Rule-based Power-balanced VLIW Instruction Scheduling with Uncertainty

Shu Xiao, Edmund M-K. Lai and A.B. Premkumar

School of Computer Engineering, Nanyang Technological University
Singapore 639798
shu_x@pmail.ntu.edu.sg, asmklai@ntu.edu.sg, asannamalai@ntu.edu.sg

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. However, in reality, there will always be some degree of imprecision involved. For power critical applications, it is desirable to find an optimal schedule which makes sure that the effects of these uncertainties could be minimized. The scheduling algorithm has to be computationally efficient in order to be practical for use in compilers. In this paper, we propose a rule based genetic algorithm to efficiently solve the optimization problem of power-balanced VLIW instruction scheduling with uncertainties in the power consumption model. We theoretically prove our rule-based genetic algorithm can produce as good optimal schedules as the existing algorithms proposed for this problem. Furthermore, its computational efficiency is significantly improved.

1 Introduction

Power-balanced instruction scheduling for very long instruction word (VLIW) processors is the task of producing a schedule of VLIW instructions so that the power variation over the execution time of the program is minimized, while the deadline constraints are met. Most currently instruction scheduling techniques for this problem are based on deterministic power models [1–3]. Since these instruction level models are estimated from empirical measurements [4, 5], there will always be some degree of imprecision or uncertainty. Furthermore, in order to reduce the complexity of the power model, some approximation techniques such as instruction clustering [6] have to be employed which contributes to the imprecision involved. While these instruction scheduling techniques using the approximate deterministic power models allow us to optimize power consumption in the average sense, it is desirable to find an optimal schedule which ensures that the effects of those uncertainties could be minimized for power critical applications.

A rough programming approach has previously been proposed to solve this problem [7–9]. This approach is shown in Fig. 1. An instruction-level power modelling technique which applies rough set theory is used to handle the uncertainties involved [7, 8]. Then the power-balanced instruction scheduling problem based on the rough set based power model is formulated as a chance-constraint rough program [7,9]. This program was previously solved by a genetic algorithm (GA) [9]. In order to rank the schedules produced in each generation of the genetic algorithm, rough simulation is used [10]. It is a simulation process that estimates the ranges of the objective function values for a given schedule given a confidence level. It is very expensive computationally. Thus while this technique is of interest academically it is not suitable for practical use.

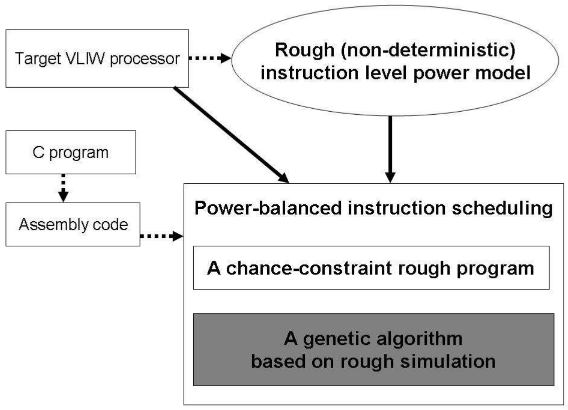
Target VLIW processor
Rough (non-deterministic) instruction level power model
C program
Assembly code
Power-balanced instruction scheduling
A chance-constraint rough program
A genetic algorithm based on rough simulation

Fig. 1. A rough programming approach proposed in [7–9].

In this paper, we propose a rule-based genetic algorithm to solve the above optimization problem much more efficiently. The steps involved in this new method is summarized in Fig. 2. It ranks the generated schedules by simply comparing the differences between the objective function values of these schedules using a set of rules instead of rough simulation. These rules are generated by learning the decision making process of rough simulations. This rule extraction process, though computationally expensive since it involves rough simulation, only needs to be performed once off-line for a particular target processor. Thus the computation time involved in the instruction scheduling phase is much reduced.

