VLIW INSTRUCTION SCHEDULING FOR DSP PROCESSORS BASED ON ROUGH SET THEORY

Shu Xiao, Edmund M-K. Lai and A. P. Vinod

School of Computer Engineering, Nanyang Technological University, Singapore 639798

ABSTRACT

Power-balanced instruction scheduling for Very Long Instruction Word (VLIW) processors is an optimization problem which requires a good instruction-level power model for the target processor. Conventionally, these power models are deterministic. In this paper, we propose a rough program problem formulation to handle the imprecision involved in the VLIW power models. A problem-specific genetic algorithm is proposed to solve it. Our experiments revealed that the actual occurrence of optimal schedules obtained by integer programming often have a large deviation of the objective function values, due to the ignorance of imprecision accumulation. The results justified our rough programming approach to find an optimal schedule which makes sure that the effects of these imprecision could be minimized.

1. INTRODUCTION

Power-aware instruction scheduling for very long instruction word (VLIW) processors is the task of producing a schedule of VLIW instructions so that the average power consumption is minimized or the power variation over the execution time of the program is minimized, while the deadline constraints are met [1–4]. An appropriate instruction-level power model is needed for this problem. In all currently published models, the power parameters given precise deterministic values [5–8]. Since these models are estimated from empirical measurements, there will always be some degree of imprecision. The variations in the measured values are usually handled by using the *mean* or *median* of a large number of measurements. Furthermore, in order to reduce the complexity of the power model, some approximation techniques such as instruction clustering [9] have to be employed which contributes to the imprecision involved.

These average value power models, however, simplifies the instruction scheduling formulation. Effective solutions can be obtained by forumulating it as a mixed-integer program which can be solved efficiently [4]. While these approximations using the average values allow us to optimize power consumption in the average sense, they are not good enough in power critical applications. For example, one cannot guarantee that the optimal schedule obtained with the average value models that a hard power variation limit for chip reliability will not be exceeded in live situations. Therefore, it is desirable to take into account the imprecision involved in the power models.

One possible approach is to formulate the scheduling program using uncertain programming. There are several approaches to uncertain programming [10]. They include stochastic programming, fuzzy programming and rough programming. In this paper, we focus on the use of rough programming to the power-aware VLIW instruction scheduling problem. Rough programming is based on rough set theory [11]. One advantage of rough set techniques is that they do not need any prior information on the data. This is in contrast with statistics which requires the assumption of prior probability distributions. Similarly, basic probability assignments are needed for those based on the Dempster-Shafer theory. For fuzzy set theory, fuzzy membership functions are required. The scheduling problem is formulated as a chance-constraint rough program. A problem-specific genetic algorithm (GA) is proposed to solve it.

The rest of the paper is organized as follows. The rough programming formulation for this optimization problem is described in Section 2. Section 3 presents our problem-specific genetic algorithm. Our experimental results are presented in Section 4.

2. PROBLEM FORMULATION

The reader is referred to [10] for a formal definition of a rough variable. A power parameter p_i can be expressed as a rough variable in the form $([a,b],[c,d])$ where $[a,b]$ is its lower approximation and $[c,d]$ its upper approximation and $c \leq a \leq b \leq d$ are real numbers. Then the scheduling problem can be formulated as the following chance-constraint rough program:

P1: $\min\ P(X,\xi)_{inf}(\alpha)$
subject to

$$\begin{array}{ll} X = \bigcup x_i^k & i=1,...,n; k=1,...,t \\ x_i^k \in \{0,1\} & i=1,...,n; k=1,...,t \end{array} \quad (1)$$

$$\begin{array}{c} G(X) \leq 0 \\ L(X) = 0 \end{array} \quad (2)$$

The objective function is defined by

$$P(X,\xi)_{inf}(\alpha) = inf\{\overline{P} | Tr\{P(X,\xi) \leq \overline{P}\} \geq \alpha\} \quad (3)$$

