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:	27.11.2016
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.

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.

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 Basics

This is basics.

2.1 Formal power model

2.2 Heuristics

There are a lot of existing heuristics. At the early time of heuristics usage, the algorithms are applied to solve one particular optimization problem. These problem-dependent heuristics cannot be adapted to other optimization processes. To improve the portability of heuristics, some algorithms are invented as parameterized interface that can be widely deployed for a variety of optimization problems. Such problem-independent heuristics usually consist of a base framework with several parameters. Only the parameters are related to the optimization problems. When using one of those heuristics for different problem sets, the algorithm framework is common while the parameters should be set up according to the problem requirements. In the recent years, there is a new trend of heuristic which is called hyper-heuristic. The hyper-heuristics provide 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. For the memory power optimization, the problem-independent heuristics are the mainly focused because of its extensive usage.

There are a variety ways to classify the heuristics. One common classification is to differentiate the algorithms according to their searching mechanisms. To be simplified, the heuristics 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 optimal solution by iteratively moving to a better solution in the neighborhood. However, the local search is greedy and cannot guarantee providing the good enough solutions because it may trap in local optimums. The idea of local search-based heuristics is to avoid the local optimum trap through some criteria for solution selection and improve the result's quality. Heuristics of this kind output only one single optimal solution. Some classical local search-based heuristics are simulated annealing, tabu search, guided local search, etc. Unlike local search-based heuristics, the non-local search-based heuristics usually seek for a set of good enough solutions. By manipulating some defined solution characteristics, it can guide the searching process to the global optimums. Some typical non-local search-based heuristics are genetic algorithm, particle swarm optimization, ant colony optimization, etc. Normally, the frameworks of non-local search-based heuristics are more complicated than that of local

search-based algorithms. And the expected result for memory power optimization is one optimal configuration not a set of configurations. Therefor, the local search-based heuristics are the main focuses in this work. In this section, the local search algorithm along with its local optimum trap is discussed first. Then, two typical local search-based heuristics, tabu search and simulated annealing, are represented. Lastly, the most promising algorithm is proposed to the memory power optimization according to the comparison between these heuristics.

2.2.1 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, the current solution is replaced by it. The searching process is repeated until there is no better solution in the current solution's neighborhood. Then it outputs the current solution as the algorithm result. Algorithm 2.2.1 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 selection criterion for solution. Local search algorithm selects the better one between the current and neighboring solutions, which is a naive criterion.

Algorithm 2.2.1: Local Search Algorithm

```
Data: an optimization problem
Result: an optimal solution
1 current solution = initial 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 2.1 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 selection 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. These two disadvantages make local search algorithm an improper choice when global optimal solution is required for the optimization problems.

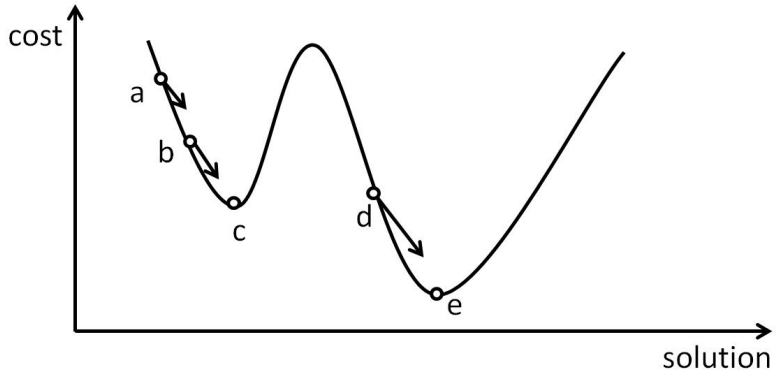

Figure 2.1: Local Optimum Trap

2.2.2 Tabu search algorithm

One of the improvements to local search is the tabu search algorithm. It is based on local search but it avoids to stuck at the local optimal trap through a different selection strategy for solutions. As discussed in section 2.2.1, once the local search algorithm is trapped at a local optimal solution, it cannot move any further due to the naive solution criterion. To solve this problem, Fred Glover proposed the concepts of tabu list and the aspiration criterion in [Glo89] and [Glo90].

The key element of tabu search is the tabu list. It imitates the memory function of human brain to guide the searching process. It is used to record the tabu objects. The tabu objects can be defined as the solutions, solution movements or values of the object function. Tabu list has a limited size which is one of the algorithm parameters. The improvement to local search algorithm is gained from the solution selection strategy that is usually called the tabu move. There are two rules in the tabu move. The first rule is to exclude the solutions recorded in the tabu list from a set of neighboring solutions. The second rule is to select the best in the rest of the neighboring solution set. The solution chosen by the tabu move is set as the current solution. Another concept of tabu search algorithm is the aspiration criterion. During the searching process, the best-so-far solution is kept recorded in the searching history. The aspiration criterion is to examine the neighboring solution set to find if there are solutions that are better than current best-so-far solution. If such solutions are found, then the best of them is selected and set as the current solution even if it is recorded in the tabu list. If no such solutions is found, the algorithm continues with the tabu move.

Algorithm 2.2.2 is the pseudo-code of tabu search framework based on the Fred Glover's proposal in [Glo89]. At the beginning of the algorithm, it generates a valid initial solution

Algorithm 2.2.2: Tabu Search Algorithm

Data: an optimization problem, algorithm parameters
Result: an optimal solution

