Energy Minimization for Real-Time Systems with Non-Convex and Discrete Operation Modes

Foad Dabiri, Alireza Vahdatpour, Miodrag Potkonjak and Majid Sarrafzadeh
Computer Science Department
University of California Los Angeles
email: {dabiri, alireza, miodrag, majid} @ cs.ucla.edu

***Abstract*—We present an optimal methodology for dynamic voltage scheduling problem in the presence of realistic assumption such as leakage-power and intra-task overheads. Our contribution is an optimal algorithm for energy minimization that *concurrently* assumes the presence of (1) *non-convex* energy-speed models as opposed to previously studied convex models, (2) *discrete* set of operational modes (voltages) and (3) intra-task energy and delay *overhead*. We tested our algorithm on MediaBench and task sets used in previous papers. Our simulation results show an average of 22% improvement in energy reduction in comparison with optimal algorithms for convex models without switching overhead and on average of 24% with consideration for energy and delay overheads. This analysis lays the groundwork for improving functionality in CAD design through non-convex techniques for discrete models.**

I. Introduction

Energy consumption is recognized as one of the most important parameters in designing modern portable electronic and wireless systems in today's VLSI circuit design. Among the various low power techniques at different levels of abstraction, dynamic voltage scheduling (DVS) is a widely-used technique for reducing power and energy consumption during system operation. DVS aims at reducing the dynamic/static power consumption by scaling down operational frequency and circuit supply voltage. Traditionally, low power research has focused on a power model where the relationship between power consumption and processor speed is convex. Convexity has a number of profound ramifications when energy is minimized using variable voltage strategies. Several researches have been performed to solve the task-scheduling problem on DVS-enabled systems which focus on dynamic energy reduction and ignore leakage energy [16][7]. In [5][17] energy efficient scheduling of periodic real time tasks in a system with DVS processor and multiple non-DVS devices is explored. Heuristics such as [1] have been proposed for periodic tasks in a multiprocessor.

As device sizes are decreasing due to advances in technological manufacturablity, leakage energy dissipation is becoming more and more important. While for the 70-nm process, leakage power is smaller than dynamic power, for the 50-nm process, they become comparable, while for the 35-nm process, leakage power is larger than dynamic power [9]. Authors in [8] introduce techniques to model subthreshold leakage current at device, circuit, and system levels.

The union of several technological, integrated circuits, architectural, operating systems, and application factors increasingly create systems where the mapping from the speed of execution (that is the inverse of the time required to complete one cycle and execute an operation) and energy consumption per operation (ES) is non-convex.

For example, in heterogeneous multiprocessor multicore system-on-chips, different cores have different ES functions and the overall relationship between processor speed and energy per operation is not convex [8][10]. More importantly, the feature scaling of CMOS devices has been and will increase the relative importance of leakage power. It is often predicted that in less than a decade, leakage power will dominate energy consumption. Total leakage and dynamic energy does not have a convex relationship with processor speed [8]. Energy savings can be accomplished by simultaneously varying the supply voltage (V_{dd}) and the threshold voltage (V_t) through adaptive body biasing [6][14][2]. Figure 1 is selected from [19] which illustrated the non convexity of energy consumption with respect to supply voltage for a chain invertor. In the first

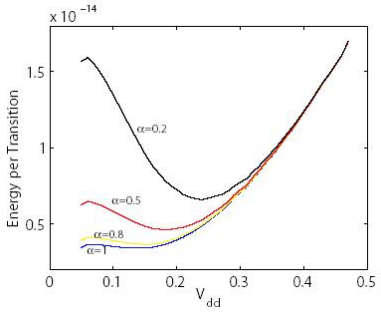

Fig. 1. Non-convexity of energy vs supply voltage. This graph appears in [19]

order approximation lowering the threshold voltage increases the processor speed at the expense of increasing the leakage current. Hence, again the energy-speed dependency is a non-convex function. The incorporation of new high bandwidth on-chip interconnect technologies, such as nanowires, RF, photonic crystals-based optical interconnect, and plasmonics communication networks compounded with a need for thermal management will have as one of ramifications a non-convex ES relationship. The level of instruction level parallelism and effective speed is a highly non-convex and non-continuous

978-3-9810801-5-5/DATE09 © 2009 EDAA

function. Hence, we expect non-convex energy-speed models to dominate the wide spectrum of pending and future energy-sensitive systems.