An instruction schedule X is a set of binary decision variables x_i^k, which has a value of 1 if instruction i is allocated to time slot k; otherwise its value is 0. n is the number of instructions in X and t is the number of time slots available. $G(X) \leq 0$ and $L(X) = 0$ are the processor resource constraints, and data dependency and instruction deadline constraints respectively. The function $P(X,\xi)$ returns the power variation of X given the set of power consumption parameters ξ of the target processor. Since the elements of ξ are rough variables, the values of this function are also rough. The rough values of this function are ranked by its α-pessimistic value $P(X,\xi)_{inf}(\alpha)$ which is is the smallest value $\overline{P}$ satisfying $Tr\{P(X,\xi) \leq \overline{P}\} \geq \alpha$. This means that, for a given X, the rough return of $P(X,\xi)$ will be below the pessimistic value $\overline{P}$ with a confidence level of α. The formal definition of the trust measure operator $Tr()$ can be found in [10]. A rough event A must hold if its trust measure $Tr(A)$ is 1, and fail if its trust measure $Tr(A)$ is 0.

Solving this program involves searching for the minimum α-pessimistic value $P(X,\xi)_{inf}(\alpha)$ among all feasible schedules X. Since the objective function to be optimized is multimodal and the search space is particularly irregular, genetic algorithm [12] is an appropriate tool for obtaining solutions to this rough program. Next, we propose a problem-specific genetic algorithm to solve the rough program.

3. IMPLEMENTATION OF GA

3.1. Chromosome Encoding and Initial Population

Each chromosome is an array of integer variables each representing an instruction. The integer value indicates the execution time slot allocated to that instruction.

An initial population of candidate schedules is a set of feasible schedules created randomly. "Seeded" with schedules obtained through conventional scheduling algorithms, the function *randomchange_1instruction()* generates a new schedule by changing the allocated time slot of an instruction randomly. This process is repeated until we have a whole population of initial feasible chromosomes.

3.2. Fitness Evaluation

Rough simulation [10] plays an important role in rough systems. In order to compute $P(X,\xi)_{\text{inf}}(\alpha)$ for a candidate X, the following rough simulation process is used. Let R be the sample size. For each power consumption parameter $p_i \in \xi$ $(i = 1,2,3,\ldots)$, randomly take R samples from its lower and upper approximations, $l_i^k (k = 1,\ldots,R)$ and $u_i^k (k = 1,\ldots,R)$, respectively. The value of the function $P(X,\xi)_{\text{inf}}(\alpha)$ is given by the minimum value of v such that

$$\frac{l(v) + u(v)}{2R} \geq \alpha \quad (4)$$

where $l(v), u(v) \leq R$ denote the number of samples that satisfy

$$P\left(X, l_1^k, \ldots, l_i^k, \ldots\right)_{\text{inf}}(\alpha) \leq v$$

and

$$P\left(X, u_1^k, \ldots, u_i^k, \ldots\right)_{\text{inf}}(\alpha) \leq v$$

respectively.

3.3. Selection, Crossover, and Mutation

Our design for chromosome selection adopts the rank-based roulette-wheel selection scheme [12]. The chromosomes are sorted in non-increasing order of fitness. The ith chromosome is assigned a probability of selection by a nonlinear function, $q(i) = a(1-a)^{i-1}$. The actual selection is done using the roulette wheel procedure as in Figure 1.

```
1 $q_0 = 0$;
2 for $i \leftarrow 1$ to pop_size do
3     Calculate accumulative probabilities for the $i$th
      chromosome $q_i \leftarrow \sum_{j=1}^{i} q(j)$;
4 end
5 Generate a random number $r$ within $[0, q_{pop\_size}]$;
6 Select the $i$th chromosome such that $p_{i-1} < r < p_i$;
```

Algorithm 1: Chromosome selection by roulette wheel.

The selection process is performed for whole population. The selected parents for crossover operation are denoted by $V_1^{"}, V_2^{"}, V_3^{"}, \ldots$ and divided into pairs:

$$(V_1^{"}, V_2^{"}), (V_3^{"}, V_4^{"}), (V_5^{"}, V_6^{"}), \ldots$$

We use a 2-point crossover operator which chooses 2 cutting points at random and alternately copies each segment out of the two parents. The crossover process may produce unfeasible schedules due to violations of constraints described in Section 2. To avoid the creation of infeasible schedules, constraints check operators have been included.