We shall review the rough power model, the chance-constraint rough program formulation of the problem and the existing GA to solve it in Sections 2, 3 and 4 respectively. In Section 5.1, our rule-based GA is presented. We proved mathematically that the solutions obtained by the rule-based GA are as good as those obtained using the existing GA. The rule extraction method is discussed in Section 5.2. Then examples are given to illustrate how the improved GA using these rules is able to improve the computational efficiency substantially.

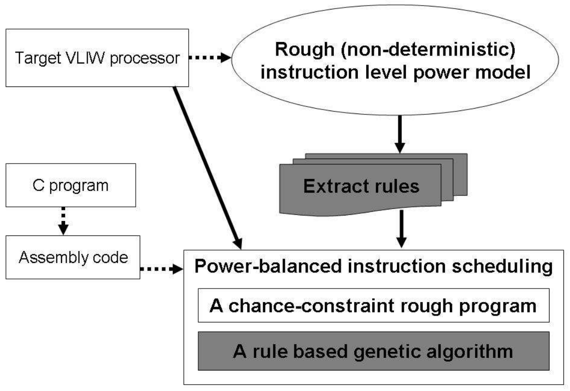
Target VLIW processor
Rough (non-deterministic) instruction level power model
C program
Extract rules
Assembly code
Power-balanced instruction scheduling
A chance-constraint rough program
A rule based genetic algorithm

Fig. 2. A rule based genetic algorithm to solve the chance-constraint rough program.

2 Rough Instruction Level Power Model

Suppose a program schedule X consists of T time slots:

$$X = < w_1, ..., w_{i-1}, w_i, ..., w_T >$$

where w_i is the long instruction word in the i-th time slot. A most common approach to estimate the power consumed in time slot i during the execution of X is

$$P_i \approx U(0|0) + \sum_{k=1}^{F} v^k_{(i|i-1)} \tag{1}$$

Here, $U(0|0)$ is the base power cost which is the power consumed by the execution of an instruction word constituted entirely by no-operation (NOP) instructions. F is the number of functional units in the target VLIW processor. the summations of $v^k_{(i|i-1)}$ is the additional power contributions on the F functional units due to the change of instructions on the same functional unit in the time slot i. The number of instruction pairs to be considered for $v^k_{(i|i-1)}$ in (1) could become too large to be characterized, since two instructions differ either in terms of functionality (i.e., opcode), addressing mode (immediate, register, indirect, etc.), or data differences (either in terms of register names or immediate values).

The complexity can be reduced by instruction clustering, that is instructions are categorized into classes, such that the instructions in a given class are characterized by very similar power cost. Then a simplified model from (1) can be obtained:

$$P_i \approx U(0|0) + \sum_{k=1}^{C} r_k c_k \tag{2}$$

where the additional power consumption to $U(0|0)$ is computed as the summations of the power due to the being executed instructions in different clusters in the time slot i. C is the number of instruction clusters. c_k is the power consumption parameter representing power consumption of instructions in the cluster k. r_k represents the number of being executed instruction belong to the cluster k in the time slot i.

However, simply by instruction clustering each c_k only represents an average power consumption for the instructions in the same cluster with different opcodes, addressing modes, operands or the preceding opcodes. In order to indicate the uncertainty involved in each power consumption parameter, each c_k is expressed as a rough variable represented by $([a, b], [c, d])$. $[a, b]$ is its lower approximation and $[c, d]$ its upper approximation and $c \leq a \leq b \leq d$ are real numbers [8]. This means that the values within $[a, b]$ are sure and those within $[c, d]$ are possible.

3 Chance-Constraint Rough Program Formulation

The total power deviation for a schedule X is proportional to the power squared and is given by

$$PV(X) = \sum_{i=1}^{T} (P_i - M)^2 \tag{3}$$

where the average power over T time slots is given by

$$M = \left(\sum_{i=1}^{T} P_i \right) / T \tag{4}$$

This is the function we seek to minimize.

If each c_k in (2) is represented as a rough variable, then the return values of P_i in (2), M in (4) and $PV(X)$ in (3) are also rough. They can be ranked by their α-pessimistic values for some predetermined confidence level $\alpha \in (0, 1]$. Our optimization problem needs to make sure that the effects of the uncertainties could be minimized. Thus a large enough confidence level is required.

