

Transactions Briefs

Dynamic and Leakage Energy Minimization With Soft Real-Time Loop Scheduling and Voltage Assignment

Meikang Qiu, Laurence T. Yang, Zili Shao, and Edwin H.-M. Sha

Abstract—With the shrinking of technology feature sizes, the share of leakage in total power consumption of digital systems continues to grow. Traditional *dynamic voltage scaling* (DVS) fails to accurately address the impact of scaling on system power consumption as the leakage power increases exponentially. The combination of DVS and *adaptive body biasing* (ABB) is an effective technique to jointly optimize dynamic and leakage energy dissipation. In this paper, we propose an optimal soft real-time loop scheduling and voltage assignment algorithm, *loop scheduling and voltage assignment to minimize energy*, to minimize both dynamic and leakage energy via DVS and ABB. Voltage transition overhead has been considered in our approach. We conduct simulations on a set of digital signal processor benchmarks based on the power model of 70 nm technology. The simulation results show that our approach achieves significant energy saving compared to that of the integer linear programming approach.

Index Terms—Assignment, dynamic voltage scaling (DVS), leakage power, loop scheduling, real-time.

I. INTRODUCTION

Energy consumption has become one of the most important design concerns for digital systems due to its significant impact on battery life, system density, cooling costs, and reliable operation [1]. In previous approaches, dynamic power was the primary contributor to total power dissipation of a CMOS design. The quadratic dependence of dynamic power on supply voltage, along with the lower order impact of supply voltage on clock frequency motivated the idea of frequency and supply voltage scaling for processors. *Dynamic voltage scaling* (DVS) is one of the most powerful techniques to reduce energy consumption. Much research has been done on DVS for real-time applications in recent years. For example, Zhang *et al.* [2] proposed an integer linear programming (ILP) model to solve DVS on multiple processor systems.

The leakage power, whose share in total power increases with the scaling of CMOS technology, is not explicitly addressed using the DVS technique. Consequently, the effectiveness of traditional DVS is limited with advancement of technology [3]. *Adaptive body biasing* (ABB) [4] is an effective technique to reduce leakage power by increasing the circuit's threshold voltage via body biasing. In this paper, we will combine DVS and ABB to minimize both dynamic and leakage energy.

Manuscript received March 01, 2008; revised June 26, 2008 and September 12, 2008. First published April 14, 2009; current version published February 24, 2010. This work was supported in part by the NSF under Grant CCR-0309461 and Grant IIS-0513669, by the Research Grants Council of the Hong Kong Special Administrative Region, China (CERG 526007, PolyU B-Q06B, and PolyU A-PH41), by the NSFC under Grant 60728206.

M. Qiu is with the Department of Electrical and Computer Engineering, University of New Orleans, New Orleans, LA 70148 USA (e-mail: mqiu@uno.edu).

L. T. Yang is with the Department of Computer Science, St. Francis Xavier University, Antigonish B2G 2W5, Canada (e-mail: ityang@stfx.ca).

Z. Shao is with the Department of Computing, Hong Kong Polytechnic University, Kowloon, Hong Kong (e-mail: cszshao@comp.polyu.edu.hk).

E. H.-M. Sha is with the Department of Computer Science, University of Texas at Dallas, Richardson, TX 75083 USA (e-mail: edsha@utdallas.edu).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TVLSI.2008.2010941

In DSP systems, some tasks may not have fixed execution time. Such tasks usually contain conditional instructions and/or operations that could have different execution time for different inputs [5]. It is possible to obtain the execution time distribution for each task by sampling or profiling [6]. Prior design space exploration methods for hardware/software codesign of embedded systems guarantee no deadline missing by considering worst-case execution time of each task. These methods are pessimistic and will often lead to overdesigned systems with high cost. There are several papers on the probabilistic timing performance estimation for soft real-time systems design [5]. They modeled each task's execution time as a random variable. In this paper, we use soft real-time approach and loop scheduling to avoid overdesigning systems. We propose an optimal algorithm to minimize the expected value of total energy consumption while satisfying timing constraints with guaranteed probabilities for real-time applications.

