Energy Efficiency Exploration on the ZYNQ Ultrascale+

Roberto Giorgi[1], Farnam Khalili[1],[2] and Marco Procaccini[1]

[1]Department of Information Engineering and Mathematics - University of Siena
[2]Department of Information Engineering - University of Florence
{giorgi,khalili,procaccini}@dii.unisi.it

***Abstract*—In the context of Cyber-Physical Systems (CPSs), Single Board Computers (SBCs) could provide adaptivity for various present and future applications, and permit scalability through clusters of SBCs while possibly save energy consumption. In this paper, we explore energy efficiency of a Zynq Ultrascale+ based board developed in the context of the AXIOM project. While an entire framework based on the Zynq Ultrascale+ is still in progress, the board is already available and capable of running a full Linux OS and it is possible to measure energy consumption. We demonstrate a possible architecture based on DataFlow-Threads (DF-Threads), a novel execution model, on the Zynq Ultrascale+ platform, in order to assess the energy efficiency of DF-Threads. We measured the power consumption, while the RAW and RDMA message types were transceived through board-to-board interconnects.**

***Index Terms*—Cyber-Physical Systems, Reconfigurable Systems, FPGA Programming, Thread Level Parallelization, Energy Evaluation**

I. Introduction

Embedded processing encircles relatively every application that exist in our lives which incorporates a wide assortment of hardware and software stages, from exceptionally basic ones to extremely complex ones, depending on the application [1]. Essentially, due to the steadily increasing interests in performance led by market, it is necessary to explore in high performance architectures for embedded computing.

In the context of the AXIOM project [2]–[5] it was realized that there is presently an extreme fragmentation of both devices and tools for embedded processing. Specifically, when more complicated functionalities are required, the entire system must be revised and a new tool-chain must be adopted. Thus, our goal was to permit the programmers to simply deploy the device with a possibly standard and open-source tool-chain based on a full Linux OS software distribution.

An important contribution of the AXIOM was the fabrication of an SBC board ("The AXIOM board") based on FPGA and embedded processor, e.g. Zynq Ultrascale+ [6] and its features are: i) a high speed reconfigurable interconnect for board-to-board communication; and ii) a user-friendly programmable environment, which allows us both to off-load partly program algorithms into accelerators (on programmable logic) and, at the same time, to distribute the computation workloads across boards via DataFlow-Threads, a novel execution model [7]–[9] and iii) the possibility of deploying an open-source tool-chain based upon easy to program concept like OmpSs [10], an OpenMP extension [11], [12]. However, scaling the performance of a computing system while retaining easy programmability is still on the headlines [13]. Additionally, tool-chains require to be integrated with suitable high level synthesis tools in order to have a higher control of the programmable logic [14], [15].

In addition, multi-processor system-on-chips (MPSoCs) are currently well-adopted, but the handling of many threads is still a source of a lot of inefficiencies. Their management must consider not only the order of execution and the quantum time per thread, but also the associations of allocating a thread on a given core. This aspect begins to be serious as the entire system grows in complexity, memory hierarchies, interconnects and distributed resources. In this paper, we propose to reduce such inefficiencies by using an efficient execution model named DF-Threads [7]–[9]. We explore energy efficiency of the AXIOM board when it comes to the distribution of DF-Threads among the AXIOM boards through well-defined and high speed board-to-board message types.

The contribution of this work is to extract the power metrics of the AXIOM board (based on the Zynq U+ FPGA) in respect to the distribution of DF-Threads through different type of board-to-board messages.

The remainder of this paper is structured as follows: In Section II we discuss some related work; in Section III we present the general architecture of the AXIOM board and its soft-IPs like the DF-Threads Scheduler (DFS) [8] on the Zynq Ultrascale+ platform; in Section V and IV, we show some experiments regarding the power consumption of when it comes to transceive high speed board-to-board messages and finally, we conclude the paper.

