An FPGA-based Scalable Hardware Scheduler for Data-Flow Models

Marco Procaccini, Farnam Khalili, Roberto Giorgi

Department of Information Engineering and Mathematics, University of Siena, Italy[1]

ABSTRACT

This paper presents a scheduler for Data-Flow threads implemented in reconfigurable logic for being deployed on Reconfigurable MPSoCs (i.e., Multi-Processing System on Chips with FPGA). "Data-Flow threads" (DF-Threads) is a novel execution model for mapping threads on local or distributed cores transparently to the programmer. This model is capable of being parallelized massively among different cores and it handles even hundreds of thousands or more Data-Flow threads, and their associated data frames, in order to distribute them both in a local node and through the network to other nodes in a transparent way. The Hardware Scheduler (HS) is designed for being used in Programmable Logic (PL) of MPSoC FPGAs and it deals with the GPP cores, providing them with Data-Flow threads ready to be executed. The overall design is modeled and tested through the HPLabs COTson simulator. Here we use the Block Matrix Multiply benchmark to analyze the potentiality of the proposed model.

Keywords: Data-Flow; Reconfigurable; FPGA; Hardware Scheduler; Thread Level Parallelism

1. Introduction

The end of the Dennard scaling [1] and the resulting difficulty to increase clock frequency forced the engineering community to shift to the multicore processors as an alternative way to improve performance at the limited power budget.

An increased core number benefits many workloads, but programming limitations to exploit full performance still remain due to the not fully exploited parallelism. According to Mondelli et al. [2], the Data-Flow execution model is capable of taking advantage of the full parallelism offered by multicore systems. Each Data-Flow thread is a node of the Data-Flow graph, each of them only executes when its inputs are available and we succeeded to impose this condition locally to each core. In a Data-Flow based execution, a program could be executed out of its linear order but in a partial order, which it depends on the data dependencies. As a result, individual partial orders are data independent and can be executed in parallel. The length of the data independent path is the expression of the granularity of the parallelism [3].

There exist many attempts of Data-Flow based architectures, which can exploit the potential of the Data-Flow execution model (explicit-Data-Flow architectures), but currently they cannot totally replace the conventional general purpose processors (GPP) due to some limitation of the execution

[1] This work is partly funded by the European Commission under the projects AXIOM (645496), HiPEAC (687698)

model. For example, the control transfer might be more expensive in the Data-Flow model, and the latency cost of explicit data communication could be prohibitive [4].

In order to overcome these limitations, a hybrid Data-Flow model is presented in this work, which is based on heterogeneous architecture composed by GPP cores and FPGA.
GPP cores allow us to be suitable for a large set of applications and FPGAs are known for their reconfigurability and power efficiency, compared to software only designs, so that they are a suitable choice for being deployed in the many-threads Data-Flow execution models as well as providing a spatial substrate for mapping Data-Flow threads. These models evolve around the optimizing of data mobility and exploiting massively parallelism among thousands of Data-Flow threads to offer more modularity and higher performance [5][6][7] [8] [9][15][16].
Here the idea is to detach the execution of the Data-Flow threads from its scheduling, reducing the latency of the data communication and increase the overall performance of the execution. We propose a scalable hardware scheduler (HS), implemented mainly on the FPGA, which provides Data-Flow threads ready to be executed to the GPP cores. The overall architecture has been modeled and tested first on the COTSon simulator [10] and the model is mapped and designed for being employed in a heterogeneous architecture.