We design a novel loop scheduling algorithm for real-time applications that produce schedules consuming minimal energy. In our algorithm, we use rotation scheduling [7] to get schedules for loop applications. The schedule length will be reduced after rotation. Then, we use DVS and ABB to assign voltages to computations individually in order to decrease the voltages of processors as much as possible within the timing constraint. In our algorithm, we jointly optimize dynamic power and leakage power with the consideration of time and energy overhead of voltage transition. To the best of our knowledge, this is the first paper to deal with voltage assignment using a soft real-time approach with the consideration of voltage transition overhead. We implement simulations on a set of benchmarks. The simulation results show that our algorithms can get better results on energy saving than the previous work. On average, our algorithm LSVAME shows a 21.8% reduction compared with the ILP technique in [2] for hard real-time and even better for soft real-time (38.1% on average with 90% guaranteed probability satisfying timing constraints).

II. MODELS AND CONCEPTS

A. Energy Model

In this subsection, we briefly overview the impact of supply voltage and body bias on processor's frequency and power consumption. We summarize the previous study by Martin *et al.* [4] that derives threshold voltage, power consumption, and the performance of the design as functions of its supply and bias voltages. Subsequently, we proceed to present the power and performance parameters of the proposed processor with DVS and ABB capabilities.

The dynamic power consumption (P_{AC}) of CMOS circuits is given by

$$P_{AC} = C_{\text{eff}} V_{dd}^2 f \quad (1)$$

where V_{dd} is the supply voltage, f is the operating frequency, and C_{eff} is the effective switching capacitance. DVS reduces the dynamic power consumption due to its quadratic dependence on voltage [8].

Different leakage sources contribute to the total static power consumption in a device. The major contributors of leakage are the subthreshold leakage and the reverse bias junction current which can increase significantly with ABB [4].

The leakage power dissipation (P_{dc}) due to subthreshold leakage (I_{subn}) and reverse bias junction current (I_j) is given by

$$P_{dc} = V_{dd} I_{\text{subn}} + |V_{bs}| I_j \quad (2)$$

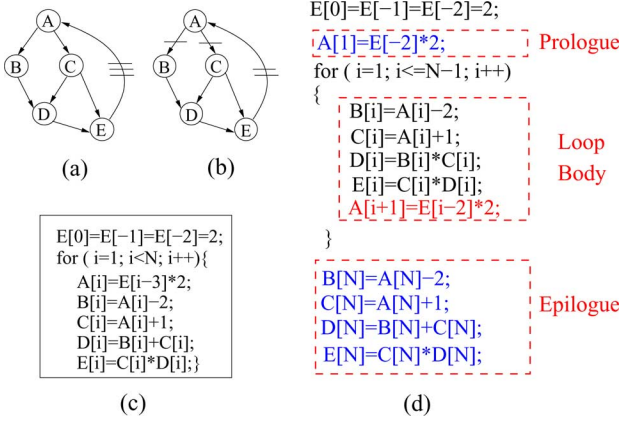


Fig. 1. (a) Original PDFG. (b) The rotated PDFG. (c) The code of loop application corresponding to (a). (d) The equivalent loop after regrouping loop body.

Equation (2) gives the leakage per device and the total leakage power consumption is $L_g \times P_{dc}$, where L_g is the number of devices in the circuit.

The power consumed in a processor is the sum of dynamic, static, and short circuit power. The short circuit power consumption occurs only during signal transitions and is negligible [9].

There is also energy required in switching the circuit between varying power modes. This switching energy, E_s , is given by

$$E_s = |\Delta V_{dd}|^2 C_r + |\Delta V_{bs}|^2 C_s \quad (3)$$

where ΔV_{dd} is the change in V_{dd} , ΔV_{bs} is the change in V_{bs} , C_r is the capacitance of the power rail, and C_s is the total capacitance of the substrate and wells of the device.