Definition 1. *Let ϑ be a rough variable given as $([a, b], [c, d])$, and $\alpha \in (0, 1]$. Then*

$$\vartheta_{\inf}^{\alpha} = \inf \{ r | Tr\{\vartheta \leq r\} \geq \alpha \} \tag{5}$$

is called the α-pessimistic value of ϑ, where Tr is the trust measure operator, defined as

$$Tr\{\vartheta \leq r\} = \begin{cases} 0, & r \leq c \\ \frac{1}{2}\left(\frac{c-r}{c-d}\right), & c \leq r \leq a \\ \frac{1}{2}\left(\frac{c-r}{c-d} + \frac{a-r}{a-b}\right), & a \leq r \leq b \\ \frac{1}{2}\left(\frac{c-r}{c-d} + 1\right), & b \leq r \leq d \\ 1, & r \geq d \end{cases} \tag{6}$$

Combining (5) and (6), we have

$$\vartheta_{\text{inf}}^{\alpha} = \begin{cases} (1-2\alpha)c + 2\alpha d, & 0 < \alpha \le \frac{a-c}{2(d-c)} \\ 2(1-\alpha)c + (2\alpha - 1)d, & \frac{b+d-2c}{2(d-c)} \le \alpha \le 1 \\ \frac{c(b-a)+a(d-c)+2\alpha(b-a)(d-c)}{(b-a)+(d-c)}, & \frac{a-c}{2(d-c)} < \alpha < \frac{b+d-2c}{2(d-c)} \end{cases} \tag{7}$$

The chance-constraint rough program is given by $\mathbf{P_{rp}}$.

$\mathbf{P_{rp}}$: $\quad \min \ PV(X,\xi)_{inf}^{\alpha}$

subject to

$$X = \bigcup x^j, \quad j = 1, ..., N \tag{8}$$

$$1 \le x^j \le T, \quad j = 1, ..., N \tag{9}$$

$$\xi = \bigcup c_k, \quad k = 1, ..., C \tag{10}$$

$$G(X) \le 0 \tag{11}$$

$$L(X) = 0 \tag{12}$$

The objective function defined by

$$PV(X,\xi)_{inf}^{\alpha} = inf\{q | Tr\{PV(X,\xi) \le q\} \ge \alpha\} \tag{13}$$

is based on its α-pessimistic value where α is the large enough confidence level. Let N denote total number of instructions. A schedule X can be represented by a set of integer variables x^j, which denote the allocated time slots for these instructions. ξ is the set of rough power consumption parameters defined in (2). (8), (11) and (12) define the constraint matrix for the processor-specific resource constraints, and data dependency constraints.

4 Existing GA Solution

Since the objective function of a rough program is multimodal and the search space is particularly irregular, conventional optimization techniques are unable to produce near-optimal solutions. Based on the general GA framework proposed in [10], a problem-specific GA has been proposed [9] to solve the formulation in Section 3. The three main elements of this algorithm are outlined as follows.

1. **Initial Population**: An initial population of candidate schedules is a set of feasible schedules created randomly and "seeded" with schedules obtained through conventional (non-power-aware) scheduling algorithms.
2. **Fitness Evaluation and Selection**: The objective function defined by (13) is used to evaluate the fitness of schedules in each generation. Then they are sorted non-decreasingly in terms of their fitness. In order to compute the objective function (13) of a given schedule X, the following rough simulation process is used. Let R be the sampling size. For each power consumption parameter $c_i \in \xi$ $(i = 1, 2, 3, \ldots)$, randomly take R samples from its lower and

upper approximations, $l_i^k (k = 1, \dots, R)$ and $u_i^k (k = 1, \dots, R)$, respectively. The value of the function $PV\,(X, \xi)_{\inf}^{\alpha}$ is given by the minimum value of v such that

