Algorithms and analysis of scheduling for loops with minimum switching

Zili Shao*

Department of Computing, Hong Kong Polytechnic University,
Hung Hom, Kowloon, Hong Kong
E-mail: cszlshao@comp.polyu.edu.hk
*Corresponding author

Qingfeng Zhuge, Meilin Liu, Chun Xue and Edwin H.M. Sha

Department of Computer Science, University of Texas at Dallas,
Richardson, Texas 75083, USA
E-mail: qfzhuge@utdallas.edu E-mail: mxl024100@utdallas.edu
E-mail: cxx016000@utdallas.edu E-mail: edsha@utdallas.edu

Bin Xiao

Department of Computing,
Hong Kong Polytechnic University,
Hung Hom, Kowloon, Hong Kong
E-mail: csbxiao@comp.polyu.edu.hk

Abstract: Switching activity and schedule length are the two of the most important factors in power dissipation. This paper studies the scheduling problem that minimises both schedule length and switching activities for applications with loops on multiple functional unit architectures. We show that, to find a schedule that has the minimal switching activities among all minimum latency schedules with or without resource constraints is NP-complete. Although the minimum latency scheduling problem is polynomial time solvable if there is no resource constraint or only one functional unit (FU), the problem becomes NP-complete when switching activities are considered as the second constraint. An algorithm, Power Reduction Rotation Scheduling (PRRS), is proposed. The algorithm attempts to minimise both switching activities and schedule length while performing scheduling and allocation simultaneously. Compared with the list scheduling, PRRS shows an average of 20.1% reduction in schedule length and 52.2% reduction in bus switching activities. Our algorithm also shows better performance than the approach that considers scheduling and allocation in separate phases.

Keywords: switching activity; loop; scheduling; low power.

Reference to this paper should be made as follows: Shao, Z., Zhuge, Q., Liu, M., Xue, C. and Sha, E.H.M. and Xiao, B. (2006) 'Algorithms and analysis of scheduling for loops with minimum switching', *Int. J. Computational Science and Engineering*, Vol. 2, Nos. 1/2, pp.88–97.

Biographical notes: Zili Shao received the BE Degree in Electronic Mechanics from University of Electronic Science and Technology of China, China, 1995. He received the MS and PhD Degrees from the Department of Computer Science at the University of Texas at Dallas, in 2003 and 2005, respectively. He has been an Assistant Professor in the Department of Computing at the Hong Kong Polytechnic University since 2005. His research interests include embedded systems, high-level synthesis, compiler optimisation, hardware/software co-design and computer security.

Qingfeng Zhuge received her PhD from the Department of Computer Science at the University of Texas at Dallas. She obtained her BS and MS Degrees in Electronics Engineering from Fudan University, Shanghai, China. Her research interests include embedded systems, real-time systems, parallel architectures, optimisation algorithms, high-level synthesis, compilers, and scheduling.

Meilin Liu received the BS and MS Degree in Electrical Engineering from Hohai University, Nanjing, China in 1992 and 2000, respectively, and the MS Degree in computer Science from University of Texas at Dallas, in 2004. She is currently a PhD Candidate of computer science at University of Texas at Dallas. Her research interests include optimisation of loop execution, loop transformations, and compiler optimisation for embedded systems.

Chun Xue received the BS Degree in Computer Science and Engineering from University of Texas at Arlington in May 1997, and MS Degree in computer Science from University of Texas at Dallas, in Dec 2002. He is currently a computer science PhD candidate at University of Texas at Dallas. His research interests include performance and memory optimisation for embedded systems, and software/hardware co-design for parallel systems.

Edwin Hsing-Mean Sha received the BSE Degree in computer science from National Taiwan University, Taiwan, in 1986, and received MA and PhD Degrees from the Department of Computer Science, Princeton University, in 1991 and 1992, respectively. Since 2000, he has been a tenured full Professor in the Department of Computer Science at the University of Texas at Dallas. He has published more than 200 research papers in refereed conferences and journals. He has been serving as an editor for many journals, and program committee members and chairs in numerous conferences. He received NSF CAREER Award and Teaching award in 1998.

Bin Xiao received the BSc and MSc Degrees in Electronics Engineering from Fudan University, China in 1997 and 2000 respectively, and PhD Degree in computer science from University of Texas at Dallas, USA, in 2003. Now he is an Assistant Professor in the department of computing of Hong Kong Polytechnic University. His research interests include computer networks and routing protocols; peer-to-peer communications; Internet security with a focus on Denial of Service (DoS) defence; embedded system design; mobile ad hoc networks; and wireless communication systems.

1 Introduction

In many portable systems, such as wireless communication and image processing systems, the DSP processor core consumes a significant amount of power and time in highly computation intensive applications. In such applications, loops are the most critical sections. An efficient loop scheduling scheme can help reduce the power consumption while still satisfying the time constraint. Switching activities play a key role in the total power consumption (Chandrakasan et al., 1992; Stan and Burleson, 1995), therefore, various techniques have been proposed to reduce power consumption by reducing switching activities (Chandrakasan et al., 1992; Tsui et al., 1993; Roy and Prasad, 1992; Alidina et al., 1994; Hachtel et al., 1994; Mehendale et al., 1995; Raghunathan and Jha, 1995; Musoll and Cortadella, 1995a, 1995b; Benini and Micheli, 1995; Macii et al., 1998; Henning and Chakrabarti, 1998; Yu et al., 1998; Masselos et al., 2000; Panda and Dutt, 1999; Sundararajan and Parhi, 2000; Parhi, 2001; Kruse et al., 2001; Kim et al., 2001; Erdogan and Arslan, 2002; Henning and Chakrabarti; 2002). This paper focuses on reducing both switching activities and schedule length of an application on multiple functional unit architectures such as VLIW (Very Long Instruction Word) processors. In a multiple functional unit architecture, several instructions can be executed in parallel. The power consumption in a clock cycle, P_{cycle}, can be computed by:

$$P_{\text{cycle}} = P_{\text{base}} + \sum_{I\,\text{nst}_i} \{P_{I\,\text{nst}_i} + SP(i, j)\} \tag{1}$$

