2012 IEEE 14th International Conference on High Performance Computing and Communications

Towards Power-Efficient Smartphones by Energy-Aware Dynamic Task Scheduling

Meikang Qiu[1,*], Zhi Chen[1], Laurence T. Yang[2], Xiao Qin[3], Bin Wang[4]
[1] Dept. of Electrical and Computer Engineering, University of Kentucky, Lexington, KY 40506, USA
[2] Dept. of Computer Science, St. Francis Xavier University, Antigonish, NS, B2G 2W5, Canada
[3] Dept. of Computer Science and Software Engineering, Auburn University, Auburn, AL 36849, USA
[4] Dept. of Computer Science and Engineering, Wright State University, Dayton, OH 45435, USA

***Abstract*—Smartphones are facing a grand challenge in extending their battery life to sustain an increasing level of processing demand while subject to miniaturized form factor. *Dynamic Voltage Scaling* (DVS) has emerged as a critical technique to leverage power management by lowering the supply voltage and frequency of processors. In this paper, based on the DVS technique, we propose a novel *Energy-aware Dynamic Task Scheduling* (EDTS) algorithm to minimize total energy consumption for smartphones while satisfying stringent time constraints for applications. This algorithm utilizes the results from a static scheduling algorithm and aggressively reduces energy consumptions on the fly. Experimental results indicate that the EDTS algorithm can significantly reduce energy consumption for smartphones, compared to the critical path scheduling method and the parallelism-based scheduling algorithm.**

***Keywords*-Smartphone, DVS, dynamic scheduling, critical path, real-time**

I. Introduction

With the unprecedented popularity of battery-powered smartphones in recent years, modern computation, communication, and entertainment are increasingly moving onto these devices. Meanwhile, the ever increasing demands in rich interactive applications, such as digital cameras, multimedia, GPS navigation units, and web browsers, have severely aggravated energy consumption problem for smartphones. Miniaturization is another vital feature of current smartphones. While the deep sub-micron to nanometer fabrication technique is an enabler to reduce the size of smartphones, this technology also exacerbates the energy consumption problem.

In order to enhance energy efficiency and process various tasks with different performances requirements, high-end smartphones are designed as heterogeneous systems, which integrate multiple processors with distinct processing power, such as PowerVR SGX 5XT from Imagination [1]. Although multiprocessors can offer greater computation per unit of power, leading to longer battery life [2], it is critical to investigate tighter energy budget strategies to guarantee functionalities of smartphones.

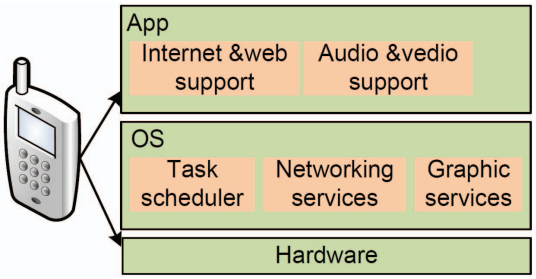


Figure 1: A typical architecture for smartphones.

Most applications on smartphones are not delay-tolerant, and acceleration of them is often at higher expenses of energy consumption. In order to balance performance and power consumption for these applications, smartphones are usually designed with *Dynamic voltage scaling* (DVS) by integrating static CMOS logic into microprocessors [3]–[5]. DVS is a powerful technique to reduce energy consumption, and widely employed in various embedded systems. With the aid of this technology, different performance levels for applications can be achieved by adjusting the operating frequency of processors. By introducing model-set instructions, the voltage supply can be switched between several voltage modes, which makes it possible to implement DVS by software. Typically, DVS can be exclusively implemented in existing *Real-time Operating Systems* (RTOSs).

In this paper, we focus on optimizing energy consumption for heterogeneous smartphones while satisfying time-

∗ Meikang Qiu is the corresponding author. mqiu@engr.uky.edu

978-0-7695-4749-7/12 $26.00 © 2012 IEEE
DOI 10.1109/HPCC.2012.214

constraints of applications.We propose an *Energy-aware Dynamic Task Scheduling* (EDTS) to examine task communications online and minimize total energy consumption based on DVS techniques. For example, Figure 1 shows the basic architecture for smartphones, we implement our techniques in the OS level. This algorithm utilizes the results from a static scheduling algorithm and attempts to aggressively reduce energy consumption. Experimental results show that compared to the critical path scheduling method and the parallelism-based scheduling approach, our online scheduling mechanism can reduce total energy consumption by 23.1% and 34.2% on average, respectively, while meeting the given timing constraints.

