Institut für Technische Informatik

Abteilung Eingebettete Systeme

Universität Stuttgart
Pfaffenwaldring 5b
D-70569 Stuttgart

Masterarbeit Nr. 01936 - 00x

Heuristics for Design Time Optimization of System-on-Chip Memory Power Consumption

Jinpeng Li

Studiengang:	INFOTECH (Information Technology)
Prüfer:	Prof. Dr.-Ing. Martin Radetzki
Betreuer:	M.Sc. Manuel Strobel
begonnen am:	20.06.2016
beendet am:	24.11.2016
CR-Klassifikation:	B.8.2

Acknowledgment

Acknowledgment goes here...

Abstract

Abstract goes here...

Contents

1 Introduction

1.1 Memory power optimization

In the field of embedded systems design nowadays, power consumption becomes one of the most important design factors especially in the domain of Systems-on-Chip. One of the important issues to design power-efficient embedded system is the power consumed by memories and memory related components. Some researchers have claimed that large fraction of power is dissipated by memories [SER16; BMP00; Mai+07]. Thus, memory power optimization plays a significant role in the design of power-efficient embedded systems. One of the most effective and common approaches to reduce memory power consumption is the memory partitioning method which is proposed in several articles and books [SER16; BMP00; Mai+07; His05; MBP02, p.43].

The rationale of memory partitioning is , on the one hand, to split one single large memory into several small memory instances which can be accessed individually. On the other hand, according to the profiled memory access patterns, frequently accessed address ranges are grouped to smaller memory instances while seldom accessed address ranges are grouped to the larger ones. Therefor, the memory power optimization can be achieved by the facts that smaller memory instances consume less power and the larger memory instances are seldom accessed. There are two central concepts for memory power optimization using the memory partitioning method. One concept is the allocation α which is a set of memory instances of certain memory types. Memory types are defined by the physical characteristic parameters such instance size, area, read current and so on. The other concept is the binding β of application code and data fragments to the selected memory instances. The code and data fragments of an application are refereed as profiles of this application. And each application is represented by a set of profiles. Every profile is characterized by some user-defined parameters. Because all the code and data should be stored in the memories, each profile should be bound to exactly one memory instance [SER16]. A configuration for the memory system is defined as the combination of an allocation of memory instances and the corresponding binding for the application profiles. The goal of memory power optimization is to find a configuration among all possible configurations such that the overall power consumed by all selected memory instances is the lowest under certain predefined constraints.

Obviously, the memory power optimization is one of the combinatorial optimizations since the process is to find the optimal solution from a finite solution space of a problem set. There are many algorithms that can be applied to solve this kind of problems in the domain of combinatorial optimization. One approach is the integer linear programming (ILP) which solves the optimization problem through a mathematical model described by certain

integer linear relationship. This approach is proposed in [SER16] to solve the memory power optimization problem. Another commonly applied approach is the heuristic algorithm which is based on searching mechanisms. Many classical combinatorial optimization problems such as traveling sales man (TSP) problem have been solved by using heuristic algorithms.

1.2 Heuristics

Heuristic is a technique that searches for a near optimal solution of a optimization problem within a reasonable time. It is often used when the exact optimal solution can not be found by the conventional algorithms. When solving a optimization problem with a very large solution space, the algorithms trying to find the exact optimal solution may be ideal. However, the computation time of such algorithms may be not acceptable in practice. In such cases, the user of heuristic can find a good solution in a reasonable time. Though the solution provided by heuristic may not be the exact optimal one, it still can be considered as a valuable solution of the optimization problem. One key feature of heuristic is the trade-off between efficiency and precision. The solution quality and the computation time can be balanced by users according to their requirements.

There are a lot of existing heuristics. Some of them are problem-dependent that cannot be applied to other problems. And the most widely used algorithms are problem-independent which are usually called metaheuristic. Such algorithm usually consists of a base framework with several parameters. The framework is independent from the optimization problem sets while the parameters should be set up according to the problems. In the recent years, there is a new trend of heuristic which is called hyper-heuristic. The hyper-heuristic provides a high-level strategy to seek one or several low-level heuristics to generate a proper algorithm for solving an optimization problem. The hyper-heuristic is a cutting-edge technique and it is beyond the knowledge of this work. The metaheuristics are the main focus for the memory power optimization problem. To be clarified, the heuristics discussed in the rest of this work are metaheuristics.

There are a variety ways to classify the metaheuristics. One common classification is to differentiate the algorithms according to the searching mechanisms. To be simplified, the metaheuristics are divided as local search-based and non local search-based in this work. The well known local search algorithm aims to seek for the best solution by moving to a neighbor solution iteratively. However, the local search cannot guarantee providing the good enough solutions because it may trap in local optimums. The ideas of local search-based heuristics is to improve the solution quality of the local search algorithm using some criteria to for solution selection. Some classical local-search-based heuristics are simulated annealing, tabu search, guided local search, etc. The non local search-based heuristics usually seek for a set of good solutions. By manipulating some defined solution characteristics, it can guide the searching process to the global optimums. The typical non local search-based heuristics are genetic algorithm, particle swarm optimization, ant colony optimization, etc. In this work, local search-based heuristics are mainly focused and they are discussed in Chapter 3.

The aim of this thesis work is trying to select one promising heuristic for the purpose of solving the memory power optimization problem. Though the heuristics have been applied to solve a variety of optimization problems, a partial success is achieved in the memory power optimization due to the twoflod feature of the problem set. The rest of this work is organized as follows. Chapter 2 discusses the related researches for the memory power optimization and heuristics. In Chpater 3, some potential heuristics are theoretical examined and compared. Simulated annealing, the most promising algorithm, is proposed for the optimization objective. A detailed introduction for the simulated annealing algorithm is given in Chpater 4. Chpter 5 represents the implementation process for optimizing memory power consumption using simulated annealing. Chapter 6 discusses the evaluation of the partial results and the conclusion is represented in Chapter 7.

