Dynamically Reconfiguring through Phase Detection on FPGA

Kang Cai[1], Stamatis Kavvadias[1], Alberto Scionti[1], Claudio Scordino[2], Paolo Gai[2], Roberto Giorgi[1]

1. Università degli Studi di Siena, via Roma 56 53100 Siena, Italy
2. Evidence, via Carducci 56 Ghezzano 56010, S. Giuliano Terme Pisa, Italy

ABSTRACT

Previous researches have shown some approaches on hardware phase detection. In this work, we propose a new framework based on Xilinx Virtex-6 platform for the implementation of task-optimized coarse-grained reconfiguration that can be reconfigured to adapt to the applications' behavior. We use MicroBlaze as a general-purpose processor and a ρ-VEX VLIW architecture as reconfigurable cores. A run-time component called supervisor, dynamically monitors the system behavior, and triggers the reconfiguration; we also propose a Profiler component that automatically obtains the phase information. The collected data can be used to guide dynamic reconfiguration on the FPGA.

KEYWORDS: RECONFIGURABLE; HARDWARE; PHASE CLASSIFICATION

1. Introduction

Nowadays, FPGAs are becoming more and more popular and powerful. However, it is still a challenge to dynamically adapt to the application needs for obtaining more efficiency. The problem is how to dynamically monitor and obtain the behavior of the running application, and try to adapt the hardware architecture to it. Therefore, it is important to accurately detect program phases on embedded hardware.

Previous efforts have demonstrated some hardware phase detection approaches. B. Vijayan [1] uses the address information of branch instructions to dynamically change the configuration of the processing elements. The key idea explored by E. Ipek [2] is the extension of the BBV (Basic block vector, T.Sherwood [8]) to incorporate information about frequencies of memory accesses performed by all the processors of the system. This is a modified version of the detection architecture originally proposed by T. Sherwood [3], who presents a hardware architecture to track and predict program phases at run-time. The architecture receives two inputs: (1) the address of a branch, and (2) the number of instructions executed since the last branch. Although these researches studied phase classification on general hardware, our work focuses on FPGA with more reconfiguration capability.

In this work, we propose a new framework, based on Xilinx Virtex-6 platform, for the implementation of task-optimized reconfiguration that can be reconfigured to adapt to the applications' behavior. We use MicroBlaze as a general-purpose processor to run the operating

system and run-time tools; the reconfigurable cores are based on **ρ**-VLIW architecture. Inside MicroBlaze we use a special component called supervisor[10] to dynamically monitor the behavior of the application and trigger the reconfiguration. This design can be used to guide the dynamic reconfiguration of embedded reconfigurable systems.

2. The Phase Detection Framework on FPGA

Figure 1 shows a view of the proposed reconfiguration framework. The general-purpose processor is the Xilinx MicroBlaze™ soft core, while the reconfigurable cores are based on a ρ-VEX VLIW [6] architecture. During the execution of the application, the supervisor dynamically monitors the application behavior, and detects the different phases that characterize the application. Detected phases are used to trigger the reconfiguration of the hardware system (e.g., reconfigurable cores).

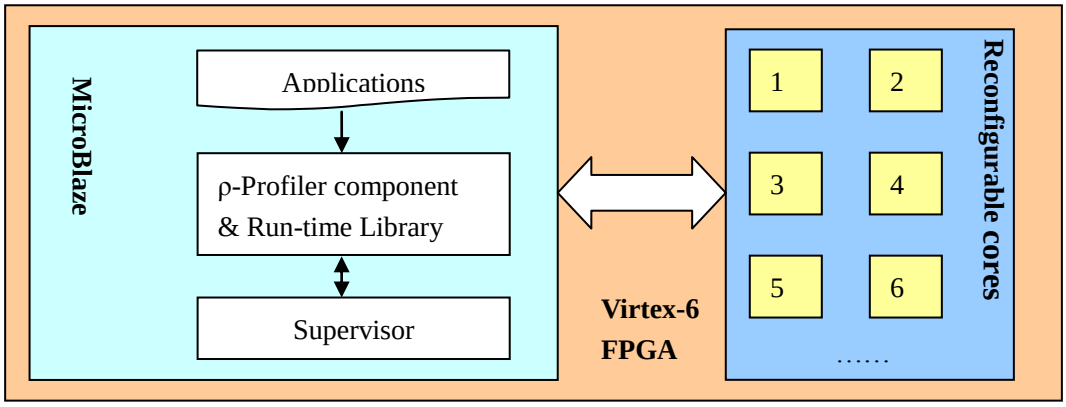

Figure 1 Overview of the reconfiguration framework

The ρ-profiler component is in charge of classifying phases. The supervisor interacts at run-time, with reconfigurable modules (e.g., reconfigurable cores), the run-time library, and the operating system. During the application execution, the supervisor uses program's behavior patterns to select code that triggers the reconfiguration, reducing as much as possible the power consumption by turning on/off reconfigurable modules without affecting performance.