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

where $l\,(v)$ denotes the number of $PV\left(X, l_1^k, \dots, l_i^k, \dots\right)_{\inf}^{\alpha} \leq v$ being satisfied when $k = 1, \dots, R$ respectively; and $u\,(v)$ denotes the number of $PV\left(X, u_1^k, \dots, u_i^k, \dots\right)_{\inf}^{\alpha} \leq v$ being satisfied when $k = 1, \dots, R$ respectively. The rough simulation process is summarized as in Algorithm 1.

```
input : A feasible schedule $X$; lower and upper approximations of each
        power consumption parameter $c_i \in \xi$; confidence level $\alpha$; let $R$ be
        the sampling size
output: Return of $PV(X, \xi)_{inf}^{\alpha}$
1 for $k \leftarrow 1$ to $R$ do
2     foreach power consumption parameter $c_i \in \xi$ do randomly sample $l_i^k$
      from its lower approximation;
3     foreach power consumption parameter $c_i \in \xi$ do randomly sample $u_i^k$
      from its upper approximation;
4 end
5 $l\,(v) \leftarrow$ the number of $PV\left(X, l_1^k, \dots, l_i^k, \dots\right)_{\inf}^{\alpha} \leq v$ being satisfied when
  $k = 1, \dots, R$ respectively;
6 $u\,(v) \leftarrow$ the number of $PV\left(X, u_1^k, \dots, u_i^k, \dots\right)_{\inf}^{\alpha} \leq v$ being satisfied when
  $k = 1, \dots, R$ respectively;
7 Find the minimal value $v$ such that
      $\frac{l\,(v) + u\,(v)}{2R} \geq \alpha$
8 Return $v$;
```

Algorithm 1: Rough Simulation algorithm.

3. **Crossover and Mutation**: The selected parents are divided into pairs and crossovered using 2-point crossover operator. The motivation for using mutation, then, is to prevent the permanent loss of any particular bit or allele (premature convergence).

When a pre-determined 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.

5 Rule-based GA Solution

5.1 Theoretical Basis

The rough simulation algorithm provides an estimate of the range of the objective function values of a schedule by simulating the possible values for every power consumption parameter for the whole duration of the program. Thus the larger the sample size, the better the simulation result. The lengthy computation makes this GA not practical for use in compilers.

However, we note that for fitness evaluation and selection, we don't have to compute the objective function values defined by (13) for each schedule. We actually only need to know the relative amounts. In this section, we shall establish the theoretical basis for obtaining the differences in the objective function values of a set of generated schedules. Lemma 1, Theorem 1 and Corollary 1 also prove that we can obtain the same result in fitness evaluation and selection as that given by rough simulation. By removing the time consuming rough simulation, the computational efficiency of the GA is significantly improved.

Lemma 1. *Given two rough variables $y = ([a_y, b_y], [c_y, d_y])$ and $z = ([a_z, b_z], [c_z, d_z])$, let $u = x + y$. Given a confidence level $\alpha \in (0, 1]$ which satisfies*

$$\alpha \geq \max\left(\frac{b_y + d_y - 2c_y}{2(d_y - c_y)}, \frac{b_z + d_z - 2c_z}{2(d_z - c_z)}\right)$$

we have

$$u^\alpha_{\text{inf}} = x^\alpha_{\text{inf}} + y^\alpha_{\text{inf}} \tag{14}$$

Proof. Since u is the sum of x and y, the lower and upper approximations of u are computed by adding the values of the corresponding limits (see rough variable arithmetics in [10]):

$$u = ([a_y + a_z, b_y + b_z], [c_y + c_z, d_y + d_z])$$

u^α_{inf}, y^α_{inf} and z^α_{inf} are the α-pessimistic values of u, y and z respectively. Based on (7), these values are given by

$$u^\alpha_{\text{inf}} = \begin{cases} (1-2\alpha)(c_y+c_z) + 2\alpha(d_y+d_z), & \alpha \leq \frac{a_y+a_z-c_y-c_z}{2(d_y+d_z-c_y-c_z)} \\ 2(1-\alpha)(c_y+c_z) + (2\alpha-1)(d_y+d_z), & \alpha \geq p_u \\ \frac{(c_y+c_z)(b_y+b_z-a_y-a_z)+(a_y+a_z)(d_y+d_z-c_y-c_z)}{(b_y+b_z-a_y-a_z)+(d_y+d_z-c_y-c_z)} \\ \quad + \frac{2\alpha(b_y+b_z-a_y-a_z)(d_y+d_z-c_y-c_z)}{(b_y+b_z-a_y-a_z)+(d_y+d_z-c_y-c_z)}, & \text{otherwise} \end{cases} \tag{15}$$

