Instruction Scheduling of VLIW Architectures for Balanced Power Consumption

Shu Xiao and Edmund M-K. Lai
School of Computer Engineering
Nanyang Technological University, Singapore 639798
Email: shu_x@pmail.ntu.edu.sg, asmklai@ntu.edu.sg

Abstract— An instruction word in VLIW (very long instruction word) processors consists of a variable number of individual instructions. Therefore the power consumption variation over time significantly depends on the parallel instruction schedule generated by the compiler. Sharp power variations across time cause power supply noises, degrade chip reliability and accelerate battery exhaustion. This paper proposes a branch and bound algorithm for instruction scheduling of VLIW architectures that effectively minimizing power variation without degrading the speed. Our experimental results demonstrate the efficiency of our algorithm compared with previously presented approaches. Finally, a new rough sets based approach to the instruction-level VLIW power model for this instruction scheduling optimization problem is discussed.

I. Introduction

A significant amount of multimedia applications exhibit high instruction-level parallelism (ILP). A single instruction word of VLIW (very long instruction word) processors contains a variable number of individual instructions which are executed on different functional units in parallel. Therefore, VLIW processors provide a means to efficiently exploit such ILP of multimedia applications because of their parallel processing power. The instruction scheduling techniques used by VLIW compilers are thus essential to improve the execution speed. However, depending on the parallel schedule generated by the compiler, power consumption variation over time can also be significantly different. Since sharp power variations across time steps cause power supply noises, degrade chip reliability and accelerate battery exhaustion, power variation reduction without compromising execution speed becomes an important instruction scheduling constraint in embedded VLIW systems.

Some instruction scheduling algorithms for ILP processor have been proposed. However, most of them produce schedules that meet deadline constraints. Published works on instruction scheduling of ILP processors for power variation minimization are still relatively few. Among them, 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. In contrast with the heuristic in [1], the integer programming formulation has the potential to derive optimal results and it can be used to evaluate any heuristic algorithm. Yang [2] 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, as shown in our experimental data in Table I.

In this paper, the mixed integer program is also used to formulate the problem. However, we propose a branch and bound algorithm to solve this problem of scheduling of VLIW instructions without compromising the speed. The algorithm adaptively adds problem-specific constraints (cuts) and applies a tight and fast lower bound algorithm to accelerate convergence of the search for the optimal solution. The algorithm is evaluated on instruction blocks of various sizes. The results show an average improvement in power variation of 20.07% compared with the classical list scheduling algorithms and an average improvement in required computation time of 68.62% compared with [2].

Finally, previously published works about power aware instruction scheduling make use of power consumption models with parameters that are assumed to be precisely known [3, 1, 2, 4]. However, in reality, the values of these parameters 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 [5, 6, 7, 8], are always imprecise. The variations in the measured values are using 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 [9]. 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.

Today's instruction-level VLIW power models seem ill-equipped to exploit this opportunity. A new rough sets based approach to the instruction-level VLIW power model is proposed. In contrast to the previous model models, what's new here is the inaccuracy in instruction-level power models are also modelled, within the rough concepts which are ranges defined by their upper approximation and lower approximation. In this way, the instruction scheduling optimization problem will become an optimization problem based on rough sets system which can meet our goal.

The rest of this paper is organized as follows. In Section II we present our mixed integer programming formulation of the problem. A customized branch and bound algorithm is developed in Section III to solve the problem. Section IV presents the experimental evaluation results. Discussion about our improvement on VLIW power model is presented in Section V.

II. Formulation

A. Power Model

VLIW processors use very long instruction words to execute multiple instructions simultaneously on seperate functional units. Each instruction takes different amount of time to execute. We divide the time line into equal length time slots. A power cost p_i^j is associated with each instruction i which represents the power consumed by this instruction in the j-th time slot.

Given a program execution schedule N, let N_i represents the very long instruction word executed at the i-th time slot of N. Let n_j be an instruction in N_i. Then power consumption at the i-th time slot P^i is the sum of power consumed by all the executing instructions, either started in this time slot or before is given by

$$P^i = \sum_{0 \le k < i} \sum_{n_j \in N_{i-k}} p_{n_j}^k \tag{1}$$

where k is an integer.