2. A New Data-Flow Hardware Scheduler

The Data-Flow execution paradigm can be exploited either completely on Hardware or it can be used in a control flow processor to improve the execution time by Thread Level Parallelism (TLP) [11]. The main task of the Data-Flow scheduler is to materialize TLP in such way that respects to Data-Flow paradigm at thread level [12] [13].
The two main actors of the model are the Processing System (PS), the control flow processor, and the Hardware Scheduler (HS) implemented into the Programmable Logic (PL). A VHDL based Interface Architecture is used to allow information exchange between PS and HS [14].
The PS is responsible to create and execute the Data-Flow threads. Whenever a new DF-thread is created, the HS is responsible to retrieve the meta-information of the thread and stores them into the associated frame. When a producer DF-Thread wants to write its outputs into the consumer DF-Thread, the HS performs the writing in a lazy and asynchronous way, without blocking the PS. After the output's writes, the HS decrease the synchronization count (SC) of the consumer DF-Thread. If the SC is equal to zero, the DF-Thread is ready to execute and it is moved into a FIFO queue named Ready Queue (RQ). When the PS asks for a new DF-Thread to execute, the HS dequeues the first element of the RQ and sends it to the PS. In order to distribute the computation among multiple nodes, the HS checks its RQ status and if the RQ size is under a certain threshold, the HS try to steal DF-Treads ready to execute from other nodes into the network.

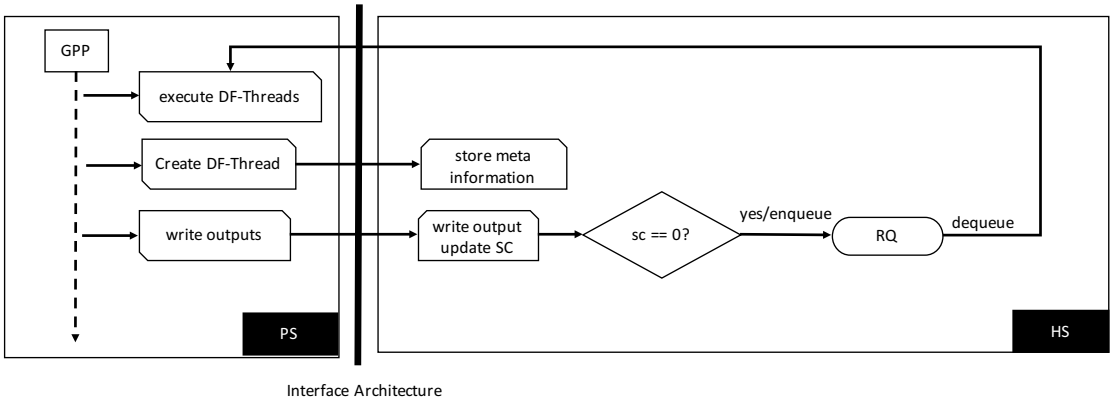

Figure 1: Interaction between System (PS) and the Hardware Scheduler (HS), exploiting the Interface Architecture, to realize the Data-Flow thread (DF-Thread) execution model. A General Purpose Processor (GPP) execute, create and write DF-Threads. On the other side, the HS collects and uses meta-information of a DF-Threads to schedule them. SC is the synchronization count, RQ is the Ready Queue.

3. Performance Analysis

In order to evaluate the proposed model, we implemented it first on the COTson simulator and we performed tests based on the Block Matrix Multiplication benchmark. Several experiments were made, varying the number of GPPs, the number of nodes, the matrix size and the block size. As we can see in figure 2, the speedup of the execution is reasonable good, specially increasing the number of nodes/GPPs and with large size of the Matrix. The block size does not affect much the overall performance. Due to the decrease available parallelism, with a small matrix size too few threads are generated and this produces a decrease of performance when the number of nodes and GPPs increases.

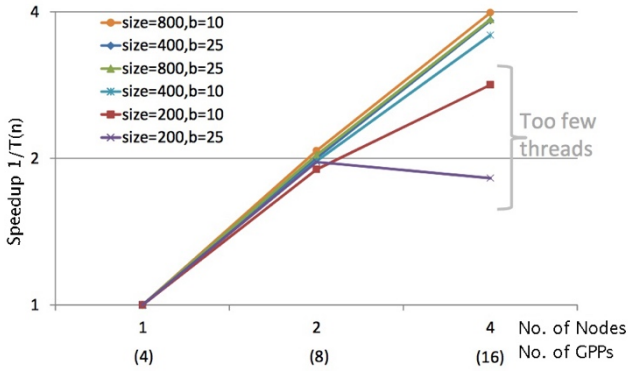

Figure 2: Speedup study of the Block Matrix Multiply test, varying the matrix size sizes and block size. Different nodes and General Purpose processors (GPP) were used.