B. System Model

PDFG: A probabilistic data-flow graph (PDFG) is used to model a digital system application. A PDFG $G = \langle U, ED, T, V \rangle$ is a directed acyclic graph (DAG), where $U = \langle u_1, \dots, u_i, \dots, u_N \rangle$ is the set of nodes; $ED \subseteq U \times U$ is the edge set that defines the precedence relations among nodes in U . There are maximum M different voltages for a node. $V = \langle V_1, \dots, V_j, \dots, V_M \rangle$ is a voltage set; the execution time of node u under voltage V_j , represented as $T_{V_j}(u)$, is a random variable; $P_{V_j}(u)$ is the corresponding probability and $E_{V_j}(u)$ is the energy consumption; there is a timing constraint L and it must be satisfied for executing the whole PDFG.

Cyclic PDFG: We use a cyclic data-flow graph (DFG) to denote a loop in our work. A cyclic DFG $G = \langle U, ED, d, T, V \rangle$ is a node-weighted and edge-weighted directed graph, where $d(e)$ is a function to represent the number of delays for any edge $e \in ED$, the edge without delay represents the intra-iteration data dependence; the edge with delays represents the inter-iteration data dependence and the number of delays represents the number of iterations involved.

For example, a PDFG is shown in Fig. 1(a). The corresponding loop application is shown in Fig. 1(b). Each node of the PDFG represents a computation task in a loop, the edge without delay represents the intra-iteration data dependence (e.g., $A \rightarrow B$), the edge with delays represents the inter-iteration data dependence (e.g., $E \rightarrow A$ has three delays which are denoted by three bars).

Retiming: Retiming [10] is an optimal scheduling technique for cyclic DFGs considering inter-iteration dependences. It can be used to optimize the cycle period of a cyclic DFG by evenly distributing the delays. Retiming generates the optimal schedule for a cyclic DFG when there is no resource constraint. Given a cyclic DFG $G = \langle U, ED, d, T, V \rangle$, retiming r of G is a function from U to integers. For a node $u \in U$, the value of $r(u)$ is the number of delays drawn from each of incoming edges of node u and pushed to all of

the outgoing edges. Let $G_r = \langle U, ED, d_r, T, V \rangle$ denote the retimed graph of G with retiming r , then $d_r(e) = d(e) + r(u_1) - r(u_2)$ for every edge $e(u_1 \rightarrow u_2) \in ED$.

Rotation Scheduling: Rotation Scheduling [7] is a scheduling technique used to optimize a loop schedule with resource constraints. It transforms a schedule to a more compact one iteratively in a PDFG. By using rotation scheduling, we can get more opportunities to reschedule nodes of DFG to better locations so that the length of a schedule can be reduced. In most cases, the minimal schedule length can be obtained in polynomial time by rotation scheduling.

Fig. 1 shows an example to explain how to obtain a new schedule via rotation scheduling. We use the schedule generated by list scheduling in Fig. 1(a) as an initial schedule. Fig. 1(c) shows the corresponding code. We get a set of nodes at the first row of the schedule, in this case, it is $\{A\}$, and we rotate node A down. The rotated graph is shown in Fig. 1(b). The equivalent loop body after rotation is shown in Fig. 1(d).

From the program point of view, rotation scheduling regroupes a loop body and attempts to reduce intradependences among nodes. In this case, after the rotation, a new loop is obtained by the transformation as shown in Fig. 1(d), where the corresponding computation for node A is rotated and put at the end of the new loop body. One iteration from the old loop is separated and put outside the new loop body: the computation for node A is put in the *prologue* and those for the other nodes are put in the *epilogue*. In the new loop body, node A performs the $(i+1)$ th computation for the iteration when the other nodes do the computation of the i th. The transformed loop body after the rotation scheduling can be obtained based on the retiming values of nodes.

