2010 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing

Power Management in Real Time Embedded Systems through Online and Adaptive Interplay of DPM and DVFS Policies

Khurram Bhatti*, Cecile Belleudy*, and Michel Auguin*
*LEAT Research Laboratory, University of Nice-Sophia Antipolis,
250-Rue Albert Einstein, 06560-Valbonne, France
{bhatti, belleudy, auguin}@unice.fr

***Abstract*— This paper[1] considers the problem of power/energy minimization for periodic real-time tasks that are scheduled over multiprocessor platforms that have dynamic power management (DPM) and dynamic voltage & frequency scaling (DVFS) capabilities. Early research reports that while both DPM and DVFS policies perform well individually for a specific set of conditions, they often outperform each other under different workload and/or architecture configuration. Thus, no single policy fits perfectly all operating conditions. Instead of designing new policies for specific operating conditions, this paper proposes a generic power management scheme, called the Hybrid Power Management (HyPowMan) scheme. This scheme takes a set of well-known existing (DPM and DVFS) policies, each of which performs well for a given set of conditions, and adapts at runtime to the best-performing policy for any given workload. We performed experiments with state-of the-art DPM and DVFS techniques and results show that HyPowMan scheme adapts well to the changing workload and always achieves overall energy savings comparable to the best-performing policy at any point in time.**

***Keywords*-Real Time Systems, Dynamic Voltage & Frequency Scaling, Machine Learning, Energy-efficient Scheduling.**

I. Introduction

The high power/energy consumption of modern processors becomes a major concern because it leads to decreased lifetime for battery-operated systems, increased heat dissipation which requires expensive packaging and cooling technology, and decreased reliability, especially for systems with many processors. To address this issue, many software techniques, especially energy-efficient scheduling techniques, have been proposed [1], [2], [3], [4], [5], [6], [7], [8], [9], [10] over the years. Dynamic Power Management (DPM) is one of such techniques that is well studied and practiced in embedded real time systems. These techniques put system components into low power states whenever they are idle due to unavailability of workload. Once applied, DPM policies eliminate both dynamic as well as static power dissipation. However, the inconvenience is that once in a power-efficient state, bringing a component back to the active/running state requires additional energy and/or latency to service an incoming task. In recent years, processors that have multiple voltage levels and frequencies (thus multiple power levels) have become available [11], allowing the design of highly flexible systems. Real time dynamic voltage and frequency scaling (DVFS) technique is also a promising technique that exploits this offered architectural flexibility. Real time applications exhibit large variations in their actual execution time and thus, often finish much earlier than their estimated worst-case execution time [12], [8]. DVFS techniques exploit these variations in actual workload for dynamically adjusting voltage and frequency to reduce dynamic power consumption of processors. The inconveniences with evolved processor technology are the significantly increased static power consumption, which becomes particularly dominant at lower operating frequencies, and the limited number of voltage and frequency levels. Both DPM and DVFS techniques perform well for a given set of conditions. However, they often outperform each other under different workloads and architecture configuration [13]. The primary motivation for our work comes from the fact that no single policy fits perfectly all operating conditions. Instead of designing new power management policies (whether DPM or DVFS policies) for specific operating conditions, we propose a generic power management scheme, called the Hybrid Power Management (HyPowMan) scheme. Our proposed scheme takes a set of well-known existing policies, each of which performs well for a given set of conditions, and adapts at runtime to the best-performing technique for any given workload. We demonstrate in this paper that our proposed scheme quickly converges to the best-performing technique at any point in time. HyPowMan scheme enhances the ability of portable embedded systems to adapt with changing workload and give an overall performance and energy savings that is better than any single policy can offer. Rest of this paper is organized as follows. In Section-II,some related research work and contributions of this work is presented. In Section-III, system model is discussed. Section-IV presents HyPowMan scheme in detail. Section-V presents experimental setup and simulation results. This paper is concluded in Section-VI.

