Efficient Computing in Cyber-Physical Systems

Peter Marwedel
TU Dortmund, Informatik 12
44221 Dortmund, Germany
Email: peter.marwedel@tu-dortmund.de

Michael Engel
TU Dortmund, Informatik 12
44221 Dortmund, Germany
Email: michael.engel@tu-dortmund.de

***Abstract*—Computing in cyber-physical systems has to be efficient in terms of a number of objectives. In particular, computing has to be execution-time and energy efficient. In this paper, we will consider optimization techniques aiming at efficiency in terms of these two objectives. In the first part, we will consider techniques for the integration of compilers and worst-case execution time (WCET) estimation. We will demonstrate, how such integration opens the door to WCET-reduction algorithms. For example, an algorithm for WCET-aware compilation reduces the WCET for an automotive application by more than 50% by exploiting scratch pad memories (SPMs). In the second part, we will demonstrate techniques for improving the energy efficiency of cyber-physical systems, in particular the use of SPMs. In the third part, we demonstrate how the optimization for multiple objectives taken into account. This paper provides an overview of work performed at the Chair for Embedded Systems of TU Dortmund and the Informatik Centrum Dortmund, Germany[1].**

I. Introduction

One of the key trends in information technology is the increasing integration of the related techniques into products which are in continuous interaction with their physical environment. The resulting integrated systems have recently been called *cyber-physical systems* [1]. Embedded systems can be defined as information processing equipment integrated into an enclosing product [2]. Many of the basic characteristics of embedded systems also apply to cyber-physical systems. Hence, the distinction between cyber-physical systems and embedded systems should be clarified. From our point of view, the term embedded system describes the information processing part of the overall system. The combination with the physical environment makes up the cyber-physical system:
Cyber physical system =
physical environment + embedded system

The integration with the physical environment has a far-reaching impact on the requirements of embedded systems. This impact includes

- The need to consider *time constraints* seriously. The lack of adequate timing models in classical computer science was stressed by E. Lee in his well-known paper [3].
- For portable applications, there is usually a very limited availability of electrical energy available. Limited supply of energy has been found to be one of the most stringent constraints for the design of portable (and sometimes also for other) systems.
- The need to analyze the resulting reliability in-depth. For safety-critical applications, even small probabilities must be considered.
- Weight, cost, electromagnetic compatibility, environmental friendliness, security and other evaluation criteria may also have to be considered, but will be ignored in the remainder of this paper.

[1]This paper covers work which has been supported by Deutsche Forschungsgemeinschaft (DFG) in the context of Collaborative Research Center SFB 876 (project B2), in the context of DFG grant MA943/10-1, and by the Commission of the European Community under grants 216008 (PREDATOR) and 216224 (MNEMEE)

The impact of these criteria is frequently underestimated. In this paper, we will demonstrate approaches for addressing the first and the second issues. A global presentation of all possible approaches is clearly impossible for a single conference contribution. This demonstration will be based on results obtained at the Chair for Embedded Systems at TU Dortmund.

The paper is structured as follows: In section 2, we will describe techniques for optimizing worst case execution times (WCETs) in order to address the presence of real-time constraints. These techniques are also linked to techniques increasing the reliability of cyber-physical systems. In section 3, we will present techniques for reducing the energy consumption of embedded systems. Sections 2 and 3 contain separate subsections on related work and results. Section 4 provides an example of how the optimizations for minimizing the WCET and the energy consumption can be combined. Section 5 comprises an overall conclusion.

II. Efficiency in terms of worst case execution times

A. Related work

For computing devices not connected to the physical environment, except possibly to (patient) human users, long response times may be inconvenient, but will not be a safety risk. This is different for computing tightly integrated with the physical environment. For such computations, we must make sure that computations are finished in the time interval available. Unfortunately, in general purpose computing, the focus frequently is on the average execution time of algorithms. Moreover, theoretical computer science typically is based on a very abstract notion of time. In this notion, time does not even have a unit. So, no distinction is made between, for example, atto-seconds and Mega-years. The *O*-notation is another example of a concept inappropriate for cyber-physical systems. Its introduction increased the trend away from modeling real time and one can come to the conclusion that *the O-notation is considered harmful* for real-time systems.