C. LSVAME Problem

Definitions: Define the *loop scheduling and voltage assignment to minimize energy* (LSVAME) problem as follows: given M different voltage levels: V_1, V_2, \dots, V_M , a PDFG $G = \langle U, ED, T, V \rangle$ with $T_{V_j}(u)$, $P_{V_j}(u)$, and $E_{V_j}(u)$ for each node $u \in U$ executed on each voltage V_j , a timing constraint L and a confidence probability P , find the voltage for each node in assignment A that gives the *minimum expected total energy consumption E with confidence probability P under timing constraint L* .

III. ALGORITHMS

A. The LSVAME Algorithm

Algorithm III.1 Optimal algorithm for the LSVAME problem (LSVAME)

Input: a PDFG, M different voltage levels, the retiming r of G , an initial schedule S of G , the rotation times R , the timing constraint L , guaranteed probability P .

Output: A schedule S , an assignment A , and the retiming r , to obtain $\text{MIN}(E)$ with $(\text{Prob.}(T \leq L)) \geq P$.

The algorithm flow chart is shown in Fig. 2

The LSVAME algorithm is shown in Algorithm III.1. In this algorithm, we first get the static schedule (a DAG) from the input PDFG. Then get scheduling graph from the DAG. Next, use the revised version of our algorithm VAP- M [11] on the schedule to obtain the minimum total energy E with $\text{Prob.}(T \leq L) \geq P$. The key idea of the algorithm VAP- M [11] is using a dynamic programming method to travel the PDFG and cancel the redundant data in each step. Finally, we use rotation scheduling (RS) to retime the original PDFG and rotate down the first row of template. LSVAME will repeat the above procedure R times, where R is a user-specified amount. From LSVAME, we can get minimal energy consumption $\text{MIN}(E)$, the corresponding schedule

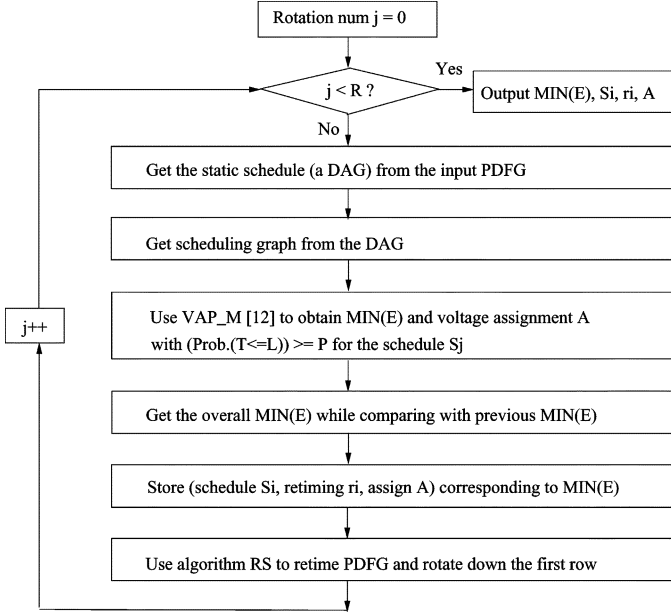


Fig. 2. Flow chart of the LSVAME algorithm.

S_i , retiming r_i , and voltage assignment A . LSVAME algorithm has several advantages: 1). Use loop scheduling to exploit the pipelining and regular pattern of an application. 2). Consider both dynamic power and leakage power with the combination of DVS and ABB. 3). Exploit multi-core architecture by using soft real-time voltage assignment to save energy. 4). Deal with voltage transition overhead in computation.

B. VAP_M Algorithm

In our algorithm LSVAME, we use the revised version of algorithm VAP_M, which is proposed by Qiu *et al.* [11], to obtain the voltage assignment of minimal energy consumption while satisfying timing constraints with guaranteed probability. The basic idea of algorithm VAP_M is to use dynamic programming method traveling the graph in a bottom up fashion. By using dynamic programming, it can cancel many redundant-pair $(P_{i,j}, E_{i,j})$ whenever finding conflicting pairs in a list during a computation.