If feasible offsprings cannot be created, the parents will not be replaced.

To prevent premature convergence, a mutation process is used by randomly change the allocated time slot of an instruction to obtain a new feasible schedule. When a predetermined number of generations is reached, the algorithm stops. The maximum number of generations depends on the size of the problem, i.e. the number of instructions and the number of available time slots.

4. EXPERIMENTAL RESULTS

The target processor is the TMS320C6711 which is a VLIW digital signal processor. We conducted measurements for each power consumption parameter using the experimental setup as in [13]. A set of experiments have been generated where each experiment is to measure the processor core current running a program composed of a sequence of instructions. Different instruction instances are considered in terms of opcode, operands, conditional registers, cross registers, functional units or inter-instruction effect. Based the measured data, we characterize the power consumption parameters as rough variables described by their lower and upper approximations which encapsulate the imprecision involved. Rosetta Toolkit [14] is a rough set theory tool to analyze data. The possible current values on real line are discretized using the Boonlean reasoning algorithm and then the lower and upper approximations for each power consumption parameter are computed.

Rough program formulation and GA are tested using the digital signal processing benchmarks from Trimaran [15]. Conventional scheduling algorithms in Trimaran is used to produce the non-power-aware schedules for comparison. The confidence level α is set to 0.9. In our problem, the crossover operators often cannot create enough feasible offsprings due to violations of the constraints. Therefore, in order to prevent premature convergence, the crossover probability is set to a low value 0.2 while the mutation rate is set to a high value 0.8. The population size pop_size are 30. Tuning these parameter values to be suitable for a specific data set may give rise to improved performance of the GA.

For any given target instruction block, we conduct instruction scheduling by means of mixed integer programming and rough programming separately. Current deviations (from the mean) of the schedules obtained by rough program formulation are compared with those of the schedules obtained by mixed integer program formulation, with respect to their deviations of their objective function values under all realization of the power parameters. Next, we give a simple example to illustrate this comparison.

Example 1 *Consider an instruction block consisting of fourteen instructions:* $\{$*addaw, add, addaw, add, ldw, mv, addaw, stw, b, addaw, cmpeq, stw, ldw, b*$\}$.

Table 1. Rough power consumption parameters in Example 1.

p_{addaw}, p_{add}, p_{mv}, p_{cmpeq}	p_{ldw},p_{stw}	p_b
$(\emptyset, [190, 214])$	$(\emptyset, [214, 233])$	$(\emptyset, [190, 207])$

Using an average value model, the scheduling problem is formulated as a mixed-integer program and solved by a branch and bound algorithm as in [4]. The optimal schedule is given by $X_{MIP} = \{x_1^1, x_2^1, x_3^4, x_4^1, x_5^2, x_6^4, x_7^1, x_8^2, x_9^4, x_{10}^5, x_{11}^5, x_{12}^5, x_{13}^6, x_{14}^6\}$ *where the superscripts indicate the time slot in which the instruction is being scheduled.*

The power parameters of the target processor are represented as rough variables as shown in Table 1. The scheduling problem is formulated as given by **P1**. *The optimal schedule obtained using the proposed genetic algorithm outlined in Section 3 is given by* $X_{RP} = \{x_1^1, x_2^1, x_3^4, x_4^1, x_5^2, x_6^5, x_7^2, x_8^3, x_9^4, x_{10}^5, x_{11}^5, x_{12}^5, x_{13}^6, x_{14}^6\}$

