Leszek CIOPIŃSKI, Stanisław DENIZIAK
DEPARTMENT OF INFORMATION SYSTEMS, KIELCE UNIVERSITY OF TECHNOLOGY
7 1000-lecia Państwa Polskiego Ave., 25-314 Kielce, Poland

A synthesis of adaptive, low-power real-time embedded systems for ARM big.LITTLE technology

Abstract

In this paper, we present a method of a synthesis of adaptive schedulers for real-time embedded systems. We assume that the system is implemented using a multi-core embedded processor with low-power processing capabilities. First, the developmental genetic programming is used to generate the scheduler and the initial schedule. Then during the system execution, the scheduler modifies the schedule whenever the execution time of the recently finished task has been shorter or longer than expected. The goal of rescheduling is to minimize the power consumption while all time constraints will be satisfied. We present a real-life example as well as some experimental results showing the advantages of the method.

Keywords: self-adaptive system, real-time embedded system, adaptive scheduler, Developmental Genetic Programming, ARM big.LITTLE.

1. Introduction

Besides the cost and performance, power consumption is one of the most important issues considered in the optimization of embedded systems. A design of energy-efficient embedded systems is important especially for battery-operated devices. Embedded systems are usually real-time systems, i.e. time constraints are defined for some tasks. Therefore, power optimization should take into consideration the fact that all time requirements should be met. It should consider the tradeoff between power, performance, cost and perhaps other attributes.

The system performance may be increased by applying a distributed architecture. The function of the system is specified as a set of tasks, then during the co-design process, the optimal architecture is searched. Recently, the advent of embedded multicore processors has created an interesting alternative to dedicated architectures. First, the co-design process may be reduced to task scheduling. Second, advanced technologies for power management, like DVFS (Dynamic Voltage and Frequency Scaling) or big.LITTLE [1], create new possibilities for designing low-power embedded systems.

Optimization of embedded systems is based on assumptions that certain system properties are known. For example, to estimate the performance of the system, execution times for all tasks should be known. Sometimes it is difficult to precisely predict all required information. Therefore, to guarantee the proper design, the worst case estimation is used. During the operation of the system it may occur that certain system properties may significantly differ from estimations or may dynamically change. It may be caused by too pessimistic estimation, by data-dependence or by some unpredictable events. In such cases the idea of self-adaptivity may be used to optimize some system properties

Although there are a lot of synthesis methods for low-power embedded systems [2], the problem of optimal mapping the task graph onto the multicore processor is rather a variant of the resource constrained project scheduling problem (RCPSP)[3] than the co-synthesis. Since the RCPSP is NP-complete only heuristic approach may be applied to real-life systems. Among the proposed heuristics for solving RCPSP, ones of the most efficient are methods based on genetic algorithms [4, 5]. For systems that may dynamically change during operation some methods of rescheduling are proposed [6, 7].

In this paper, we present a novel method for synthesis of a power-aware scheduler for real-time embedded systems. We assume that the function of the system is specified using the task graph that should be executed by the multicore processor supporting the big.LITTLE technology. The scheduler is generated automatically using the developmental genetic programming (DGP). The scheduler is self-adaptive, i.e. it dynamically reschedules tasks whenever any task finished its execution earlier or later than expected. In the first case, the goal of the rescheduling is the reduction in power consumption by moving some tasks to low-power cores. In the second case, the system is rescheduled to satisfy all time constraints by moving some tasks to high-performance cores. The example shows the benefits of using our methodology. The big.LITTLE technology is quite new and is mainly used in mobile devices. According to our best knowledge, there are no applications of this technology to design low-power real-time embedded systems as well as the adaptive scheduling method for such systems.