C. The RS Algorithm

The rotation scheduling (RS) algorithm with our PDFG model is shown in Algorithm III.2. The main idea of this algorithms is: first, get a shorter schedule by rotating an original schedule. Then use the revised version of algorithm VAP_M [11] to minimize energy consumption via better voltage level selection.

Algorithm III.2 Rotation scheduling for the LSVAME problem (RS)

Input: M different voltage levels, a PDFG, and the timing constraint L .

Output: a voltage assignment to minimize energy while satisfying L .

- 1: Input the PDFG G and environment parameter.
 - 2: Create a schedule S using list scheduling.
 - 3: For $i = 1$ to n ,
 - 4: Get a set of nodes U to be rotated;
 - 5: Delete nodes in U from S ;
 - 6: For each $v \in S$, retime v , and get new graph G_r ;
 - 7: Compact S according to G_r ;
 - 8: Insert nodes in U into S according to G_r ;
 - 9: Output a new graph.
-

TABLE I
FREQUENCY LEVELS, CORRESPONDING V_{dd} , V_{bs} ,
AND POWER BASED ON 70 NM TECHNOLOGY

Frequencies	f	GHz	10	12	15
Supply Voltage	V_{dd}	V	0.756	0.885	1.000
Bias Voltage	V_{bs}	V	-0.658	-0.661	-0.675
Power	P	μ W	14.8	22.6	30.5

D. Complexity of LSVAME

The complexity of the algorithm LSVAME is $O(R * |V|^2 * L * M * K^{t+1})$, where R is the rotation time and we set it as $10 * |V|$. In the algorithm VAP_M, there are K^t loops and each loop needs $O(|V|^2 * L * M * K)$ running time. The complexity of VAP_M is $O(K^t * |V|^2 * L * M * K)$. $|V|$ is the number of nodes, L is the given timing constraint, which will decide the number of steps, M is the maximum number of voltage levels for each node, and K is the maximum number of execution time variation for each node. $t = \min(t_{mp}, t_{mc})$, where t_{mp} is the number of nodes with multi-parent, and t_{mc} is the number of nodes with multi-child. The result is optimal since the algorithm covers all possible cases.

IV. SIMULATIONS

We conduct simulations with our algorithm on a set of benchmarks including 2-D filter, wave digital filter (WDF), infinite impulse filter (IIR), differential pulse-code modulation device (DPCM), Floyd-Steinberg algorithm (Floyd), and all-pole filter. The proposed runtime system has been implemented and a simulation framework to evaluate its effectiveness has been built. The distribution of execution times of each node of PDFG is Gaussian. In the simulations, we set the rotation times as $10 * \text{Node_Num}$, where Node_Num is the number of nodes in the PDFG. The results show that the rotation times to generate the best schedule is usually less than $1 * \text{Node_Num}$ (one rotation) and close to the times when all nodes have been rotated one time.

We conducted simulations on five methods: method 1: list scheduling, with DVS; method 2: list scheduling, with DVS and ABB; method 3: ILP in [2], with DVS; method 4: ILP in [2], with DVS and ABB; method 5: our LSVAME algorithm. In method 5, we consider the hard real-time case ($P = 1.00$) and soft real-time case ($P = 0.90$). In the list scheduling, the priority of a node is set as the longest path from this node to a leaf node. The simulations are performed on a PC with a P4 2.1 G processor and 512 MB memory running Red Hat Linux 9.0. The dynamic processor loads were obtained through measurements on a 600 MHz Crusoe processor. We implement our algorithm in the SPAM compiler environment to replace the simulated annealing algorithm originally used by the Princeton project (see Website <http://www.idiom.com/free-compilers/TOOL/SPAMComp-1.html>).