where P_{base} is the base power needed to support instruction execution, $P_{I\,\text{nst}_i}$ is the basic power to execute an instruction I_i on a functional unit, and $SP(i, j)$ is the switching power caused by switching activities between $I\,\text{nst}_i$ (current instruction) and $I\,\text{nst}_j$ (last instruction) executed on the same functional unit (FU). Let S be a schedule for an application and L the schedule length of S. Then the energy E_S for Schedule S can be computed by

$$E_S = \sum_{k=1}^{L} P_{\text{cycle}}^{(k)} = L \times P_{\text{base}} + \sum_{k=1}^{L} \sum_{I\,\text{nst}_i^{(k)}} P_{I\,\text{nst}_i^{(k)}} + \sum_{k=1}^{L} \sum_{I\,\text{nst}_i^{(k)}} SP^{(k)}(i, j) \tag{2}$$

$\Sigma\Sigma P$ is the summation of basic power consumption for all instructions of an application. It does not change with different schedules. L and $\Sigma\Sigma SP(i, j)$ will change with different schedules, though. Therefore, in order to minimise the energy consumption of an application, schedule length and switching activity both need to be considered in scheduling.

Low power scheduling to reduce switching activities has been extensively studied in high level synthesis (HLS) and compiler optimisation. In HLS, a lot of approaches have been proposed to minimise switching activities. In Su et al. (1994), an instruction scheduling technique called cold scheduling is proposed to reduce the switching activities on the control path. In Raghunathan and Jha (1995), Kruse et al. (2001) and Chang and Pedram (1995), a low power resource allocation approach is proposed to find an allocation for a fixed schedule in such a way that the total switching activities can be reduced. In Musoll and Cortadella (1995a, 1995b), an operand sharing scheduling technique is proposed to schedule the operation nodes with the same operands as closely as possible to reduce the switching activities on the functional units. In Mehendale et al. (1995), a scheduling algorithm for optimising coefficients of a FIR filter is proposed to minimise the switching activities on memory data bus and functional units. In recent works Masselos et al. (2000) and Choi and Chatterjee (2001), the power efficient scheduling problem is formulated as the Travelling Salesman's Problem (TSP) and solved by heuristics of TSP. The above techniques are either based on single FU architecture (Mehendale et al., 1995;

Musoll and Cortadella, 1995a, 1995b; Masselos et al., 2000; Su et al., 1994; Choi and Chatterjee, 2001) or a fixed schedule (Raghunathan and Jha, 1995; Kruse et al., 2001; Chang and Pedram, 1995). So optimising schedule length is not considered in these techniques.

In compiler optimisation, various instruction level scheduling techniques have been proposed to reduce power consumption. In Tiwari et al. (1994a, 1994b) and Lee et al. (1997), several revised list scheduling techniques are proposed to minimise energy, based on the instruction level energy models for the specific processors. Using similar energy models, in Parikh et al. (2000), several energy oriented instruction scheduling approaches are presented and compared with performance oriented scheduling. In Toburen et al. (1998), an instruction scheduling technique is proposed to limit the number of instructions that can be scheduled in a given cycle based on some predefined per cycle energy dissipation threshold. In Lee et al. (2003), a two phase scheduling approach is proposed to optimise transition activity in the instruction bus on a VLIW architecture. These techniques are based on DAG (Directed Acyclic Graph) Scheduling, in which an application is modelled as DAG and only the DAG parts of loops are considered. The loop pipelining techniques (Lam, 1988; Rau et al., 1992; Huff, 1993; Chao et al., 1997) cannot be applied to optimise schedule length when loops are represented as DAGs.

Several low power loop compilation optimisation techniques have been proposed (Yun and Kim, 2001; Yang et al., 2002). However, with the focus on reducing power variations of applications, they cannot be directly applied to optimise the energy consumption. In HLS, based on operand sharing approach, a loop pipelining methodology to reduce both latency and power is first proposed in Yu et al. (1998). Using a similar approach, a loop pipelining technique is proposed to first minimise power and then maximise throughout in Kim et al. (2001). These techniques are based on operand sharing and cannot be directly used on multiple functional unit architectures. Therefore, in this paper, we propose a low power scheduling scheme for multiple functional unit architectures to reduce both schedule length and switching activities for an application with loops. The scheme is constructed based on a general model and can be applied in either HLS or compiler optimisation.

In the paper, we first analyse the complexity of the low power loop scheduling problem. We formally prove that the loop scheduling problem with minimum latency and minimum switching activities is NP-complete with or without resource constraints. While the minimum latency loop scheduling problem is polynomial time solvable if there is only one FU or no resource constraints, the problem becomes NP-complete when considering switching activities as the second constraint.

We then design an algorithm, Power Reduction Rotation Scheduling (PRRS), to minimise both switching activities and schedule length for loop applications by performing scheduling and allocation simultaneously. In the PRRS algorithm, the schedules are generated by repeatedly rotating down and reallocating nodes with minimum schedule length and switching activities based on rotation scheduling (Chao et al., 1997) and a best schedule is selected that has the minimal switching activities among all schedules with the minimal schedule length.

Finally, we conduct experiments on a VLIW simulator similar to TI C6000 DSP. The experimental results show significant reduction in switching activities and schedule length. Compared with the list scheduling, PRRS shows an average 20.1% reduction in schedule length and 52.2% reduction in bus switching activities. The experimental results also show that PRRS has better performance in switching activities reduction than the algorithm based on the approach that considers low power allocation with a fixed schedule (Kruse et al., 2001).

In the next section, we introduce necessary background. Section 3 presents complexity analysis of our scheduling problem. The algorithm is discussed in Section 4. Experimental results and concluding remarks are provided in Section 5 and 6, respectively.

2 Basic concepts and models

In this section, we introduce some basic concepts which will be used in the later sections.