$$y^\alpha_{\text{inf}} = \begin{cases} (1-2\alpha)c_y + 2\alpha d_y, & \alpha \leq \frac{a_y-c_y}{2(d_y-c_y)} \\ 2(1-\alpha)c_y + (2\alpha-1)d_y, & \alpha \geq p_y \\ \frac{c_y(b_y-a_y)+a_y(d_y-c_y)+2\alpha(b_y-a_y)(d_y-c_y)}{(b_y-a_y)+(d_y-c_y)}, & \text{otherwise} \end{cases} \tag{16}$$

$$z^\alpha_{\text{inf}} = \begin{cases} (1-2\alpha)c_z + 2\alpha d_z, & \alpha \leq \frac{a_z-c_z}{2(d_z-c_z)} \\ 2(1-\alpha)c_z + (2\alpha-1)d_z, & \alpha \geq p_z \\ \frac{c_z(b_z-a_z)+a_z(d_z-c_z)+2\alpha(b_z-a_z)(d_z-c_z)}{(b_z-a_z)+(d_z-c_z)}, & \text{otherwise} \end{cases} \tag{17}$$

where

$$p_u = \frac{b_y + b_z + d_y + d_z - 2(c_y + c_z)}{2(d_y + d_z - c_y - c_z)}$$
$$p_y = \frac{b_y + d_y - 2c_y}{2(d_y - c_y)}$$
$$p_z = \frac{b_z + d_z - 2c_z}{2(d_z - c_z)}$$

Note that $p_u < \max(p_y, p_z)$. Hence if $\alpha \geq \max(p_y, p_z)$, then $\alpha \geq p_u$. In this case, we have

$$y^{\alpha}_{inf} + z^{\alpha}_{inf} = u^{\alpha}_{inf} \tag{18}$$

This completes the proof.

Remark 1. We do not need to consider the cases $\alpha \leq p_y$ and $\alpha \leq p_z$ in (16) and (17) in Lemma 1. Our optimization problem requires that the effects of the uncertainties could be minimized. Thus a large enough confidence level is needed in which case $\alpha \geq \max(p_y, p_z)$ is required. In the rest of this paper, we assume that α satisfies this condition.

Definition 2. *Consider an instruction schedule X_1. Schedule X_2 is obtained by rescheduling a single instruction from time slot i to time slot j. Then we say that instruction schedules X_1 and X_2 exhibit a 'OneMove' difference.*

If there is a 'OneMove' difference between X_1 and X_2, then the power consumption of the two schedules are exactly the same for every time slot except i and j as shown in Table 1 where c is any one of the power consumption parameters in (2).

Table 1. Symbolic power consumption values to illustrate Definition 2.

time slot	...	i	...	j	...
X_1	...	$A + c$	...	B	...
X_2	...	A	...	$B + c$	...

Theorem 1. *Suppose there is a 'OneMove' difference in time slots i and j between two instruction schedules X_1 and X_2 as shown in Table 1. Let $A = a_1 + a_2 + ... + a_n$ and $B = b_1 + b_2 + ... + b_m$. Then the difference of the objective function values defined by (13) for X_1 and X_2 is given by*

$$PV(X_2, \xi)^{\alpha}_{inf} - PV(X_1, \xi)^{\alpha}_{inf} = \sum_{i=1}^{m} (2b_i c)^{\alpha}_{\text{inf}} - \sum_{i=1}^{n} (2a_i c)^{\alpha}_{\text{inf}} \tag{19}$$