3. Phase Classification Based on FPGA

In this section we will discuss two issues regarding phase classification based on FPGA: what program structures will be used and the work flow of phase classification on FPGA.

3.1. Program Structures for Capturing Phase Behavior on FPGA Platform

We mainly chose micro-architecture independent structures for capturing phase behavior. As test case, we considered VLIW cores as the reconfigurable parts, and based on many research results [5, 7, 8, 9], we employ: basic blocks, loops, branches, procedures, and memory address information including global stride, local strides, working set size.

3.2. Program Phase Classification

The program phase classification on FPGA is used to dynamically reconfigure the hardware resources. In this work, we propose an on-line phase classification work-flow, depicted in figure 2. The work-flow is composed of:

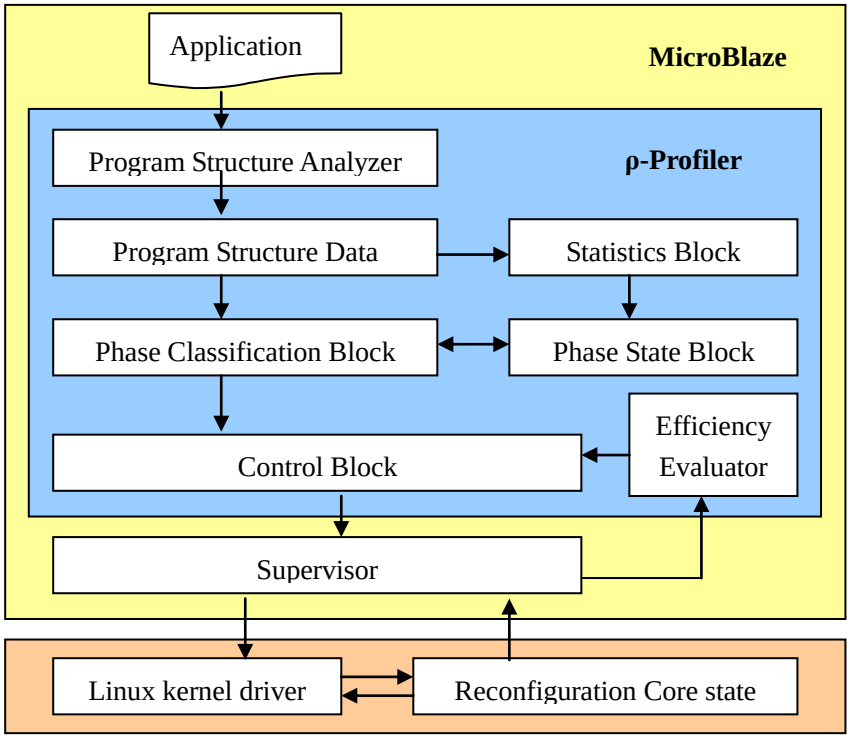

Figure 2 Overview of the phase classification work flow

- **Program Structure Analyzer：** According to the structure being used, it analyzes the structure, marking its start and the end points. It provides this information as input to the next block.
- **Program Structure Data：** Gather input metric for phase classification.
- **Phase classification Block：** Extracts the phases according to the selected metrics. If a phase is not present in the phase state block, name a new ID, otherwise the code segment is considered.
- **Phase state Block：** Record phases and the corresponding code segment IDs.
- **Efficiency Evaluator:** Compute power consumption and execution time, providing the efficiency evaluation to support supervisor making decisions, make the system working with the optimal configuration.
- **Control block:** Interacts from one side phase classification block, efficiency evaluator, from the other side with supervisor.
- **Supervisor:** Triggers the hardware reconfiguration, according to the phase state and the efficiency evaluation.