Recently, commercial WCET-analysis tools (for example, aiT [4]) became available which compute safe upper bounds on the execution times. These tools help to prove that a system will meet given time constraints. However, such proofs are only possible after the software is generated. As a result, software is not optimized for efficiency in terms of WCET as the objective function. Hence, optimization potential may be missed.

In order to predict WCET values easily, it has also been proposed to design computing systems which exhibit exactly the same execution time independently of any input values. The resulting PRET (Precision Timed) machine is described in [5].

B. WCC: A WCET-aware compiler

Requiring all computations to execute in exactly the same amount of time is a strong restriction of the design space. For most applications, it is sufficient to guarantee that the WCET does not violate real-time constraints. Hence, computations may finish faster, and this provides additional design options not available with the PRET machine. Consequently, the resulting system may be more efficient than a PRET-based machine. In order to obtain an efficient system, software generation should also be aiming at the production of software which is efficient in terms of WCET as an objective. This leads us to consider WCET as an objective considered during compiler optimizations.

This idea was implemented in the worst-case execution time aware compiler WCC. In order to avoid the redesign of WCET-estimation tools, WCC is based upon an integration of a standard compiler structure with a commercial WCET-estimation tool, in this case aiT [4]. Fig. 1 shows the structure of the resulting WCET-aware compiler.

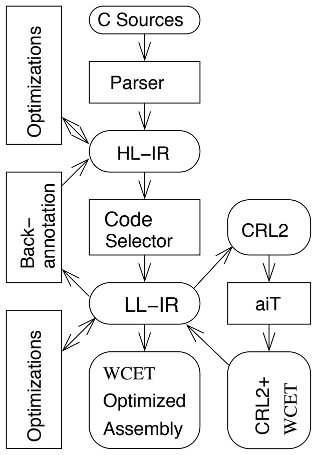

Fig. 1. Structure of WCC

The compiler components are displayed on the left, aiT on the right. WCC reads the source code (standard C code), parses it and stores it in a high-level intermediate representation HL-IR. This representation is used for the mapping to machine instructions (in our case exemplified for the TriCore™ architecture). As a result, we obtain blocks of code represented as machine instructions in a "low level intermediate representation" LL-IR. In order to enable WCET-based optimizations, this representation is converted into the format accepted by aiT, CRL2. This conversion is relatively easy, since HL-IR and CRL2 are both representations of blocks of machine instructions. Calls to aiT generate WCET information for the converted blocks and attach this information to the output of aiT. This output can then be converted back to LL-IR. This enhanced LL-IR information is the basis for optimizations aiming at a reduction of the WCET. Also, the WCET information can be back-annotated to blocks in the HL-IR representation. In this way, high-level optimizations like inlining and unfolding can take WCET information into account as well. The most time consuming part of this cooperation is the execution of aiT.

C. Results

The integration of a compiler with timing analysis opens new ground for optimizations regarding the WCET. All existing compiler algorithms can be reconsidered for their potential to reduce the WCET. From our point of view, this is urgently needed. But it is also sufficient and there is no need to guarantee the same execution time for all input data. We analyzed all the common optimizations like loop unrolling, inlining, etc. We found the largest optimization potential in an unexpected case, for register allocation. The studied register allocation is based on graph coloring. The graph coloring approach was made WCET-aware. Fig. 2 displays the results. Averaged over the various benchmarks, slightly over 30% speedup was observed.

Results for an "almost" industrial example were obtained in the PREDATOR project [6]. In this project, project partner Bosch provided the DEMOCAR software. This software is a prototype of software used in real automobiles, modeled after the real software (which was not available due to intellectual property issues). The DEMOCAR software includes a component controlling the ignition sequence of a combustion engine (IgnitionSWCSync). This runnable was compiled with WCC for the TriCore™ processor. The TriCore™ processor is frequently used in the automotive domain. It features a scratch pad memory (SPM). This SPM is exploited by WCC, but not by the gcc compilers. As a result, the WCET of this runnable could be reduced by about 50% compared to gcc.