[1]This work is partially supported by European project COMCAS [CA501], the Higher Education Commission of Pakistan, and French cluster of Secured Communicating Solutions (SCS).

II. Related Work

A. Dynamic Power Management

Dynamic power management policies include simple timeout policies [1], predictive [2], stochastic-modeling-

978-0-7695-4322-2/10 $26.00 © 2010 IEEE
DOI 10.1109/EUC.2010.35

based [3], session clustering [4], online [5], and adaptive learning-based policies [14]. Lu and De Micheli present in [4] a quantitative comparison of various existing policies. Existing policies can be broadly classified into predictive schemes and stochastic schemes according to authors in [6], [7], [15]. Predictive policies predict the duration of upcoming idle period as soon as a system component goes idle and the decision to transition state can be made if the prediction indicates a long idle period. Stochastic policies take into account both power consumption as well as performance penalty. They model the request arrival and device's power state changes as stochastic processes. Minimizing power consumption and delays then become stochastic optimization problems [13]. In [16], authors assume the arrival of requests as a stationary geometric distribution and model power management as a discrete-time markov decision process. In [14], the work is extended to handle non-stationary request arrival. Authors in [10] propose to accumulate idle time intervals on some processors before applying state-transition while other processors work more to meet deadline guarantees in a multiprocessor platform.

B. Dynamic Voltage & Frequency Scaling

Numerous researchers have explored DVFS on single-processor systems such as [17], [18], [12]. However, fewer have considered the problem of applying DVFS on multiprocessor platforms [8]. In [9], authors have broadly classified DVFS techniques into intra-task and inter-task strategies. In intra-task strategies, available dynamic slack time is reallocated inside the same task. These techniques often require insertion points in application's code to measure the evolution of a task over its execution time. Drawback of such techniques is the complexity and cost of insertion points for measurement [17]. Inter-task DVFS techniques redistribute dynamic slack either among all ready tasks [9] or to appropriate priority single task only at task's boundaries (release, termination). Authors in [19] consider dynamic slack sharing among multiple processors and reducing the speed globally. In [20], Aydin et al. propose their solution for periodic hard real time tasks on identical multiprocessors with DVFS support when only partitioned scheduling is used. In [21], authors propose an inter-task multiprocessor DVFS technique in which execution of tasks in real schedule mimics the canonical schedule (produced at runtime) and dynamic slack is fully consumed by a single appropriate priority task.

C. Interplay of DPM and DVFS Techniques

Some recent research work that focuses on system-wide energy efficiency, applies both, DPM and DVFS techniques together on different system components such that certain components apply DVFS only while others apply DPM only. Authors in [13], [22], [23] propose to apply DVFS on processors only while DPM for I/O devices to save overall energy consumption. Authors in [13] propose an approach to apply multiple existing DPM policies on a single processor system in order to achieve best performance and adaptability to the varying workload. They take a set of existing DPM policies and design a control mechanism that selects, in an online fashion, the best-suited policy for a given idle period. However, none of the previous research work has attempted to apply an entirely online and adaptive interplay of multiple DVFS as well as DPM techniques together on an identical multiprocessor platform.

D. Our Contributions

1) This work proposes a novel and generic scheme for power management, called HyPowMan scheme. Instead of designing new power management policies (DPM or DVFS) to target specific operating conditions, HyPowMan scheme takes a set of well-known existing policies, each of which performs well for a given set of conditions, and proposes a machine-learning mechanism to adapt at runtime to the best-performing policy for any given workload. The decision of applying suitable policy (whether DPM or DVFS) is online and adaptive to the varying workload. HyPowMan scheme enhances the ability of portable embedded systems to work with larger set of conditions by adapting with the changing workloads and gives an overall performance that is better than any single policy can offer.
2) Our proposed scheme is generic in the sense that it permits to integrate existing as well as new power management techniques (DPM and/or DVFS). Moreover, HyPowMan scheme can be applied under the control of global as well as partitioning-based scheduling algorithms.
3) HyPowMan scheme is intended mainly for multiprocessor real time systems. However, applying this scheme on single-processor systems is rather trivial.

