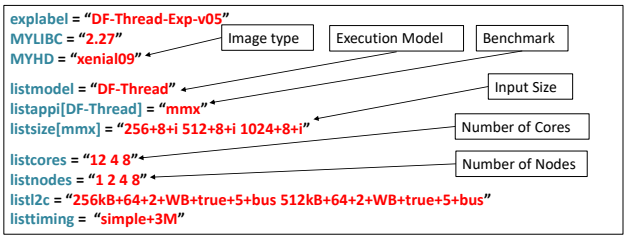

Figure 4: INFOFILE example, which describes a Design Space Exploration experiment.

3.2.2 *Experiment Description*

In this section, we want to introduce the experiment description file, named "INFOFILE", which makes the DSE easy to manage a clear identification of the Design Space. As depicted in Figure 4, we can describe the experiment

through a simple file that uses "Bash syntax": <variable> = "<string>". Each DSE variable is defined with the prefix "list", while "<string>" represents a set of value where elements can be separated with the character "+" (i.e. 256+8+i represent matrix size = 256, block size = 8 and matrix element type = integer). Moreover, we can define multiple configurations of the architecture, in order to find the best organization for a given application.

As we can see in Figure 5, each architecture configuration is composed of high-level blocks and we can specify the organization using the bash syntax of the INFOFILE. The names identify a block and the "main" block is the root of the configuration. The "-" character specifies the link between two blocks and the "+" character separates the different links. The "." character separates a first part which represents a single instance of the implicitly defined architectural blocks and a second part which represents multiple instances of the implicitly defined architectural blocks. For example, in the "listarch" variable of Figure 5 the part *.ic-cpu+busT-t2+l2-ic+l2-dc+t2-it+t2-dt* will be instantiated C times (where C is the number of Cores). Also, there is the possibility to insert a tracing module between two blocks of the architecture, snooping the data and the information exchanged among the blocks ("trace" block in Figure 5).

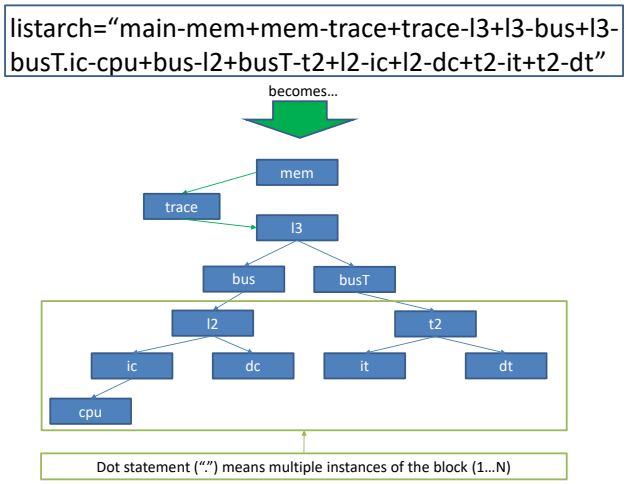

Figure 5: Architecture specification using the INFOFILE syntax, which it is used by the MYDSE tool to configure the COTSon simulation framework.

3.2.3 *Customizable devices*

The SimNow tool allows us to personalize the device architecture of the virtual machine that we want to run, by selecting a device from a list and placing it into the device tree. Moreover, the device list could be extended in order to introduce new customizable devices. As depicted in Figure 6, we were able to introduce a new PCI device, named XNIC, in order to emulate the behavior of a hardware accelerator device (e.g. FPGA) and at the bottom we show the kernel boot of such device.

4. EXAMPLE OF EVALUATIONS

In this section, we want to show some experiments that we made on the COTSon simulator during the AXIOM and the TERAFLUX European projects. The evaluations that we