B. Mixed Integer Program

The following notations will be used:

- N is the set of n target instructions to be scheduled.
- T is the set of t time slots, where t is the performance deadline.
- $x_i^k = \begin{cases} 1, \; if\, instruction\, i\, is\, allocated\, to\, time\, slot\, k \\ 0, \; otherwise. \end{cases}$
- X is the set of n variables x_i^k, which equal to 1.
- U is the set of u functional unit types.
- c_j is the number of functional units of type j.
- $a_i^j = \begin{cases} 1, \; if\, instruction\, i\, corresponds\, to \\ \quad functional\, unit\, type\, j \\ 0, \; otherwise \end{cases}$
- E is the set of v dependency pairs $< l, m >$, where instruction m depends on instruction l.
- D_i is the number of the execution stages of instruction i.
- P^k is the total power consumption in time slot k.
- M is the average power consumption over all the t time slots.

For the sake of simplicity we introduce the function

$$\varepsilon(x) = \begin{cases} 1 & if\, x \ge 1 \\ 0 & otherwise \end{cases}$$

where x is an integer.

According to the power estimation formula (1), power consumption at each time slot P^k and the average power consumption M can be computed by

$$P^k = \sum_{f=0}^{\min(D^{\max}-1, k-1)} \sum_{i=1}^{n} x_i^{k-f} \varepsilon(D_i - f) p_i^f \tag{2}$$

$$M = \left(\sum_{k=1}^{t} P^k \right) / t \tag{3}$$

where $D^{\max} = \max_{\forall i \in N}(D_i)$. Then, the power deviation at any given time slot k is computed as

$$PV^k(P^k) = |P^k - M| \tag{4}$$

with the total deviation given by

$$PV(X) = \sum_{k=1}^{t} PV^k(P^k) \tag{5}$$

The optimization problem of instruction scheduling of VLIW architectures for balanced power consumption can be formulated as a mixed integer program **P1** with objective function (6) and constraints (7)-(12).

$$\mathbf{P1}: \qquad \min\; PV(X) \tag{6}$$

subject to

$$X = \bigcup_{i,k: x_i^k = 1} \{x_i^k\} \tag{7}$$

$$x_i^k \in \{0, 1\} \quad for\, each\; i = 1, ..., n; k = 1, ..., t \tag{8}$$

$$\sum_{k=1}^{t} x_i^k = 1 \quad for\, each\; i = 1, ..., n \tag{9}$$

$$\sum_{k=1}^{t} kx_i^k + D_i - 1 \leq t \quad for\, each\ i = 1, ..., n; k = 1, ..., t \tag{10}$$

$$\sum_{i=1}^{n} a_i^j x_i^k \leq c_j, \quad for\, each\ j = 1, ..., u; k = 1, ..., t \tag{11}$$

$$\sum_{k=1}^{t} kx_m^k - \sum_{k=1}^{t} kx_l^k \geq D_l \quad \forall < l, m > \in E \tag{12}$$

The objective function (5) is given as a sum of the power deviations defined by (4) over all time slots. The set X is called a schedule. Condition (9) states that each instruction can only be issued once. Finally, (12) are dependency constraints, (10) are the deadline constraints and (11) are the resource constraints.

For the program **P1**, we need to find a feasible schedule X that minimize the value of the function $PV(X)$ with all the constraints satisfied. This problem is NP-complete. In the next section, we describe a branch and bound method that allows us to find a solution within a reasonable amount of time.

III. Branch and Bound Algorithm

Starting with an initial schedule X_1, a sequence of schedules X_r are generated until the optimal schedule is found. The initial schedule X_1 can be one produced through a conventional list scheduling algorithm.

The branch and bound algorithm requires three important elements:

1. the rules for branching to a set of new schedule X_s from a certain schedule X_r, and
2. the rules for selecting a certain schedule X_r from the produced schedule pool, and
3. the lower bound estimate for a schedule X_r.

A. Branching Rules

First, the larger the power consumed by an instruction, the earlier it is rescheduled. This helps to reduce the impact caused by relaxation of the integer constraints (8) and thus the lower bound obtained remains tight.

Second, once an instruction i is selected to be rescheduled, we propose four conditions to decide whether rescheduling to time slot j is feasible. Let X_s be the new schedule after i is rescheduled to j. The first condition is that for X_s, the power consumption for time slot j should not be larger than the peak power of the best schedule to date, expressed as

$$P_u^j \leq Peak^{X_z} \tag{13}$$

where $Peak^{X_z}$ is the peak power of the best schedule to date X_z.