978-1-5386-8167-1/18/$31.00

II. Related Work

Data-flow execution models have been reviewed recently [7], [16], [17] as they promise an elegant way to effectively move data from one computational thread to another one [18]–[20]. Importantly, in such models, the computations can be mostly performed in a producer-consumer manner, while for mutable shared data, the memory model offered by Data-flow Threads (DF-Threads) [7] is enclosing Transactional Memory [21], which is a concurrency control mechanism analogous to database transactions for controlling access to shared memory by replacing locks with atomic execution units, so that user can focus on where atomicity is required.

In this context, the TERAFLUX project [22]–[25] accomplished such data-flow modality while extending to multiple nodes which are executing seamlessly through an appropriate memory semantic [7], [26]. In this semantics a compound of consumer-producer patterns [27], [28] and transactional memory [21], [29] allows a novel combination of data-flow paradigm and transactions in order to solve the consistency issues across nodes, where each node is supposed to be cache-coherent like in a classical multi-core. Additionally, such distributed systems could support fault-tolerance [30], [31], and in this context a data-flow thread may be re-executed without harming the computing program since the thread inputs are maintained before scheduling the corresponding thread.

Nowadays, FPGAs are widely used in prototyping Embedded Computers and more recently have become a significant component as the accelerators in the HPC and CPS field, since they undertake tasks with higher reliability, reconfigurability and energy efficiency [32]. Reconfigurable logics like FPGAs propose outstanding ways to boost specific functions, but need enough tools in order to moderate the complicated programming [32]–[34].

Considering reconfigurable computing platform based on FPGAs, many works have offered solutions to address the issues of dynamic allocation of tasks to general-purpose-multi-core processors [35], [36], or reconfigurable logic [37]. Nevertheless, these approaches have been effectively investigated only on single and multi-core super-scalar architectures.

Recent papers discuss further the details of DF-Threads hardware framework [3], [5], [38]–[40], the software layers [41], [42] and application use-cases [4], [43], [44].

III. DF-Thread Management for the Zynq Ultrascale+

Recently, there has been a huge exertion to move forward general programming models with thread management such as P-threads, Cilk, OpenMP. But in most of these models, synchronization and distribution of data between cores of different nodes need to be managed manually by programmers and imposes an extra effort [10].

Instead, DF-Threads execution model proposes better scalability by re-managing the distribution of threads based upon the data-flow paradigm [7], [45]. For this reason, a Distributed Thread Scheduler (DTS) is offered by [8], which is a hardware implementation of DF-Threads modality.

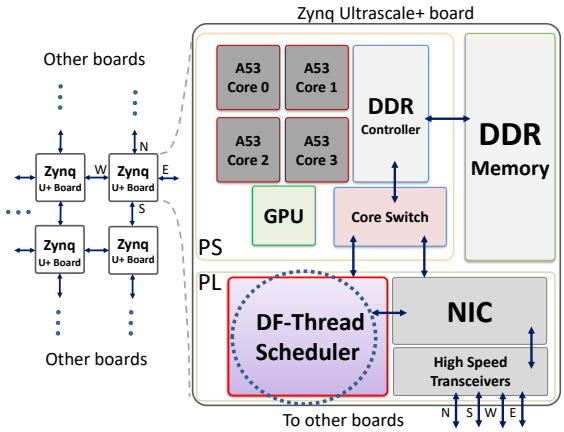

Fig. 1: Proposed scalable DF-Threads architecture mapped on the Zynq Ultrascale+ based board like AXIOM board (left side). The detailed bock designs for each node (board) are depicted at the right side. The proposed DF-Threads scheduler is completely designed on PL (dotted circle). U+: Ultrascale+, PS: Processing System, PL: Programmable Logic, NIC: Network Interface Card [46].

Figure 1 shows the block designs exploited to materialize and map the DF-Threads management on the PL part of the Zynq U+ platforms as an individual soft IP. DF-Threads Scheduler (DFS) is tightly coupled (i.e., based on the AXI Stream protocol and proper buffering) to the NIC [46] module to be able to transceive appropriate messages in order to distribute the workloads among the network.

