Power-Aware VLIW Instruction Scheduling

Shu Xiao and Edmund M-K. Lai

School of Computer Eng., Nanyang Technological University, Singapore 639798

shu_x@pmail.ntu.edu.sg, asmklai@ntu.edu.sg

Abstract

This paper presents an efficient branch-and-bound algorithm for VLIW instruction scheduling that miminizes power consumption variations. Our experimental results show that our algorithm is much more efficient compared with previously presented approaches. We also briefly discuss a rough set approach to model imprecision inherent in an instruction-level power model.

Keywords: VLIW instruction scheduling, branch-and-bound algorithm, rough set.

1 Introduction

For VLIW (very long instruction word) processors, instantaneous power consumption can vary significantly different depending on how the parallel schedule is generated by the compiler. It is desired that instantaneous power variations in time be minimized in order to avoid excessive power supply noise and reduce battery exhaustion. Previously published works in this area are few. Yun [1] extended iterative modulo scheduling by adding a heuristic for power-aware scheduling of VLIW processor cores. Yang [2] proposed a mixed integer programming formulation to derive the optimal schedules. He used a commercial library (ILOG CPLEX) to obtain the solutions. However, due to lack of problem-specific information, the average time used to solve the mixed integer problem is quite unacceptable for ILP compilers. In this paper, we also formulate the instruction scheduling problem as a mixed-integer program. However, we developed a branch-and-bound algorithm to solve it efficiently. The algorithm is evaluated on a set of signal processing benchmarks of various sizes. The results show that our algorithm obtained the same optimal results as [2] but with much shorter computation times.

2 Mixed Integer Program

Given a power model for the processor and an instruction schedule X, the power deviation $PV(X)$ from the mean value can be calculated. The objective is to minimize this deviation. Assuming that we have an initial schedule generated from a speed optimized compiler, the number of instructions n and the number of time slots t for execution are known. We may then formulate problem as the mixed-integer program **P1**.

$$\mathbf{P1}: \qquad \min \ PV(X) \tag{1}$$

subject to

$$X = \bigcup_{i,k:x_i^k=1} \{x_i^k\} \tag{2}$$

$$\sum_{k=1}^{t} x_i^k = 1 \quad for\, each\ i = 1, ..., n \tag{3}$$

$$\sum_{k=1}^{t} kx_i^k + D_i - 1 \leq t \quad for\, each\ i = 1, ..., n; k = 1, ..., t \tag{4}$$

$$\sum_{i=1}^{n} a_i^j x_i^k \leq c_j, \quad for\, each\ j = 1, ..., u; k = 1, ..., t \tag{5}$$

$$\sum_{k=1}^{t} kx_m^k - \sum_{k=1}^{t} kx_l^k \geq D_l \quad \forall < l, m > \in E \tag{6}$$

The binary integer variables $x_i^k = 1$ if instruction i is allocated to time slot k. Constraint (3) makes sure that each instruction can only be issued once. (4) are deadline constraints, where D_i is the number of the execution stages of instruction i. For the resource constraints (5), u is the number of functional unit types and c_j is the number of functional units of type j. The binary variable $a_i^j = 1$ if instruction i corresponds to functional unit type j. (6) are the program flow dependency constraints, where v is the number of dependency pairs $< l, m >$.

3 Branch and Bound Algorithm

Starting with an initial schedule X_1, a sequence of schedules X_r are generated using the branch-and-bound algorithm until the optimal solution is found. The initial schedule X_1 can be generated by a conventional performance optimized scheduling algorithm. There are three important elements in the branch-and-bound algorithm: the rules for branching to a set of new schedule X_s from a certain schedule X_r, the lower bound estimate for a schedule X_r, and the rules for selecting a certain schedule X_r from the produced schedule pool.

Branching Rules: The task of branching is to generate a sequence of new schedules (branches) from a selected active leaf schedule (node) X_r, by rescheduling an instruction to all of its feasible time slots. The rescheduled instruction is one of the non-rescheduled instructions along the path from the initial schedule X_1 to the current selected one X_r. Therefore, the rules for branching can be divided into two categories. One category is used to determine which instruction to reschedule among the non-rescheduled instructions along the path from X_1 to X_r. The other is used to identify the feasible time slots of the selected instruction. These rules help to greatly cut infeasible branches, and thus the size of the produced schedule pool is greatly reduced.

Lower Bounds: Given a selected active leaf schedule X_r, a lower bound of the objective function is estimated to see if a better schedule may be found in its successors. If the lower bound of the objective function for X_r is worst (larger power variation) than the best objective value to date, then X_r is marked as "inactive" and this branch is cut. Otherwise, generate branches from X_r according to the branching rules.