The major contributions of this paper are four-fold: (1) we propose a dynamic scheduling algorithm for dealing with the runtime variations; (2) we use a critical path based static scheduling algorithm *Data Flow Graph Critical Path* (DFGCP) to obtain near-optimal solutions; (3) we propose an optimal algorithm *Critical Path Assignment* (CPA) for the critical path with a dynamic programming approach; (4) we consider both overheads for the voltage level transfer and the communication cost between different tasks, based on DVS power models.

II. Related Work

In the past few years, numerous methodologies for low-power smartphone system design have been proposed at operating system level [2], [6] as well as architecture level [7]. A scheduler was proposed in [2] to monitor workloads for systems and adaptively schedule real-time tasks while considering the worst-case CPU demands. Through modification of the real-time scheduler and task management services in operating systems, this scheduler can boost system performance and save power consumption for heavy workloads and critical tasks. Targeting multimedia applications, the authors in [6] proposed a soft real-time CPU scheduler for mobile devices to reduce energy consumption. While these studies focus on independent tasks, we consider dependencies and real-time constraints between tasks.

A myriads of endeavors have been put forward in tackling runtime variations with DVS. For example, Gruian [8] applied a stochastic DVS technique on hard real-time systems by taking into account task dependencies. Depending on the probability distribution of the execution conditions for tasks, Lorch and Smith [9] proposed an approach to modify scaling algorithms while maintaining their performance. However, these methods assume task priorities and estimate CPU requirements off-line. We propose a two-phase scheduling algorithm, EDTS, which schedules tasks online based on the static scheduling results of an initial scheduling.

Energy-aware static scheduling is usually based on the information of the average case or worst case task execution estimation [10]. At runtime, the real execution time and energy consumption may exhibit high variations [11], [12], due to process variability, physical faults, and voltage/frequency changes. In our model, we expect each core in the same processor can adjust its voltage and frequency independently.

In [13], the authors jointly presented a host of runtime and compilation techniques to conceal the heterogeneity of smartphones from developers. By investigating various features of HTC and Apple, Li and Ortiz, et al. [14] pointed out that the most significant challenge of reuse in smartphones is the design of software to accommodate heterogeneity of these devices. However, our work focuses on using a dynamic programming task scheduling technique to reduce energy consumption for smartphones with DVS enabled.

III. Basic Concepts and Models

In this section, we introduce basic concepts that will be used in later sections.

A. Data Flow Graph (DFG)

In general, the tasks in smartphone applications are not stand-alone. A certain number of tasks will have precedence relationships due to different functionality of each task and communications between them. We use a *Directed Acyclic Graph* (DAG) to model the precedence constraints of smartphone applications.

Definition 3.1: *Data Flow Graph (DFG):* A DFG G = $\langle U, ED, T, E, W\rangle$ is a node-weighted DAG, where $U = \langle u_1, \cdots, u_i, \cdots, u_N\rangle$ is a set of task nodes; $ED \subseteq U \times U$ is an edge set that defines the precedence relations among nodes in U. For example, an edge $e(u \rightarrow v)$ in the graph indicates that task v cannot be executed until task u completes. T and E are sets of execution time and energy consumption for all nodes in U, respectively. W is a set of communication cost between tasks.

The execution time T of a task can be profiled by *average case execution time* (ACET) or *worst case execution time* (WCET) when the task is executed on a processor core. We assume that the WCET and ACET of a task are always measured at the highest voltage level (i.e., with fastest speed). Our approach uses ACET for the static

scheduling. An edge $e \in ED$ is associated with a weight w that represents the worst-case communication cost between two dependent tasks when they are scheduled on two different processors. Generally, the communication cost between two tasks is negligible when they are executed on the same processor. There is a timing constraint TC for the whole task graph, which defines the time bound to finish execution the entire task graph.