Scaling the supply voltage, in order to reduce the power consumption, has a side-effect on the circuit delay and hence the operational frequency [3][14]. In addition, transition overhead is another important issue, which often is ignored in previous voltage scaling techniques. Each time the processor's supply voltage is switched, the change requires a certain amount of extra energy and time. Consequently, simplified convex energy models assumed in [18][12] for tackling DVS problem cannot be utilized to achieve the best energy reduction anymore. Authors in [12] present an optimal technique for DVS in the presence of discrete voltage levels for quadratics power models without overhead consideration. Authors in [15] have studied a non-convex method but their approach is not optimal nor polynomial time.

In this paper we study DVS problem where energy-speed is a non-convex function with discrete set of operation modes Furthermore, we include the intra-task delay and energy overhead caused by voltage switching. Although intra-task overheads and discrete operation modes has been studied, our approach is the first one that concurrently addresses them all in non-convex models.

II. Preliminaries

In this section we study models we use for power, energy, delay and switching overheads in CMOS based processors and state the scheduling problem accordingly.

A. Power and Energy Model

Power dissipation of digital CMOS based processors is composed of I) dynamic power which is dissipated whenever the processor is active, and II) static power which is consumed regardless of the activity and as long as the device is on as shown in Equation 1.

$$P = P_{dynamic} + P_{leakage} \tag{1}$$

Dynamic power consumption during execution of a given task corresponds to the power dissipated in charging and recharging internal capacitors in every gate, given by Equation 2:

$$P_{dynamic} = C \times \Phi V_{DD}^2 \tag{2}$$

where C is the average switching capacitance per clock cycle for a given task. V_{DD} is the power supply voltage and Φ represents technology dependant factors. Leakage power is modeled according to Equation 3:

$$P_{leakage} = V_{DD}.I_{sub} \tag{3}$$

where I_{sub} is the subthreshold current which according to [4] can be stated as:

$$I_{sub} = A.e^{\frac{v_{gs}-v_{th_0}-\gamma v_s+\eta v_{ds}}{nkT/q}} \times [1 - e^{\frac{-v_{ds}}{kT/q}}] \tag{4}$$

This equation can be re-written as:

$$I_{sub} = K_1 e^{K_2 V_{DD}} e^{K_3 V_{bs}} = \Psi e^{K_2 V_{DD}} \tag{5}$$

where K_is are constant fitting parameters and V_{bs} is the bias voltage [14] and finally Ψ represents the constants in Equation 5. The total power consumption as a function of supply voltage can be stated as:

$$P = \Phi V_{DD}^2 + \Psi V_{DD} e^{K_2 V_{DD}} \tag{6}$$

Assuming R is the number of clock cycles required for a given task and f_{clock} is the clock frequency for the corresponding given supply voltage, total energy consumption during the execution of a task is:

$$E = R \times (C \times \Phi V_{DD}^2 + \Psi V_{DD} e^{K_2 V_{DD}}) f_{clock} \tag{7}$$

In this paper, we use energy per clock cycle ($\frac{E}{R}$) vs speed curves in our optimization process.

B. Speed Model

The delay of CMOS based processors can be stated as:

$$d_i = K_i \times \frac{V_{DD}}{(V_{DD} - V_t)^\alpha} \tag{8}$$

where K_i and α are technology dependent parameters for the i^{th} unit. Discarding V_t in the above equation is no longer a reasonable assumption since with the current low voltage technologies, V_{DD} and V_t are in the same order of magnitude.

C. Problem Statement

The problem of dynamic voltage scheduling can be stated as follows:

We are given a set of tasks $J = \{\tau_1, ..., \tau_n\}$ where each task is represented by a tuple $\tau_i = (a_i, b_i, R_i, C_i)$:

- a_i is the arrival time of task i
- b_i is the deadline of task i
- R_i is the required clock cycles to process task i
- C_i is the average switching capacity of task i

Also, we assume that the target processor operates under a finite set of modes [1], $\Xi = \{m_1, ..., m_k\}$, where each mode is a pair $m_j = (e_j, s_j)$:

- e_i is the energy consumption per clock cycle in the j^{th} mode
- s_i is the processor speed when running in the j^{th} mode

The objective of this scheduling problem is to find mapping functions $\xi(t)$ and $\chi(t)$ which define the processor mode and the scheduled task at time t such that the total energy consumption during the $[0, T]$ period is minimized and all task meet their deadlines. The objective can be stated as:

$$minimze \int_0^T P(\xi(t))dt = \sum_0^R e(s_i)\Delta R \tag{9}$$

We make no assumptions about power and speed relation. In other words, we are assuming a non-convex model and therefore *any* (e_j, s_j) can be used in our algorithm.