Since the DFS is offloaded on the PL, all overheads regarding the thread management are reduced. As such, this leads to better energy efficiency and accelerates the distribution tasks through the PL as well.

IV. Methodology

A reliable and precise method to measure and monitor the power consumption of the system is necessary in order to enable optimization towards the energy efficiency. Additionally, the ability to estimate power consumption in a design is mandatory for efficient part selection and system reliability. Referring to the AXIOM board, there are specifically dedicated eight INA219 power monitor integrated circuits to monitor the crucial power rails of the board, and are reported by Table I. These INA219 ICs communicate with the FPGA through an I2C bus connected to the PS, and more detailed information on the INA219 can be found in [47].

In order to distribute threads among the network, there are two types of transceiving messages: 1) RAW and 2) Remote

TABLE I: AXIOM board's power supply rail adopting dedicated power monitors

Power supply rail	Nominal Voltage [V]	Description
VCC_INTFP	0.85	PS full-power domain supply voltage
VCCINT	0.85	PS internal power supply
INTFP_DDR	0.85	PS DDR controller and PHY supply voltage
1V2_DDR_PS	1.2	PS DDR supply
1.2V_DDR_PL	1.2	PL DDR supply
MGTAVCC	0.9	Analog supply voltage for GTH transceiver
MGTAVTT	1.2	Analog supply voltage for GTH transceiver termination circuits

Direct Memory Access (RDMA). Data of RAW messages are sourced by the DFS, while RDMA ones include a certain portion of memory to be moved between source and destination nodes. As a result, we measure power consumption of the AXIOM board for each of these message types, when the DFS is running at run-time using NIC. For this purpose, two boards have been interconnected, and one board is configured in server mode (DFS is sender) and the other in client mode (DFS is receiver). As such, a tool is designed to acquire power data by making IOCTL calls to the INA219s' driver that return the current value sunk from each monitored rail.

V. Experiments

In order to extract power values for the crucial rails of the board, we performed the experiments while DFS issues the RAW and RDMA messages of 1000M length in 10 cycles. The duration of the test was 240s with 200ms sampling time. Essentially, the total power consumption of the board remained between 1W and 1.6W (sum of the seven crucial power rails). Figure 2 illustrates the maximum power variations for the crucial voltage rails during RAW and RDMA transactions.

As can be seen from Figure 2, the MGTAVTT voltage rail has the highest power consumption since the gigabit transceivers' termination circuits with 1.2V supply voltage sink larger amount of current. The average power consumption in client mode has 10.15% larger value in comparison with server mode due to an extra processing effort to re-compose the acknowledge message and send it back to the server. Moreover, since in our DFS implementation we did not utilize any access to the PL DDR (we access to the PS DDR), the average power consumption for the 1V2_DDR_PL voltage rail remained below 5.5mW.

Finally, comparing power consumption between RAW and RDMA message, the RDMA message type consumes in average 9.7% less than the RAW message types. This arises from the extra dedicated logics to deal with data of RAW messages while for RDMA messages, the data are efficiently moved to the PS DDR by using the Xilinx Data Mover soft IP.