Since constraints (12) only describe direct dependencies in the whole dependence graph, we need to extend them to include all indirect dependence information. Let

$$D_{i,j} = \begin{cases} L_{i,j}\ if\ j\ (in)directly\ depends\ on\ i \\ 0\ otherwise \end{cases} \tag{14}$$

where $L_{i,j}$ is the length of the maximum path from i to j in the dependence graph, if j (in)directly depends on i. Then, constraints (12) can then be extended as

$$\sum_{k=1}^{t} kx_j^k - \sum_{k=1}^{t} kx_i^k \geq D_{i,j} \quad \forall i, j \in N,\ if\ D_{i,j} > 0 \tag{15}$$

The second condition is that instruction i in time slot j should satisfy constraints (15).

Given constraints (10) and (15), those time slots in which instruction i definitely can not be allocated to can be expressed as

$$\begin{array}{r} x_i^k = 0 \quad k = 1, \ldots, D_i^{front}, D_i^{front} \geq 1 \\ x_i^k = 0 \quad k = t - D_i^{back} + 2, \ldots, t, D_i^{back} \geq 1 \end{array} \tag{16}$$

where

$$\forall i \in N, D_i^{front} = \max_{\forall h \in N}(D_{h,i})$$

$$D_{i,K} = \max_{\forall j \in N}(D_{i,j})$$

$$D_i^{back} = \max_{\forall j \in N}(D_{i,j}) + D_{i,K}$$

The third condition is that instruction i in time slot j should satisfy constraints (16).

The fourth condition is that instruction i in time slot j should satisfy the resource constraints (11). If any of these constraints is violated,instruction i cannot be scheduled to time slot j. Therefore the branch to reschedule i to j will not be included in the schedule pool.

These rules help to greatly cut infeasible branches, and thus the size of the produced schedule pool is greatly reduced.

B. Selection Rules

Given a pool of possible schedules, a random selection strategy is adopted. This means that we assume that the schedules in the pool has equal probability of leading to the optimal solution.

C. Lower Bounds

Given the current schedule X_r, its lower bound is estimated to check if a better schedule may be found in its successors which are generated by rescheduling those non-rescheduled instructions from X_1 to X_r. If the lower bound for X_r is larger than the best objective value to date, then X_r is removed from the schedule pool. Otherwise, replace X_r, in the schedule pool, with its branches produced according to the rules in Section A.

Let P_{total} denote the sum of power consumption over all the time slots. It can be expressed as

$$P_{total} = tM \tag{17}$$

Let

$$P_u^k = \sum_{f=0}^{\min(D^{\max}-1,k-1)} \sum_{i:x_i^{k-f} \in U_r} x_i^{k-f} \varepsilon(D_i - f) p_i^f \tag{18}$$

denote the power consumption in time slot k due to all the rescheduled instructions of the schedules from X_1 to X_r.

Now we introduce two constraints

$$\sum_{k=1}^{t} P^k - P_{total} = 0 \tag{19}$$

$$P^k - P_u^k \geq 0 \quad for\, k = 1, ..., t \tag{20}$$

Constraint (19) requires that the total power consumption over all time slots in X_r or any of its successors equals P_{total}. Condition (20) guarantees that each P^k must at least be the power consumption P_u^k associated with the rescheduled instructions in each time slot k.

The lower bound for the current schedule X_r can therefore be the minimum of (5) using the values of (17)-(20). Because integer constraints (8) are relaxed, execution of an instruction may be divided and scheduled to multiple time slots. Execution time in each time slot is less than a whole time slot. As a result, power consumption at each time slot P^k can be any values satisfying constraints (19) and (20). Then the obtained minimum of total power deviation (5) will not be a tight lower bound.

We reduce the impact caused by this relaxation by restricting how execution of an instruction may be divided. Among the non-rescheduled instructions in X_r, suppose the smallest power consumption of a single instruction is $P_{X_r}^{\Delta}$. Execution of an non-rescheduled instruction is divided into equal smaller slices with a residue if any. Each slice has a power consumption equal to $P_{X_r}^{\Delta}$. In this way, the values of P^k are limited to sums of some smaller execution slices. The algorithm to obtain the lower bound for the current schedule X_r is outlined as follows.