2. Developmental Genetic Programming

Genetic algorithms (GA) [8] are very commonly used in a wide spectrum of optimization problems. The main advantage of GA approach is the possibility of getting out from the local minima of the optimization criterion. Thus, GA is efficient for global optimization of complex problems, like RCPSP. Although GA approach usually gives satisfactory results, it may be inefficient for hard constrained problems. In these cases, a lot of individuals obtained using genetic operators correspond to not feasible solutions (e.g. schedule that exceeds the required deadline or incorrect schedule, in the RCPSP). Such individuals should not be considered during the evolution. It is obtained by defining the constrained genetic operators that produce only correct solutions. But such constrained operators may create infeasible regions in the search space. Such regions may contain optimal or close to optimal solutions.

The above problem may be eliminated by using the Developmental Genetic Programming (DGP)[9]. DGP is an extension of the GA by adding the developmental stage. This method was applied for the first time to optimize analog circuits. The main difference between DGP and GA is that in the DGP genotypes represent the method building the solution, while in the GA genotypes describe the solution. Thus, during the evolutionary process, a method of building a target solution is optimized, instead of a solution itself. In the DGP the search space (genotypes) is separated from the solution space (phenotypes). The search space is not constrained, all individuals are evolved. Thus, all of them may take part in the reproduction, crossover or mutation. There is no "forbidden" genotypes. Phenotypes are created by using the genotype-to-phenotype mapping function, which always produces a valid solution. During the evolution, the fitness of the genotype is evaluated according to the quality of the corresponding phenotype.

This idea of DGP is taken from biology, where the genotype corresponds to the chromosome containing information used for synthesis of proteins. The application of DGP has been successful in many domains, where human-competitive results have been obtained. The high efficiency of the DGP-based optimization has been also proved for hardware-software co-design and cost minimization in real-time cloud computing [10].

3. Synthesis of a power-aware adaptive scheduler

We use ARM multicore processors with big.LITTLE technology for implementation of the target systems. Such a system consists of two processors, usually quad-core. The first of them has higher

performance (about 40%), but consumes more power. The second one is slower, but is optimized to use much less energy (about 75%), to execute the same task. The goal of optimization is to find a makespan for which the power consumption is as small as possible, while all time constraints are met.

We assume that the system is specified as a task graph. This is a very widely used method of specification of real-time embedded systems. We also assume that for each task, the time of execution and the average power consumption is known for each available processor core. Usually these parameters are estimated using the worst case estimation methods. During the system operation, the scheduler will dynamically modify the schedule to minimize the power consumption, whenever it will be possible to move some tasks to low-power cores, i.e. when the execution time of the finished tasks will be shorter than estimated. In our method, it is also possible to use the average execution time, instead of the worst case estimation. When some task will be delayed, then the scheduler will try to find a new schedule that satisfies the time requirements. A sample task graph, describing the JPEG encoder, is given in Fig.1, while Table 1 presents the ARM Cortex-A15/Cortex-A7 database. For each core the execution time and the energy consumption is given.

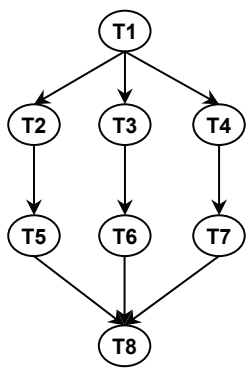


Fig. 1. Task graph for the JPEG encoder

Tab. 1. Resource library

Task	Low-power core		High-efficiency core	
	Time, ms	Energy, mJ	Time, ms	Energy, mJ
T1	60	7	32	23
T2	168	34	66	102
T3	168	34	66	102
T4	168	34	66	102
T5	64	8	35	26
T6	64	8	35	26
T7	64	8	35	26
T8	192	36	103	114

The idea of our approach is based on the observation that when the DGP will be applied for the RCPSP problem, then besides the final schedule we also obtain the scheduler dedicated to the optimized system. Thus, instead of the implementation of static schedule we may implement this scheduler, which may adapt to any perturbation during the system operation.

First, the DGP is used to find the optimal makespan. The method starts with an initial population that consists of randomly generated genotypes. The genotype has a form of a binary tree corresponding to the certain procedure of task assignment and task scheduling. In opposite to the classical genetic approach, genes do not correspond to task implementation but they specify preferences used by the scheduler. The details are given in [11].