III. Efficiency in terms of energy consumption

A. Related work

The amount of energy which is available for portable systems is considered as a very stringent constraint for the design of such systems. Many approaches for the reduction of the energy consumption have been published in recent

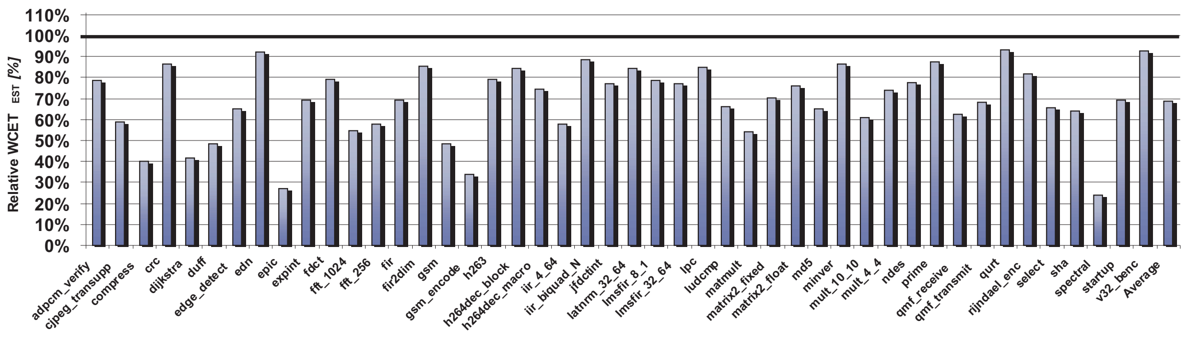

Fig. 2. Results for WCET-aware register allocation

years. Energy minimization techniques can be found at all levels of abstraction, from chip fabrication processes up to the application level. It is not possible to provide a representative overview in this paper. In our work, we found that a large potential exists in the exploitation of the memory hierarchy. As a rule of thumb, the smaller the memory, the less energy is consumed per access. Smaller memories are frequently also faster. This very naturally leads to the attempt to store frequently accessed objects in small but fast memories. In addition, techniques for speeding up the execution of algorithms are also reducing the energy consumption, unless they require an increase in the power consumption. Of course, the underlying reason is the fact that energy is computed as the integral of the power consumption over time.

Scratchpad memories (SPMs) are small and fast memories which are mapped into the address space of the system. Accesses to SPMs are inferred by the use of appropriate addresses, rather than by comparing tags (as for caches). Our own work led to a detailed identification of the benefits of SPMs [7]. However, SPMs require software techniques for mapping objects to these memories. Barua [8] and Kandemir [9] have performed extensive work on the exploitation of SPMs. In general, we distinguish between non-overlaying (frequently called "static") and overlaying (frequently called "dynamic") approaches to SPM allocation. For non-overlaying approaches, memory objects are allocated to space in the SPM before applications are executed and stay there during the lifetime of the application. For overlaying approaches, memory objects are copied back and forth between the layers in the hierarchy while the application is running.

B. *Exploitation of the memory hierarchy*

Early algorithms for the exploitation of SPMs typically use non-overlaying allocation, frequently also called static allocation. Simple versions of this allocation problem can be mapped to a knapsack problem. For more complex versions of the allocation problem, it is more appropriate to model the allocation problem as an *integer linear programming* (ILP) problem [10], [2].

For overlaying approaches, algorithms are typically based on an analysis of control-flow graphs (CFGs). For each of the edges of CFGs, the cost and the benefit of spilling variables to slower layers in the memory hierarchy is considered. Based on this information, heuristics or optimal approaches for spilling decisions can be taken. Our own work [11] is based on a global ILP model, which is optimal for the considered cost function. We consider the migration of data and code in order to maximize the optimization potential. For code, we consider the migration of entire functions and of basic blocks. Functions can be migrated easily by adding the corresponding pragmas in a pre-compiler phase. Popular compilers accept such pragmas and use them for the allocation of address space to functions. Basic blocks can be allocated to SPM address space by a pre-compiler run as well, if procedure-exlining (turning basic blocks into functions) is used.