2.1 Data flow graph (DFG)

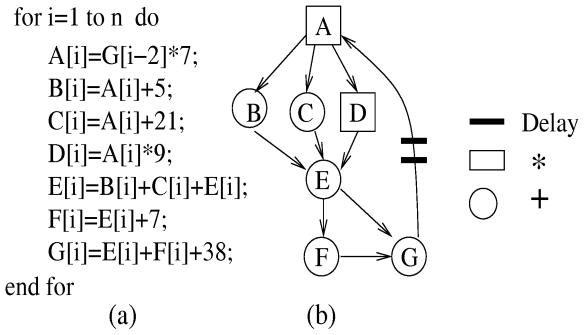
Data flow graph is used to model loops and is defined as follows. A *Data Flow Graph* (*DFG*) $G = \langle V, E, \mathrm{OP}, d\rangle$ is a node-weighted and edge-weighted directed graph, where V is the set of operation nodes, $E \subseteq V \times V$ is the edge set that defines the precedence relations for all nodes in V, $\mathrm{OP}(u)$ is a binary string associated with each node $u \in V$, $d(e)$ represents the number of delays for an edge e. Nodes in V can be various operations, such as addition, subtraction, multiplication, logic operation, etc.

In DFG, $\mathrm{OP}(u)$ is a binary string that denotes the state of the signal associated with node u. It may represent different values in different optimisation environments. For example, $\mathrm{OP}(u)$ can be used to represent the operand of node u in optimising switching activities in functional units (Musoll and Cortadella, 1995a, 1995b), or it can be used to represent the binary code of node u in optimising switching activities in instruction buses (Lee et al., 2003).

In our case, a DFG can contain cycles. The intraiteration precedence relation is represented by the edge without delay and the interiteration precedence relation is represented by the edge with delays. The *cycle period* of a DFG corresponds to the minimum schedule length of one iteration of the loop when there are no resource constraints.

An example is shown in Figure 1. The DFG in Figure 1(b) models the loop in Figure 1(a). In this example there are two kinds of operations: multiplication and addition. They are denoted by the rectangle and circle as shown in Figure 1(b).

Figure 1 A loop and its corresponding DFG



2.2 The static schedule

A *static* schedule of a cyclic DFG is a repeated pattern of an execution of the corresponding loop. In our work, a schedule implies both control step assignment, and functional unit allocation. A static schedule must obey the precedence relations of the *directed acyclic graph (DAG)* portion of the respective DFG. The DAG is obtained by removing all edges with delays in the DFG.

Figure 2 shows a static schedule for the DFG in Figure 1(b) when there are three FUs. The schedule is obtained by list scheduling. In the schedule, the binary string in the parenthesis beside each node denotes the states of the signals associated with nodes. To make it simple, we assume that all multiplication operation nodes are associated with the same state of signal, 001 and all addition operation nodes are with the same state of signal, 110. These assumptions here are only for demonstration purposes. In practice, nodes with the same operation may have different states of signal.

Figure 2 The static schedule for the DFG in Figure 1(b)

Step	FU1	FU2	FU3
1	A(001)		
2	B(110)	C(110)	D(001)
3	E(110)		
4	F(110)		
5	G(110)		

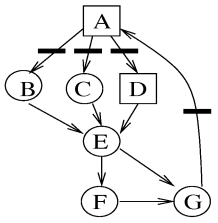
We use $[i, j]$ to denote the location of a node in a schedule, where i is the row (control step) and j is the column (FU). For example, location [2, 1] in the schedule refers to node B scheduled at control step 2 and assigned to FU_1 in Figure 2.

2.3 Retiming and rotation scheduling

Retiming (Veen and Woeginger, 1998) can be used to optimise the cycle period of a DFG by evenly distributing the delays in it. Given a DFG $G = \langle V, E, \text{OP}, d\rangle$, retiming r of G is a function from V to integers. For a node $u \in V$, the value of $r(u)$ is the number of delays drawn from each of its incoming edges of node u and pushed to all of its outgoing edges. Let $G_r = \langle V, E, \text{OP}, d_r\rangle$ denote the retimed graph of G with retiming r, then $d_r(e) = d(e) + r(u) - r(v)$ for every edge $e(u \rightarrow v) \in V$ in G_r.

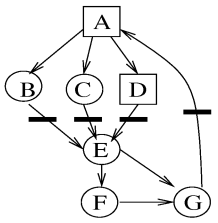
Rotation Scheduling presented in Chao et al. (1997) is a scheduling technique used to optimise a loop schedule with resource constraints. It transforms a schedule to a more compact one iteratively. In most cases, the minimal schedule length can be obtained in polynomial time by rotation scheduling. In each step of rotation, nodes in the first row of the schedule are rotated down. By doing so, the nodes in the first row are rescheduled to the earliest possible available locations. From the retiming point of view, each node gets retimed once by drawing one delay from each of the incoming edges of the node and adding one delay to each of its outgoing edges in the DFG. The new location of the node in the schedule must also obey the precedence relation in the new retimed graph. The retimed graphs and schedules after the first and second rotation are shown in Figure 3(a) and Figure 3(b) respectively, which is based on the original schedule in Figure 2. The minimal schedule length is obtained by the schedule in Figure 3(b).

Figure 3 (a) The retimed graph and the schedule after the first rotation and (b) The retimed graph and the schedule after the second rotation



Step	FU1	FU2	FU3
1	B(110)	C(110)	D(001)
2	E(110)	A(001)	
3	F(110)		
4	G(110)		

(a)



Step	FU1	FU2	FU3
1	E(110)	A(001)	
2	F(110)	B(110)	C(110)
3	G(110)	D(001)	

(b)

2.4 The power cost model

Switching activity is used as the indicator of the power consumption in our work. The switching activity of node u bound to functional unit FU_i, called *Switch_Node*(u, FU_i), is defined as the hamming distance between *LAST_OP*(FU_i) and OP(u), where OP(u) is the state of signal of u and *LAST_OP*(FU_i) is the state of signal of the node executed on FU_i before u. The switching activity of a static schedule for a DFG is defined as the summation of the switching activities of all nodes bound to FUs. Since the static schedule is repeatedly executed for the loop, the initial value of *LAST_OP*(FU_i) is set as OP(u) where u is the last node executed on FU_i in the previous iteration. For example, for the static schedule shown in Figure 2, the initial value of *LAST_OP*(FU_1) is 110(*OP*(G) of G) and the initial value of *LAST_OP*(FU_2) is 110(*OP*(C) of C) and the initial value of *LAST_OP*(FU_3) is 001 (*OP*(D) of D).