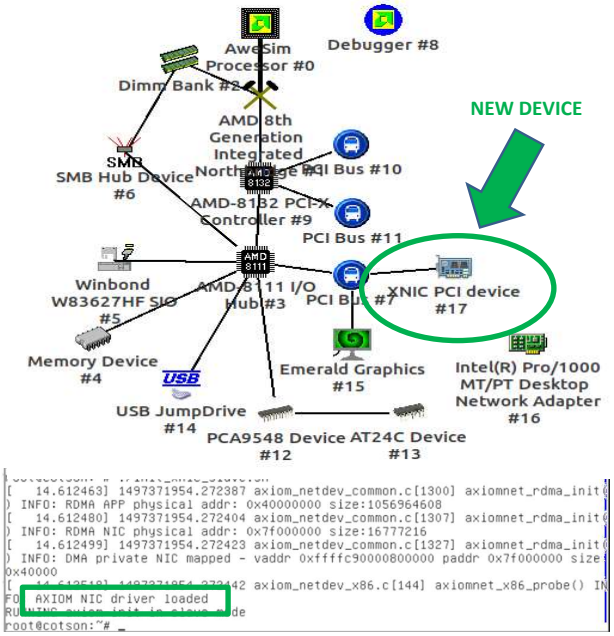

Figure 6: Architecture configuration of a SimNow virtual machine with the additional custom PCI device named XNIC. In the bottom part, the new device is loaded into SimNow.

show in this section are based on the DF-Threads execution model, focusing on execution time, OS impact and cache usage (we don't show results about temperature and power consumption because it is out of the scope of this paper).

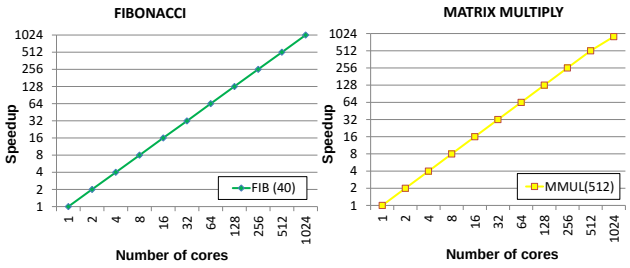

Figure 7: Mutli-node, multi-core simulation up to 1024 cores using Fibonacci, with input set to 40, and Matrix Multiply with matrix size 512x512 based on DF-Threads execution model.

TEST CASE 1: as can be seen in Figure 7, we tested the DF-Threads execution model through two well-known benchmarks like Fibonacci (input n=40) and Matrix Multiply (with a matrix size of 512). We are able to simulate different nodes/cores configuration, from 1 to 1024 cores. Each node is configured to have from 1 to 32 cores and the node range is from 1 to 32.

TEST CASE 2: thanks to our DSE tools, we were able to study the scaling factor while varying the input size and the number of nodes, demonstrating that the DF-Threads execution model has good scaling in every tested Operating System. We tested the performance of different OS distribution with a different size of the Matrix Multiplication benchmark. As depicted in Figure 8, the performance vary as the OS distribu-

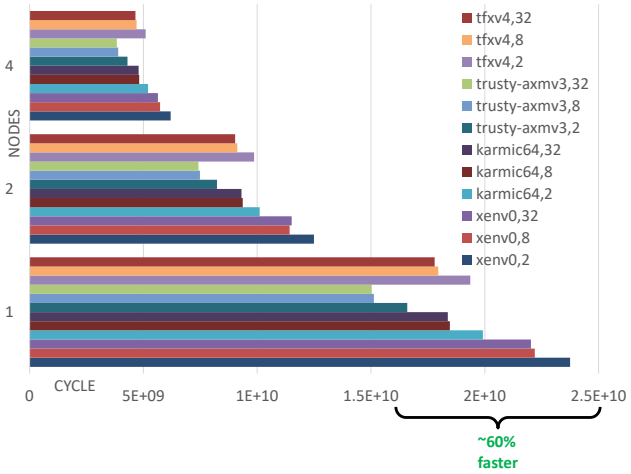

Figure 8: Study of the variation in performance (execution cycles) in the different OS distribution as we vary the L2 Cache size from 2KiB to 32KiB. The benchmark is Matrix Multiply with matrix size of 512 . Kernel activity is responsible from 10% to 50% on the execution time.

tion is changed and the trusty-axmv3 seems to be the best among the tested ones in most cases. According to [11], the Kernel activity has a huge impact on the performance of the different operating system, varying from 10% to 50%.