B. Energy Model

The dynamic power consumption (P_{AC}) of CMOS circuits integrated in smartphones is calculated by,

$$P_{AC} = C_{eff} V_{dd}^2 f \tag{1}$$

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

The frequency f is represented as in Equation (2).

$$f = \frac{(V_{dd} - V_{th})^{\alpha}}{kV_{dd}} \tag{2}$$

where V_{th} represents the threshold voltage, and k is a device-dependent constant. α is a technology-dependent constant, which varies between 1 and 2.

IV. An Motivational Example

Figure 2(a) shows a simple application on a smartphone with 2 different voltage levels, namely V_1 and V_2. This application includes 3 tasks, and the execution time and energy consumption of each voltage mode are shown in Figure 2(b). Our objective is to schedule all tasks in the graph with the minimum energy consumption while satisfying a given time constraint.

Based on Figure 2(a), the *critical path* (CP) of the task graph is $u_1 \rightarrow u_2 \rightarrow u_3$. Assuming the *timing constraint* (TC) of the smartphone application is 9 time units. Figure 2(c) illustrates the procedure to achieve the minimal energy consumption by our proposed static scheduling algorithm. The voltage level assignment for each task is recorded in a two-dimensional matrix $E_i[j]_k$ (i represents a task, j represents a time period, and k represents the voltage mode assignment to task i, respectively). From Figure 2(c), we can see that the minimum energy consumption, 46, is achieved, assigning the voltage mode $V_2 \rightarrow u_1, V_1 \rightarrow u_2$, and $V_2 \rightarrow u_3$, respectively.

From the result of $E_3[9]$ we can obtain the assignment as follows: (1) Starting from the minimum energy consumption at $E_3[9]$, we know V_2 is assigned to u_3 and its execution time $t_3(2)$ is 4 (for $t_i(k)$, i represents a node number, k represents a voltage level), as shown in Figure 2(b). (2) Calculating the sub-optimal combination of task modes before adding task u_3. We can get the index for $E_2[j]$ by subtracting $t_3(2)$ from TC: $TC - t_3(2) = 9 - 4 = 5$. Then we arrive at the location $E_2[5]$, which means that the optimal energy consumption to execute all the tasks from the root to u_2 is 38. By checking the mode assignment, we can see that V_1 is allocated to u_2. Therefore, the execution time of task u_2 is $t_2(1) = 2$. (3) In a similar way, we can determine that V_2 is assigned to u_1 and its execution time is $t_1(2) = 3$. Thus, the total execution time from u_1 to u_3 is $3 + 2 + 4 = 9$ (which is not greater than 9) and the total energy consumed is $10 + 28 + 8 = 46$.

V. Algorithms

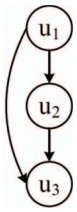
In this section, an algorithm, *Energy-aware Dynamic Task Scheduling* (EDTS), is devised to minimize the total energy consumption while satisfying the timing constraint.

For real-time applications on smartphones, we use the following major steps to implement the energy-aware scheduling. First, we partition and map the tasks in a DAG G onto the microprocessors of a smartphone platform. Then, an initial schedule of DAG G with the task execution order and communication links is obtained. Second, we identify the *critical path* (CP) by finding the path with the longest execution time. If there are more than one longest path in the graph, we select the one with the largest energy consumption in the DAG G. Third, based on the ACETs for all tasks in the graph, we can obtain a static schedule by our static scheduling algorithm. Finally, within each scanning period, the whole task graph is dynamically scheduled and the execution order of each task is determined by our dynamic scheduling algorithm.

During partitioning and mapping the tasks in a DAG, we consider related architectural constraints, heterogeneity, and resource capacities of smartphone platforms. The available energy of each processor may vary over time for different applications. Whenever the resource availability varies too much, the DAG needs to be repartitioned and re-mapped onto processors to maintain energy efficiency. We adopt the partitioning scheme, VPIS, proposed in [15] to schedule tasks onto microprocessors, with the consideration of various constraints and conditions. Our objective is to balance the load and minimize the total system energy consumption.

A. The Critical Path Assignment (CPA) Optimal Algorithm

We use a dynamic programming method to solve the energy-aware scheduling problem for smartphone systems.