References

[1] Frank, D. J., Dennard, R. H., Nowak, E., Solomon, P. M., Taur, Y., & Wong, H. S. P. (2001). Device scaling limits of Si MOSFETs and their application dependencies. *Proceedings of the IEEE*, *89*(3), 259-288.

[2] Mondelli, Andrea, et al. "Dataflow support in x86_64 multicore architectures through small hardware extensions." *Digital System Design (DSD), 2015 Euromicro Conference on*. IEEE, 2015

[3] Dennis, J. B. (1980). Data flow supercomputers. *Computer*, (11), 48-56.

[4] Budiu, M., Artigas, P. V., & Goldstein, S. C. (2005, March). Dataflow: A complement to superscalar. In *Performance Analysis of Systems and Software, 2005. ISPASS 2005. IEEE International Symposium on* (pp. 177-186). IEEE.

[5] Giorgi, R., & Faraboschi, P. (2014, October). An introduction to DF-Threads and their execution model. In *Computer Architecture and High Performance Computing Workshop (SBAC-PADW), 2014 International Symposium on* (pp. 60-65). IEEE.

[6] Stavrou, K., Pavlou, D., Nikolaides, M., Petrides, P., Evripidou, P., Trancoso, P., ... & Giorgi, R. (2009, June). Programming abstractions and toolchain for dataflow multithreading architectures. In *Parallel and Distributed Computing, 2009. ISPDC'09. Eighth International Symposium on* (pp. 107-114). IEEE.

[7] Verdoscia, L., Vaccaro, R., & Giorgi, R. (2014, August). A clockless computing system based on the static dataflow paradigm. In *Data-Flow Execution Models for Extreme Scale Computing (DFM), 2014 Fourth Workshop on* (pp. 30-37). IEEE.

[8] Solinas, M., Badia, R. M., Bodin, F., Cohen, A., Evripidou, P., Faraboschi, P., ... & Goodman, D. (2013, September). The TERAFLUX project: Exploiting the dataflow paradigm in next generation teradevices. In *Digital System Design (DSD), 2013 Euromicro Conference on* (pp. 272-279). IEEE.

[9] Zuckerman, S., Suetterlein, J., Knauerhase, R., & Gao, G. R. (2011, June). Using a codelet program execution model for exascale machines: position paper. In *Proceedings of the 1st International Workshop on Adaptive Self-Tuning Computing Systems for the Exaflop Era* (pp. 64-69). ACM.

[10] Argollo, E., Falcón, A., Faraboschi, P., Monchiero, M., & Ortega, D. (2009). COTSon: infrastructure for full system simulation. *ACM SIGOPS Operating Systems Review*, *43*(1), 52-61.

[11] Giorgi, R., Popovic, Z., & Puzovic, N. (2007, October). DTA-C: A decoupled multi-threaded architecture for CMP systems. In *Computer Architecture and High Performance Computing, 2007. SBAC-PAD 2007. 19th International Symposium on* (pp. 263-270). IEEE.

[12] Kavi, K. M., Giorgi, R., & Arul, J. (2001). Scheduled dataflow: Execution paradigm, architecture, and performance evaluation. *IEEE Transactions on Computers*, *50*(8), 834-846.

[13] Giorgi, R., & Popovic, Z. (2006). Core Design and Scalability of Tiled SDF Architecture. *HiPEAC ACACES-2006*, 145-148.

[14] Khalili, F., Procaccini, M., Giorgi,R. (2018). Reconfigurable Logic Interface Architecture for CPU-FPGA Accelerators. *HiPEAC ACACES-2018.*

[15] Kyriacou, C., Evripidou, P., & Trancoso, P. (2006). Data-driven multithreading using conventional microprocessors. IEEE Transactions on Parallel and Distributed Systems, 17(10), 1176-1188.

[16] Alves, T. A., Marzulo, L. A., França, F. M., & Costa, V. S. (2011). Trebuchet: exploring TLP with dataflow virtualisation. International Journal of High Performance Systems Architecture, 3(2-3), 137-148.