Power-Balanced VLIW Instruction Scheduling Using Rough Programming

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

Abstract

Current techniques for power-aware VLIW instruction scheduling assumed that the power consumption parameters are precisely known. In reality, there will always be some degree of imprecision. In this paper, we propose to apply rough programming to handle the imprecision involved. The power-aware instruction scheduling problem is formulated as a chance-constraint rough program. A problem-specific genetic algorithm implementation is proposed to solve it. Experiments with various target instruction sequences 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 finding a globally optimal schedule with all the possible realization of the power parameters considered.

Keywords: Genetic algorithm, instruction scheduling, power-aware, rough programming, VLIW.

1 Introduction

Power-balanced instruction scheduling is the task of producing a schedule of microprocessor 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. Previously published works in this area make use of power consumption models with parameters that are assumed to be precisely known [1–4]. 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 [5–8], are inherently 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 given the same power figure [9].

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, the optimal schedule obtained with the average values fails to guarantee that a hard power variation limit for chip reliability will not be exceeded in real life situations. Therefore, it is desirable to find schedules which guarantee good power variations for all possible realizations of the power consumption parameters. So instead of formulating the scheduling problem as a mixed-integer program [4], some kind of uncertain programming formulation and solution will be needed.

There are several approaches to uncertain programming [10], e.g. stochastic programming, fuzzy programming and rough programming. We propose to use rough programming approach to power-aware VLIW instruction scheduling. Rough programming [10] 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 [12]. For fuzzy set theory, fuzzy membership functions are required [13].

This paper focuses on the optimization problem of Very Long Instruction Word (VLIW) instruction scheduling for power variation minimization. In general, a VLIW processor is a pipelined CPU that can execute a set of explicitly parallel instructions on different functional units. This set of instructions is statically scheduled by the compiler. The problem is formulated as a chance-constraint rough program. Since the objective function to be optimized is multimodal and the search space is particularly irregular, conventional optimization techniques are unable to produce near-optimal solutions. We proposed a problem-specific genetic algorithm (GA) to solve it. The advantage of genetic algorithm [14] is highly robust to avoid getting stuck at a local optimal solution. Experiments with various target instruction sequences 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 finding a globally optimal schedule with all the possible realization of the power parameters considered.

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 conventional mixed-integer program (MIP) for the scheduling of VLIW instructions for minimal power variations is given by **P1** and efficient techniques have been proposed to solve it [4].

P1: $\min\ P(X,\xi)$

subject to

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

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

where $P(X,\xi)$ is the power variation of a given schedule X over time which we seek to minimize, and ξ denotes the set of the power consumption parameters. For the two constraints given by (1), n is the number of instructions in X and t is the number of time slots available. The binary decision variables x_i^k has a value of 1 if instruction i is rescheduled in time slot k; otherwise its value is zero. $G(X) \leq 0$ and $L(X) = 0$ in (2) denote the constraint matrix for processor-specific resource constraints, data dependence constraints and performance deadline constraints.

As said in the introduction, the power-aware VLIW instruction scheduling problem has been investigated, but very few studies are concerned with uncertainty of the power consumption parameters ξ in the above formulation. In real life situations, the power consumption parameters may vary a lot. Therefore, classical approaches replying on precise data are disrupted. We propose to encapsulate the imprecision of the power consumption parameters by expressing them as rough variables and then formulate the problem as a rough program. Definitions for rough space and rough variables are given as follows [10].

Definition 2.1 *Let Λ be a nonempty set, $\overline{A}$ be a σ-algebra of subsets of Λ, Δ be an element in $\overline{A}$, and π be a set function satisfying the following axioms:*

1. *$\pi\{\mathrm{A}\} \geq 0$ for any $A \in \overline{A}$.*
2. *For every countable sequence of mutually disjoint events $\{\mathrm{A}_i\}_{i=1}^{\infty}$, we have $\pi\{\bigcup_{i=1}^{\infty} A_i\} = \sum_{i=1}^{\infty} \pi\{A_i\}$.*

Then $(\Lambda, \Delta, \overline{A}, \pi)$ is called a rough space.

Definition 2.2 *A rough variable ζ on the rough space $(\Lambda, \Delta, \overline{A}, \pi)$ is a function from Λ to the real line $\Re$ such that for every Borel set O of $\Re$, we have $\{\lambda \in \Lambda | \xi(\lambda) \in O\} \in \overline{A}$. The lower and the upper approximations of the rough variable ζ are then defined as $\underline{\zeta} = \{\zeta(\lambda) | \lambda \in \Delta\}$ and $\overline{\zeta} = \{\zeta(\lambda) | \lambda \in \Lambda\}$ respectively.*

We need to encapsulate the variations of the power consumption parameters by expressing them as rough variables which can be represented by $([a,b],[c,d])$ as in Example 2.3.