The simulations are conducted based on the power model of 70 nm processor [4], [8]. According to the power model, we can obtain the optimal supply and body bias voltage for a given frequency. Then, the energy consumption per cycle can be calculated by using (9) in paper [4], and the power is derived from the formula $E_{cyc} = P/f$. In the simulations, we use M different voltage types, V_1, \dots, V_M , in which a voltage with level V_1 is the quickest with the highest energy consumption and a voltage with level V_M is the slowest with the lowest energy consumption. Table I shows the case where $M = 3$, i.e., the three V_{dd} levels as well as their corresponding frequency levels, V_{bs} , and power consumptions. The energy overhead during a voltage transition among the above three voltage levels is calculated based on (3) [4], [12].

For each benchmark, we conduct a set of simulations based on three processor cores. The simulation results are shown in Table II. Column “E” represents the minimum total energy consumption obtained from five different scheduling algorithms: method 1 (Field “Med. 1”), method 2 (Field “Med. 2”), method 3 (Field “Med. 3”), method

TABLE II

COMPARISON OF TOTAL ENERGY CONSUMPTION WITH FIVE METHODS WHILE SATISFYING TIMING CONSTRAINT $T = 600$ NS FOR VARIOUS BENCHMARKS

Bench.	3 processor cores, $T = 600$ ns											
					LSVAME(1.0)				LSVAME(0.9)			
	Med. 1 E(nJ)	Med. 2 E(nJ)	Med. 3 E(nJ)	Med. 4 E(nJ)	E(nJ)	(% M2)	(% M3)	(% M4)	E(nJ)	(% M2)	(% M3)	(% M4)
2D(2)	383	315	272	225	176	44.1	35.3%	21.8%	140	55.6%	48.5%	37.8%
WDF(1)	386	918	781	647	514	44.0	34.2%	20.6%	405	55.9%	48.1%	37.4%
MDFG1	781	1227	1052	865	672	45.2	36.1%	22.3%	538	56.2%	48.9%	37.8%
MDFG2	875	1206	1048	842	651	46.0	37.9%	22.7%	521	56.8%	50.3%	38.1%
WDF(2)	1118	2631	2279	1878	1452	44.8	36.3%	22.7%	1139	56.7%	50.0%	39.4%
IIR	1475	321	265	214	172	46.4	35.1%	19.6%	135	57.9%	49.1%	36.9%
DPCM	1486	645	551	459	351	45.6	36.3%	23.5%	281	56.4%	49.0%	38.8%
Floyd	1549	728	628	504	402	44.8	36.0%	20.2%	317	56.5%	49.5%	37.1%
All-pole	2703	1283	1085	897	701	45.4	35.4%	21.9%	546	57.4%	49.7%	39.1%
2D(1)	3198	2227	1926	1569	1210	45.7	37.2%	22.9%	968	56.5%	49.7%	38.3%
Average Reduction (%)						45.2	36.0%	21.8%	–	56.6%	49.3%	38.1%

4 (Field “Med. 4”), and method 5 (Field “LSVAME”). method 5 contains two cases: the hard real-time ($P = 1.00$) and soft real-time ($P = 0.90$). Column “% M4” and “% M3” represent the percentage of reduction in total energy consumption, compared to method 4 and 3, respectively. The total average improvement of LSVAME is shown in the last row of the tables.

From the simulation results, we can see that our algorithm achieves significant energy saving compared with the ILP in [2]. For example, with three processor cores, on average, LSVAME shows 21.8% reduction in hard real-time, and reductions of 38.1%, with probability 0.9, for soft real-time scenario. The reason of such big improvement is because we use loop scheduling and DVS+VBB to shrink the schedule length and assign the best possible voltage level to minimize energy while satisfying time constraint with guaranteed probabilities.