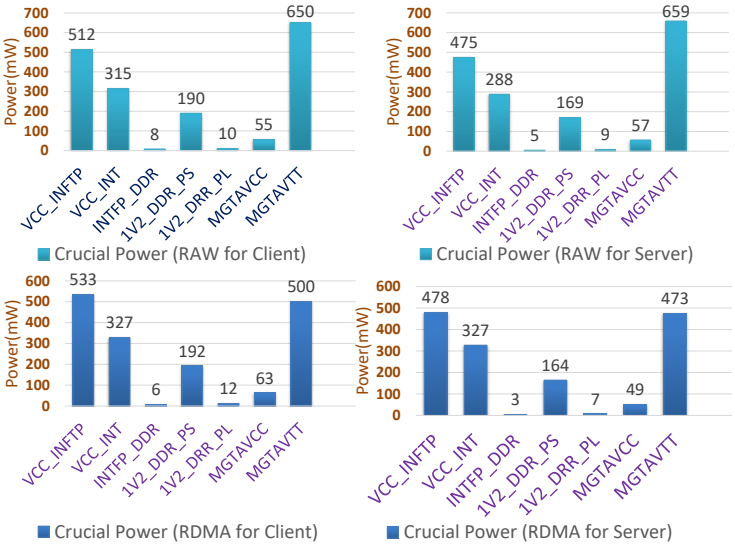

Fig. 2: Maximum variations in power consumption of crucial voltage rails for the Zynq Ultrascale+ when RAW and RDMA messages are issuing.

VI. Conclusion

In this paper, we present the deployment of Scalable Embedded Computers by using DF-Threads to distribute computations across multiple Zynq Ultrascale+ based board (The AXIOM board). We discussed several solutions in the High-Performance Computing domain as well as embedded worlds. We proposed a DF-Threads Scheduler which permits efficient scalability across the boards, and we presented some power measurements in the context of the AXIOM project as well. In order to be able to optimize the efficiency of the board, we explored the energy consumption of the important voltage rails while transceiving is performed across the nodes by the DFS.

Future work invokes further exploration of power consumption for more number of nodes (boards) with more benchmarks.

VII. Acknowledgment

The authors would like to thank with gratitude Davide Catani of SECO (s.r.l) [48] for his support in developing the experiments. This work has been partially supported by the European Commission under the AXIOM H2020 project (id. 645496) and HiPEAC (id. 779656).

References

[1] L. Jóźwiak, "Embedded computing technology for highly-demanding cyber-physical systems," *IFAC-PapersOnLine*, vol. 48, no. 4, pp. 19 – 30, 2015. 13th IFAC and IEEE Conference on Programmable Devices and Embedded Systems.

[2] D. Theodoropoulos, S. Mazumdar, E. Ayguade, *et al.*, "The axiom platform for next-generation cyber physical systems," *ELSEVIER Microprocessors and Microsystems*, pp. 540–555, 2017.

[3] R. Giorgi, S. Mazumdar, S. Viola, P. Gai, S. Garzarella, B. Morelli, D. Pnevmatikatos, D. Theodoropoulos, C. Alvarez, E. Ayguade, J. Bueno, A. Filgueras, D. Jimenez-Gonzalez, and X. Martorell, "Modeling multi-board communication in the axiom cyber-physical system," *Ada User Journal*, vol. 37, pp. 228–235, December 2016.

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

[5] 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*, pp. 262–269, July 2015.

[6] Xilinx Inc., "Xilinx UltraScale Architecture."

[7] R. Giorgi and P. Faraboschi, "An introduction to DF-Threads and their execution model," in *IEEE MPP*, (Paris, France), pp. 60–65, Oct. 2014.

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

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

[10] J. Bueno, L. Martinell, A. Duran, M. Farreras, X. Martorell, R. Badia, E. Ayguade, and J. Labarta, "Productive cluster programming with OmpSs," *Euro-Par Parallel Processing*, pp. 555–566, 2011.

[11] L. Dagum and R. Menon, "Openmp: An industry standard api for shared-memory programming," in *IEEE International Conference on Computational Science and Engineering*, pp. 46–55, Jan 1998.

[12] J. J. Costa, T. Cortes, X. Martorell, E. Ayguade, and J. Labarta, "Running OpenMP applications efficiently on an everything-shared SDSM," in *Proceedings of IPDPS*, pp. 35–42, 2004.

[13] S. Wesner, L. Schubert, R. Badia, A. Rubio, P. Paolucci, and R. Giorgi, "Special section on terascale computing," *ELSEVIER Future Generation Computer Systems*, vol. 53, pp. 88–89, July 2015.