To the best of our knowledge, this is the first research work which applies a completely online and adaptive interplay of multiple DVFS and DPM techniques together through a machine-learning mechanism on identical multiprocessor systems that run real time preemptive tasks.

III. System Model & Notations

We consider a multiprocessor platform composed of m identical processors such that P = $\{P_1, P_2, ..., P_m\}$. All processors in P possess DPM and DVFS support and voltage/frequency on every processor can be scaled independently and over a continuous spectrum. This assumption can be easily lifted for processors with discrete operating frequencies. We consider that a statically specified optimum supply voltage (V_{dd}) and operating frequency (F_{ref}) are available. F_{op} and V_{op} refer to all other frequencies and corresponding supply voltages, respectively. To simplify our discussion, we consider that voltage and frequency

are always adjusted together. We consider a frame-based system (of application) in which a frame of certain length is executed repeatedly. A finite set of n tasks such that TS = $\{T_{a1}, T_{a2}, ..., T_{an}\}$ is executed in each frame. Each member task of TS is characterized by at least a quadruplet (r_i, C_i, d_i, T_i) where the elements of quadruplet refer to release time, worst-case execution time, relative deadline, and periodicity of a task, respectively. Moreover, a task may have runtime parameters such as its actual execution time (AET_i) which is smaller than C_i and greater than its best-case execution time B_i. Deadlines of all tasks are considered equal to their period (d_i=T_i). Utilization factors of a single task as well as that of complete task set are given by ratio u_i=C_i/T_i and Eq.1, respectively.

$$u_{sum}(TS) = \sum_{T_i \in (TS)} u_i \quad (1)$$

Note that a *necessary* condition for schedulability of a task set on a system of *m* identical processors is to have the aggregate utilization less than or equal to the computing capacity of the system. Consequently, we assume that the condition $u_{sum}(TS) \leq m$ holds throughout this paper. We adopt a global approach to multiprocessor scheduling in which no task is permanently assigned to any processor. All tasks are preemptive and support migration as in case of SMP architectures [24]. Power consumption of CMOS technology-based processors, represented as a function of speed (S) in variable speed settings, can be decomposed into static and dynamic components which relate to supply voltage V_{op}, operating frequency F_{op}, and leakage-current (Iq) through an approximate relation given by Eq.2.

$$Pwr(S) = \gamma C_{eff} V_{op}^2 F_{op} + I_q V_{op} \quad (2)$$

Here, γ and C_{eff} refers to the switching activity and effective load capacitance, respectively. The first addend in Eq.2 corresponds to the amount of dynamically dissipated power caused by switching circuitry while second addend models statically dissipated power caused by leakage current. If a task occupies a processor during an execution interval of $[t_1, t_2]$ then the energy consumed by the processor during this time interval is given by Eq.3.

$$E(t_1, t_2) = \int_{t_1}^{t_2} Pwr(S(t))dt \quad (3)$$

Eq.2 and Eq.3 demonstrate that striking power/energy savings can be achieved by either putting system components in power-efficient state or scaling the F_{op} and V_{op} simultaneously. However, g a processor's state or scaling F_{op} & V_{op} consumes energy and increases latency to service an incoming task which cannot be ignored.