Proof. The objective function defined by (13) can be computed as a sum of the contributions from time slots i and j and that of the rest of the time slots. According to Lemma 1, with a suitable α,

$$\begin{aligned}
PV(X_1,\xi)^{\alpha}_{inf} &= \left(\sum_{k=1,k\neq i,k\neq j}^{T} (P_k - M)^2\right)^{\alpha}_{inf} + \left((P_i - M)^2 + (P_j - M)^2\right)^{\alpha}_{inf} \\
&= \left(\sum_{k=1,k\neq i,k\neq j}^{T} (P_k - M)^2\right)^{\alpha}_{inf} + \left((A + c - M)^2 + (B - M)^2\right)^{\alpha}_{inf} \\
PV(X_2,\xi)^{\alpha}_{inf} &= \left(\sum_{k=1,k\neq i,k\neq j}^{T} (P_k - M)^2\right)^{\alpha}_{inf} + \left((P_i - M)^2 + (P_j - M)^2\right)^{\alpha}_{inf} \\
&= \left(\sum_{k=1,k\neq i,k\neq j}^{T} (P_k - M)^2\right)^{\alpha}_{inf} + \left((A - M)^2 + (B + c - M)^2\right)^{\alpha}_{inf}
\end{aligned}$$

The difference is given by

$$\begin{aligned}
&PV(X_2,\xi)^{\alpha}_{inf} - PV(X_1,\xi)^{\alpha}_{inf} \\
&= \left((A - M)^2 + (B + c - M)^2\right)^{\alpha}_{inf} - \left((A + c - M)^2 + (B - M)^2\right)^{\alpha}_{inf} \\
&= (2Bc)^{\alpha}_{inf} - (2Ac)^{\alpha}_{inf} \\
&= \sum_{i=1}^{m} (2b_i c)^{\alpha}_{\text{inf}} - \sum_{i=1}^{n} (2a_i c)^{\alpha}_{\text{inf}}
\end{aligned}$$

Hence proved.

Corollary 1. *Suppose two schedules X_1 and X_2 have K ($K > 1$) 'OneMove' differences. Then the difference in the objective function values of X_1 and X_2 equals to the sum of the differences caused by each of the K 'OneMoves'.*

Proof. First consider $K = 2$. There are two 'OneMove' differences between X_1 and X_2. We construct an intermediate schedule X_3 with one 'OneMove' difference compared with X_1 and another 'OneMove' difference compared with X_2. Then

$$\begin{aligned}
&PV(X_2,\xi)^{\alpha}_{inf} - PV(X_1,\xi)^{\alpha}_{inf} \\
&= PV(X_2,\xi)^{\alpha}_{inf} - PV(X_1,\xi)^{\alpha}_{inf} + PV(X_3,\xi)^{\alpha}_{inf} - PV(X_3,\xi)^{\alpha}_{inf} \\
&= (PV(X_2,\xi)^{\alpha}_{inf} - PV(X_3,\xi)^{\alpha}_{inf}) + (PV(X_3,\xi)^{\alpha}_{inf} - PV(X_1,\xi)^{\alpha}_{inf})
\end{aligned}$$

This completes the proof for $K = 2$. $K > 2$ can be proved in the same way.

5.2 Rule Extraction

Corollary 1 tells that the difference between two schedules depends on the 'OneMove' differences between them. A 'OneMove' difference is characterized by two sets of α-pessimistic values $(2b_ic)^{\alpha}_{inf}$ and $(2a_ic)^{\alpha}_{inf}$ as shown in Theorem 1. Therefore we abstract the rough simulation processes that were used for computing these α-pessimistic values by a set of rules. Then by matching the 'OneMove' differences between two given schedules with our rules, the difference between their objective function returns can be obtained.

A rule corresponds to the rough simulation process of a α-pessimistic value $(2b_ic)^{\alpha}_{inf}$ (or $(2a_ic)^{\alpha}_{inf}$). Its format is as follows:

1. The premise defines a possible combination of a_ic (or b_ic) given the target VLIW processor. Suppose the instruction set of the processor is divided into C clusters, we have C^2 combinations for (c, a_i) (or (c,b_i)). Because $(2a_ic)^{\alpha}_{inf}$ and $(2ca_i)^{\alpha}_{inf}$ are equal, the reciprocal ones are excluded. Thus we totally have $\frac{1}{2}(C^2 + C)$ rules.
2. The conclusion part is the value of $(2a_ic)^{\alpha}_{inf}$ or $(2b_ic)^{\alpha}_{inf}$ obtained through rough simulation.

