Pareto based Run-time Manager for Overlapped Resource Sharing

Narasinga Rao Miniskar^{*,†,1},
Satyakiran Munaga^{*,†,1}, Roel Wuyts^{*,1},
Francky Catthoor^{*,†,1},

* *IMEC, Kapeldreef 75, 3001 Leuven, Belgium*
† *ESAT Department, K.U.Leuven, 3001 Leuven, Belgium*

ABSTRACT

Mapping applications on a heterogenous platform in energy efficient way is a challenging problem. The goal is to find at run-time, the best assignment and schedule which minimizes the energy consumed to execute the given set of (sub-)tasks, while satisfying all given constraints. This notoriously complex problem requires a mixed design-time/run-time approach: a low and scalable overhead run-time phase assisted by an extensive design-time preparation. However, the existing approaches are limited to either two dimensional pareto handling approaches or sequential based sharing of resources or homogeneous multiprocessors. In this paper we present, for the first time, a low-overhead and multi-dimensional pareto based run-time mapping algorithm which can share the platform resources in overlapped basis.

KEYWORDS: mapping; heterogeneous; TCM, sequential, overlapped

1 Introduction

Modern nomadic systems need to perform heavy data-processing and deliver high throughput but within very limited power and energy budgets. As a result it is common today that hardware platforms for nomadic embedded systems are highly heterogeneous and composed of multiple processors from the whole spectrum of flexibility: parametrized hardware accelerator, ASIP, FPGA, VLIW DSP, and GP-RISC. Given such an adaptive heterogeneous multiprocessor platform, the process of mapping decides for each (sub-)task: when it should execute, the platform resource(s) on which it should execute, and the knob-settings of the resources during its execution[2]. The goal of task mapping is to find the best assignment and scheduling which minimizes energy consumed to execute the given set of tasks while satisfying timing, precedence and resource constraints of all (sub-)tasks.

[1]E-mail: {miniskar,satyaki,wuytsr,catthoor}@imec.be
[2]Thus, mapping refers to the combined problem of assignment and scheduling.

As the set of tasks to be run and the performance-related timing constraints like individual tasks deadlines are only known at run-time, final mapping decisions can only be done at run-time on-chip. The application dynamics can be fast, can only be done during the run-time and may require the run-time manager to be invoked every few tens of milliseconds. This implies that run-time mapping algorithms should be very fast without significantly compromising the energy-efficiency of the solution.

In a multiprocessor platform, one can share the resources (processors, memories, interconnect, etc...) either the sequential strategy or overlapped strategy or fully interleaved strategy. In Sequential resource strategy[Yang04], all platform resources will be given to only one task at a time even though it doesn't use all and they will be released when the task completes its execution. In the Overlapped resource sharing strategy, only the requested portion of the platform resources are assigned to the task, and they will be released only after the task's completion. Even if the task doesn't require those requested portion of resources till to its completion, those resources will be in idle. There is another resource sharing Strategy, which can interleave the tasks based on processor idle state exploitation, which is the out-of the scope of this paper.

The problem of energy-aware dynamic application mapping on adaptive heterogeneous MPSoC is very challenging. In related literature, there are solutions to only simplified versions of this complex problem. The majority of the work optimizes the makespan of tasks and not the energy consumption[Bake05]. Existing energy-aware solutions suffer from one or more of the following issues: (i) can only handle uni-processor[Sinh01] and homogeneous multiprocessor systems[Zhan02] (ii) can only handle DVFS knobs[Pras02, Chen07] and few can exploit body bias knobs - they are based on approximate models (iii) purely design-time techniques involving slow selection heuristics[Pras02], which can not be used directly at run-time (iv) for tractable run-time complexity, they hide task internals and can assign each task to one processor[Chen07] - can not fully exploit the mapping choices available in a multi-processor system and thus greatly sub-optimal.

There is some literature on the mixed design-time/run-time approach or two phase approach[Yang04][YC06]: extensive design-time preparation and low-overhead run-time final decision making selection. A design-time-exploration leads to a intermediate representation that can be used at run-time to make the final scheduling decision. Each pareto point in a pareto curve annotate with a mapping of sub-tasks on a heterogenous platform. The run-time decision making involves the selection of one operating point for each active application based on the time constraints, resource constraints, and environment changes. The run-time decision making has two components, the Pareto Point Selection (to select the operating pareto point from each application) and the Feasibility check (to check the feasibility of timing and resource constraints). However, the existing literature limits either to limited dimensions, sequential resource sharing strategy.

In this paper, for the first time, we present a low-overhead energy-aware pareto-based run-time mapping algorithm which can share the platform resources in overlapped basis. This requires a suitable Pareto Point selection heuristic and its associated Feasibility Check heuristic too. We will assess the effectiveness of our improved mapping heuristics on a heterogeneous MPSoC platform, which includes RISC, VLIW. In Section 2, we will discuss the Related work for this problem. Section 3 details the problem definition. Section 4 details the Experimental Setup. In Section 5, we present the heuristics to solve this problem. In Section 6, we will discuss about the results. Finally we conclude in Section 7.

2 Experimental Setup

The generic heterogeneous multiprocessor platform that we have considered in this paper has a five RISC processors (Strong ARM 1100x) and three VLIWs with eight FUs each (TI-C64X+). The four StrongARM 1100x processors run at 1.48V, 0.1632A, with clock frequency of 133MHz and the TI-C64X+ processors run at 1.2V with clock frequency of 500MHz. We have dedicated one Strong ARM 1100x processor for run-time-mapping which runs at 1.97V, with a clock frequency of 500 MHz. The L1 memories of all processors are of same size 64KB and the shared main memory size is 256MB. All processors and memories are connected with Bus interconnect.