Selection Rules: Compute the lower bounds of the objective function for the successors of each active leaf schedule. The active leaf schedule with smaller lower bound value of the objective function (lower power variation value) has higher priority.

4 Performance Evaluation

The proposed algorithm is compared with the general integer program solver in CPLEX90 used by [2], in terms of time used to find the optimal schedule. For comparison, we use the same power model as in [2]. We load the performance optimized schedule from Trimaran [3] ILP compiler and execute our algorithm to reschedule instructions to minimize the power variation across time steps. All instances used for testing were taken from the benchmarks with Trimaran. Instruction blocks with various problem dimensions (characterized by the number of instructions and the number of time slots as the performance deadline), ranging from $(6, 14)$ to $(35, 34)$, are selected. The results show, in comparison to using the integer program solver in CPLEX90 adopted by [2], the proposed algorithm obtains the same optimized power variation values while an improvement in required computation time ranging from 18.97% to 100.00% with an average of 80.20%.

5 Instruction-level VLIW Power Model

The scheduling problem formulation and our solution discussed above are independent of the power model of the processor. However, it does assume that the power is precisely known. In reality, the values of the parameters in the model are not precise for two main reasons. Firstly, physical measurements, which has been an important approach to instruction-level power modelling and estimation for microprocessors [4–6], are inherently imprecise. The variations in the measured values are usually handled by using the *mean* or *median* of a large number of measurements. Secondly, in order to reduce the complexity of the power model, those instruction with consume similar amounts of power are typically clustered together and given a the same power figure [7]. While these approximations allow us to optimize power consumption in the average sense, we are not able to get any idea of the deviations from the average that may actually occur. The recently developed rough set theory [8] approach can be applied to model the uncertainty inherent in the power model parameters. The instruction scheduling problem can be formulated as a rough program [**?**]. We can substitute (1) in **P1** by min $PV(X, \xi)$, where ξ denotes the set of the power consumption parameters described as rough variables rather than precise numbers. Then the rough returns of $PV(X, \xi)$ may be ranked by *1)* the expected value; *2)* the α-optimistic value or the α-pessimistic value, for some predetermined confidence level $\alpha \in (0, 1]$; *3)* the trust measure for some predetermined level $\overline{r}$.

6 Conclusion

The power balanced VLIW instruction scheduling problem is formulated as a mixed integer program. A branch-and-bound algorithm has been presented to solve this problem. This algorithm arrives at the optimal solution much faster than previously reported results for a set of example programs. The imprecision inherent in an instruction-level power model is briefly discussed. A rough set based approach is suggested to overcome the problems of parameter imprecision. It is a first attempt to apply rough set theory to instruction-level power modelling. Further work is currently being carried out in this direction.

References

[1] H. Yun and J. Kim, "Power-aware modulo scheduling for high-performance VLIW processors," in *Proc. Int. Symp. on Low Power Electronics and Design*, Huntington Beach, California, USA., Aug. 2001, pp. 40–45.

[2] H. Yang, G. R. Gao, and C. Leung, "On achieving balanced power consumption in software pipelined loops," in *Proc. Int. Conf. on Compilers, Architecture, and Synthesis for Embedded Systems*, Grenoble, France, Oct. 2002, pp. 210–217.

[3] Trimaran: An infrastructure for research in instruction-level parallelism. Hewlett Packard Laboratories, University of Illinois and Georgia Institute of Technology. [Online]. Available: http://www.trimaran.org

[4] J. T. Russell and M. Jacone, "Software power estimation and optimisation for high performance, 32-bit embedded processors," in *Proc. Int. Conf. on Computer Design*, Oct. 1998, pp. 328–333.

[5] C. Gebotys, "Power minimization derived from architectural-usage of VLIW processors," *Proc. Design Automation Conf.*, pp. 308–311, 2000.

[6] N. Julien, J. Laurent, E. Senn, and E. Martin, "Power consumption modeling and characterization of the TI C6201," *IEEE Micro*, vol. 23, no. 5, pp. 40–49, Sep.-Oct. 2003.

[7] A. Bona, M. Samit, D. Sciutos, C. Silvano, V. Zaccaria, and R.Zafalon, "Energy estimation and optimization of embedded VLIW processors based on instruction clustering," *Proc. Design Automation Conf.*, vol. 39, pp. 886–891, 2002.

[8] Z. Pawlak, *Rough Sets: theoretical aspects of reasoning about data.* Boston, MA: Kluwer Academic Publisher, 1991.