Example 1. Suppose the instruction set of the target VLIW processor is divided into two clusters. The rough variables representing the two associated power consumption parameters c_1 and c_2 are given in Table 2.

Table 2. Power consumption parameters for Example 1.

c_1	c_2
$([19.0, 20.2], [19.0, 20.7])$	$([22.0, 23.0], [21.5, 23.3])$

The premise of the rules are all combinations of c_1 and c_2: $\{c_1, c_2\}$: (c_1, c_2), (c_1, c_1), (c_2, c_2) and (c_2, c_1). (c_2, c_1) is actually a repetition of (c_1, c_2) because $(2c_1c_2)^{\alpha}_{inf}$ and $(2c_2c_1)^{\alpha}_{inf}$ are equal. Therefore only three rules are needed.

Let $\alpha = 0.95$. We compute $(2c_1c_2)^{\alpha}_{inf}$, $(2c_1c_1)^{\alpha}_{inf}$ and $(2c_2c_2)^{\alpha}_{inf}$ by rough simulation. The three rules are summarized as a decision table shown in Table 3.

Table 3. Decision table for Example 1.

	c	$a_i(b_i)$	DiffObj
1	c_1	c_2	853.0
2	c_1	c_1	744.1
3	c_2	c_2	987.9

The next example illustrates how the rules in Example 1 can be used to rank two given schedules.

Example 2. There are seven instructions $\{s_1, s_2, s_3, s_4, s_5, s_6, s_7\}$ to be scheduled in five time slots. For simplicity, let all of them be instructions with single cycle functional unit latency. Further, assume there are no dependencies among them and there are no resource usage constraints by the target VLIW processor. Suppose s_1, s_2, s_3 and s_7 belong to cluster c_1 and s_4, s_5 and s_6 belong to cluster c_2.

Two schedules X_1 and X_2 as shown in Table 4 are to be ranked according to their "fitness".

Table 4. Instruction Schedules for Example 2.

X_1	TimeSlot	1	2	3	4	5
	Instructions	s_1	s_2,s_3	s_4	s_5,s_6	s_7

X_2	TimeSlot	1	2	3	4	5
	Instructions	s_1	s_2	s_3,s_4	s_5	s_6,s_7

Using the power equation (2), we substitute instructions with their corresponding (symbolic) power consumption parameters for these two schedules (For simplicity $U(0|0)$ is ignored since this part is the same in every time slot.) Then we have the power data for each time slot for the two schedules as in Table 5.

Table 5. Power consumptions for instruction schedules in Example 2.

X_1	TimeSlot	1	2	3	4	5
	Power	c_1	$c_1 + c_1$	c_2	$c_2 + c_2$	c_1

X_2	TimeSlot	1	2	3	4	5
	Power	c_1	c_1	$c_1 + c_2$	c_2	$c_1 + c_2$

Comparing the power data shown in Table 5, the two schedules exhibit two 'OneMove' differences – one between slots 2 and 3 and another between slots 4 and 5. Therefore the difference in the objective function values of X_2 and X_1 depends on the four α-pessimistic values according to Corollary 1 and Theorem 1, i.e.

$$PV(X_2, \xi)^{\alpha}_{inf} - PV(X_1, \xi)^{\alpha}_{inf} = (2c_1c_2)^{\alpha}_{inf} - (2c_2c_2)^{\alpha}_{inf} + (2c_1c_2)^{\alpha}_{inf} - (2c_1c_1)^{\alpha}_{inf}$$

The values of $(2c_1c_2)^{\alpha}_{inf}$, $(2c_2c_2)^{\alpha}_{inf}$ and $(2c_1c_1)^{\alpha}_{inf}$ can be found in Table 3. Therefore,

$$PV(X_2, \xi)^{\alpha}_{inf} - PV(X_1, \xi)^{\alpha}_{inf} = 2 \times 853.0 - 987.9 - 744.1 = -26$$

Hence X_2 is worse than X_1; its power variation defined by the objective function (13) is increased by 26 compared with X_1.