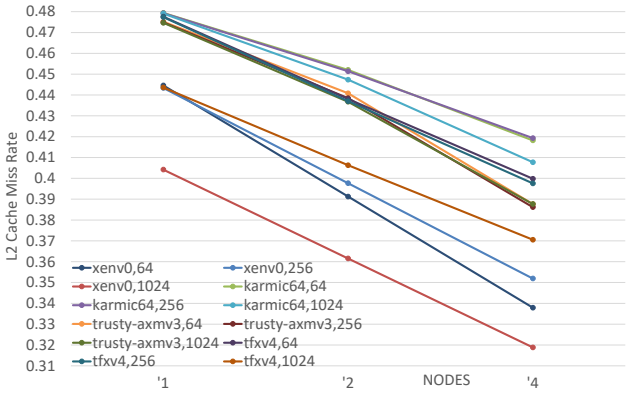

Figure 9: Example metric (L2 Cache Miss Rate). We vary the number of nodes (from 1 to 4), the OS distribution and the input size of the Matrix Multiply benchmark (matrix size = 256,512,and 1024).

TEST CASE 3: in this test we want to exploit the output traces of the simulator in order to analyze different aspects of the experiments. As we can see in Figure 9, we study the L2 cache miss rate behavior among the different OS distribution. We choose the input size at 512x512 in order to have enough parallelism and we vary the L2 cache size from 32 to 1024 KiB. The results show that the L2 cache size has a strong influence in the performance and therefore it may play a crucial role in the system. For example, we discovered that one of the optimization points of the execution model implementation should be the L2 cache usage.

VALIDATION TEST: Finally, we validated our results obtained through the utilization of the simulator, comparing the execution time (sequential execution) of the Matrix Multiplication Benchmark both in simulator and on a real board FPGA based (AXIOM-Board [9, 15, 17, 30]). As can be seen in Figure 10, the results of the simulator and the board are very close, confirming our performance predictions, despite some architectural differences.

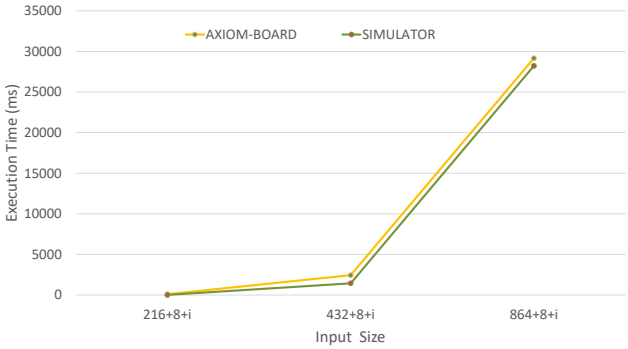

Figure 10: Preliminary comparison of the execution time between the simulator and the AXIOM-Board. We used the Matrix Multiply benchmark three different size: 216,432 and 864.

5. RELATED WORK

Several types of research have been recently carried out and focused on Design Space Exploration (DSE) and simulators like MULTICUBE [28], which proposed a multi-level simulation approach using an approximate analytic metamodel for MP-SoC architectures based on Artificial Neural Networks. An iterative multi-objective optimization technique for MP-SoC is proposed in [20], which is able to model the correlation of the multi-processor configurations with appropriate analytical functions. In order to accurately simulate more complex processor designs, conventionally, architectural simulators like Trace-Factory [16] and GEMS [21] have been adopted. GEMS provides a very detailed and accurate simulation by using a timing first approach to model timing. In our case, we relay on COTSon, which uses functional directed approach allowing simulation of several thousands full system cores. GRAPHITE [22] is a multi-core simulator designed to support more than 1000 cores at higher-level of abstraction using distribution workload techniques. These simulators lack efficient parallelization capabilities due to fine-grained synchronization difficulties, which limit the speedup. To mitigate these limitations, the user has to trade speed with accuracy [22, 23].

Sniper [4] is a parallel and scalable multi-core simulator which compromises between accurate high-abstraction analytical models and fast parallel simulation, and covers a larger portion of the hardware design space. Even though it covers operating system runtime simulations, they simulate only up to 16-cores, while in our case we successfully are able to simulate a 1000 general purpose multi-core architecture [10]. However, some of these approaches miss portability and require considerable time to be ported to other frameworks.