[14] J. M. P. Cardoso, T. Carvalho, J. G. F. Coutinho, R. Nobre, R. Nane, P. C. Diniz, Z. Petrov, W. Luk, and K. Bertels, "Controlling a complete hardware synthesis toolchain with LARA aspects," *Microprocessors and Microsystems - Embedded Hardware Design*, vol. 37, no. 8-C, pp. 1073–1089, 2013.

[15] J. M. P. Cardoso, J. G. F. Coutinho, T. Carvalho, P. C. Diniz, Z. Petrov, W. Luk, and F. Gonçalves, "Performance-driven instrumentation and mapping strategies using the lara aspect-oriented programming approach," *Software: Practice and Experience*, pp. n/a–n/a, 2014.

[16] F. Yazdanpanah, C. Alvarez-Martinez, D. Jimenez-Gonzalez, and Y. Etsion, "Hybrid dataflow/von-neumann architectures," *IEEE Trans. on Parallel and Distrib. Systems*, vol. 25, pp. 1489–1509, June 2014.

[17] D. Orozco, E. Garcia, R. Pavel, J. Arteaga, and G. Gao, "The design and implementation of tideflow: A dataflow-inspired execution model for parallel loops and task pipelining," *International Journal of Parallel Programming*, vol. 44, no. 2, pp. 278–307, 2016.

[18] S. Arandi, G. Matheou, C. Kyriacou, and P. Evripidou, "Data-driven thread execution on heterogeneous processors," *International Journal of Parallel Programming*, vol. 46, pp. 198–224, Apr 2018.

[19] G. Matheou and P. Evripidou, "Data-driven concurrency for high performance computing," *ACM Trans. Archit. Code Optim.*, vol. 14, pp. 53:1–53:26, Dec. 2017.

[20] G. Matheou and P. Evripidou, "Architectural support for data-driven execution," *ACM Trans. Archit. Code Optim.*, vol. 11, pp. 52:1–52:25, Jan. 2015.

[21] R. Giorgi, "Transactional memory on a dataflow architecture for accelerating haskell," *WSEAS Trans. Computers*, vol. 14, pp. 546–558, 2015.

[22] M. Solinas, R. Badia, *et al.*, "The teraflux project: Exploiting the dataflow paradigm in next generation teradevices," in *IEEE Proc. 16th EUROMICRO-DSD*, (Santander, Spain), pp. 272–279, 2013.

[23] R. Giorgi, R. Badia, *et al.*, "TERAFLUX: Harnessing dataflow in next generation teradevices," *ELSEVIER Microprocessors and Microsystems*, vol. 38, no. 8, Part B, pp. 976–990, 2014.

[24] A. Mondelli, N. Ho, A. Scionti, M. Solinas, A. Portero, and R. Giorgi, "Dataflow support in x86-64 multicore architectures through small hardware extensions," in *IEEE Proc. DSD*, pp. 526–529, August 2015.

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

[26] R. Giorgi, "Teraflux: Exploiting dataflow parallelism in teradevices," in *ACM Computing Frontiers*, (Cagliari, Italy), pp. 303–304, May 2012.

[27] 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*, (Madrid, Spain), pp. 264–269, Nov. 2014.

[28] N. Ho, A. Mondelli, A. Scionti, M. Solinas, A. Portero, and R. Giorgi, "Enhancing an x86_64 multi-core architecture with data-flow execution support," in *ACM Computing Frontiers*, (Ischia, Italy), pp. 1–2, May 2015.

[29] R. Giorgi, "Accelerating haskell on a dataflow architecture: a case study including transactional memory," in *Proc. Int.l Conf. on Computer Eng. and Applications*, (Dubai, UAE), pp. 91–100, Feb. 2015.

[30] S. Weis, A. Garbade, J. Wolf, B. Fechner, A. Mendelson, R. Giorgi, and T. Ungerer, "A fault detection and recovery architecture for a teradevice dataflow system," in *Proc. IEEE Int.l Workshop on Data-Flow Execution Models for Extreme Scale Computing (DFM)*, pp. 38–44, Oct. 2011.