To validate the effectiveness of our approaches, we have considered the TGFF[Dick98] generated task graphs. Each task graph represents an application task and the nodes inside the task graph represents a sub-task. We have generated 9 TGFF task graphs, where each task graph has sub-tasks varying from 6 to 30. We have played with the period_mul, period_laxity, and task_cnt parameters of TGFF to generate these different task graphs. In order to characterize the workload (in cycles), memory (l1 and main) size and Bandwidth (in %) requirements[YC06], we have used type_attrib of TGFF for each type of processor type and for each sub-task.

From the Design-Time-Exploration[Yang04], we have obtained the execution time with SimItARM and CCStudio simulators and energy consumptions with models [Sinh01], [Laur04] for the StrongARM and TI-C64X+ respectively. We have obtained multi-dimensional pareto curves each of the task graph obtained from TGFF.

3 Run-Time Mapping Heuristics

For the overlapped resource sharing based run-time mapping, we have proposed first the combination of Gradient Descent heuristic[Tack05] as a pareto selection algorithm and List Scheduling heuristics as a feasibility check algorithm to solve this problem. Even though this run-time scheduling heuristic can be stopped at any moment of time, it requires 71 m.secs on the StrongARM processor running at 500MHz with 1.97v for the reference case. This is too much overhead for the run-time mapping as it needs to take decision in 1 to 15 m.secs, for the Wireless and Multimedia applications.

For this purpose, we have feedbacked in the List Scheduler. The feedback List Scheduler, i. compute the ALAP time for only the updated task, ii. it will place the updated task in the right position of the priority selection queue, iii. For all tasks before the updated Task, the schedule is guaranteed, so, the List Scheduler will run for the rest of the tasks. However the run-time overhead is still too high, $\sim$53.43 m.secs.

For the initial solutions of Gradient Descent, the initial pareto points of all tasks are more performance efficient but with high energy consumption. As they can finish much faster, even if we share all platform resources in sequentially (the tasks will execute one after the other), it can still run. If the platform is shared sequentially, the Feasibility Check is much simpler as in [Yang04] or EDF. This 2-phase pareto selection significantly reduced the run-time overhead by $\sim$13 m.secs with unoptimized run-time manager.

When compare to Sequential resource sharing strategy run-time mapping[Yang04], we have obtained 3x better energy efficient solution with an increase of small overhead(5 m.sec.).

4 Conclusions

In this paper, we have proposed for the first time a low-overhead multi-dimensional pareto based run-time mapping algorithm for the overlapped resource sharing strategy. We have also shown how the overhead can be managed by exploiting the different correlations w.r.t. different Feasibility Check algorithms and pareto-selection phases. We have also shown that our Overlapped resource sharing based heuristic is providing much better solution than the Sequential resource sharing based heuristics when compare to the overhead of the Run-time-mapping.

References

[Bake05] T. BAKER. An Analysis of EDF Schedulability on a Multiprocessor. *IEEE Transactions on Parallel and Distributed Systems*, 2005.

[Chen07] J. CHEN, C. YANG, T. KUO, AND C. SHIH. Energy-efficient Real-time Task Scheduling in Multiprocessor DVS Systems. In *Proceedings of the Asia South Pacific Design Automation Conference*, pages 342–349, Yokohama, Japan, jan 2007.

[Dick98] R. DICK. TGFF: task graphs for free. In *CODES/CASHE '98*, 1998.

[Laur04] J. LAURENT. Functional Level Power Analysis: An Efficient Approach for Modeling the Power Consumption of Complex Processors. In *DATE*, 2004.

[Pras02] V. PRASANNA. Power-Aware Resource Allocation for Independent Tasks in Heterogeneous Real-Time Systems. In *ICPADS '02*, page 341. IEEE, 2002.

[Sinh01] A. SINHA AND A. CHANDRAKASAN. JouleTrack - A Web Based Tool for Software Energy Profiling. In *DAC*, 2001.

[Tack05] N. TACK, G. LAFRUIT, F. CATTHOOR, AND R. LAUWEREINS. Pareto based optimization of multi-resolution geometry for real time rendering. In *Web3D*. ACM, 2005.

[Yang04] P. YANG, P. MARCHAL, C. WONG, S. HIMPE, F. CATTHOOR, P. DAVID, J. VOUNCKX, AND R. LAUWEREINS. *Multiprocessor Systems-on-Chip*, Chapter: Cost-efficient mapping of dynamic concurrent tasks in embedded real-time multimedia systems, pages 46–58. Morgan-Kaufmann, 2004. Eds W. Wolf, A. Jerraya.

[YC06] C. YKMAN-COUVREUR, V. NOLLET, F. CATTHOOR, AND H. CORPORAAL. Fast Multi-Dimension Multi-Choice Knapsack Heuristic for MP-SoC Run-Time Management. In *Proceedings of the International Symposium on System-on-Chip*, pages 195–198, Tampere, Finland, nov 2006.

[Zhan02] Y. ZHANG, X. HU, AND D. CHEN. Task Scheduling and Voltage Selection for Energy Minimization. In *Proceedings of the Design Automation Conference*, pages 183–188, 2002.