(a)

Node	V_1		V_2	
	T_1	E_1	T_2	E_2
u_1	1	20	3	10
u_2	2	28	3	22
u_3	2	30	4	8

(b)

E[j]	j								
	1	2	3	4	5	6	7	8	9
$E_1[j]$	20_1	20_1	10_2	10_2	10_2	10_2	10_2	10_2	10_2
$E_2[j]$	--	--	48_1	42_2	38_1	32_1	32_1	32_1	32_1
$E_3[j]$	--	--	--	--	78_1	72_1	56_2	50_2	46_2

(c)

Figure 2: An motivational example. (a) A simple DAG. (b) The execution time and energy consumption for each task at different voltage modes. (c) The procedures to derive the minimal energy consumption for the DAG. $E_i[j]_k$ represents that assigning voltage level k to task i is optimal, when the time constraint is j.

Given the timing constraint TC, a DAG G, and an assignment A, we give several definitions as follows:

Definition 5.1: Assignment A: An allocation scheme assigns a specific voltage mode to each task in a DAG.

Definition 5.2: G^i: A subgraph i, which starts from the root of the task graph till the node v_i.

Definition 5.3: $E_A(G^i)$ and $T_A(G^i)$: The total energy consumption and the total execution time of G^i under the assignment A.

In our algorithm, each step achieves a currently minimum total energy consumption of G^i while satisfying various timing constraints.

A table $D_{i,j}$ (i represents a node number, and j represents time) will be built, where each entry of this table stores the smallest energy E that has been obtained.

In every step of our algorithm, we will consider at least one task. When two tasks are added together, total energy consumption is the sum of their energy consumption, $E'_{i,j}$ = $E^1_{i,j}$ + $E^2_{i,j}$. For each entry, we only keep the smallest total energy consumption and the corresponding voltage level assignment. When there is more than one solution, we keep the one with the smallest total execution time. If the total execution times are also the same, all solutions will be kept. When a critical path is found, we will use the optimal algorithm, CPA, to get the optimal solution for the energy-aware scheduling problem. The algorithm is shown in Algorithm V.1.

In algorithm CPA, we first build a local table $B_{i,j}$ for each node. The table $B_{i,j}$ only stores energy consumption of a node under different voltage levels. In the next step of the algorithm, when $i = 1$, there is only one node. We set the initial value, and let $D_{1,j} = B_{1,j}$ (line 1). Then we build the table $D_{i,j}$ (line 3-7), by using the dynamic programming method. For each node u_i for each j, we vary time k ($1 \leq k \leq j$) in table $B_{i,j}$ (line 4). We add energy consumption together in the two tables $B_{i,k}$ and $D_{i-1,j-k}$ (line 6). We also consider the communication cost w_i when tasks i and $i+1$ are scheduled on different microprocessors. If the two tasks are implemented on the same core, α is 0; otherwise, α is 1. Finally, we keep the smallest total energy and the corresponding voltage selection. The energy in $D_{i,j}$ is the minimum total energy for graph G^i under the timing constraint j.

Algorithm V.1 The *CPA* Algorithm for Critical Path.

Require: M different voltage levels, a critical path, and a timing constraint TC.
Ensure: An optimal assignment for the critical path.
Build a local table $B_{i,j}$ for each node on the critical path;

1: Let $D_{1,j} = B_{1,j}$
2: Start from $u_1 \rightarrow u_{|U|}$, compute D step by step;
3: **for** each time j of node $u_i, i > 1$ **do**
4: **for** each time k in $B_{i,k}$, $1 \leq k \leq j$ **do**
5: **if** $D_{i-1,j-k}! = NULL$ **then**
6: $D_{i,j} = D_{i-1,j-k} + B_{i,k} + \alpha w_i$;
7: For $D_{i,j}$, always keep the minimum one if there are multiple results;
8: **else**
9: Continue;
10: **end if**
11: **end for**
12: **end for**
13: **return** $D_{|U|,j}$;

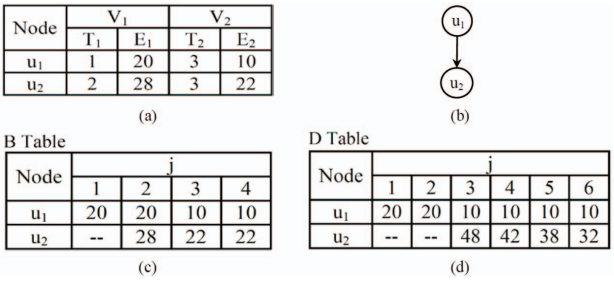


Figure 3: (a) Initial parameters. (b) A DFG. (c) The corresponding B table. (d) Part of corresponding D table.