Conducting rough simulation of these two schedules with the possible realization of the power parameters, we have

$$P(X_{RP}, \xi)_{inf}(0.9) = 14798 \qquad (5)$$

$$P(X_{MIP}, \xi)_{inf}(0.9) = 17713 \qquad (6)$$

where ξ *denotes the set of power consumption parameters described in Table 1. (5) means, with confidence level* 0.9, *the current deviations (from the mean) of schedule* X_{RP} *are less than 14798 under all the possible realization of the power parameters in* ξ. *(6) means, with confidence level* 0.9, *the current deviations (from the mean) of schedule* X_{MIP} *are only less than 17713 under all the possible realization of the power parameters in* ξ. *We can see that the schedule obtained by integer programming is far from a globally optimal one if considering all the possible realization of the power parameters.*

Table 2 shows the comparison results of the schedules obtained from more problem instances with different problem dimensions. For each problem instance, the problem dimension (Dim.) indicates the number of time slots and the number of instructions respectively in the instruction block. The objective function values of the optimal schedules obtained through MIP (Column ”MIP”) are generally much larger than those obtained through rough programming (Column ”RP”). This implies that the optimal schedules obtained by integer programming often have a larger deviation of the objective function values, with all the possible realization of the power parameters considered. It is a result of the ignorance of uncertain in the power parameters. The rough programming approach takes parameter imprecision into account in a natural way.

Table 2. Experimental results on instruction blocks of various sizes from Trimaran's benchmark program.

Dim.	Source	MIP(mA)	RP(mA)	Improvement (%)
(6,14)	Wave	17713	14798	16.5
(9,14)	Wave	48225	30552	36.6
(13,14)	Fir	99643	82177	17.5
(10,20)	Bmm	94124	71452	24.1
(14,15)	Bmm	106120	72826	31.4
(10,22)	Bmm	72757	45416	37.6
(12,19)	Fir	107432	83827	22.0
(12,22)	Bmm	137611	104337	24.2
(17,16)	Fib-mem	141390	130557	7.7
(19,23)	Fir	255739	230823	9.7
(21,21)	Bmm	240921	212049	12.0
(13,35)	Bmm	228399	134942	40.9
(23,22)	Bmm	273203	221874	18.8
(25,24)	Bmm	286887	168803	41.2
(27,24)	Fir	316431	152204	51.9
(31,30)	Bmm	537007	491859	8.4
(35,34)	mm-dyn	704769	666751	5.4

5. CONCLUSIONS

Rough programming has been applied to the problem of power-balanced VLIW instruction scheduling with power model uncertainties. We formulated the scheduling problem as a chance-constraint rough program and a problem-specific genetic algorithm is developed to solve it. The experimental results show that the schedules obtained through rough programming have real power variations which are guaranteed to be within the optimal values with the desired level of confidence. This is in contrast with the conventional approach using mixed-integer programming where the power variations of the obtained instruction schedules cannot be guaranteed.

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

[4] S. Xiao and E. M-K. Lai, "A branch and bound algorithm for power-aware instruction scheduling of VLIW architecture," in *Proc. Workshop on Compilers and Tools for Constrained Embedded Syst.*, Washington DC, USA, Sept. 2004.

[5] H. Mehta, R. M. Owens, and M. J. Irwin, "Instruction level power profiling," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, 1996, vol. 6, pp. 3326–3329.

[6] V. Tiwari, S. Malik, A. Wolfe, and M. T. Lee, "Instruction level power analysis and optimization of software," in *Proc. Int. Conf. on VLSI Design*, Jan. 1996, pp. 326–328.

[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] V. Zaccaria, M. Sami, D. Sciuto, and C. Silvano, *Power estimation and optimization methodologies for VLIW-based embedded systems*, Kluwer, Boston, 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.*, New Orleans, USA, 2002, pp. 886–891.

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

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

[12] D. E. Goldberg, *Genetic algorithms in search, optimization and machine learning*, Addison-Wesley, Boston, MA, 1989.

[13] 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: VLSI in Computers & Processors*, Oct. 1998, pp. 328–333.

[14] J. Komorowski, A. Skowron, and A. Øhrn, "The Rosetta toolkit," in *Handbook of Data Mining and Knowledge Discovery*, W. Kl Ed. Oxford University Press, 2000.

[15] "Trimaran: An infrastructure for research in instruction-level parallelism," .