IV. Hybrid Power Management Scheme

HyPowMan scheme takes a set of well-known existing power management policies, each of which performs well for a given set of conditions, and on top of them, devises a *policy selection mechanism* which could select best-performing policy for a given type of workload. HyPowMan scheme implements this policy selection mechanism through a machine-learning algorithm. Machine-learning algorithm provides theoretical guarantee on overall performance converging to that of the best-performing policy among the available ones. This is somewhat similar to that of hybrid branch predictors employed in microprocessors and used in [13] as well. We refer to each participating power management policy as an *expert* and the set of all participating policies collectively as *expert set*. Any multiprocessor DPM or DVFS policy is eligible to become member of expert set. When a processor is busy in executing tasks, all DPM-based experts are inactive and are said to be *dormant* experts. However, DVFS-based experts can (or cannot) be in inactive state depending on the workload. Conversely, whenever the processor is idle, all DVFS-based experts are dormant. Any expert, which is currently active, is said to be *working* expert. A working expert makes the power management decisions on processors under the control of applied scheduling policy. For instance, expert3 in Fig.1 is the working expert. Working expert returns to dormant state, which is the default state for all experts, once it finishes its job or another working expert replaces it. The most critical task for HyPowMan scheme is to select an appropriate expert for a given power management opportunity. Before we present our machine-learning algorithm in detail, we would like to emphasis a fundamental difference between DPM-based and DVFS-based experts. As mentioned earlier in Section-1, the power management opportunities (also referred as *input* hereafter) for DPM-based experts are the idle time intervals which are inherently present in an application's schedule. Whereas, input for DVFS-based experts is dynamic slack, which is generated at runtime due to the variations in actual workload (to which, DPM-based experts can also exploit while working expert). Thus, challenges for HyPowMan scheme are, how to measure the performance (at runtime) for different experts which work for different kind of inputs, how to evaluate them, and how to employ them in a multiprocessor context. Note that the objective of HyPowMan scheme is to converge towards the best-performing technique *within given expert set* only and not to find *the best possible* energy savings under given operating conditions.

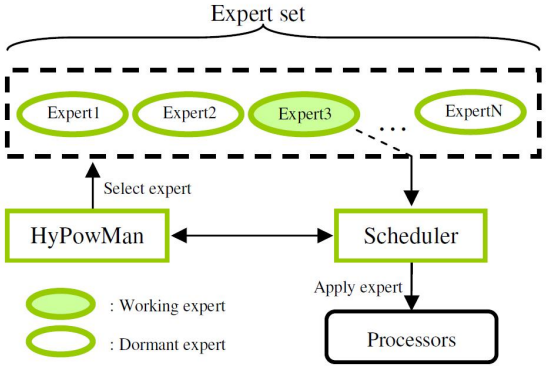

Figure 1. HyPowMan scheme working mechanism

A. Machine-Learning Algorithm

Machine-learning algorithm employed in HyPowMan considers at most *N* experts are present in expert set. The algorithm associates a weight vector W^{input} with expert set, where W^{input}= $(w_1^{input}, w_2^{input}, ..., w_N^{input})$ consists of weight factors corresponding to each expert *k* ($\forall k, 1 \leq k \leq N$) for a given input. Every time an input arrives, this weight vector is updated. Initially, weight factors of all individual experts are equal and sum to one in order to provide equal opportunity for all experts to perform well. However, these weights may not sum to one later during execution. HyPowMan scheme maintains a probability vector H^{input} associated with expert set, where H^{input}= $(h_1^{input}, h_2^{input}, ..., h_N^{input})$ consists of probability factors corresponding to each expert k such that, $(0 \leq h_k^{input} \leq 1)$. This probability reflects the performance of an expert based on its weight factor. It is obtained by normalizing weight factors as shown in Eq.4. The probability factor provides a measure on each expert's performance on previous input such that, at any point in time, the best-performing expert has the highest probability.

$$H^{input} = W^{input} / \sum_{k=1}^{N} w_k^{input} \tag{4}$$

HyPowMan selects expert with highest probability amongst all experts to become a working expert on next input. In case the probability factors of multiple experts are equal, working expert is chosen randomly. Once selected, a working expert governs all decisions related to power management under the control of scheduling policy. When an input is finished, the performance of all experts is evaluated. Working expert is evaluated based on how much energy was saved and how much performance degradation was incurred under that particular expert. Dormant experts are evaluated based on how they would have performed had they be selected as working expert. This evaluation is based on the *loss* factor of each expert. Loss factor is evaluated with respect to an *ideal* (offline) power management policy that offers maximum possible energy savings and zero performance degradation. The loss factor incurred by an expert *k* is referred as l_k^{input}. The value of loss factor is a composition of loss in energy saving and performance degradation and it is influenced by their relative importance which is expressed by factor α such that $(0 \leq \alpha \leq 1)$. We refer to the loss factors corresponding to energy and performance as l_{ke}^{input} and l_{kp}^{input}, respectively, for expert k. Eq.5 represents joint loss factor of individual experts.