[1] In this paper we use the term 'operation mode' rather than 'supply voltage' or 'speed'. The reason being the fact that current technologies can reduce energy dissipation not by only changing supply voltage but with also reducing bias-voltage etc.

D. Switching Overhead

An important overhead caused by dynamic scheduling techniques is delay and energy overheads. Specially when the operation mode of a processor is changing while a task is under execution, this overhead becomes more significant because of the delay and energy dissipations resulting from memory access and recovery. Therefore, for each pair of operation modes, there are two overhead measures associated with: ϵ_{ij} and δ_{ij} where ϵ_{ij} is the energy overhead when switching from mode i to j. δ_{ij} is the delay caused by this operation in which the scheduled task becomes idle. When the mode switching is a result of supply (V_{dd}) and body-bias voltage (V_{bs}) change, the delay and energy overheads can be stated as [2][14]

$$\epsilon_{ij} = C_r|V_{dd_i} - V_{dd_j}|^2 + C_s|V_{bs_i} - V_{bs_i}|^2 \quad (10)$$

$$\delta_{ij} = max(p_{V_{dd}}|V_{dd_i} - V_{dd_j}|, p_{V_{bs}}|V_{bs_i} - V_{bs_i}|) \quad (11)$$

where C_r represents power rail capacitance, and C_s is the total substrate and well capacitance. Since transition times for V_{dd} and V_{bs} are different, the two constants $p_{V_{dd}}$ and $p_{V_{bs}}$ are used to calculate both time overheads independently. Finally, if there exists any other overhead in state switching, additional terms can be added to equations 10 and 11.

III. Optimal Dynamic Speed Selection Algorithm

In this section we cover the preprocessing phases of our method along with the optimal scheduling algorithm. Throughout different steps of our developed algorithm, we use similar definitions and terms as used in [18].

Definition Intensity of the interval $[a, b]$ is defined to be:

$$g(I) = \frac{\sum R_i}{b-a} \ \forall i : [a_i, b_i] \subseteq I \quad (12)$$

which is the average speed required to execute all the tasks which are completely dominated by I.

[2] $g(I)$ is the lower bound on the average speed in the interval I. In order to find an optimal scheduling, we need to set the processor such that in runs at the average speed no less that $g(I)$ with an energy consumption which would yield to an optimal solution.

Our approach to solve this scheduling problem is to make a virtual surjective mapping of speed to energy in the $[0, s_{max}]$ without compromising (s_{max} is the maximum possible speed that the processor can run at). Using such a curve we can virtually run at any speed.

A. Virtually Continuous Speed Operation

The DVS problem where the target processor runs at discrete speeds is well known to be NP-hard in general. In order to show how a processor with discrete operation speed can virtually run at any speed, we give a simple example. Assume our processor has two operation modes as shown in Figure 2. In order to run the processor at speed s^* ($s_1 \leq s^* \leq s_2$) for a given interval $[a, b]$, we run the processor at s_1 for t_1 seconds and s_2 for t_2 seconds where for t_1 and t_2 we have:

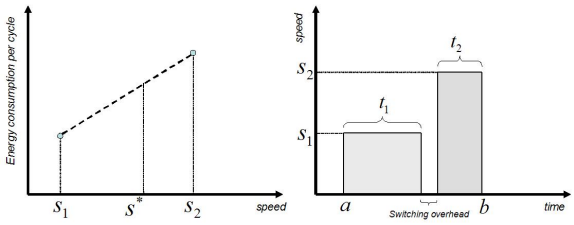

Fig. 2. A processor with discrete speeds can virtually run at any speed by partially running at different pre-set speeds

$$t_1 = \frac{s_2 - s^*}{s_2 - s_1} \times (b-a) \quad (13)$$

$$t_2 = \frac{s^* - s_1}{s_2 - s_1} \times (b-a) \quad (14)$$

The average speed using these two speeds is equal to s^*. Therefore, we have been able to run the the processor at the virtual speed s^*. Now, considering the intra-task delay overhead caused by switching processor operation, the target speed would be $s^*_\delta = \frac{b-a}{b-a-\delta_{ij}} s^*$ and we can re-drive t_1 and t_2 as follows:

$$\begin{aligned} &\text{if } s^*_\delta = \tfrac{b-a}{b-a-\delta_{ij}} s^* < s_2 \\ &t_1 = \frac{s_2 - s^*_\delta}{s_2 - s_1} \times (b-a) \\ &t_2 = \frac{s^*_\delta - s_1}{s_2 - s_1} \times (b-a) \quad (15) \\ &\text{else if } s^*_\delta \geq s_2 \\ &t_1 = 0 \\ &t_2 = \frac{s^*}{s_2} \times (b-a) \quad (16) \end{aligned}$$

Equation 16 shows the case where due to switching delay overhead, the virtual speed is larger than s_2 and therefore the processor only runs at s_2 for the portion of time.