Algorithm III.1 *Lower bound of the current schedule* X_r

Let bound, $f^k (k = 1$ *to* $t)$*,* p *and* f^W *be variables used in this algorithm. The lower bound equals the value of bound when this algorithm stops.*

Step 1. $bound = 0$*. For* $k = 1$ *to* t*, calculate* P_u^k *according to (18). For* $k = 1$ *to* t*,* $f^k = P_u^k$*.*

Step 2. Calculate $P_{X_r}^{\Delta}$*, which is the smallest power consumption of a single instruction among the non-rescheduled instructions in* X_r*.*

Step 3. Given an non-rescheduled instruction in X_r*, let* p_i *be its power consumption.* $p = p_i$*.*

Step 4. $p = p - P_{X_r}^{\Delta}$*. Let* f^W *be the smallest among* f^k $(k = 1$ *to* $t)$*.* $f^W = f^W + P_{X_r}^{\Delta}$*. If* $p >= P_{X_r}^{\Delta}$*, then go to Step 4. Otherwise go to Step 5.*

Step 5. If all the non-rescheduled instructions in X_r *have been processed by Step 4, then go to Step 6. Otherwise go to Step 3.*

Step 6. If, for $k = 1$ *to* t*,* $f^k > M$*, then* $bound = f^k - M + bound$*.* $bound = 2 * bound$*. Stop the algorithm.*

IV. Performance Evaluation

The VLIW processor we used for evaluating our algorithm is the TMS320C6711 [10] which is a VLIW digital signal processor. The input to our algorithm is a schedule of an instruction block produced by Trimaran's ILP compiler. Trimaran [11] is an integrated compilation and performance monitoring infrastructure for research in instruction-level parallelism. Processor architecture can be specified through the machine description language HMDES. Our algorithm reschedules the one produced through Trimaran to minimize its power variation across time steps. Our branch and bound algorithm was implemented in C. All our computational experiments were conducted on an Intel Pentium 4 2.80GHz personal computer with 523,256KB RAM under Microsoft Windows 2000. All instances we used for testing our algorithms were taken from the benchmarks with Trimaran.

Table I shows the results of 18 problem instances. For each problem instance, the problem dimension (Dim.) indicates the number of time slots and the number of instructions involved in the instruction block. Power variation of the original schedule produced by Trimaran's ILP compiler is shown in the column "Root". The performance of our branch and bound algorithm is indicated by the optimal value of the objective function obtained ("Opt"), the number of nodes visited before the optimal schedule is found ("Nodes") and the total CPU time in seconds before the algorithm terminates ("TT"). Experiments of using CPLEX90 to solve this integer programming problem are also conducted. The computation time required are shown in the column "CPLEX".

"Ratio" is the percentage of power variation improvement obtained after optimization relative to the original schedule. "TOpt." is the percentage of computation time required by our algorithm relative to that of using CPLEX90 to solve this integer programming problem as done in [2]. We can see that power variation is greatly reduced through our optimization method. At the same time, the average computation time required by our problem specific branch and bound algorithm is much lower.

V. Discussion:A Rough Set Based Approach to Instruction-level VLIW Power Model

There are several approaches to deal with imprecision or uncertainty. In this paper, we propose to use the rough set

TABLE I
Experimental results on Trimaran's benchmark program

Dim.	Source	Root(mA)	Opt.(mA)	Ratio	Nodes	TT(sec.)	CPLEX(sec.)	TOpt.
(6,14)	Wave	146.00	127.33	12.79%	1070	0.03	0.05	40.00%
(11,11)	Fib	229.09	229.09	0	182	0.01	0.07	85.71%
(9,14)	Wave	224.89	130.67	41.90%	5965	0.11	0.13	15.38%
(13,13)	Wave	271.38	271.38	0	156	0.01	0.07	85.71%
(10,22)	Bmm	360.00	119.6	66.78%	907	0.10	0.19	47.37%
(15,15)	Fib-mem	313.6	313.6	0	482	0.01	0.10	90.00%
(12,19)	Fir	313.67	201.67	35.71%	537	0.06	0.10	40.00%
(12,22)	Bmm	386.00	64.00	83.42%	4108	0.54	2.16	75.00%
(17,16)	Fib-mem	234.35	206.35	11.95%	386	0.03	0.57	94.74%
(20,21)	Wc	408.60	408.60	0	3179	0.18	0.22	18.18%
(13,35)	Bmm	710.77	250.62	64.74%	4269	0.47	0.58	18.97%
(23,22)	Bmm	248.35	220.35	11.27%	753	0.09	5.43	98.34%
(23,24)	Bmm	474.43	474.43	0	782	0.06	0.27	77.78%
(25,24)	Bmm	251.52	223.52	11.13%	3095	0.34	19.29	98.24%
(29,29)	Bmm	608.28	608.28	0	396	0.04	1.44	97.22%
(31,30)	Bmm	258.58	230.58	10.83%	1499	0.26	10.04	97.41%
(33,33)	mm-dyn	692.36	692.36	0	4377	0.85	1.96	56.63%
(35,34)	mm-dyn	261.94	233.94	10.69%	6914	1.59	108.47	98.53%

theory [12] approach to model the uncertainty inherent in the power model parameters. The instruction scheduling problem can then be formulated as a rough program [13]. One of the main advantages of rough set is that it does not need any prior information on the data, such as probability distributions in statistics, basic probability assignment in the Dempster-Shafer theory [14], or grade of membership in fuzzy set theory [15]. Therefore, we propose to deal with the inaccuracy in instruction-level VLIW power modelling using rough set theory.