For a static schedule S, Switch_Act(S) is used to denote its switching activity, where:

$$\text{Switch\_Act}(S) = \sum_{\text{FU}_i} \sum_{u \text{ assigned to FU}_i} \text{Switch\_Node}(u, \text{FU}_i).$$

For example, Switch_Act(S) = 6 for the static schedule S in Figure 2, where the switching activities are 3 + 3 + 0 + 0 + 0 = 6 on FU_1 and 0 on FU_3 and FU_4. The switching activity remains 6 for both schedules in Figure 3(a) and Figure 3(b). Here, in order to make it simple, we assume that the state on a FU will not change with an empty slot. It may not be true for some optimisation problems. For example, when the problem is to optimise switching activities on an instruction bus, an empty slot will represent a 'NOP' instruction and will cause switching activities. As shown in Section 4, our algorithm is general and can be easily extended to deal with all cases.

The problem we intend to solve is defined as follows. Given a cyclic DFG $G = \langle V, E, \text{OP}, d \rangle$ that models a loop and a set of FUs, find a static schedule S of G such that S has the minimum switching activities in all possible minimum latency schedules. We call the problem as the min-latency-switching-activity scheduling problem.

3 Complexity analysis

In this section, we analyse the complexity of the min-latency-switching-activity scheduling problem. In previous work such as (Masselos et al., 2000; Choi and Chatterjee, 2001), the power efficient scheduling problem is formulated as the Travelling Salesman Problem (TSP) and solved by heuristics of TSP when there is one FU. However, because a problem can be transformed to TSP, it does not necessarily mean that it is NP-complete. For example, the problem to sequence jobs that require common resources on a single machine (Veen and Woeginger, 1998) can be transformed to TSP but still is polynomial time solvable. In this section, we formally prove that the min-latency-switching-activity scheduling problem is NP-complete with or without the resource constraints. Note that the minimum latency loop scheduling problem is polynomial time solvable if there is only one FU or no resource constraints. We show that it becomes NP-complete when switching activities are considered as the second constraint. We categorise the problem into three cases and give proofs as follows.

3.1 1 < the number of resources < infinite

When the number of resources is greater than one but not infinite, it is known that the minimum latency loop scheduling is NP-complete (Garey and Johnson, 1979). So the min-latency-switching-activity scheduling problem is also NP-complete.

Theorem 3.1: *Let U be the number of resources, where U > 1 and U < ∞, min-latency-switching-activity scheduling problem is NP-complete.*

Proof 3.1: When $U > 1$ and $U < \infty$, the minimum latency loop scheduling problem is NP-complete (Garey and Johnson, 1979). Given an instance of the minimum latency loop scheduling problem, we can assigning all nodes with the same OP(u) to get an instance of our problem. Thus, we transform the minimum latency loop scheduling problem to our problem in polynomial time. □

3.2 The number of resources = 1

When the number of resources equals one, it is known that the minimum latency loop scheduling is trivially polynomial time solvable. However, this is not the case when switching activities are considered as the second constraint.

Theorem 3.2: *Let U be the number of resources, when U = 1, min-latency-switching-activity scheduling problem is NP-complete.*

In order to prove Theorem 3.2, we first define the decision problem (DP1) of min-latency-switching-activity scheduling problem when $U = 1$.

DP1: Given a cyclic DFG $G = \langle V, E, \text{OP}, d \rangle$, one FU and two constants D and K, does there exist a static schedule that has the schedule length at most D and has the switching activity at most K?

In our proof, we will transform the L_1 Geometric Travelling Salesman Problem (GTSP) to our problem. GTSP is defined as follows (Garey and Johnson, 1976).

The L_1 geometric travelling salesman problem (GTSP): Given a set S of integer coordinate points in the plane and a constant L, does there exist a circuit passing through all the points of S which, with edge length measured by L_1, has total length less than or equal to L?

Proof 3.2: It is obvious DP1 belongs to NP. Assume $S = \{[x_1, y_1], [x_2, y_2], \ldots, [x_n, y_n]\}$ is an instance of GTSP. Construct DFG $G = \langle V, E, \text{OP}, d \rangle$ as follows. $V = \langle v_1, v_2, \ldots, v_n \rangle$ where v_i corresponds to a point $[x_i, y_i]$ in S. $E = \phi$. Assume that $X = \max(x_i)$ and $Y = \max(y_i)$ for $1 \le i \le n$, then $\text{OP}(v_i) = (X - x_i)0's \bullet x_i1's \bullet (Y - y_i)0's \bullet y_i1's$ for each $v_i \in V (1 \le i \le n)$, where '•' denotes concatenation. For example, if $X = Y = 3$, $x_1 = 2$ and $y_1 = 1$, then $\text{OP}(v_1) = 011\ 001$. Set $D = n$ and $K = L$. Since GTSP is NP-complete and the reduction can be done in polynomial time, DP1 is NP-Complete. □

3.3 No resource constraints

When there are no resource constraints, the minimum latency loop scheduling problem is polynomial time solvable. Retiming (Leiserson and Saxe, 1991) can be used to find an optimal solution. However, when switching activities are considered, the problem becomes NP-complete.

Theorem 3.3: *Let U be the number of resources, when U = ∞, min-latency-switching-activity scheduling problem is NP-complete.*

The decision problem (DP2) of min-latency-switching-activity scheduling problem when $U = \infty$ is similar to DP1 except

that there is one FU in DP1 while no resource constraint in DP2. The proof of Theorem 3.2 is as follows.