```
current solution = initial solution;
best-so-far solution = initial solution;
set tabu list as empty;
set algorithm parameters;
while not terminate do
    generate a set of neighboring solutions;
    evaluate neighboring solutions;
    if aspiration criterion satisfied then
        update current solution;
        update tabu list;
        update best-so-far solution;
    else
        tabu move;
        update current solution;
        update tabu list;
output current solution;
```

and set it as the current solution and the best-so-for solution. The parameters are set up according to the algorithm inputs. And a tabu list is created as empty. After the initialization, the algorithm generates a set of neighboring solutions by some certain mechanism and evaluate it by an object function. After this step, there are two different branches. One branch is the execution of aspiration criterion. If the condition of the criterion is satisfied, the solution selected by the criterion is set as the current solution and the best-so-far solution. Also, the updated current solution is added to the tabu list. There are two cases for the tabu list updating. At the early stage of the algorithm, the list is not full. The solutions are added into the list sequentially. When there is no space for a new recorded solution, the oldest solution in the list is replaced by the new one. The other branch is executed when the condition of aspiration criterion is not satisfied, the algorithm continues with the tabu move. The current solution and the tabu list are updated with the solution chosen by the tabu. And the best-so-far solution is not updated because there is no solutions better than it. The same searching process is repeated until the termination condition is satisfied and the algorithm outputs the current or the best-so-far solution as the optimization result. Some algorithm parameters are related to the termination condition. One simple method to terminate the algorithm is to set a fixed iteration number. Thus this fixed number is one parameter of the algorithm. However, this method cannot guarantee the solution quality. Another common termination mechanism is to count the appearance of the current solution. If the current solution dose not change for a max number iterations, the algorithm can terminate. Therefor, this max number is also one algorithm parameter.

Figure 2.2 illustrates how the tabu search algorithm avoids the local optimum trap. The solid line represents the execution of the aspiration criterion and the dotted line is the tabu

move. The same problem set in section 2.2.1 is used. The tabu search algorithm starts from initial solution a with an empty tabu list (assume the list size is large enough). After the neighboring solutions are generated, it finds solution b satisfies the aspiration criterion. Then b becomes the current solution and the best-so-far solution is updated to b as well. Also, b is added to the tabu list. Known from the figure, solution b is a local optimum and there is no neighboring solution satisfies the aspiration criterion. Thus, the algorithm continues with the tabu move. During the tabu move, solution b is excluded from the generated neighboring solution set because it is stored in the tabu list. And solution c is found to be the best among the rests of the set. Thus, c becomes the current solution and it is added to the tabu list. In the next iteration, solution b and d both are c's neighboring solutions. However, b is still in the tabu list and the aspiration criterion is not satisfied. Thus, the current solution d is selected by the tabu move as it is the best among the rest of neighboring solutions. The same searching process is repeated until the algorithm reaches solution e which is the global optimum. By selecting a worser solution in the tabu move, the tabu search algorithm can move to a new searching region and avoid the local optimum trap.

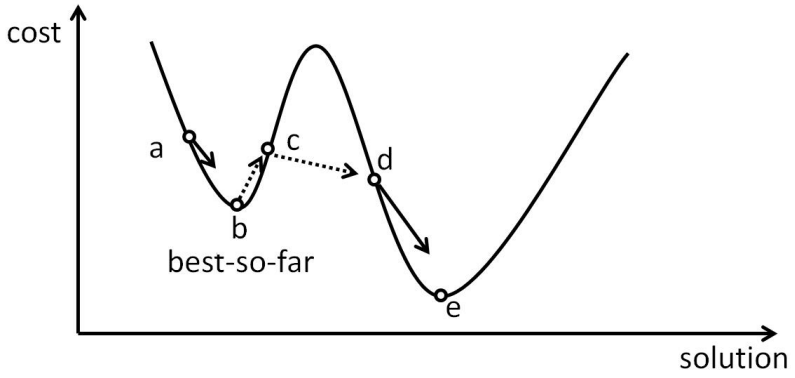

Figure 2.2: Tabu Search's Avoidance to Local Optimum Trap

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.

[Glo89] Fred Glover. “Tabu Search—Part I”. In: *ORSA Journal on Computing* 1.3 (1989), pp. 190–206. DOI: 10.1287/ijoc.1.3.190. eprint: http://dx.doi.org/10.1287/ijoc.1.3.190. URL: http://dx.doi.org/10.1287/ijoc.1.3.190.

[Glo90] Fred Glover. “Tabu Search—Part II”. In: *ORSA Journal on Computing* 2.1 (1990), pp. 4–32. DOI: 10.1287/ijoc.2.1.4. eprint: http://dx.doi.org/10.1287/ijoc.2.1.4. URL: http://dx.doi.org/10.1287/ijoc.2.1.4.

[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.

[S K83] M. P. Vecchi S. Kirkpatrick C. D. Gelatt. “Optimization by Simulated Annealing”. In: *Science* 220.4598 (1983), pp. 671–680. ISSN: 00368075, 10959203. URL: http://www.jstor.org/stable/1690046.

[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, 27.11.2016

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, 27.11.2016