B. Generation of the Optimal Surjective Energy-Speed Curve

Phase I: Continuous Convex Curve Fit. Given a set of operation modes, the first step of our methodology is to find a lower convex curve on the energy-speed points as illustrated in Figure 3 and extend the curve on lower-left side to cover all the speed axis.

Let the points in this curve be $\mathcal{M} = \{(e'_1, s'_1), ..., (e'_q, s'_q)\}$ sorted in non-decreasing order with respect to s'_is. The resulting energy-speed curve can be stated as:

$$\mathcal{E}_\rfloor(s) = \begin{cases} s'_1 & \text{if } 0 < s \leq s'_1 \\ \frac{e'_i - e'_{i-1}}{s'_i - s'_{i-1}}(s - s'_{i-1}) + e'_{i-1} & s'_{i-1} < s < s'_i \ \forall 1 < i \leq q \end{cases} \quad (17)$$

Phase II: Energy Overhead Insertion. In the second phase of our algorithm, we will include the intra-task energy overhead caused by mode switching. Each line segment

[2]This is for uniform switching capacity, for non-uniform, $g(I)$ should be modified accordingly

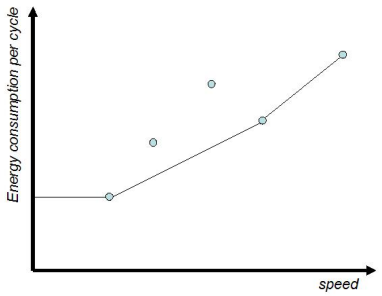

Fig. 3. Continuous Convex Curve Fit: A lower bound convex curve for energy vs speed

$[(e_i, s_i), (e_j, s_j)]$ in the $\mathcal{E}_\rfloor(s)$ is replaced by $[(e_i, s_i), (e_j, s_j)+\epsilon_{ij}]$ where ϵ_{ij} is the normalized switching overhead, we call this function $\mathcal{E}'_\rfloor(s)$. Finally we find a monotone curve fit of $\mathcal{E}'_\rfloor(s)$ as follows:

$$\mathcal{E}_\mathcal{V}(s) = \begin{cases} e'_1 & \text{if } 0 < s \leq s'_1 \\ min(\mathcal{E}'_\rfloor(s), e'_i) & s'_{i-1} < s < s'_i \ \forall 1 < i \leq q \end{cases} \quad (18)$$

Figure 4 illustrates phase II and the final curve is drawn in bold line.

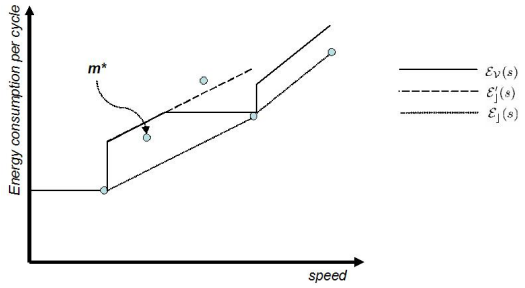

Fig. 4. Result of preprocessing phases: Final energy vs. speed curve: $\mathcal{E}_\mathcal{V}(s)$

Phase III:Refinement Process. $\mathcal{E}_\mathcal{V}(s)$ can potentially introduce new points in the energy-speed domain which where disregarded as a result of phase I. For example, m' is a point where as a result of energy overhead consideration, it can be used to achieve better performance. As the last step of energy-speed curve generation, we repeat phase II with $\mathcal{M} = \mathcal{M} \bigcup \{m^*\}$ (considering new points such as m^*). In the rest of the paper, we will refer to the final virtual energy-speed curve resulted from phase III as $\mathcal{E}_\mathcal{V}(s)$.

C. Proposed Energy Optimization Algorithm

In this section we present our scheduling algorithm.

Definition Interval I^* is said to be critical if $g(I^*)$ is maximum for all feasible intervals in $[0, T]$. R^* is the number of clock cycles required to execute all tasks that lie inside I^*. It is easy to verify that $I^* = [a_i, b_j]$ for some tasks i and j.

Lemma 3.1: Let $s^* = g(I^*) = \frac{R^*}{(b_j, a_i)}$. Assume that in order to achieve minimum energy consumption during I^*, task(s) in I^* is (are) run at $S = \{s^*_1, ..., s^*_p\}$ for the time percentages of $(\alpha_1, ..., \alpha_p)$ respectively $(\sum \alpha_i = 100\%)$. Then the operation modes are consecutive in $\mathcal{E}_\mathcal{V}(s)$.