[31] S. Weis, A. Garbade, B. Fechner, A. Mendelson, R. Giorgi, and T. Ungerer, "Architectural support for fault tolerance in a teradevice dataflow system," *Springer Int.l Journal of Parallel Programming*, vol. 44, pp. 208–232, Apr 2016.

[32] V. Milutinovic, J. Salom, N. Trifunovic, and R. Giorgi, *Guide to DataFlow Supercomputing Basic Concepts, Case Studies, and a Detailed Example Postscript*. Berlin, DE: Springer, Apr 2015.

[33] W. A. Najjar, E. A. Lee, and G. R. Gao, "Advances in the dataflow computational model," *Parallel Comput.*, vol. 25, pp. 1907–1929, Dec. 1999.

[34] S. Windh, X. Ma, R. J. Halstead, P. Budhkar, Z. Luna, O. Hussaini, and W. A. Najjar, "High-level language tools for reconfigurable computing," *Proceedings of the IEEE*, vol. 103, no. 3, pp. 390–408, 2015.

[35] W. Ahmed, M. Shafique, L. Bauer, and J. Karlsruhe, "Adaptive resource management for simultaneous multitasking in mixed-grained reconfigurable multi-core processors," in *Hardware/Software Codesign and System Synthesis (CODES+ISSS), 2011 Proc. of the 9th Int'l Conference on*, pp. 365–374, Oct 2011.

[36] A. Portero, A. Scionti, Z. Yu, P. Faraboschi, C. Concatto, L. Carro, A. Garbade, S. Weis, T. Ungerer, and R. Giorgi, "Simulating the future kilo-x86-64 core processors and their infrastructure," in *45th Annual Simulation Symp. (ANSS12)*, (Orlando, FL), pp. 62–67, Mar 2012.

[37] J. Clemente, V. Rana, D. Sciuto, I. Beretta, and D. Atienza, "A hybrid mapping-scheduling technique for dynamically reconfigurable hardware," in *Field Programmable Logic and Applications (FPL)*, pp. 177–180, Sept 2011.

[38] P. Burgio, C. Alvarez, E. Ayguade, A. Filgueras, D. Jimenez-Gonzalez, X. Martorell, N. Navarro, and R. Giorgi, "Simulating next-generation cyber-physical computing platforms," *Ada User Journal*, vol. 37, pp. 59–63, Mar. 2016.

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

[40] 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 (EUC 2015)*, pp. 148–153, Oct. 2015.

[41] C. Alvarez, E. Ayguade, J. Bueno, *et al.*, "The AXIOM software layers," in *IEEE Proc. 18th EUROMICRO-DSD*, pp. 117–124, Aug. 2015.

[42] C. Alvarez, E. Ayguade, *et al.*, "The AXIOM software layers," *ELSEVIER Microprocessors and Microsystems*, vol. 47, Part B, pp. 262–277, 2016.

[43] G. Burresi and R. Giorgi, "A field experience for a vehicle recognition system using magnetic sensors," in *IEEE MECO 2015*, (Budva, Montenegro), pp. 178–181, IEEE, June 2015.

[44] A. Rizzo, G. Burresi, F. Montefoschi, M. Caporali, and R. Giorgi, "Making iot with udoo," *Interaction Design and Architecture(s)*, vol. 1, pp. 95–112, Dec. 2016.

[45] Y. Wu, L. Zheng, B. Heilig, and G. R. Gao, "Hamr: A dataflow-based real-time in-memory cluster computing engine," 2017.

[46] Vasileios17, "Efficient network interface design for low cost distributed systems," 2017.

[47] "http://www.ti.com/lit/ds/symlink/ina219.pdf," Dec. 2015.

[48] SECO, s.r.l., "http://www.seco.com," Dec. 2017.