For example, for the DFG shown in Figure 3(b), the initial parameters are shown in Figure 3(a). We compute the corresponding B table of node u_1 and u_2 as follows.

For node u_1, after sorting, we can get (T (time): E (energy)) pairs: (1: 20), (2: 20), (3: 10), and (4: 10), as shown in Figure 3(c). Figure 3(d) shows the corresponding $D_{i,j}$ table. For instance, we can compute entry $D_{2,3}$ for the path $u_1 \rightarrow u_2$ with the time constraint $TC = j = 4$ as follows. Using the CPA algorithm, two cases can satisfy this time constraint. Case 1: $4 = 2 + 2$, with $D_{2,4} = D_{1,2} + B_{2,2}$. Energy consumption for this assignment is: $20 + 28 = 48$. Case 2: $4 = 1 + 3$ with $D_{2,4} = D_{1,1} + B_{2,3}$. Energy consumption under this assignment mode is: $20 + 22 = 42$. Energy consumption of Case 2 is less than that of Case 1. Hence, we fill 42 into entry $D_{2,4}$.

Time complexity: It takes $O(M)$ to compute one value of $D_{i,j}$, where M is the maximum number of voltage levels. Thus, the complexity of the CPA algorithm CPA is $O(|U| * TC * M)$, where $|U|$ is the number of nodes and TC is the given timing constraint. Usually, the execution time of each node is bounded by a constant. So TC equals $O(|U|^c)$ (c is a constant). In this case, CPA is a polynomial algorithm.

B. The DFGCP Static Scheduling Algorithm

In this section, we propose a highly efficient algorithm, DFGCP, to solve the static scheduling problem. The algorithm is shown in Algorithm V.2.

Algorithm V.2 The *DFGCP* Algorithm

Require: M different voltage levels, a DFG $G=\langle U, ED, T, E, W\rangle$, and a timing constraint TC.
Ensure: An assignment for the DFG.
Find CP, a critical path of DFG G;
if Time(CP) $> TC$ **then**
  Flag $\leftarrow$ No;
  Use our *CPA* to find the minimal total energy consumptions and corresponding voltage assignments;
  When we found a solution, then set Flag $\leftarrow$ Yes;
end if
if Flag $==$ Yes **then**
  Output the assignment of G;
else
  Output "No Solution"; exit;
end if
For the nodes on the non-critical path (non-CP), we will use CPA algorithm to find the minimal energy consumptions and keep the corresponding voltage levels.
Add together the energy of CP and non-CP, we get the minimal total energy consumptions.

In DFGCP algorithm, we first find a *critical path* (CP) of the DFG G. If the total execution time of the CP is larger than the timing constraint TC, we will use the CPA algorithm to find the minimal total energy consumption and the corresponding voltage selections. In each step, we will consider the voltage level transfer overheads when using DVS. For each node, if it is not on the same processor with its parent nodes, the communication cost with the parents w_i will be considered. Finally, if we find a solution for CP within TC, the algorithm continues using CPA to find the optimal solution for non-CP paths. At this time, we fix the assignments of the overlapping nodes of CP and non-CP paths.

Time Complexity: DFGCP is a polynomial time algorithm. The complexity of the CPA algorithm is $O(|U| * TC * M)$, where $|U|$ is the number of nodes and TC is the given timing constraint. M is the maximum number of voltage levels. We use CPA to compute every path once. The total number of paths is bounded by $O(|U|^2)$. Hence, CPA is a polynomial time algorithm. For a sparse graph, the number of paths is very small, assuming a constant c, then the complexity is approximately linear and the amount of computation time is very small.