Example 2.3 *Suppose a rough space $(\Lambda, \Delta, \overline{A}, \pi)$ where $\Lambda = \{x | c \leq x \leq d\}$, $\Delta = \{x | a \leq x \leq b\}$, with $c \leq a \leq b \leq d$. Then the function $\zeta(x) = x$ for all $x \in \Lambda$ is a rough variable, also expressed as $([a,b],[c,d])$, where $[a,b]$ is its lower approximation and $[c,d]$ is its upper approximation. This means that the values within $[a,b]$ are sure and those within $[c,d]$ are possible.*

A power parameter $p_i \in \xi$ 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. In **P1**, if ξ is a set of rough variables, then the values of the objective function $P(X,\xi)$ for any given schedule X are also rough. For our power balanced VLIW instruction scheduling problem, we choose to rank the rough returns of $P(X,\xi)$ by the α-optimistic value $P(X,\xi)_{sup}(\alpha)$ or the α-pessimistic value $P(X,\xi)_{inf}(\alpha)$ for some predetermined confidence level $\alpha \in (0,1]$. There are also other measures to rank rough returns, e.g. the expected value $E[P(X,\xi)]$; the trust measure $Tr\{P(X,\xi) > \overline{r}\}$ for some predetermined level $\overline{r}$. However, ranking the rough returns of $P(X,\xi)$ by these measures cannot show the power variation value ranges of the obtained optimal schedule with all the possible realization of the power consumption parameters considered.

Definition 2.4 *let ϑ be a rough variable, and $\alpha \in (0,1]$. Then*

$$\vartheta_{inf}(\alpha) = inf\{r | Tr\{\vartheta \leq r\} \geq \alpha\} \quad (3)$$

is called the α-pessimistic value to ϑ, where Tr is the trust measure operator.

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. That is, the trust measure plays the role of probability measure and credibility measure.

The rough program corresponding to **P1** is given by **P2**.

P2: $\min\ P(X,\xi)_{inf}(\alpha)$

subject to

$$\begin{array}{ll} X = \bigcup\ x_i^k & i = 1,...,n; k = 1,...,t \\ x_i^k \in \{0,1\} & i = 1,...,n; k = 1,...,t \end{array} \quad (4)$$

$$\begin{array}{c} G(X) \leq 0 \\ L(X) = 0 \end{array} \quad (5)$$

The objective function is given by

$$P(X,\xi)_{inf}(\alpha) = inf\{\overline{P} | Tr\{P(X,\xi) \leq \overline{P}\} \geq \alpha\} \quad (6)$$

is based on the α-pessimistic value where α is the specified confidence level and ξ is the set of rough power consumption parameters. (4) and (5) are the same as those in **P1** since there are no rough variables involved. $P(X,\xi)_{inf}(\alpha)$ 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. Solving this program involves searching for the minimum α-pessimistic value $P(X,\xi)_{inf}(\alpha)$ among all feasible schedules X. Next, we propose a problem-specific genetic algorithm to solve the rough program.

3 Implementation of GA

The standard Genetic Algorithm was given in [14]. The specific implementation details for obtaining optimal solutions to the rough program **P2** are discussed below.

3.1 Chromosome Encoding

Each chromosome is an array of integer variables each representing an instruction. The integer value indicates the execution time slot allocated to that instruction. As an example, chromosome $\{1,1,4,1,2,4,1,2,4,5,5,5,6,6\}$ means that there are 14 instructions: the first, second, fourth and seventh instructions are allocated to the first time slot; the fifth and eighth instructions are allocated to the second time slot; the third, sixth and ninth instructions are allocated to the fourth time slot; the tenth, eleventh and twelfth instructions are allocated to the fifth time slot; the thirteenth and fourteenth instruction are allocated to the sixth time slot.

3.2 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. The function *random-change_1instruction()* generates a new schedule by randomly change the allocated time slot of an instruction. Repeating this process *pop_size* times, we can make *pop_size* initial feasible chromosome. Function *constraint_check()* returns 1 if the generated new schedule does not violate any constraints; otherwise returns 0.