The scheduling is performed during the genotype to phenotype mapping. This process is very fast, therefore it can be executed during the system operation. So, instead of implementing the final schedule we implement the method which creates this schedule. We observed that such an approach had great self-adaptivity capabilities, since the DGP had to consider only valid makespans. The genotype to phenotype mapping is a constrained process. If it is not possible to obtain the valid phenotype for a set of preferences defined by the genotype, then the mapping selects the next matching resource. Thus, the preferences specified by the genotype need not be strictly adhered. For example, if for a given task, the preference suggests assigning this task to the low-power core, then if this decision leads to an infeasible makespan, the scheduler will choose another, faster core that best matches to this preference. Therefore, the scheduler is able not only to build a correct solution, but also modify it if any unpredictable events occur. E.g. if the task execution is longer than expected, then the scheduler can move some tasks from a slower to a faster core, to fulfill time requirements. Similarly, if the task is finished before its predicted end time, the scheduler can move other tasks from a faster core to slower one, to save some energy.

4. Experimental results

We verified advantages of the presented method using the example from Fig.1 with the task characteristics given in Table 1. Quad-core and double-core processors were used. To evaluate the capabilities of self-adaptivity of the scheduler, we performed some simulations of different changes in execution times for some tasks. Table 2 presents the results obtained for the cases when the execution times were longer than estimated. In all the cases our scheduler was able to adapt to that situation and new makespans that satisfied the deadline were created. Table 3 presents the results obtained for other cases, where the execution times for some tasks were shorter than expected, i.e. estimation was too pessimistic. In such a case, the scheduler had an opportunity to reduce the power consumption.

Tab. 2. .Self-adaptation for task delays

Case	Delay	Deadline [ms]	Time without resche-duling, ms	Time after resche-duling, ms	Time-out, %	Energy, mJ
Quad-core						
0	(none)	300	293	-	0.0	451
1	T1+25%	300	308	279	2.67	505
2	T2+33%	300	315	286	5.00	505
3	T1 + 25% T3 + 25%	300	325	296	8.33	505
0	(none)	350	338	-	0.0	317
1	T1+40%	350	351	344	0.29	467
0	(none)	400	395	-	0.0	247
1	T1+25%	400	410	381	2.50	301
2	T2+33%	400	417	395	4.25	283
3	T1 + 25% T3 + 25%	400	452	381	13.00	301
Double-core						
0	(none)	400	395	-	0.0	383
1	T4+10%	400	413	383	3.25	401
2	T2 +50% T3 + 50%	400	458	395	14.50	401

Tab. 3. Self-adaptation for energy minimization

Case	Time decrease	Deadline, ms	Time, ms	Energy, mJ	Energy without rescheduling, mJ
Quad-core					
0	(none)	300	293	451	-
1	T1-50%	300	263	451	451
2	T1-T7 - 45%	300	297	373	451
0	(none)	350	338	317	-
1	T2-T4 - 33%	350	312	263	317
2	T3 - 40%	350	338	299	317
0	(none)	400	395	247	-
1	T2-T4 - 50%	400	400	169	247
2	T1 – T7 -30%	400	397	169	247
Double-core					
0	(none)	400	395	383	-
1	T4 – 50% T6 - 50%	400	400	305	383
2	T1 – T7 -30%	400	396	305	383

5. Conclusions

In this paper, a method of automatic synthesis of power-aware schedulers for real-time distributed embedded systems has been presented. Starting from the system specification in the form of the task graph, we use the developmental genetic programming to optimize the scheduling strategy that minimizes the power consumption. Finally, the best makespan as well as the optimized scheduler are generated. The scheduler has powerful capabilities of self-adaptation. This feature may be used to dynamically minimize the power consumption as well as to increase the system performance.

The presented method is dedicated to ARM big.LITTLE technology, developed for low-power systems. But, since we use a general optimization method, it could be easily adapted to other energy-efficient architectures. The computational experiments confirm that the schedulers, generated using DGP, are efficient and flexible. Our approach is also scalable. The experimental results show that the method may be also applied to complex embedded systems [12].