$$l_k^{input} = \alpha l_{ke}^{input} + (1 - \alpha) l_{kp}^{input} \tag{5}$$

Computation of loss factor slightly differs for DPM and DVFS experts. We shall elaborate this difference shortly. Once the joint loss factor l_k^{input} is calculated, final step in algorithm is to update weight factors for each expert based on the loss they have incurred as shown by Eq.6.

$$w_k^{input+1} = w_k^{input} \beta^{l_k^{input}} \tag{6}$$

Here, β is a constant such that $(0 \leq \beta \leq 1)$. The value of β should be set between 0 and 1 depending on the granularity of weight factors, i.e. the higher the value of β is set, the lower the variation in weight occurs for a given input. Eq.6 depicts that the weight factors corresponding to experts with higher loss factors are reduced while for the experts with lower loss factors are increased by simple multiplicative rule. This gives higher probability of selecting better performing experts for the next input. Note that weight and probability factors for all experts are updated once the input is terminated. Calculations related to selecting the working expert (for next input) are performed during the *active* time and hence no additional overhead or latency is incurred when the inputs actually occur. HyPowMan scheme itself has linear time-complexity of the order *O(N)*, where N refers to the size of expert set. For known size of expert set, HyPowMan scheme has a constant time-complexity, i.e. *O(1)*. For experts have different time-complexities within expert set, the overall time-complexity of HyPowMan scheme would be bounded by the expert having largest time-complexity.

1) Loss Factor for DPM and DVFS Experts: For DPM experts, the input is an idle time interval. Energy loss factor (l_{ke}^{input}) is calculated by comparing the length of idle period with the time a processor has or would have spent in power-efficient state. If this time is less than the break-even time[2] of processor, then there is no savings on energy and loss is maximum (l_{ke}^{input}=1). Otherwise, for sleep time greater than break-even time, Eq.7 is used to compute l_{ke}^{input}.

$$l_{ke}^{input} = 1 - (T_{sleep-k} / T_{idle}) \tag{7}$$

Here, $T_{sleep-k}$ and T_{idle} refer to the time a processor passed in power efficient state and available idle time, respectively. Performance loss is computed based on whether a processor switched to a power efficient state or not. If processor was transitioned to power efficient state, the loss on performance is incurred ($l_{kp}^{input} = 1$). Otherwise, loss of performance is zero ($l_{kp}^{input} = 0$). For DVFS experts, the input is dynamic slack time. Similar to DPM experts, whenever there is an input available, the time it takes to scale F_{op} and V_{dd} is compared to available slack time. If the amount of slack justifies the scaling of F_{op} and V_{op} to lower values and back to F_{ref} and V_{dd}, then the loss on energy is considered zero ($l_{ke}^{input} = 0$). Otherwise, no gain on energy can be achieve by applying DVFS and thus, the loss is maximum ($l_{ke}^{input} = 1$). Performance loss is computed based on whether F_{op} and V_{op} were scaled or not. If scaling is applied, performance loss is incurred ($l_{kp}^{input} = 1$). Otherwise, no loss would have incurred ($l_{kp}^{input} = 0$).
Algorithm-1 presents our machine-learning algorithm. Suitable values for parameters (α and β corresponding to DPM

[2]Given that, a device is associated with non-zero transition costs, break-even time denotes the minimum length of idle interval which justifies a device's transition from active state to an energy-efficient state.