Proof: Let the duration of $|I^*| = T$ and assume $s_q \in S$ where $s^*_i < s_q < s^*_j$ and $(j-i)$ is minimum. In other words s_q lies between two operation speeds in S and all the intermediate speeds are not in S. Now, we will show that adding s_q to S can decrease the total energy consumption during I. Since $s^*_i < s_q < s^*_j$, there exist $\beta and \gamma$ such that:

$$s_q = \beta s^*_i + \gamma s^*_j \quad (19)$$

Without the loss of generality, assume $\alpha_i \leq \alpha_j$. Therefore, the normalized energy consumption (i.e. per clock cycle) during s^*_i and s^*_j can be stated as:

$$E_{i,j} = \alpha_i e^*_i + \alpha_j e^*_j = \alpha_i e^*_i + \frac{\gamma\alpha_i}{\beta} e^*_j + \frac{\alpha_j\beta - \gamma\alpha_i}{\beta} e^*_j + \epsilon_{ij} \geq \alpha_i e_q + \frac{\alpha_j\beta - \gamma\alpha_i}{\beta} e^*_j + \epsilon_{qj} \quad (20)$$

Which means that if the processor is run at s_q it can reduces the energy consumption during virtual speed of s_q when the processor is sun at s^*_i and s^*_j. Therefore we conclude that S contains consecutive operation modes in $\mathcal{E}_\mathcal{V}(s)$. ■

Lemma 3.2: Let $s^* = g(I^*)$. Assume minimum energy consumption during I^*, requires running the task in I^* at consecutive speeds: $S = \{s^*_1, ..., s^*_p\}$ for the time percentages $(\alpha_1, ..., \alpha_p)$ respectively then $p \leq 2$.

Proof: The proof is very similar to the proof of Lemma 3.1. Assume $s^*_i < s_q < s^*_{i+1}$. For all $s^*_j < s^*_i$ we can follow the proof in Lemma 3.1 and eliminate operation at s^*_j by increasing α_i and α_{i+1} accordingly and reduce total power consumption during I. ■

Theorem 3.3: In the optimal scheduling, the critical interval I^* is run at virtual mode of $e(g(I^*)), g(I^*)$.

Proof: If $\mathcal{E}_\mathcal{V}(s)$ was a convex curve the optimality would be followed [18]. Although $\mathcal{E}_\mathcal{V}(s)$ is not convex in principle but we show that it has the convexity property. The reason why $\mathcal{E}_\mathcal{V}(s)$ is not visually convex is the fact that $\mathcal{E}_\mathcal{V}(s)$ is a conditional graph. In other words $\mathcal{E}_\mathcal{V}(s)$ is minimum energy consumption at different speeds when each speed is virtually achieved through one or two operation modes.

$$\mathcal{E}_\mathcal{V}(\alpha s_1 + (1-\alpha)s_2) \leq \alpha\mathcal{E}_\mathcal{V}(s_1) + (1-\alpha)\mathcal{E}_\mathcal{V}(s_2) \quad (21)$$

The important point here is the fact that, the right hand side of the Equation 21 is missing the energy overhead term. The interpretation of Equation 21 is that energy consumption at speed $\alpha s_1 + (1-\alpha)s_2$ is less the weighted energy consumption if the processor is running at αs_1 and $(1-\alpha)s_2$. Therefore we need to take into account the overhead energy ϵ_{12} as well. According to Equation 18 in Phase II:

$$\begin{aligned} &\mathcal{E}_\mathcal{V}(\alpha s_1 + (1-\alpha)s_2) = \\ &\quad min(\alpha\mathcal{E}_\mathcal{V}(s_1) + (1-\alpha)\mathcal{E}_\mathcal{V}(s_2) + \epsilon_{12}, \mathcal{E}_\mathcal{V}(s_2)) \\ &\leq \quad \alpha\mathcal{E}_\mathcal{V}(s_1) + (1-\alpha)\mathcal{E}_\mathcal{V}(s_2) + \epsilon_{12} \end{aligned} \quad (22)$$

Equation 22 proves the correctness of our claim and the convexity of $\mathcal{E}_{\mathcal{V}}$ in use. ■

Finally, according to Theorem 3.3 the recursive algorithm in 1 yields the optimal solution.