Definition V.1 *Let R^+ the set of nonnegative real, $S \subseteq R^+$ be a set of real $x_1, x_2, \ldots, x_i, \ldots$ such that $x_1 < x_2 < \ldots < x_i < \ldots$. $\pi(S)$ is a partition of R^+: $\pi(S) = \{0, (0, x_1), x_1, (x_1, x_2), x_2, (x_2, x_3), x_3, \ldots, x_i, (x_i, x_{i+1}), x_{i+1}, \ldots\}$. For $\forall x \in R^+$, we define the following mapping*

$$S(x) = \begin{cases} \{x\},\ if\ x \in S \\ \{(x_i, x_{i+1})\},\ if\ x \notin S,\ x_i < x < x_{i+1} \end{cases} \tag{21}$$

Then, (R^+, S) is called an approximation space defined by $\pi(S)$ on R^+.

Then, the basic idea to deal with inaccuracy in instruction level VLIW power model is described in the following definition.

Definition V.2 *Let (R^+, S) be the approximation space. Let X be a power consumption parameter of the instruction-level VLIW power model. Instead of being expressed as an accurate value, the estimated value of X is expressed as a range limited by a pair of concepts, called its lower and upper approximation in approximation space (R^+, S). Its lower approximation is defined as*

$$S_*(X) = \{x \in R^+ : S(x) \subseteq X\} \tag{22}$$

Its upper approximation is defined as

$$S^*(X) = \{x \in R^+ : S(x) \cap X \neq \Phi\} \tag{23}$$

All the power consumption parameters in the instruction-level VLIW power model are expressed as ranges defined with rough sets (22) and (23). As a result, the instruction scheduling optimization problem **P1** in Section II is based on rough sets. Traditionally, the quantities involved in an optimization problem are given in precise values. For a rough set problem, the conventional optimization techniques have to be adapted. However, there are few published work for this problem [13]. Future work involves detailed power consumption modelling method using this theory and algorithms to solving the instruction scheduling optimization problem based on this formulation.

VI. Conclusion

The VLIW instruction scheduling for optimized power balance problem without degrading the speed of the program is being formulated as a mixed integer programming problem. A branch and bound algorithm has been presented to solve this problem. The effectiveness of this algorithm is demonstrated through a set of example programs. The results show an average improvement in power variation of 20.07% compared with the classical list scheduling algorithms and an average improvement in required computation time of 68.62% compared with [2].

A rough set based approach to instruction-level VLIW power estimation with the inaccuracy modelled is discussed. The benefits and initial model are outlined.

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] M. T. Lee, V. Tiwari, S. Malik, and M. Fujita, "Power analysis and minimization techniques for embedded DSP software," *IEEE Trans. on Very Large Scale Integration Systems*, vol. 5, no. 1, pp. 123–135, Mar. 1997.

[4] C. Lee, J. K. Lee, T. T. Hwang, and S. C. Tsai, "Compiler optimization on VLIW instruction scheduling for low power," *ACM Trans. on Design Automation of Electronic Systems*, vol. 8, no. 2, pp. 252–268, Apr. 2003.

[5] V. Tiwari, S. Malik, and A. Wolfe, "Power analysis of embedded software a first step toward software power minimization," *IEEE Trans. on Very Large Scale Integration Systems*, vol. 2, no. 4, pp. 437–445, Dec. 1994.

[6] 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.

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

[8] 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.

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

[10] *TMS320C6000 CPU and instruction set reference guide*, Texas Instruments, Oct. 2000, reference Guide, SPRS088E.

[11] 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

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

[13] B. Liu, *Theory and practice of uncertain programming.* Heidelberg: Physica-Verlag, 2002.

[14] G. Shafer, *A Mathematical Theory of Evidence.* Princeton: Princeton University press, 1976.

[15] L. A. Zadeh, "Outline of a new approach to the analysis of complex systems and decision processes," *IEEE Trans. Syst., Man, Cybern.*, vol. 3, pp. 28–44, Jan. 1973.