In [25] authors propose a full stack simulation system targeting heterogeneous kilo-core architectures which includes a customized extended version of x86/64 ISA [19,26] to sup-

port DataFlow-Threads (DF-Threads) execution model [14]. In [13,27] COTSon [2] simulator has been leveraged in order to provide Distributed Scheduler to support many-node architectures, and significantly improve scalability and power estimation.

6. CONCLUSION

We have presented a set of tools for the Design Space Exploration, based on the COTSon simulator framework, with the aim of supporting large set of experiments of a multi-node multi-core platform with full OS execution (e.g., 1000 general purpose processors and real OS activity). Thanks to our tools, we setup the simulation environment in less than ten minutes, including several regression tests, saving hours in comparison with manual installation. We can run several experiments by using a simple configuration file, handle possible failures or errors during the simulations. Finally, results are automatically collected and presented in the desired graphical view.

We showed several test cases and a validation test against a FPGA platform. The results permitted us to derive information early in the design process.

The DSE tools that we presented in this paper were massively used during two European projects (TERAFLUX and AXIOM), facilitating the exploration of a large design space and the test of a new execution model (DF-Threads).

In the future, we want to exploit machine learning techniques to better select design points and in order to improve the statistical characterization of the collected data.

7. ACKNOWLEDGMENT

The authors would like to thank Stefano Viola of SECO (s.r.l) for his support in developing the XNIC PCI device, The European Commission under the AXIOM H2020 project (id. 645496), TERAFLUX (id. 249013), and HiPEAC (id. 779656), and importantly, the anonymous reviewers for their helpful comments.

8. REFERENCES

[1] C. Alvarez et al. The AXIOM software layers. *ELSEVIER Microprocessors and Microsystems*, 47, Part B:262–277, 2016.

[2] E. Argollo, A. Falcón, P. Faraboschi, M. Monchiero, and D. Ortega. COTSon: infrastructure for full system simulation. *SIGOPS Oper. Syst. Rev.*, 43(1):52–61, 2009.

[3] S. Borkar and A. A. Chien. The future of microprocessors. *Communications of the ACM*, 54(5):67–77, 2011.

[4] T. E. Carlson, W. Heirman, and L. Eeckhout. Sniper: exploring the level of abstraction for scalable and accurate parallel multi-core simulation. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, page 52. ACM, 2011.

[5] Debootstrap website. https://wiki.debian.org/Debootstrap.

[6] G. Gibeling, A. Schultz, and K. Asanovic. The ramp architecture & description language. In *2nd Workshop on Architecture Research using FPGA Platforms*, 2006.

[7] R. Giorgi. Teraflux: Exploiting dataflow parallelism in teradevices. In *ACM Computing Frontiers*, pages 303–304, Cagliari, Italy, May 2012.

[8] R. Giorgi. Scalable embedded systems: Towards the convergence of high-performance and embedded computing. In *Proc. 13th IEEE/IFIP Int.l Conf. on Embedded and Ubiquitous Computing (2015)*, pages 148–153, Oct. 2015.

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

[10] R. Giorgi. Exploring future many-core architectures: The TERAFLUX evaluation framework. In *Advances in Computers*, Advances in Computers, pages 33–72. Elsevier, 2017.

[11] R. Giorgi. Scalable embedded computing through reconfigurable hardware: comparing df-threads, cilk, OpenMPI and jump. *ELSEVIER Microprocessors and Microsystems*, 63:66–74, Aug. 2018.

[12] R. Giorgi, N. Bettin, P. Gai, X. Martorell, and A. Rizzo. *AXIOM: A Flexible Platform for the Smart Home*, chapter 3, pages 57–74. Springer Int.l Pub., Cham, 2016.

[13] R. Giorgi et al. TERAFLUX: Harnessing dataflow in next generation teradevices. *Microprocessors and Microsystems*, 38(8, Part B):976–990, 2014.

[14] R. Giorgi and P. Faraboschi. An introduction to DF-Threads and their execution model. In *IEEE MPP*, pages 60–65, Paris, France, Oct. 2014.