Algorithm 1 Optimal Dynamic Voltage Scheduling Algorithm

1: Find critical interval $I^* = [a_i, b_j]$, assume $s_i \geq g(I^*) \leq s_{i+1}$.
2: Order the tasks in I^* according to the earliest deadline first (EDF).
3: If $g(I^*)$ results in using Equation 16 or $e(g(I^*)) = e_{i+1}$, execute each task in I^* at (e_{i+1}, s_{i+1}) else if $g(I^*)$ results in using Equation 15 run each task at virtual mode of $(e(g(I^*)), g(I^*))$.
4: Remove I^* and modify arrival and deadline of remaining task accordingly
5: Recursively repeat the above procedure.

1) Time Complexity: The preprocessing steps of the algorithm which include the three phases have polynomial running time. Convex curve fit in Phase I can be done in $O(nlogn)$ using Skiena's convex hull algorithms where n is the number of operation modes. Phase II is a linear procedure which takes $O(n)$ and therefore, the worst-case runtime of Phase III where only one point is added to $\mathcal{M}$ can be done in $O(n^2logn)$. The straight forward implementation of the scheduling algorithm itself requires $O(n^2)$ and therefore our proposed methodology can be implemented with $O(n^2logn)$ complexity.

IV. Simulation Results

We implemented our algorithm in Matlab and tested its effectiveness using both randomly generated task sets and real-life benchmarks. Our methodology results in optimal energy reduction and simulations are carried out to compare the effectiveness of our algorithms. We compared our simulation with Kim's optimal voltage allocation technique [11] which is an optimal methodology for DVS enabled processors with discrete set of voltages. Their method assumes quadratic power model and therefore convex energy vs speed curve. Furthermore, in Kim's algorithms, they don't take into account energy and delay overheads. Therefore, first we compared our results with their assuming no overhead so that we could illustrate the effects of non-convexity in energy-speed curves.

The randomly generated tasks are selected from [11] and we added four more task sets with the same characteristics. As for the real-life benchmarks, we applied our optimization techniques on the processes from MediaBench test cases [13].

The statistical behavior of tasks determine the effectiveness of energy managements techniques not justin addition to the number of task. Task set can easily be scaled up or down without effecting the performance. Table I summarizes the characteristics of the benchmarks. First column is the task number where $J1, J2, J3$ and $J4$ are random task sets from [11] and $J5, J6, J7$ and $J8$ are generated in a similar fashion. Later rows are for 6 different benchmarks from Media bench.

TABLE I
Statistical characteristics of the task set: First two columns show the mead and std of average rate of all tasks where the third and forth column are the mean and std of intensity over time

	Per Task		Over Time		# of tasks
	Mean	Std	Mean	Std	
J1	160.5	40.9	393.3	198.1	10
J2	147.1	39.8	558.3	330.8	15
J3	111.7	24.5	556.2	293.0	20
J4	93.3	19.2	619.1	341.2	30
J5	101.7	42.1	395.9	288.4	35
J6	136.2	58.3	442.3	210.0	40
J7	181.5	38.1	588.7	171.7	45
J8	174.6	126.6	238.5	272.8	50
JPEG	221.2	95.5	281.9	242.3	12
MPEG	110.3	48.6	344.2	113.2	14
GSM	189.8	99.1	427.7	341.7	11
G721	174.6	126.6	238.5	272.8	15
PGP	91.5	79.5	82.4	148.9	14
EPIC	138.6	98.7	513.9	284.6	17

TABLE II
Energy reduction comparison with respect to Kim's algorithm and single speed processor for P1 and P2

	P1: 2 modes			P2: 3 modes			
Task	No O.H.	w/ Overhead		No Overhead		w/ Overhead	
Set	SS	Kim	SS	Kim	SS	Kim	SS
J1	28.6	13.4	25.6	26.6	54.4	30.1	50.2
J2	11.7	6.4	9.5	5.1	30.1	9.3	20.8
J3	14	4.5	13.3	3	42.7	5.5	31.4
J4	7.8	5.2	7	3.8	18.2	6	9.3
J5	12.8	9.7	8.5	14.4	34.6	18.1	30.1
J6	23.5	12	17.3	21.9	48.6	25.5	45.4
J7	19.3	7.8	15.3	9.5	52.3	12.3	48.7
J8	17.7	13.9	17.1	13.9	26.4	17.3	22.1
JPEG	31.9	28	27.4	33.1	57.2	33.1	52.5
MPEG	34.9	23.4	26.9	38.1	35.2	38.1	31.1
GSM	17.2	6.3	15.3	6.7	48.9	5.9	40.9
G721	26.7	18	22.2	22	44.2	22	39.4
PGP	28.32	32.9	21.9	44.4	44.4	44.4	44.4
EPIC	13.8	27.4	12.1	22.6	46.4	28.7	33.2
Mean	19.9	14.9	16.4	18.3	40.7	20.4	34.5

The second and third column represent the mean and standard deviation of the average speed ($\frac{R_i}{b_i - a_i}$) per task where the third and forth column represent the mean and standard deviation of intensity per time unit over the duration of the whole task set.