We use k-means [7] as the phase classification algorithm. The phases and program structures are recorded in the format of figure 3:

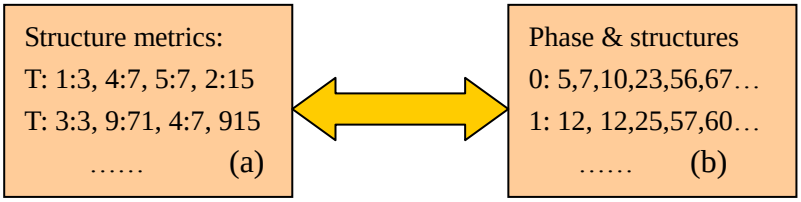

Figure 3 possible representation of phase and the program structures

In figure 3 (a), the "T" symbol delimits a new execution interval represented by a fixed number of instructions. For each line, the data are represented by a list of pair values: the first value is the program structure ID and the second value indicates how many times it was executed. In figure 3 (b) the first number of each line represents the phase ID assigned by the classification algorithm. Data in

each line are represented by a list of program structure IDs belonging to phases.

Whenever a program is loaded on the FPGA, the supervisor starts monitoring the program structure being used from start to end, tracking the structure ID and the execution status, and forms a metric as an input to the Phase classification block to get the phase. Then, efficiency evaluator provides feedback on impact on reconfiguration based on the selected metrics, finally the supervisor triggers the system reconfigurable elements based on the phases, power consumption efficiency, and different program structures execution time. The phase classification and the program execution are in a parallel module.

4. Conclusions

To help FPGA platform reconfiguring its resources dynamically, we studied the online phase classification on FPGA problem. We use the supervisor to monitor the behavior of program and form a strategy based on the program phases to trigger reconfiguration. Also we take power consumption and execution time as the reference parameters to help supervisor to make the decisions, keeping a minimum reconfiguration time in order to have no effect on the system performance. We expect our approach can help dynamic reconfiguration of components in FPGA.

5. Acknowledgements

This work is partially funded by the European FP7 project ERA, ID: 249059, http://www.era-project.org/., and HiPEAC IST-217068.

References

[1] B. Vijayan, D. V. 2008. Accurate and Low-Overhead Dynamic Detection and Prediction of Program Phases Using Branch Signatures. 20th International Symposium on Computer Architecture and High Performance Computing. IEEE.

[2] E. Ipek, J. F. 2006. Dynamic Program Phase Detection in Distributed Shared-Memory Multiprocessors. IEEE International Symposium of Code Generation and Optimization (CGO'06). IEEE.

[3] T. Sherwood, S. S.2003. Phase Tracking and Prediction. Proceedings of the 30th Annual International Symposium on Computer Architecture (ISCA'03). IEEE.

[4] N.Puzovic, S.McKee, R.Eres, Ayal.Zaks P.Gai, S.Wong, R.Giorgi. 2010. A Multi-Pronged Approach to Benchmark Characterization",IEEE Int.lConf. on Cluster Computing, ISBN: 978-1-4244-8396-9. p. 1-4.

[5] J.Kim, S.Y.M. 2011. Program Phase-Aware Dynamic Voltage Scaling Under Variable Computational Workload and Memory Stall Environment. IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems, 30 (1). p. 110-123.

[6] RAE Seedorf, F. Anjam, A.A.C. Brandon, S. Wong.2012. Design of a Pipelined and Parameterized VLIW Processor: r-VEX v.2.0. HiPEAC-2012.

[7] J. Lau, S. Schoemackers, B. Calder.2004. Structures for phase classification.Proceeding. Proceedings of the 2004 IEEE International Symposium on Performance Analysis of Systems and Software. p.57 – 67.

[8] S.Wong, L. Carro, M. Rutzig, D. M. Matos, G.Roberto, N.Puzovic, S. Kaxiras, M. Cintra, G. Desoli, P. Gai, S. A.Mckee, A. Zaks, "ERA - Embedded Reconfigurable Architectures", Springer New York, ISBN:978-1-4614-0061-5, 2011. p. 239-259.

[9] M. B. C. Alioto, P. Bennati, R. Giorgi. 2010. Exploiting Locality to Improve Leakage Reduction in Embedded Drowsy I-Caches at Same Area/Speed. IEEE Int.l Symp. on Circuits and Systems (ISCAS), ISBN:978-1-4244-5309-2. p. 37-40.

[10] http://www.evidence.eu.com/era-supervisor.html