Approaches mentioned so far are constrained to single threads. Extending the scope of optimizations for SPMs, we can take multiple threads into account. Thread context information comprises the information stored in SPMs. As a rule, this information must be saved and restored during context switches. As an exception, we can allocate dedicated space in the SPM for the different threads and avoid such copy operations. Verma et al. explore this idea and study the tradeoffs between different sizes of dedicated areas [12].

Verma's approach is still based on a fixed number of threads known at design time. These days, embedded applications are also becoming more and more dynamic, especially for smart phones. Hence, it makes sense to consider threads which are added during the life-time of systems. Research in our group also led to algorithms for such situations [13]. However, these algorithms require a level of indirection in order to access information which may be in the SPM. This indirection creates some overhead and also has a negative impact on timing predictability. Fortunately, smart phones can hardly be considered as cyber-physical systems and timing predictability is less important in this case.

The additional level of indirection can be avoided, if memory management units (MMU) can be used to map memory references dynamically either to SPM or some slower memory [14]. Unfortunately, page sizes supported by MMUs tend to become too large to be used with SPMs.

Achievable energy savings depend very much on the amount of memory accesses to the slow and large main memory. A variant of Amdahls law [15] can be used to compute the relative saving of energy:

$$\text{saving} = \frac{1}{(1-P)+\frac{P}{S}} \tag{1}$$

where P is the fraction of memory references replaced by more efficient ones and S is the improvement of the energy efficiency due to the smaller memory. For example, if 90% of the references can be replaced by more efficient ones and the smaller memory needs just $\frac{1}{100}$ times the energy of the larger memory, then

$$\text{saving} = \frac{1}{0.1+\frac{0.9}{100}} = \frac{1}{0.109} \tag{2}$$

The new energy consumption would be 10.9% of the original energy consumption. In currently (2012) available technologies, S may indeed be in the order of 100. In equation 1, the first term is the remaining energy consumption of the unoptimized part and the second is the energy consumption of the optimized part. The reciprocal value denotes the improvement over the initial version.

The correspondence to Amdahl's law highlights an important issue: just like for the original law, $(1-P)$ limits achievable improvements. Hence, it is important to enable the optimization of all memory references, including references to code, heap and stack data. Barua's approach includes techniques for handling heap and stack data. Nevertheless, the fraction $(1-P)$ of remaining references will frequently be the limiting term. Reductions of $(1-P)$ may be more important than increases of S.

C. *Energy efficiency of GPU computing*

In a project on resource-constrained machine learning, we have been analyzing images generated by a sensor for detecting (biological) viruses. Sophisticated image analysis techniques are required in order to achieve sufficient detection quality, since a high noise level is present in the sensor output. Due to the high computational load, these algorithms have been mapped to a graphics processing unit (GPU) [16]. We expected such a mapping to reduce the execution time of the algorithms, if compared to a general purpose CPU.

D. *Results*

For their mapping to SPMs, Steinke et al. [10] found energy reductions between 12% and 43% compared to caches. The reductions depended on the benchmark and were smallest for quicksort and largest for me_ivlin. The average reduction was 23%. The average performance gain was 16%.

For our approach considering multiple threads [12], energy reductions between 9% and 20% have been reported. In this case, the base line is an approach allocating the SPM space to the thread which leads to the maximum reduction in the energy consumption of the application.

Despite the overhead resulting from indirect addressing, our strategy for a varying set of threads [13] outperforms caches for most of the benchmarks. Typically, the SPM-based system was about 30% more energy efficient.

For air pollution simulation running on GPUs, we found a speed up of up to 132× and a reduction of the energy consumption to just 1.5%, compared to a CPU [16] (the corresponding values for image analysis still need to be computed).