We considered four different processors in term of number of operation modes. The speed ranges from $300Mz$ to $700Mz$ and modes are equidistantly separated for different cases. Number of modes are $2, 3, 5$ and 13 for $P1, P2, P3$ and $P4$ respectively. Our simulation results are carried out to illustrated two aspects: First, to show the improvement of our algorithm in comparison to Kim's method and second to show the effectiveness of our algorithm in energy reduction in comparison with single speed processors. Tables II and III summarize the simulation results. Tables are divided into two separate columns representing different processors. For each processor, we have included four different results: First column shows the percentage of energy reduction in percentage in comparison with Kim's algorithms without considering any overhead. Second column shows power reduction percentage in comparison with a single speed processor (SS) where task are executed at the minimum speed such that all deadlines are

TABLE III
ENERGY REDUCTION COMPARISON WITH RESPECT TO KIM'S ALGORITHM AND SINGLE SPEED PROCESSOR FOR P3 AND P4

	P3: 5 modes				P4: 13 modes			
Task	No Overhead		w/ Overhead		No Overhead		w/ Overhead	
Set	Kim	SS	Kim	SS	Kim	SS	Kim	SS
J1	26.6	57.5	25.1	52.4	43.3	61.4	55.2	58.4
J2	3.4	33.2	13.7	22.5	16.6	34.1	28.2	25.4
J3	9	12.2	22	9.8	20.1	25.2	24.2	9.9
J4	2.3	20.8	16	10.5	16.2	18.9	33.7	10.5
J5	10.1	39.5	10.9	32.4	14.2	39.5	16.3	32.4
J6	22.2	53.4	25.4	50.2	33.7	61.5	38.7	57.9
J7	7.3	57.3	11.4	49.2	12.5	65.3	17.4	60.2
J8	18.3	35.4	23.8	28.9	13.2	44.5	21	41.3
JPEG	37.3	61	36.1	58.9	46	30.9	45.8	26.93
MPEG	28.2	29.3	28.2	23.9	50.2	44.6	50.5	44.62
GSM	12.3	21.8	20.8	17.4	24.3	34.8	27.3	22.97
G721	40.9	53.6	39.4	50.6	29.3	13	35.2	8.88
PGP	41.9	41.9	41.9	41.9	57.7	57.7	57.7	57.79
EPIC	20	11.6	29	6.2	31.6	30.8	30.7	14.56
Mean	19.4	36.2	24.5	30.9	28.1	38.5	32.8	31.7

met. Third and forth columns are similar scenarios but with consideration for intra-task energy and delay overhead. Since Kim's algorithm does not consider any overhead, we modified their method to include delay overhead in order to make more fair comparison.

Since $P1$ has only two operation modes, our results without overhead consideration are identical to Kim's because the energy-speed domain is convex for any two point. Therefore we have removed the corresponding column. Our simulation results show an average of 22% improvement in energy reduction in comparison with Kim's algorithms without switching overhead and on average of 24% with consideration for intra-task energy and delay overheads. An important observation is that energy reduction improvements with respect to Kim's algorithm can vary from 3% to 58%. The reason lies in the task characteristics and the speed values in the optimal scheduling. If the optimal speed is near an operation mode in $\mathcal{M}$, both techniques are close together. However if the optimal speeds vary from points in $\mathcal{M}$, our technique performs much better.

V. CONCLUSION

In this paper we present a polynomial time optimal methodology for dynamic voltage scheduling problem for energy minimization in the presence of realistic assumption. We have developed an optimal methodology for energy minimization in the presence of *non-convex* energy-speed models as opposed to previously studied convex models. Our methodology results in minimum energy consumption for arbitrary energy vs speed models and therefore can be utilized to consider dynamic and static energy dissipation. Our developed algorithm is designed for processors with *discrete* set of operational modes and we make no continuity assumption on supply voltage and prove the optimality of the result. Further more, we extend our results to include intra-task energy and delay *overhead* caused by the scheduling algorithms. We simulated our algorithm on randomly generated task sets from previous papers as well as real-life MediaBench benchmarks for different processor scenarios. Our simulation results show an average of 22% improvement in energy reduction without intra-task switching overheads and an average of 24% with consideration for overheads in comparison with the optimal algorithms for convex models .