C. The EDTS Dynamic Scheduling Algorithm

Algorithm V.3 The EDTS Algorithm

Require: M different voltage levels, a DFG $G=\langle U, ED, T, E, W\rangle$, and a timing constraint TC.
Ensure: A dynamic scheduling for the DFG.
Get the initial scheduling by DFGCP algorithm;
Topologically sort the nodes, getting node sequence $u_i \in U$;
for each node $u_i, 1 \leq i \leq |U|$, not visited, get the one with the earliest start time **do**
  if required execution time is substantially different from ACET **then**
    Mark it as visited;
    Run DFGCP algorithm for the remaining nodes and find the new static schedule with minimal energy consumption while satisfying the new timing constraint ($TC = TC - \sum_{k=1}^{i} T_k$, where $\sum_{k=1}^{i} T_k$ is the time used);
  else
    Continue;
  end if
  Finish node u_i, and update system energy overhead and the information (such as the starting time) of nodes that are dependent on u_i;
  if current static schedule is not followed **then**
    Run DFGCP algorithm for the remaining nodes and find the new static schedule with minimal energy consumption while satisfying the new timing constraint ($TC = TC - \sum_{k=1}^{i} T_k$, where $\sum_{k=1}^{i} T_k$ is the time used);
  else
    Continue to the next node;
  end if
end for

The DFGCP static scheduling algorithm gives a solution by assuming all tasks run at ACETs. However, in real-

life scenarios, we do not know in advance the actual execution time of a task for smartphone applications. The information of these tasks will change greatly in runtime, thus even an optimal static schedule can become invalid in the dynamic case. In this subsection, we present an aggressive dynamic programming based online scheduling algorithm, called *Energy-aware Dynamic Task Scheduling* (EDTS). EDTS algorithm uses the results from DFGCP static scheduling algorithm, which obtains a near-optimal schedule based on the knowledge of ACET of each task.

The actual execution time of a task may be greater or less than its ACET, we first obtain a static schedule with DFGCP by assuming every task takes its ACET. However, if every task aggressively runs at this statically computed average case speed during runtime, some of them may miss their deadlines. Our EDTS algorithm uses the path information to track any changes of tasks in smartphone applications. When a task node is finished, EDTS checks whether the schedule is followed. If not, then the remaining task graph will be recomputed with the DFGCP static scheduling algorithm. Also, in the course of the implementation of each node, whenever the variation of execution time exceeds the pre-specified threshold value, DFGCP will be used to recompute. For example, we set difference ratios to be $\pm 5\%$ between the real execution time and its ACET used previously in DFGCP. The new computation will only implement the remaining subgraph with the updated ACET values.

Time Complexity: Our dynamic scheduling algorithm, EDTS, progressively improves performance based on the schedule obtained by the static scheduling, DFGCP. The EDTS algorithm is shown in Algorithm V.3. For a sparse graph, the complexity of this algorithm is $O(|U|(|U| * TC * M))$, where $|U|$ is the number of nodes, TC is the given timing constraint, and M is the maximum number of voltage levels. Hence, EDTS is a polynomial time algorithm. For general task graphs, since DFGCP is a polynomial time algorithm and EDTS calls $O(|U|)$ times of DFGCP, EDTS is also polynomial.

VI. Experiments

In this section, we conduct experiments with the EDTS algorithm on a set of benchmarks including *Wave Digital filter* (WDF), *Infinite Impulse filter* (IIR), D*ifferential Pulse-Code Modulation device* (DPCM), *Two dimensional filter* (2D), *Floyd-Steinberg algorithm* (Floyd), and *All-pole filter*. The number of tasks for these benchmarks has been augmented with the unfolding technique (the unfolding rate is 5). The proposed run-time system has been implemented and a simulation framework to evaluate its effectiveness has been built. The dynamic processor loads are obtained through measurements on a 600MHz Crusoe processor. The execution time (ACET and WCET) and energy consumption are based on the profiling. The execution time of each node follows a Gaussian distribution.