Regarding the overall life cycle of PCs, we found that the energy consumed during the fabrication of PCs usually exceeded the energy consumed during the use of the PC [17]. We conclude that efforts for reducing the impact of chip fabrication on the environment need to continue.

IV. Efficiency in terms of multiple objectives and requirements

The discussion in sections 2 and 3 lead to the question: can techniques for achieving energy efficiency be combined with techniques for achieving WCET-efficiency? The latter should tend to reduce execution times. If such a reduction does not come with an increased power consumption, it should also tend to reduce the energy consumption.

In a similar way, we can also try to look at another combination of requirements: reliability and real-time constraints. Many error correction techniques are based on the assumption that once an error is detected, some amount of time is available to correct it. This assumption may be wrong for cyber-physical systems and unconditional attempts to correct errors might violate time constraints. However, correction may also not be necessary in cases where only a (possibly not noticeable) deterioration of result quality is effected. Therefore, error correction should not be attempted in such cases. In our work, we could demonstrate that we are indeed able to tag computations for which errors will only result in a deteriorated image quality [18].

In an effort to consider tradeoffs between multiple objectives, Lokuciejewski et al. analyzed dependencies between objectives during the selection of good combinations of compiler optimizations [19], [20]. Machine learning was used to find such good combinations. For the first time, an instance of strong correlation between average and worst case execution times could be demonstrated.

Multiple objectives have also been considered in recent work on the automatic parallelization of sequential software [21] for cyber-physical systems. Parallelization techniques are rapidly gaining importance since the high computing demands of future cyber-physical systems can only be satisfied using multiple processor cores. In contrast to conventional parallelization approaches, in cyber-physical systems it is often sufficient to achieve a given amount of speedup for an application, e.g., in order to adhere to given timing constraints. However, in many cases additional constraints, like energy consumption or code size, have to be considered. The work aims at load balanced exploitation of multi-processor platforms with a smaller number (typically < 10) of processors. It could be shown that speed-ups close to the number of processors could be obtained. Using this multi-objective parallelization approach, the designer of a cyber-physical system is enabled to consider the tradeoff between gains in speedup and increased energy consumption. An extension of this work considers different parallelization approaches, pipeline (loop-level) parallelization and task-level parallelization in common in order to enable finding improved solutions for the multi-objective optimization problems.

V. Summary and Future Work

In this paper, we demonstrated the fact that computations in cyber-physical systems have to be performed while taking the requirements imposed by the physical environment into account has far-reaching consequences for the implementation of these computations. In particular, time must be considered a first-class citizen. As a result, traditional approaches, for example for compiling, have to be questioned. We demonstrated that an integration of compilers with timing analysis is possible and that it can be achieved while leveraging recent results on timing analysis. Furthermore, energy efficiency remains a primary concern for portable systems and (considering green computing) beyond. We have also shown that energy efficiency and WCETs can be considered in combination.

Future work will include more optimizations considering multiple objectives. In addition to the objectives listed above, we will also take the quality of the algorithm results into account. For example, the quality of videos may be compromised due to the lack of time for error correction and the quality of pattern recognition techniques may be compromised due to the lack of available energy.

Acknowledgment

The authors would like to thank current and past PhD students of the Chair for Embedded Systems at TU Dortmund (Informatik 12) and Informatik Centrum Dortmund (ICD) upon whose work this paper is based.

References

[1] E. A. Lee, "Computing foundations and practice for cyber-physical systems: A preliminary report," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2007-72, May 2007. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2007/EECS-2007-72.html

[2] P. Marwedel, *Embedded System Design – Embedded Systems Foundations of Cyber-Physical Systems*. Springer, 2011.

[3] E. A. Lee, "Absolutely positively on time," *IEEE Computer*, Jul. 2005.

[4] Absint, "aiT worst-case execution time analyzers," *http://www.absint.de/ait*, 2010.