REFERENCES

[1] T. A. AlEnawy and H. Aydin. Energy-aware task allocation for rate monotonic scheduling. In *RTAS '05: Proceedings of the 11th IEEE*, pages 213–223, Washington, DC, USA, 2005. IEEE Computer Society.

[2] A. Andrei, M. Schmitz, P. Eles, Z. Peng, and B. M. Al-Hashimi. Overhead-conscious voltage selection for dynamic and leakage energy reduction of time-constrained systems. In *DATE '04: Proceedings of the conference on Design, automation and test in Europe*, pages 28–38, Washington, DC, USA, 2004. IEEE Computer Society.

[3] A. P. Chandrakasan and R. W. Brodersen. *Low Power Digital CMOS Design*. Kluwer Academic Publishers, Norwell, MA, USA, 1995.

[4] F. Farbiz, M. Farazian, M. Emadi, and K. Sadeghi. Sizing consideration for leakage control transistor. In *VLSID '04: Proceedings of the 17th*, page 639, Washington, DC, USA, 2004. IEEE Computer Society.

[5] C.-M. Hung, J.-J. Chen, and T.-W. Kuo. Energy-efficient real-time task scheduling for a dvs system with a non-dvs processing element. In *RTSS '06: Proceedings of the 27th IEEE International Real-Time Systems Symposium*, pages 303–312, Washington, DC, USA, 2006. IEEE Computer Society.

[6] T. Ishihara and H. Yasuura. Voltage scheduling problem for dynamically variable voltage processors. In *ISLPED '98: Proceedings of the 1998 international symposium on Low power electronics and design*, pages 197–202, New York, NY, USA, 1998. ACM.

[7] R. Jejurikar and R. Gupta. Energy aware task scheduling with task synchronization for embedded real time systems. In *CASES '02: Proceedings of the 2002 international conference on Compilers, architecture, and synthesis for embedded systems*, pages 164–169, New York, NY, USA, 2002. ACM.

[8] J. Kao, S. Narendra, and A. Chandrakasan. Subthreshold leakage modeling and reduction techniques. In *ICCAD '02: Proceedings of*, pages 141–148. ACM, 2002.

[9] H.-S. Kim. Impact of scaling on the effectiveness of dynamic power reduction schemes. In *ICCD '02: Proceedings of the IEEE Int. Conf. on Computer Design: VLSI in Computers and Processors*, page 382. IEEE Computer Society, 2002.

[10] D. J. N. Kumar, R. Tullsen and P. Ranganathan. Heterogeneous chip multiprocessors. *Computer*, 38(11):32–38, 2005.

[11] W.-C. Kwon and T. Kim. Optimal voltage allocation techniques for dynamically variable voltage processors. pages 125–130, 2003.

[12] W.-C. Kwon and T. Kim. Optimal voltage allocation techniques for dynamically variable voltage processors. *Trans. on Embedded Computing Sys.*, 4(1):211–230, 2005.

[13] C. Lee, M. Potkonjak, and W. H. Mangione-Smith. Mediabench: a tool for evaluating and synthesizing multimedia and communicatons systems. In *MICRO: Proceedings of the 30th*, pages 330–335. IEEE Computer Society, 1997.

[14] S. M. Martin, K. Flautner, T. Mudge, and D. Blaauw. Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads. In *ICCAD '02: Proceedings of the*, pages 721–725. ACM, 2002.

[15] A. Nahapetian, F. Dabiri, M. Potkonjak, and M. Sarrafzadeh. Optimization for real-time systems with non-convex power versus speed models. In *Integrated Circuit and System Design: PATMOS*, pages 443–452, 2007.

[16] P. Rong and M. Pedram. Power-aware scheduling and dynamic voltage setting for tasks running on a hard real-time system. In *ASP-DAC '06: Proceedings of the 2006 conference on Asia South Pacific design automation*, pages 473–478, Piscataway, NJ, USA, 2006. IEEE Press.

[17] T. Wei, P. Mishra, K. Wu, and H. Liang. Online task-scheduling for fault-tolerant low-energy real-time systems. In *ICCAD '06: Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design*, pages 522–527, New York, NY, USA, 2006. ACM.

[18] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced cpu energy. In *FOCS '95: Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS'95)*, page 374. IEEE Computer Society, 1995.

[19] B. Zhai, D. Blaauw, D. Sylvester, and K. Flautner. Theoretical and practical limits of dynamic voltage scaling. In *DAC '04: Proceedings of the 41st annual conference on Design automation*, pages 868–873, New York, NY, USA, 2004. ACM.