[15] R. Giorgi, F. Khalili, and M. Procaccini. Energy efficiency exploration on the zynq ultrascale+. In *The 30th International Conference on Microelectronics (ICM)*, December 2018.

[16] R. Giorgi, C. Prete, G. Prina, and L. Ricciardi. Trace factory: Generating workloads for trace-driven simulation of shared-bus multiprocessors. *IEEE Concurrency*, 5(4):54–68, Oct. 1997.

[17] R. Giorgi, M. Procaccini, and F. Khalili. AXIOM: A scalable, efficient and reconfigurable embedded platform. In *Design, Automation and Test in Europe, the european event for electronic system design and test (DATE)*, March 2019.

[18] R. Giorgi and A. Scionti. A scalable thread scheduling co-processor based on data-flow principles. *Future Generation Computer Systems*, 53:100–108, Dec. 2015.

[19] N. Ho, A. Portero, M. Solinas, A. Scionti, A. Mondelli, P. Faraboschi, and R. Giorgi. Simulating a multi-core x86-64 architecture with hardware isa extension supporting a data-flow execution model. In *IEEE Proc. AIMS-2014*, pages 264–269, Madrid, Spain, Nov. 2014.

[20] G. Mariani et al. A correlation-based design space exploration methodology for multi-processor systems-on-chip. In *DAC*, pages 120–125. ACM, 2010.

[21] M. M. Martin, D. J. Sorin, B. M. Beckmann, M. R. Marty, M. Xu, A. R. Alameldeen, K. E. Moore, M. D. Hill, and D. A. Wood. Multifacet's general execution-driven multiprocessor simulator (gems) toolset. *ACM SIGARCH Computer Architecture News*, 33(4):92–99, 2005.

[22] J. E. Miller, H. Kasture, G. Kurian, C. Gruenwald, N. Beckmann, C. Celio, J. Eastep, and A. Agarwal. Graphite: A distributed parallel simulator for multicores. In *High Performance Computer Architecture (HPCA), 2010 IEEE 16th International Symposium on*, pages 1–12. IEEE, 2010.

[23] S. S. Mukherjee, S. K. Reinhardt, B. Falsafi, M. Litzkow, M. D. Hill, D. A. Wood, S. Huss-Lederman, and J. R. Larus. Wisconsin wind tunnel ii: a fast, portable parallel architecture simulator. *IEEE*, 8(4):12–20, 2000.

[24] D. Patterson. 50 years of computer architecture: From the mainframe cpu to the domain-specific tpu and the open risc-v instruction set. In *Solid-State Circuits Conference-(ISSCC), 2018 IEEE International*, pages 27–31. IEEE, 2018.

[25] A. Portero et al. Simulating the future kilo-x86-64 core processors and their infrastructure. In *45th Annual Simulation Symp. (ANSS12)*, pages 62–67, Orlando, FL, Mar 2012.

[26] A. Portero, Z. Yu, and R. Giorgi. T-star (t*): An x86-64 isa extension to support thread execution on many cores. In *HiPEAC ACACES-2011*, pages 277–280, Fiuggi, Italy, July 2011. poster.

[27] A. Portero, Z. Yu, and R. Giorgi. Teraflux: Exploiting tera-device computing challenges. *ELSEVIER Procedia Computer Science*, 7:146–147, 2011. Proc. 2nd European Future Technologies Conf. and Exhibition 2011 (FET 11).

[28] C. Silvano et al. Multicube: Multi-objective design space exploration of multi-core architectures. In *IVLSI*, pages 488–493. IEEE, 2010.

[29] D. Theodoropoulos et al. The AXIOM project (agile, extensible, fast i/o module). In *IEEE Proc. 15th Int.l Conf. on Embedded Computer Systems: Architecture, MOdeling and Simulation*, pages 262–269, July 2015.

[30] D. Theodoropoulos et al. The AXIOM platform for next-generation cyber physical systems. *ELSEVIER Microprocessors and Microsystems*, pages 540–555, 2017.

[31] L. Verdoscia and R. Giorgi. A data-flow soft-core processor for accelerating scientific calculation on FPGAs. *Mathematical Problems in Engineering*, 2016(1):1–21, Apr. 2016. article ID 3190234.