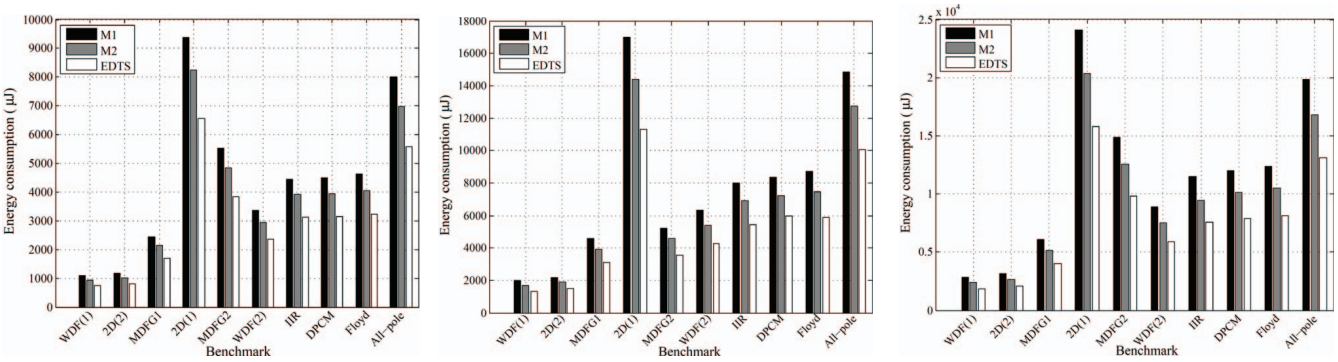
We conducted experiments using three different methods: Method 1: Dynamic version *parallelism-base* (PS) algorithm [16]; Method 2: *Critical path dynamical scheduling* (CPDS) [17]; Method 3: Our EDTS algorithm. Method 1 uses a greedy technique to further reclaim the slack generated during runtime. Initially all tasks are assigned with a statically computed processing speed. All the available slacks from a task due to its earlier completion are given to the next expected task running on the same processor. The speed for the next expected task will be adjusted based on its ready time [17].

The experiments are conducted based on the power model of 70nm processor [18]. Then energy consumption per cycle can be calculated by using Equation (9) proposed in [18]. The power is derived from the formula $E_{cyc} = P/f$. In experiments, we use M different voltage types with a descending processing speed in V_1, V_2, $\cdots$, V_M. The time and energy overheads during a voltage transition among the above voltage levels are calculated based on Equations (15) and (20) in paper [19]. We compare our results with those from Method 1 and Method 2 on a PC with a P4 2.1G processor running on Red Hat Linux 9.0.

The experimental results are shown in Figure 4(a) to Figure 4(c) when the number of "TC" is 2000, 3000, and 4000, respectively. In these figures, M1, M2, and EDTS represent the Method 1, Method 2, and our proposed dynamic scheduling algorithm, respectively.

As shown in these figures, our algorithm achieves significant energy reduction, compared to Method 1 and Method 2. For example, with 3 voltage levels, compared with Method 1 and Method 2, EDTS shows 30.1% and 20.2% reduction in energy consumption, respectively. This is mainly because our method uses the optimal algorithm CPA to implement the energy-aware static scheduling.

Hence, our EDTS algorithm can significantly improve the performance of smartphone systems. We can see that with more voltage-level selections, the reduction in total energy consumption is more prominent. For example, with 3 voltage levels, compared to Method 1, EDTS shows an average 30.1% reduction in total energy consumption, while using 5 voltage levels, the reduction can be achieved



(a) $TC = 2000ns, M = 3$ (b) $TC = 3000ns, M = 4$ (c) $TC = 4000ns, M = 5$

Figure 4: The comparison of total energy consumption for Method 1, Method 2 and EDTS when (a) TC = 2000 ns and $M = 3$, (b) $TC = 3000$ and $M = 4$, (c) $TC = 4000$ and $M = 5$ across a set of benchmarks.

up to 34.2%.

VII. Conclusion

Smartphones are power-hungry devices. This paper studied how to minimize total energy consumption while satisfying application timing constraints for smartphone systems. We proposed a highly efficient algorithm, *Energy-aware Dynamic Task Scheduling* (EDTS), which utilizes the results from a static scheduling algorithm and aggressively reduces energy consumption. Experimental results across a suite of benchmarks showed that our algorithm can achieve significantly higher energy efficiency for smartphones.

Acknowledgement

This work was supported in part by the University of Kentucky Start Up Fund and NSFC 61071061; the NSF CNS-0915762 (CSR), CCF-08452578 (CAREER), CNS-0917137 (CSR).