3.3 Fitness Evaluation

The function $-P\left(X,\xi\right)_{\inf}\left(\alpha\right)$ is used to evaluate the fitness of a candidate X. Rough simulation [10] plays an important role in rough systems. In order to compute $P\left(X,\xi\right)_{\inf}\left(\alpha\right)$ of a candidate X, the following rough simulation process is used. Let R be the sampling 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 $P\left(X,\xi\right)_{\inf}\left(\alpha\right)$ is given by the minimum value of v such that

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

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

input : A feasible schedule X; lower and upper approximations of each power consumption parameter $p_i \in \xi$; confidence level α; let R be the sampling size
output: Return of $P(X,\xi)_{inf}(\alpha)$

1 **for** $k \leftarrow 1$ ***to*** R **do**
2 **foreach** *power consumption parameter* $p_i \in \xi$ **do** randomly sample l_i^k from its lower approximation;
3 **foreach** *power consumption parameter* $p_i \in \xi$ **do** randomly sample u_i^k from its upper approximation;
4 **end**
5 $l\left(v\right) \leftarrow$ the number of $P\left(X, l_1^k, \ldots, l_i^k, \ldots\right)_{\inf}\left(\alpha\right) \leq v$ being satisfied when $k = 1,\ldots,R$ respectively;
6 $u\left(v\right) \leftarrow$ the number of $P\left(X, u_1^k, \ldots, u_i^k, \ldots\right)_{\inf}\left(\alpha\right) \leq v$ being satisfied when $k = 1,\ldots,R$ respectively;
7 Find the minimal value v such that

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

8 Return v;

Figure 1: Rough Simulation algorithm.

3.4 Selection, Crossover, and Mutation

The chromosomes in the population are sorted non-increasingly in terms of their fitness. The chromosome selection for the next generation is done on the basis of fitness. Our design adopts the rank-based roulette-wheel selection scheme [14]. 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 Fig. 2.

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

Figure 2: Chromosome selection by roulette wheel (*pop_size*: population size).

The above selection process is done repeatedly from $i = 1$ to *pop_size*. 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 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 unfeasible schedules, constraints check operators have been included. If feasible offsprings cannot be created, the parents will not be replaced.

The motivation for using mutation, then, is to prevent the permanent loss of any particular bit or allele (premature convergence). This may be particularly be a problem if one is working with a small population. Suppose the function *randomchange_1instruction()* generates a new schedule by randomly change the allocated time slot of an instruction. Let X be the schedule to be mutated. Then *randomchange_1instruction(X)* is repeated until a new feasible schedule is created.

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.

4 Experimental Results

The target processor of our scheduling experiments is the TMS320C6711 [15] which is a VLIW digital signal processor. Fig. 3 shows the internal organization of the processor. The functional units are classified into four types: L, M, S and D. There are two functional units for each type in data path A and B separately. In order to describe the knowledge about the power consumption parameters by rough variables, we use the following method of obtaining the rough quantities from the experiments. Repeated measurements for each parameter are randomly conducted using the experimental setup as in [6]. Based the measured data, the possible current values on real line are discretized using the Boonlean reasoning algorithm. After discretization, the lower and upper approximations for each power parameter are generated using the Rosetta Toolkit [16].

GA and rough program formulation are tested using the digital signal processing benchmarks from Trimaran [17]. Trimaran is an integrated compilation and performance monitoring infrastructure for research in instruction-level parallelism. Instruction blocks with various problem dimensions (characterized by the number of instructions and the number of time slots as the performance deadline) are selected.

The confidence level α is set to 0.9 in all of our experiments. For the genetic algorithm part, no systematic parameter optimization process has so far been attempted, but the following parameter set was used in our experiments. In our problem, the crossover operators often cannot create enough feasible offsprings due to violations of the constraints. Therefore, in order to prevent the permanent loss of any particular bit or allele, the crossover probability $P\_CROSSOVER$ is set to a low value while the mutation rate $P\_MUTATION$ is set to a high value. Tuning the values to be suitable for a specific data set may give rise to improved performance.

POP_SIZE (population size) = 30
P_CROSSOVER (crossover probability) = 0.2
P_MUTATION (mutation rate) = 0.8
a (in rank-based selection) = 0.05
GEN (maximum generation): dependent on data set

For any given target instruction block, we conduct instruction scheduling by means of 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 4.1 *The target instruction block consisting of fourteen instructions is given as {addaw, add, addaw, add, ldw, mv, addaw, stw, b, addaw, cmpeq, stw, ldw, b}. The data dependence graph is shown in Fig. 4.*

If the power consumption parameters of target processor are given as average values, the scheduling problem is formulated as a mixed integer program