Proof 3.3: It is obvious DP2 belongs to NP. Assume $S = \{[x_1, y_1], [x_2, y_2], \ldots, [x_n, y_n]\}$ is an instance of GTSP. Construct DFG $G = \langle V, E, \text{OP}, d\rangle$ as follows. $V = V^{(1)} \cup V^{(2)}$, where $V^{(1)} = \langle v_1^{(1)}, v_2^{(1)}, ..., v_n^{(1)}\rangle$ and $V^{(2)} = \langle v_1^{(2)}, v_2^{(2)}, ..., v_n^{(2)}\rangle$. The nodes in $V^{(1)}$ correspond to the points in S. Assume that $X = \max(x_i)$ and $Y = \max(y_i)$ for $1 \le i \le n$, then $\text{OP}(v_i^{(1)}) = (X + Y + 2)1's \bullet (X - x_i)0's \bullet x_i 1's \bullet (Y - y_i)0's \bullet y_i 1's$ for each node $v_i^{(1)} \in V^{(1)} (1 \le i \le n)$. For example, if $X = Y = 3$, $x_1 = 2$ and $y_1 = 1$, then $\text{OP}(v_1^{(1)}) = 11111111\ 011\ 001$. The nodes in $V^{(2)}$ construct a cycle. Set $\text{OP}(v_1^{(1)})$ all 0's for $1 \le i \le n$. Add edge $e(v_i^{(2)} \to v_{i+1}^{(2)})$ to E and set $d(e(v_i^{(2)} \to v_{i+1}^{(2)})) = 0$ for $1 \le i \le (n-1)$. Add edge $e(v_n^{(2)} \to v_1^{(2)})$ to E and set $d(e(v_n^{(2)} \to v_1^{(2)})) = 1$. Set $D = n$ and $K = L$. Set the initial state of signal of each FU to all 0' □

With the construction of $V^{(2)}$, the assignment of nodes in $V^{(2)}$ does not introduce switching activities and the minimum schedule length equals n. The construction of $V^{(1)}$ makes all nodes in $V^{(1)}$ to be assigned to the same FU for minimising switching activities. Since the reduction can be done in polynomial time, DP2 is NP-complete.

4 The PRRS algorithm

In this section, an algorithm, Power Reduction Rotation Scheduling (PRRS), is designed to solve the min-latency-switching-activity scheduling problem based on rotation scheduling. The basic idea is to generate the schedules by repeatedly rotating down and reallocating nodes with minimising schedule length and switching activities based on Rotation Scheduling, and then select a best schedule that has the minimal switching activities The PRRS algorithm is shown in Algorithm 4.1.

Theorem 4.1: *Power-Reduction-Rotation-Scheduling (PRRS)*

DFG $G = \langle V, E, OP, d\rangle$, the retiming r of G, an initial schedule S of G, the rotation times N

A schedule S and the retiming rk = 1 to N

R ← All nodes in the first row in S;

Delete the first row from S;

Shift S up by 1 control step;

$u \in R$

$r(u) \leftarrow r(u) + 1$;

$u \in R$

$T \leftarrow$ All available locations of u from Row 1 to Row L in S based on the precedence relation in G_r;

$E = \phi$

$T \leftarrow$ All available locations of u in Row $L + 1$ in S;

$[a, b] \leftarrow$ *The location with the minimum switching activities among all locations in T;*

Put u into $[a, b]$;

$S_k \leftarrow S$; $r_k \leftarrow r$;

Select S_j *from* $S_1, S_2, \ldots, S_N$ *such that* S_j *has the minimum switching activities among all minimum-latency schedules;*

Output S_j *and* r_j;

In this algorithm, we first put all nodes in the first row of S into set R. Then we delete the first row of S and shift S up by one control step. Variable L is used to record the schedule length of S. After that, we retime each node $u \in R$ such that $r(u) \leftarrow r(u) + 1$. Then based on the precedence relation in the retimed graph G_r, we rotate each node $u \in R$ by putting u into the location with the minimum switching activities among all available empty locations in T, where T is the set containing all available locations of u.

We obtain the best location for a rotated node by the following strategy. For a location $[i, j] \in T$, we define a function, *Switch_Location*$(u, [i, j])$, to compute the switching activities if u is assigned to location $[i, j]$. Assume that u' is the node in the first nonempty location above $[i, j]$ and u'' is the node in the first nonempty location below $[i, j]$ both in column j of S, then *Switch_Location*$(u, [i, j]) = \text{HD}(\text{OP}(u'), OP(u)) + \text{HD}(\text{OP}(u), \text{OP}(u'')) - \text{HD}(\text{OP}(u'), \text{OP}(u''))$, where HD$(x, y)$ represents the hamming distance of x and y. When computing T, the available locations from row 1 to row L are considered first. If there are no available locations in this field, we assign the node to the locations in row $L + 1$. Using this strategy, the schedule length is minimised as a first priority. After all nodes in R are scheduled, the schedule S and the retiming r are recorded. *PRRS* will repeat the above procedure N times, where N is a user specified amount. A best schedule is selected from the generated N schedules, which has the minimum switching activities among all minlatency schedules.

An example is shown in Figure 4, where the schedules shown in Figure 2 in Section 2 are rotated. Figure 4(a) shows the schedule obtained by removing the first row from the original schedule (Figure 2). There is only one node A in the rotated node set. Figure 4(b) shows the rotated node A and the available empty location set T. The number above the line between A and a location in T is the number of bit switches if A is put into the location. The best location, [2, 3], is selected and it is the earliest location with the minimum switches. So A is put into location [2, 3] in the new schedule. The schedules generated by PRRS after the first and second rotation is shown in Figure 5. The switching activity is 0 for both schedules while it is 6 for both schedules in Figure 3 generated by the traditional rotation scheduling. This shows that our PRRS can significantly reduce

switching activities compared to the traditional rotation scheduling.