6. References

[1] big.LITTLE Processing with ARM Cortex™-A15 & Cortex-A7, ARM Holdings, September 2013, http://www.arm.com/files/downloads/big.LITTLE_Final.pdf.

[2] Luo J., Jha N.K.: Low Power Distributed Embedded Systems: Dynamic Voltage Scaling and Synthesis. Proc. 9th Int. Conference High Performance Computing — HiPC 2002, Lecture Notes in Computer Science, vol. 2552, 2002, pp. 679-693.

[3] Hartmann S., Briskorn D.: A survey of variants and extensions of the resource-constrained project scheduling problem. European journal of operational research: EJOR. Amsterdam: Elsevier, Vol. 207., 1 (16.11.), pp. 1-15 (2010).

[4] Xiang Li,Lishan Kang,Wei Tan: Optimized Research of Resource Constrained Project Scheduling Problem Based on Genetic Algorithms. Lecture Notes in Computer Science, Vol. 4683, 2007, pp. 177-186.

[5] Hossein Zoulfaghari, Javad Nematian, Nader Mahmoudi, and Mehdi Khodabandeh: A New Genetic Algorithm for the RCPSP in Large Scale. Int. J. Appl. Evol. Comput. 4, 2 (April 2013), 29-40.

[6] Van de Vonder S., Demeulemeester E.L., Herroelen W.S.: A classification of predictive-reactive project scheduling procedures. Journal of Scheduling 10 (3) (2007) 195–207.

[7] Al-Fawzan M., Haouari M.: A bi-objective model for robust resource constrained project scheduling. International Journal of Production Economics 96 (2005) pp.175-187.

[8] Michalewicz Z.: Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag Berlin Heidelberg, 1996.

[9] Koza J.R., Poli R.: Genetic Programming. In Edmund Burke and Graham Kendal, editors: Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques. Chapter 5. Springer, 2005.

[10] Deniziak S., Ciopiński L., Pawiński G.: Design of Real-Time Computer-Based Systems using Developmental Genetic Programming. In Handbook of Genetic Programming Applications, eds. Amir H. Gandomi, Amir H. Alavi, and Conor Ryan, Springer, 2015, in print.

[11] Sapiecha K., Ciopiński L., and Deniziak S.: An application of developmental genetic programming for automatic creation of supervisors of multi-task real-time object-oriented systems. IEEE Federated Conference on Computer Science and Information Systems (FedCSIS), 2014.

[12] Deniziak S. and Ciopiński L.: Synthesis of Power Aware Adaptive Schedulers for Embedded Systems using Developmental Genetic Programming. IEEE Federated Conference on Computer Science and Information Systems (FedCSIS), 2015.

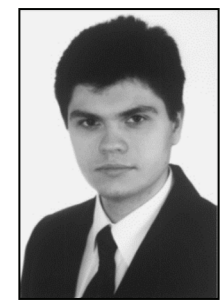
Received: 16.04.2015 Paper reviewed Accepted: 02.06.2015

Stanisław DENIZIAK, DSc, eng.

He is graduated of Faculty of Electronics of the Warsaw University of Technology, defended his doctoral thesis in 1994, and habilitation in 2006. He is the head of the Division of Computer Science of the Kielce University of Technology. His research interests include design of embedded systems, Internet of things, logic synthesis for FPGAs. He is a member of IEEE and IEEE Computer Society.

e-mail: s.deniziak@tu.kielce.pl

Leszek CIOPIŃSKI, MSc, eng.



He graduated of Faculty of Electrical Engineering, Automatic Control and Computer Science of the Kielce University of Technology in 2008. He is a research assistant in the Division of Computer Science of the Kielce University of Technology. His research interests include Developmental Genetic Algorithms, Resource-Constrained Project Scheduling Problem, Embedded Systems and Self-Adaptive Systems.

e-mail: l.ciopinski@tu.kielce.pl