2 Related Work

3 Heuristics Selection

As introduced in Chapter 1, the local search-based heuristics are the main focus in this work. In this chapter, the local search algorithm along with its local optimum problem is discussed first. Then, the two typical local search-based heuristics, simulated annealing and tabu search, are represented. Last, the most promising algorithm is proposed to the memory power optimization according to the comparison between these heuristics.

3.1 Local search algorithm

Local search algorithm is one of the simplest heuristics. Given a optimization problem, it starts from an initial solution and searches in the current solution's neighborhood. If a better solution is found, it moves to that one. The searching process is repeated until the termination condition is satisfied. Then it outputs the current solution as the optimization result. Algorithm 3.1.1 shows the pseudo-code of local search process. There are four main steps in the algorithm. First step is finding a initial solution and set it as the current solution. The initial solution should be valid for the problem. In the second step, a neighboring solution is generated by certain mechanism. And the third step is to compare the neighboring solution with the current one through a object function. The object function is a method to indicate how good the solution is. The last step is the select criterion for solution. Local search selects the better one between the current and neighboring solution, which is a naive criterion. These two disadvantages make local search algorithm an improper choice when global optimal solution is required for the optimization problems.

Algorithm 3.1.1: Local Search Algorithm

Data: an optimization problem
Result: an optimal solution

1 initial solution = current solution;
2 **while** *not terminate* **do**
3   generate a neighboring solution;
4   evaluate the neighboring solution;
5   **if** *neighboring solution is better than current solution* **then**
6   current solution = neighboring solution;

7 output current solution;

Though local search algorithm is simple, the solution it provides may be the local optimal one. This is the major problem of local search algorithm. Figure 3.1 illustrates this local

optimum trap. Suppose the optimization problem is to find the solution with minimum cost, the local search algorithm starts with the initial solution a. The cost of neighboring solution b is lower than cost of a, then b becomes the current solution. The same searching process is repeated until the current solution reaches c. There is no better solution in c's neighborhood, thus the algorithm outputs solution c and terminates. However, solution c is only the local optimum and the global optimum is solution e which is not in c's neighborhood. In order to reach solution e, the algorithm has to move to solution d whose cost is higher than c's cost. And this violates the select criterion of the algorithm. Another drawback of local search algorithm is that the result quality is dependent on the initial solution. If the algorithm starts with solution d, the output will be the global optimum e.
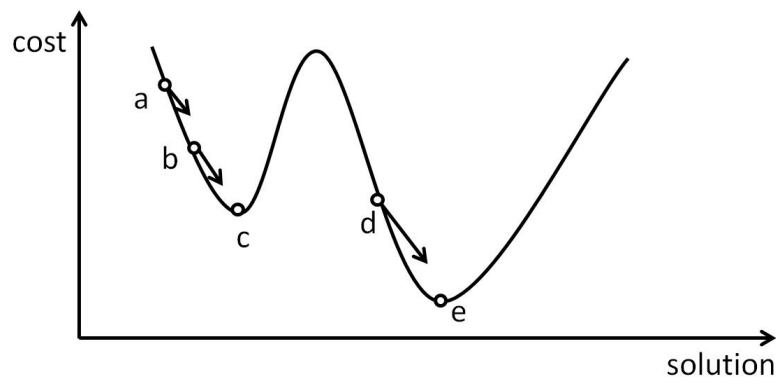

Figure 3.1: Local Optimum Trap

4 Simulated Annealing

This chapter introduces the simulated annealing.

A Appendix

Appendix goes here...

B List of Figures

C List of Abbreviations

MPSoC Multiprocessor System-on-Chip

Bibliography

[BMP00]   L. Benini, A. Macii, and M. Poncino. "A recursive algorithm for low-power memory partitioning". In: *Low Power Electronics and Design, 2000. ISLPED '00. Proceedings of the 2000 International Symposium on.* July 2000, pp. 78–83. DOI: 10.1145/344166.344518.

[His05]   Jason D. Hiser. "Effective Algorithms for Partitioned Memory Hierarchies in Embedded Systems". AAI3169653. PhD thesis. Charlottesville, VA, USA, 2005. ISBN: 0-542-05748-4.

[Mai+07] Songping Mai1 et al. "An application-specific memory partitioning method for low power". In: *2007 7th International Conference on ASIC.* Oct. 2007, pp. 221–224. DOI: 10.1109/ICASIC.2007.4415607.

[MBP02]   Alberto Macii, Luca Benini, and Massimo Poncino. *Memory Design Techniques for Low Energy Embedded Systems.* Springer Science & Business Media, 2002.

[SER16]   Manuel Strobel, Marcus Eggenberger, and Martin Radetzki. "Low power memory allocation and mapping for area-constrained systems-on-chips". In: *EURASIP Journal on Embedded Systems* 2017.1 (2016), p. 2. ISSN: 1687-3963. DOI: "10.1186/s13639-016-0039-5". URL: http://dx.doi.org/10.1186/s13639-016-0039-5.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben.

Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet.

Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht.

Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Unterschrift:

Stuttgart, 24.11.2016

Declaration

I hereby declare that the work presented in this thesis is entirely my own.

I did not use any other sources and references that the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations.

Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before.

The electronic copy is consistent with all submitted copies.

Signature:

Stuttgart, 24.11.2016