Algorithm 4.1 Power-Reduction-Rotation-Scheduling (PRRS)

Require: DFG $G = <V, E, \text{OP}, d>$, the retiming r of G, an initial schedule S of G, the rotation times N

Ensure: A schedule S and the retiming r

for all $k = 1$ to N **do**

$R \leftarrow$ All nodes in the first row in S;

Delete the first row from S;

Shift S up by 1 control step;

for all $u \in R$ **do**

$r(u) \leftarrow r(u) + 1$;

end for

for all $u \in R$ **do**

$T \leftarrow$ All available locations of u from Row 1 to Row L in S based on the precedence relation in G_r;

if $E = \emptyset$ **then**

$T \leftarrow$ All available locations of u in Row $L + 1$ in S;

end if

$[a, b] \leftarrow$ The location with the minimum switching activities among all locations in T;

Put u into $[a, b]$;

end for

$S_k \leftarrow S$; $r_k \leftarrow r$;

end for

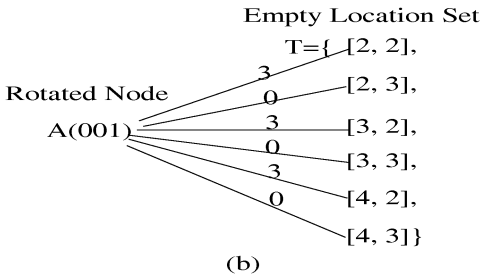
Select S_j from $S_1, S_2, \ldots, S_N$ such that S_j has the minimum switching activities among all minimum-latency schedules;

Output S_j and r_j;

Figure 4 (a) The schedule obtained by removing the first row from the schedule in Figure 2 and (b) The rotated node A and the available empty location set T

Step	FU1	FU2	FU3
1	B(110)	C(110)	D(001)
2	E(110)		
3	F(110)		
4	G(110)		

(a)



(b)

Figure 5 The schedules generated by *PRRS* algorithm both with the switching activity of 0; (a) the schedule after the first rotation and (b) the schedule after the second rotation

Step	FU1	FU2	FU3
1	B(110)	C(110)	D(001)
2	E(110)		A(001)
3	F(110)		
4	G(110)		

(a)

Step	FU1	FU2	FU3
1	E(110)		A(001)
2	F(110)	B(110)	D(001)
3	G(110)	C(110)	

(b)

Let M be the number of functional units and n be the number of nodes in G. Then the number of nodes in a row in a schedule is at most M and the total number of empty locations is at most $M \times (n - 1)$. Considering the rotation times N, the complexity of *PRRS* algorithm is $O(N \times M \times M \times (n - 1) = O(N \times M^2 \times n)$.

5 Experiments

In this section, we conduct experiments with the PRRS algorithm on a set of benchmarks including 4-stage lattice filter, 8-stage lattice filter, differential equation solver, elliptic filter and voltera filter. The experiments are performed on a VLIW simulator with architecture similar to TI C6000 DSP. The optimisation problem for reducing switching activities on the instruction bus is used in the experiments and the real binary code of instructions from TI TMS320C6000 Instruction Set (2000) is used as OP(u) for each node u.

We compare our results with those from list scheduling, the traditional rotation algorithm and the low power allocation approach in Kruse et al. (2001). In the list scheduling, the priority of a node is set as the longest path from this node to a leaf node (Micheli, 1994). In the low power allocation approach, the schedule is fixed and the allocation is performed to reduce switching activities. We implement an algorithm, LPAllocation, based on this approach. LPAllocation uses the schedule generated by traditional rotation scheduling and performs the allocation by bipartite matching.

The experiments are performed on a Dell PC with a P4 2.1 G processor and 512 MB memory running Red Hat Linux 9.0. In the experiments, the running time of PRRS on each benchmark is less than one minute.

The experimental results for the list scheduling, rotation scheduling, and our PRRS algorithm, are shown in Table 1 when the number of FUs is 4, 5 and 6, respectively. Column 'SA' presents the switching activity of the static schedule and Column 'SL' presents the schedule length obtained from three different scheduling algorithms: the list scheduling (Field 'List'), the traditional rotation scheduling

(Field 'Rotation') and our PRRS algorithm (Field 'PRRS'). Column 'SL (%)' and 'SA (%)' under 'PRRS' present the percentage of reduction in schedule length and switching activities respectively compared to the list scheduling algorithm. The average reduction is shown in the last row of the table. PRRS shows an average 20.1% reduction in schedule length and 52.2% reduction in bus switching activities compared with the list scheduling.

Table 1 The comparison of bus switching activities and schedule length for list scheduling, rotation scheduling and PRRS

	List		*Rotation*		*PRRS*			
Bench	*SA*	*SL*	*SA*	*SL*	*SA*	*SA* (%)	*SL*	*SL* (%)
The number of FUs = 4								
4-Lattice	68	9	72	7	38	44.1	7	22.2
8-Lattice	108	17	118	11	68	37.0	11	35.3
DEQ	30	5	32	4	14	53.3	4	20.0
Elliptic	136	14	136	14	86	36.8	14	0.0
Voltera	70	12	68	12	38	45.7	12	0.0
The number of FUs = 5								
4-Lattice	74	9	80	6	32	56.8	6	33.3
8-Lattice	106	17	112	9	68	35.8	9	47.1
DEQ	30	5	36	4	10	66.7	4	20
Elliptic	136	14	136	14	58	57.4	14	0.0
Voltera	72	12	72	12	26	63.9	12	0.0
The number of FUs = 6								
4-Lattice	76	9	68	5	34	55.3	5	44.4
8-Lattice	104	17	116	7	68	34.6	7	58.8
DEQ	30	5	36	4	6	80.0	4	20.0
Elliptic	136	14	136	14	40	70.6	14	0.0
Voltera	66	12	72	12	36	45.5	12	0.0
Average reduction (%) *over list*						52.2	–	20.1

We conduct experiments to compare the performance of PRRS with that of LPAllocation, the algorithm based on the approach in Kruse et al. (2001), The experimental results on the various benchmarks are shown in Table 2 when the number of FUs is 4, 5 and 6, respectively. In the table, 'LPAlloc' presents algorithm LPAllocation. PRRS shows an average 20.7% reduction in bus switching activity compared with LPAllocation.