We also see that our algorithm achieves significant energy reduction compared with the traditional DVS method. For example, with three processor cores, compared with Method 3 (the ILP in [2] with DVS), LSVAME shows 36.0% reduction in hard real-time, and reductions of 49.3%, with probability 0.9, for soft real-time scenario. Our method consider energy reduction for both dynamic and leakage with the combination of DVS and ABB. Compared with Method 2 (DVS/ABB with the initial schedule obtained from list scheduling, and without rotation scheduling), LSVAME shows 45.2% reduction in hard real-time, and reductions of 56.6%, with probability 0.9, for soft real-time scenario.

We also unfold all the benchmarks 5–15 times, which means the number of nodes increases 5–15 times. Originally, the largest benchmark 2-D (1) originally has only 34 nodes. After unfolding 10 times, the benchmark has 340 nodes. In the simulations, the running time of LSVAME on each benchmark is less than 15 minutes. But the running time of ILP [2] on each benchmark needs more than 4 hours.

V. CONCLUSION

This paper combined loop scheduling and voltage assignment via DVS and ABB to minimize both dynamic and leakage power consumption with the consideration of voltage transition overhead. An optimal algorithm, LSVAME, is proposed. By taking advantage of the uncertainties in execution time of tasks, our approach gives tasks scheduling and voltage assignments and to minimize the expected total energy consumption while satisfying timing constraint with guaranteed probabilities.

REFERENCES

- [1] P. Huang and S. Ghiasi, “Leakage-aware intraprogram voltage scaling for embedded processors,” in *Proc. DAC*, 2006, pp. 364–369.
- [2] Y. Zhang, X. Hu, and D. Z. Chen, “Task scheduling and voltage selection for energy minimization,” in *Proc. DAC*, 2002, pp. 183–188.
- [3] D. Duarte, N. Vijaykrishnan, M. J. Twin, H.-S. Kim, and G. McFarland, “Impact of scaling on the effectiveness of dynamic power reduction schemes,” in *Proc. ICCD*, 2002, pp. 382–387.
- [4] S. Martin, K. Flautner, T. Mudge, and D. Blaauw, “Combined DVS and ABB for lower power microprocessors under dynamic workloads,” in *Proc. ICCAD*, 2002, pp. 721–725.
- [5] S. Hua, G. Qu, and S. S. Bhattacharyya, “Energy reduction techniques for multimedia applications with tolerance to deadline misses,” in *Proc. DAC*, 2003, pp. 131–136.
- [6] T. Tia, Z. Deng, M. Shankar, M. Storch, J. Sun, L. Wu, and J. Liu, “Probabilistic performance guarantee for real-time tasks with varying computation times,” in *Proc. RTAS*, 1995, pp. 164–173.
- [7] L.-F. Chao, A. LaPaugh, and E. H.-M. Sha, “Rotation scheduling: A loop pipelining algorithm,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 16, no. 3, pp. 229–239, Mar. 1997.
- [8] R. Jejurikar, C. Pereira, and R. Gupta, “Leakage aware DVS for real-time embedded systems,” in *Proc. DAC*, 2004, pp. 275–280.
- [9] H. Veendrick, “Short-circuit dissipation of static CMOS circuitry and its impact on the design of buffer circuits,” *IEEE J. Solid-State Circuits*, vol. SSC-19, no. 4, pp. 468–473, Aug. 1984.
- [10] C. E. Leiserson and J. B. Saxe, “Retiming synchronous circuitry,” *Algorithmica*, vol. 6, no. 1, pp. 5–35, 1991.
- [11] M. Qiu, Z. Jia, C. Xue, Z. Shao, and E. H.-M. Sha, “Voltage assignment with guaranteed probability satisfying timing constraint for real-time multiprocessor DSP,” *J. VLSI Signal Process. Syst.*, vol. 46, no. 1, pp. 55–73, Jan. 2007.
- [12] T. Burd, “Energy-efficient processor system design,” Ph.D. dissertation, Dept. Elect. Eng. Comput. Sci., Univ. California, Berkeley, May 2000.