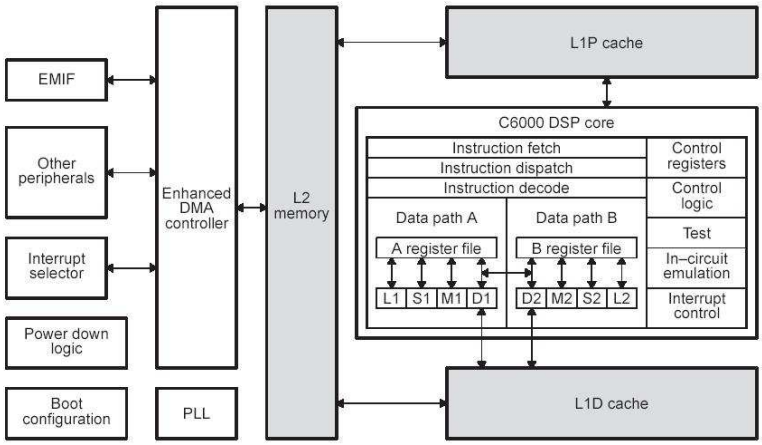

Figure 3: Internal Organization of the TMS320C6711 DSP processor.

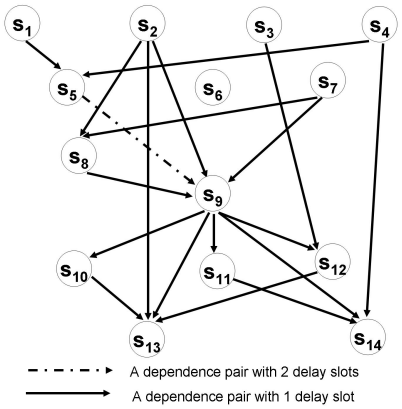

Figure 4: Data dependency graph for instructions in Example 4.1.

and solved by a branch and bound algorithm as in [4], we obtain the optimal schedule $X_{MIP} = \{x_1^1, x_2^1, x_3^4, x_4^1, x_5^2, x_6^4, x_7^1, x_8^2, x_9^4, x_{10}^5, x_{11}^5, x_{12}^5, x_{13}^6, x_{14}^6\}$ *where the superscripts indicate the time slot in which the instruction is being scheduled.*

If the power parameters of target processor are described by rough variables as in Table 1, the scheduling problem is formulated as a rough program as proposed in this paper, we obtain another schedule using the proposed genetic algorithm $X_{RP} = \{x_1^1, x_2^1, x_3^4, x_4^1, x_5^2, x_6^5, x_7^2, x_8^3, x_9^4, x_{10}^5, x_{11}^5, x_{12}^5, x_{13}^6, x_{14}^6\}$.

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

$$P(X_{RP}, \xi)_{inf}(0.9) = 14798 \tag{7}$$

$$P(X_{MIP}, \xi)_{inf}(0.9) = 17713 \tag{8}$$

where ξ *denotes the set of power consumption parameters described in Table 1. (7) 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*. (8) 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 18 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. Compared with the MIP solutions, the results show that the rough programming approach produces schedules with the better objective function values.

5 Conclusions

Rough programming has been applied to the problem of power-balanced VLIW instruction scheduling with 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 better instruction schedules are obtained using the rough programming approach compared to the conventional mixed-integer programming approach. Fur-

Table 1: Rough power consumption parameters in Example 4.1.

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

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
(11,11)	Fib	52646	52642	0
(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
(15,15)	Fib-mem	113961	113961	0
(12,19)	Fir	107432	83827	22.0
(12,22)	Bmm	137611	104337	24.2
(17,16)	Fib-mem	141390	130557	7.7
(20,21)	Wc	212336	212256	0
(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
(23,24)	Bmm	320020	319856	0
(25,24)	Bmm	286887	168803	41.2
(27,24)	Fir	316431	152204	51.9
(29,29)	Bmm	507000	506639	0
(31,30)	Bmm	537007	491859	8.4
(33,33)	mm-dyn	661606	661038	0
(35,34)	mm-dyn	704769	666751	5.4

thermore, the schedules obtained through rough programming have actual power variations which are guaranteed to the desired level of confidence.

References

[1] 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.

[2] 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.

[3] 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.

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

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

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

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

[13] 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.

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

[15] *TMS320C621x/C671x DSP Two-Level Internal Memory Reference Guide*, Texas Instruments, Aug. 2002, application Report, SPRU609.

[16] 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.

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

Shu Xiao obtained her B.E.and M.S. degrees from Wuhan University, China in 1998 and 2001 respectively. She is currently a PhD student in the School of Computer Engineering, Nanyang Technological University, Singapore. Her research interests are in power-aware VLIW instruction scheduling.

Edmund M-K. Lai obtained his B.E.(Hons) and PhD degrees from the University of Western Australia in 1982 and 1991 respectively. He is currently an associate professor in the School of Computer Engineering, Nanyang Technological University, Singapore. His research interests are in digital signal processing and information theory.