To demonstrate the influence of the number of FUs, Table 3 shows the switching activity and schedule length for 8-stage Lattice filter for different scheduling algorithms when the number of FUs varies from 3 to 12. The experimental results show that when the number of FUs increases, the percentage of reduction on switching activities increases correspondingly.

In summary, from Tables 1–3, we found that the list scheduling shows inferior performance in both schedule length and switching activities for applications with loops. The traditional rotation scheduling can effectively reduce schedule length but not switching activities. The LPAllocation algorithm can reduce switching activities for a fixed schedule. Our PRRS can reduce both schedule length and switching activities, and it yields greater reduction on switching activities compared with the LPAllocation algorithm based on the approach in Kruse et al. (2001).

Table 2 The comparison of bus switching activities for PRRS and LPAllocation

	LPAlloc	*PRRS*	
Bench	*SA*	*SA*	%
The number of FUs = 4			
4-Lattice	50	38	24.0
8-Lattice	94	68	27.7
DEQ	16	14	12.5
Elliptic	86	86	0.0
Voltera	42	38	9.5
The number of FUs = 5			
4-Lattice	58	32	44.8
8-Lattice	82	68	17.1
DEQ	16	10	37.5
Elliptic	68	58	14.7
Voltera	40	26	35.0
The number of FUs = 6			
4-Lattice	38	34	10.5
8-Lattice	76	68	10.5
DEQ	14	6	57.1
Elliptic	44	40	9.1
Voltera	36	36	0.0
Average reduction (%)			20.7

Table 3 Comparison of switching activities and schedule length for 8- lattice filter when no of FUs varies from 3 to 12

	List		*Rotation*		*LPAlloc*		*PRRS*		
FUs	*SA*	*SL*	*SA*	*SL*	*SA*	*SL*	*SA*	*SL*	%
3	106	17	118	14	90	14	86	14	27.1
4	108	17	118	11	94	11	68	11	42.4
5	106	17	112	9	82	9	68	9	39.3
6	104	17	116	7	76	7	68	7	41.4
7	96	17	120	6	58	6	58	6	51.7
8	110	17	120	6	58	6	30	6	75.0
9	110	17	120	5	84	5	38	5	68.3
10	114	17	110	5	66	5	20	5	81.8
11	112	17	120	4	44	4	30	4	75.0
12	102	17	106	4	76	4	26	4	75.5
Average reduction (%)									57.7

6 Conclusion

This paper studied low power loop scheduling problem and attempted to minimise both the schedule length and the power consumption for applications with loops on multiple-functional-unit architectures. We showed that to find a schedule that has the minimal switching activity among all minimum-latency schedules with or without resource constraints is NP-complete. An algorithm, Power Reduction Rotation Scheduling, was proposed. The algorithm minimises both the switching activity and the schedule length based on rotation scheduling when performing the scheduling and allocation simultaneously. The experimental results show that our algorithm can greatly reduce switching activities and schedule length compared to the existing approaches.

Acknowledgements

This work is partially supported by TI University Program, NSF EIA-0103709, Texas ARP 009741-0028-2001, NSF CCR-0309461, USA and HK POLYU A-PF86 and COMP 4-Z077, HK.

References

Alidina, M., Monteiro, J., Devadas, S., Ghosh, A. and Papaefthymiou, M. (1994) 'Precomputation-based sequential logic optimization for low power', *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, December, Vol. 2, No. 4, pp.426–436.

Benini, L. and Micheli, G.D. (1995) 'State assignment for low power dissipation', *IEEE J. Solid-State Circuit*, March, Vol. 30, No. 3, pp.258–268.

Chandrakasan, A., Sheng, S. and Brodersen, R. (1992) 'Low-power cmos digital design', *IEEE Journal of Solid-State Circuits*, April, Vol. 27, No. 4, pp.473–484.

Chang, J. and Pedram, M. (1995) 'Register allocation and binding for low power', in *Proc. of the 32nd ACM/IEEE Design Automation Conference*, June, pp.29–35.

Chao, L-F., LaPaugh, A.S. and Sha, E.H-M. (1997) 'Rotation scheduling: A loop pipelining algorithm', *IEEE Trans. on Computer-Aided Design*, March, Vol. 16, No. 3, pp.229–239.

Choi, K. and Chatterjee, A. (2001) 'Efficient instruction-level optimization methodology for low-power embedded systems', in *Proc. of the IEEE Int. Symp. on System Synthesis,* October, pp.147–152.

Erdogan, A. and Arslan, T. (2002) 'On the low power implementation of fir filtering structures on single multiplier DSPs', *IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing*, March, Vol. 49, No. 3, pp.223–229.

Garey, M.R. and Johnson, D.S. (1979) *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, San Francisco, CA.

Garey, M.R. and Johnson, D.S. (1976) 'Some NP-complete geometric problems', in *Proc. of the ACM Symp. on Theory of Computing,* May, pp.10–22.

Hachtel, G.D., Hermida, M., Pardo, A., Poncino, M. and Somenzi, F. (1994) 'Re-encoding sequential circuits to reduce power dissipation', in *the 1994 IEEE/ACM international conference on Computer-aided design*, November, pp.70–73.

Henning, R. and Chakrabarti, C. (1998) 'Relating data characteristics to transition activity in high-level static cmos design', in *13th International Conference on VLSI Design*, January, pp.38–43.

Henning, R. and Chakrabarti, C. (2002) 'An approach to switching activity consideration during high level low power design space exploration', *IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing*, May, Vol. 49, No. 5, pp.339–351.

Huff, R.A. (1993) 'Lifetime-sensitive modulo scheduling', in *the ACM SIGPLAN 1993 conference on Programming language design and implementation,* June, pp.258–267.

Kim, D., Shin, D. and Choi, K. (2001) 'Low power pipelining of linear systems: a common operand centric approach', in *Proc. of the IEEE/ACM Int. Symp. on Low Power Design*, August, pp.225–230.