[5] S. A. Edwards and E. A. Lee, "The case for the precision timed (PRET) machine," in *Proceedings of the 44th annual conference on Design automation*. SESSION: Wild and crazy ideas (WACI), June 2007, pp. 264 – 265. [Online]. Available: http://chess.eecs.berkeley.edu/pubs/517.html

[6] Predator project team, "Predator – design for predictability and efficiency (home page)." [Online]. Available: http://www.predator-project.eu

[7] R. Banakar, S. Steinke, B.-S. Lee, M. Balakrishnan, and P. Marwedel, "Scratchpad memory: a design alternative for cache on-chip memory in embedded systems," *10th Intern. Symp. on Hardware/software Codesign (CODES)*, pp. 73–78, 2002.

[8] S. Udayakumaran, A. Dominguez, and R. Barua, "Dynamic allocation for scratch-pad memory using compile-time decisions," *ACM Transactions in Embedded Computing Systems*, vol. V, pp. 472–511, 2006.

[9] G. Chen, O. Ozturk, M. Kandemir, and M. Karakoy, "Dynamic scratch-pad memory management for irregular array access patterns," *Design, Automation and Test in Europe (DATE)*, pp. 931–936, 2006.

[10] S. Steinke, L.Wehmeyer, B.-S. Lee, and P. Marwedel, "Assigning program and data objects to scratchpad for energy reduction," *Design, Automation and Test in Europe (DATE)*, pp. 409–417, 2002.

[11] M. Verma and P. Marwedel, "Overlay techniques for scratchpad memories in low power embedded processors," *IEEE Trans. on Very Large Scale Integration*, vol. 14, no. 8, pp. 802–815, 2006.

[12] M. Verma, K. Petzold, L. Wehmeyer, H. Falk, and P. Marwedel, "Scratchpad sharing strategies for multiprocess embedded systems: A first approach," in *IEEE 3rd Workshop on Embedded System for Real-Time Multimedia (ESTIMedia)*, Sep. 2005, pp. 115–120.

[13] R. Pyka, C. Fabach, M. Verma, H. Falk, and P. Marwedel, "Operating system integrated energy aware scratchpad allocation strategies for multi-process applications," *Int. Workshop on Software & Compilers for Embedded Systems (SCOPES)*, pp. 41–50, 2007.

[14] B. Egger, J. Lee, and H. Shin, "Scratchpad memory management for portable systems with a memory management unit," *9rd ACM Intern. Conf. on Compilers, Architectures and Synthesis for Embedded Systems (CASES)*, pp. 321–330, 2006.

[15] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," *AFIPS spring joint computer conference*, 1967.

[16] C. Timm, A. Gelenberg, F. Weichert, and P. Marwedel, "Reducing the energy consumption of embedded systems by integrating general purpose GPUs," TU Dortmund, Faculty of Computer Science 12, Tech. Rep. 829, 2010.

[17] P. Marwedel and M. Engel, "Plea for a holistic analysis of the relationship between information technology and caron-dioxide emissions," in *Workshop on Energy-aware Systems and Methods (GI-ITG)*, Hanover, Germany, Feb. 2010.

[18] M. Engel, F. Schmoll, A. Heinig, and P. Marwedel, "Unreliable yet useful – reliability annotations for data in cyber-physical systems," in *Proceedings of the 2011 Workshop on Software Language Engineering for Cyber-physical Systems (WS4C)*, Berlin, Germany, Oct. 2011.

[19] P. Lokuciejewski and P. Marwedel, *WCET-aware Source Code and Assembly Level Optimization Techniques for Real-Time Systems*. Springer, 2010.

[20] P. Lokuciejewski, S. Plazar, H. Falk, P. Marwedel, and L. Thiele, "Multi-objective exploration of compiler optimizations for real-time systems," in *Proceedings of the 13th International Symposium on Object/Component/Service-oriented Real-time Distributed Computing (ISORC)*, Carmona, Spain, May 2010, pp. 115–122.

[21] D. Cordes and P. Marwedel, "Multi-objective aware extraction of task-level parallelism using genetic algorithms," in *Proceedings of Design, Automation and Test in Europe (DATE 2012)*, Dresden, Germany, Mar. 2012.