5.3 Computational Efficiency of Rule-based GA

The computational advantage of the rule-based method can be evaluated using a real VLIW processor. Our target processor is the TMS320C6711 [11] which is a VLIW digital signal processor. The instruction set of TMS320C6711 is partitioned into four clusters as in [8]. Therefore, we have ten rules for this VLIW processor to abstract the rough simulation results of the ten α-pessimistic values.

We perform power-balanced instruction scheduling on five programs taken from MediaBench [12]. For any given target instruction block, scheduling is performed by means of GA using rough simulation and the proposed rule-based approach for fitness evaluation and selection. The sample size for rough simulation, the population size and the number of generations are 50, 20 and 20 respectively. The crossover probability, mutation rate, population size and generations are same in both cases. All our computational experiments were conducted on an Intel Pentium 4 2.80GHz personal computer with 512MB RAM running under Microsoft Windows 2000.

Table 6 shows the computation time required by the two GAs. For each problem instance, the problem size refers to the number of time slots and the number of instructions in the instruction block. The results show a significant reduction in computation time. The shorter time required by the rule-based GA make this approach practical for implementation is real compilers.

Table 6. Computation time of GAs on benchmarks from Mediabench.

Prob. Size	Source	Rough Simulation GA (sec.)	Rule-based GA (sec.)
(28,30)	epic	55.98	0.015
(37,30)	g721	73.09	0.031
(44,23)	gsm	66.53	0.015
(38,34)	jpeg	81.83	0.031
(70,58)	mpeg2	245.90	0.046

6 Conclusions

This paper presents our continuing research on power-balanced instruction scheduling for VLIW processors using rough set theory to model the uncertainties involved in estimating the power model of the processor. In this paper, we proposed an efficient rule-based genetic algorithm to solve the scheduling problem which has been formulated as a rough program. The rules are used to rank the schedules produced in each generation of the GA so that selection decisions can be made. Therefore the computational efficiency of this GA is significantly improved compared with those in [9, 10]. The theoretical basis of our method is derived rigorously. The short computation time required makes the rule-based approach practical for use in production compilers.

References

1. Yun, H., Kim, J.: Power-aware modulo scheduling for high-performance VLIW processors, Huntington Beach, California, USA. (2001) 40–45
2. Yang, H., Gao, G.R., Leung, C.: On achieving balanced power consumption in software pipelined loops, Grenoble, France (2002) 210–217
3. Xiao, S., Lai, E.M.K.: 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 (2004)
4. Tiwari, V., Malik, S., Wolfe, A., Lee, M.T.: Instruction level power analysis and optimization of software. (1996) 326–328
5. Gebotys, C.: Power minimization derived from architectural-usage of VLIW processors, Los Angeles, USA (2000) 308–311
6. Bona, A., Sami, M., Sciutos, D., Silvano, C., Zaccaria, V., R.Zafalon: Energy estimation and optimization of embedded VLIW processors based on instruction clustering, New Orleans, USA (2002) 886–891
7. Xiao, S., Lai, E.M.K.: A rough programming approach to power-aware vliw instruction scheduling for digital signal processors, Philadelphia, USA (2005)
8. Xiao, S., Lai, E.M.K.: A rough set approach to instruction-level power analysis of embedded VLIW processors. In: Proc. Int. Conf. on Information and Management Sciences, Kunming, China (2005)
9. Xiao, S., Lai, E.M.K.: Power-balanced VLIW instruction scheduling using rough programming. In: Proc. Int. Conf. on Information and Management Sciences, Kunming, China (2005)
10. Liu, B.: Theory and practice of uncertain programming. Physica-Verlag, Heidelberg (2002)
11. : TMS320C621x/C671x DSP two-level internal memory reference guide. Application Report SPRU609, Texas Instruments Inc. (2002)
12. Lee, C., Potkonjak, M., Mangione-Smith, W.H.: MediaBench: a tool for evaluating and synthesizing multimedia and communications systems. (1997) 330–335