References

[1] http://www.imgtec.com.

[2] P. Pillai and K. G. Shin, "Real-time dynamic voltage scaling for low-power embedded operating systems," in *SOSP*, Oct. 2001, pp. 89–102.

[3] T. D. Burd, T. A. Pering, A. J. Stratakos, and R. W. Brodersen, "A dynamic voltage scaled microprocessor system," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 11, pp. 1571–1580, Nov. 2000.

[4] M. Qiu, L. T. Yang, Z. Shao, and E. H.-M. Sha, "Dynamic and leakage energy minimization with soft real-time loop scheduling and voltage assignment," *IEEE TVLSI*, vol. 18, no. 3, pp. 501–504, Mar. 2010.

[5] M. Qiu, C. Xue, Z. Shao, and E. H.-M. Sha, "Energy minimization with soft real-time and DVS for uniprocessor and multiprocessor embedded systems," in *IEEE/ACM DATE*, Acropolis, Nice, France, Apr. 2007.

[6] W. Yuan and K. Nahrstedt, "Energy-efficient soft real-time cpu scheduling for mobile multimedia systems," in *SOSP*, Oct. 2003, pp. 149–163.

[7] M. Qiu, E. H.-M. Sha, M. Liu, M. Lin, S. Hua, and L. T. Yang, "Energy minimization with loop fusion and multi-functional-unit scheduling for multidimensional DSP," *JPDC*, vol. 68, no. 4, pp. 443–455, Apr. 2008.

[8] F. Gruian, "Hard real-time scheduling for low-energy using stochastic data and DVS processors," in *ISLPED*, Aug. 2001, pp. 46–51.

[9] J. R. Lorch and A. J. Smith, "Improving dynamic voltage scaling algorithms with PACE," in *SIGMETRICS*, Jun. 2001, pp. 50–61.

[10] J. Luo and N. K. Jha, "Static and dynamic variable voltage scheduling algorithms for real-time heterogeneous distributed embedded systems," in *ASP-DAC*, Jan. 2002, pp. 719–726.

[11] M. Qiu and E. H.-M. Sha, "Cost minimization while satisfying hard/soft timing constraints for heterogeneous embedded systems," *ACM TODAES*, vol. 14, no. 2, Article 25, pp. 1–30, Mar. 2009, (TODAES 2011 Best Paper Award).

[12] M. Qiu, Z. Jia, C. Xue, Z. Shao, and E. H.-M. Sha, "Voltage assignment with guaranteed probability satisfying timing constraint for real-time multiproceesor DSP," *Journal of Signal Processing Systems (JSPS)*, vol. 46, no. 1, pp. 55–73, Jan. 2007.

[13] F. Lin, Z. Wang, R. Likamwa, and L. Zhong, "Transparent programming of heterogeneous smartphones for sensing," Technical Report 0310-2011, Rice University, Tech. Rep., March 2011.

[14] X. Li, P. J. Ortiz, J. Browne, D. Franklin, and J. Y. Oliver, et al, "Smartphone evolution and reuse: Establishing a more sustainable model," in *ICPPW*, Sep, 2010, pp. 476–484.

[15] Z. Wang and X. S. Hu, "Energy-aware variable partitioning and instruction scheduling for multibank memory architectures," *ACM TODAES*, vol. 10, no. 2, pp. 369–388, Apr. 2005.

[16] R. Mishra, N. Rastogi, D. Zhu, D. Mosse, and R. Melhem, "Energy aware scheduling for distributed real-time systems," in *IEEE IPDPS*, Apr. 2003.

[17] Y. Liu, B. Veeravalli, and S. Viswanathan, "Critical-path based low-energy scheduling algorithms for body area network systems," in *IEEE RTCSA*, Aug, 2007, pp. 301–308.

[18] S. Martin, K. Flautner, T. Mudge, and D. Blaauw, "Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads," in *IEEE/ACM ICCAD*, 2002, pp. 721–725.

[19] J. Park, D. Shin, N. Chang, and M. Padram, "Accurate modeling and calculation of delay and energy overheads of dynamic voltage scaling in modern high-performance microprocessors," in *ACM ISLPED*, Aug. 2010, pp. 419–424.