Kruse, L., Schmidt, E., Jochens, G., Stammermann, A., Schulz, A., Macii, E. and Nebel, W. (2001) 'Estimation of lower and upper bounds on the power consumption from schedule data flow graphs', *IEEE Trans. on VLSI Systems*, February, Vol. 9, No. 1, pp.3–14.

Lam, M. (1988) 'Software pipelining: an effective scheduling technique for vliw machines', in *the ACM SIGPLAN 1988 conference on Programming Language design and Implementation,* June, pp.318–328.

Lee, C., Lee, J-K. and Hwang, T. (2003) 'Compiler optimization on VLIW instruction scheduling for low power', *ACM Transactions on Design Automation of Electronic Systems*, April, Vol. 8, No. 2, pp.252–268.

Lee, M.T-C., Fujita, M., Tiwari, V. and Malik, S. (1997) 'Power analysis and minimization techniques for embedded dsp software', *IEEE Transactions on VLSI Systems*, March, Vol. 5, No. 1, pp.123–135.

Leiserson, C.E. and Saxe, J.B. (1991) 'Retiming synchronous circuitry', *Algorithmica*, Vol. 6, pp.5–35.

Macii, E., Pedram, M. and Somenzi, F. (1998) 'High-level power modeling, estimation and optimization', *IEEE Trans. on Computer-Aided Design*, November, Vol. 17, pp.1061–1079.

Masselos, K., Theoharis, S., Merakos, P.K., Stouraitis, T. and Goutis, C.E. (2000) 'Low power synthesis of sum-of-products computation', in *Proc. of the IEEE/ACM Int. Symp. on Low Power Electronics and Design,* July, pp.234–237.

Mehendale, M., Sherlekar, S. and Venkatesh, G. (1995) 'Coefficient optimization for low power realization of fir filters', in *IEEE Workshop on VLSI Signal Processing*, pp.352–361.

Micheli, G.D. (1994) *Synthesis and Optimization of Digital Circuits*, McGraw-Hill, New York, NY.

Musoll, E. and Cortadella, J. (1995a) 'Scheduling and resource binding for low power', in *Proc. of the IEEE Int. Symp. on System Synthesis*, April, pp.104–109.

Musoll, E. and Cortadella, J. (1995b) 'High-level synthesis techniques for reducing the activity of low power', in *Proc. of the IEEE/ACM Int. Symp. on Low Power Design*, April, pp.99–104.

Panda, P. and Dutt, N. (1999) 'Low power memory mapping through reducing address bus activity', *IEEE Trans. on VLSI Syst.*, September, Vol. 7, No. 3, pp.309–320.

Parhi, K.K. (2001) 'Low-power implementation of DSP systems', *IEEE Trans. on Circuits and Systems, Part-I: Fundamental Theory and Applications*, October, Vol. 48, No. 10, pp.1214–1224.

Parikh, A., Kandemir, M., Vijaykrishnan, N. and Irwin, M.J. (2000) 'Instruction scheduling based on energy and performance constraints', in *IEEE Computer Society Annual Workshop on VLSI*, April, pp.37–42.

Raghunathan, A. and Jha, N.K. (1995) 'An ILP formulation for low power based on minimizing switched capacitance during data path allocation', in *Proc. of the IEEE Int. Symp. on Circuits & Systems*, May, pp.1069–1073.

Rau, B.R., Schlansker, M.S. and Tirumalai, P.P. (1992) 'Code generation schema for modulo scheduled loops', in *The 25th annual international symposium on Microarchitecture,* December, pp.158–169.

Roy, K. and Prasad, S. (1992) 'SYCLOP: Synthesis of CMOS logic for low power applications', in *The 1991 IEEE International Conference on Computer Design on VLSI in Computer & Processors*, October, pp.464–467.

Stan, M.R. and Burleson, W.P. (1995) 'Bus-invert coding for low-power i/o', *IEEE Trans. on VLSI Syst.*, March, Vol. 3, No. 1, pp.49–58.

Su, C-L., Tsui, C-Y. and Despain, A.M. (1994) 'Saving power in the control path of embedded processors', *IEEE Design & Test of Computers*, Winter, Vol. 11, No. 4, pp.24–30.

Sundararajan, V. and Parhi, K.K. (2000) 'Synthesis of low power folded programmable coefficient fir digital filters', in *2000 IEEE Asia Pacific Design Automation Conference*, January, pp.153–156.

Tiwari, V., Malik, S. and Wolfe, A. (1994a) 'Compilation techniques for low energy: An overview', in *the Symposium on Low Power Electronics*, pp.38–39.

Tiwari, V., Malik, S. and Wolfe, A. (1994b) 'Power analysis of embedded software: a first step towards software power minimization', *IEEE Transactions on VLSI Systems*, December, Vol. 2, No. 4, pp.437–445.

TMS320C6000 CPU and Instruction Set Reference Guide (2000) Texas Instruments, Inc. (literature number SPRU189F).

Toburen, M.C., Conte, T.M. and Reilly, M. (1998) 'Instruction scheduling for low power dissipation in high performance processors', in *The Power Driven Micro-architecture Workshop in conjunction with the ISCA'98*, June.

Tsui, C-Y., Pedram, M. and Despain, A.M. (1993) 'Technology decomposition and mapping targeting low power dissipation', in *The 30th international on Design automation conference,* June, pp.68–73.

Veen, J.V.D. and Woeginger, S.Z.G.J. (1998) 'Sequencing jobs that require common resources on a single machine: a solvable case of the TSP', *Mathematical Programming,* No. 82, pp.235–254.

Yang, H., Gao, G.R. and Leung, C. (2002) 'On achieving balanced power consumption in software pipelined loops', in *International Conference on Compilers, Architectures and Synthesis for Embedded Systems*, pp.210–217.

Yu, T.Z., Chen, F. and Sha, E.H-M. (1998) 'Loop scheduling algorithms for power reduction', in *Proc. of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, May, Vol. 5, pp.3073–3076.

Yun, H-S. and Kim, J. (2001) 'Power-aware modulo scheduling for high-performance vliw processors', in *the 2001 International Symposium on Low Power Electronics and Design,* August, pp.40–45.