and DVFS experts) should be specified by the user in accordance with the relative importance of energy and performance loss factors (line 1). Initial weights, as described earlier, are equal and sum to one in the beginning. Then the expert which offers highest probability is selected to become working expert at the start of next input (line 4). If all probability factors are equal, (which is the case at start up) then any expert is chosen randomly to become working expert. At the end of an input, each expert is evaluated (line 5) and its weight and probability is updated (line 6 and 7). Note that HyPowMan scheme has no direct control on the decision-making process of either experts or of the scheduling algorithm itself. HyPowMan scheme rather functions as a top-level entity which evaluates the performance of different experts and based on previous performance, selects best-performing expert. Once selected, its the responsibility of an expert to provide temporal guarantees for real time tasks as long as it is a working expert. Whenever a DVFS-based expert is substituted by any other expert, all parameters of running/preempted tasks are reinstated with respect to nominal operating conditions (F_{op} & V_{op}). Similarly, when a DPM-based expert is substituted by any other expert, all processors from power-efficient state (if any) are recovered to running state. Only those techniques providing real time guarantees are chosen to be member of expert set.

Algorithm 1 Machine-learning

Set Parameters
$\{\alpha^{dpm}, \alpha^{dvfs}\} \in [0,1]$
$\{\beta^{dpm}, \beta^{dvfs}\} \in [0,1]$
Initialize weight factors: $w_k^1 \in [0,1]; (\forall k, 1 \leq k \leq N)$
for all future inputs **do**
Select expert having maximum probability: $h^{input+1} = max\left[h_k^{input}\right]$
if input arrives **then**
Apply selected expert as working expert
else if input terminates **then**
Update weight vector (W^{input})
Update probability vector (H^{input})
end if
end for

B. Selection of Experts

To clearly demonstrate the working of HyPowMan scheme, we have selected a DVFS expert [21] and a DPM expert [10]. Authors in [21] suggest that their DVFS technique is an online slack reclamation technique which permits the dynamic slack, produced by the precedent task, to be fully consumed by single subsequent task at the appropriate priority level. Such greedy allocation of slack allows large variations in F_{op} and V_{op} which eventually results in larger gains on energy consumption. Authors in [10] suggest that, in a multiprocessor real time system composed of *m* processors, their DPM technique aggressively extracts out idle intervals from some processors (let us say *k* processors) and clusters them on some other processors (let us say *m-k* processors) to elongate the duration of idle period. Once the idle period is accumulated, transitioning *m-k* processors to power-efficient states then become a matter of comparing the length of idle period against the break-even time of particular processor in question. Further details on individual experts can be found in [21] and [10].

V. Experimental setup & results

In this section, we provide an experimental evaluation of HyPowMan scheme. Penalties associated with state-transition of processor and changes in F_{op} and V_{op} are considered in presented results. We have evaluated the performance of HyPowMan scheme using a freeware java-based simulation tool called STORM (Simulation TOol for Real-time Multiprocessor Scheduling) [25]. Energy consumption in processors is measured under the control of each expert alone as well as under the control of HyPowMan scheme. These measurements are performed as a function of variations in three parameters, i.e. variations in total utilization $u_{sum}(TS)$, number of tasks, and best-case to worst-case execution time ratio (bcet/wcet) of each task. Moreover, our results show that the variations in α and β significantly alter the convergence of HyPowMan scheme towards best-performing expert. For each data point, a task set is randomly generated, in which, each task has a uniform probability to have small (5-25ms), medium (25-75ms), or long (75-120ms) periods. All task periods are uniformly distributed among these three ranges. Note that a similar period generation scheme is used in [18], [26]. Individual utilization of each task u_i is also generated randomly such that the Eq.1 always holds. We have used hardware parameters from Marvell's XScale technology-based processor PXA270 [11] in experiments. PXA270 processor supports six voltage-frequency levels as shown in table-I and five power-efficient states as shown in table-II. All results on energy consumption characteristics are scaled between 0.0 and 1.0 with respect to the non-optimized values. All tasks are scheduled under global Earliest Deadline First (EDF) scheduling algorithm. Note that our proposed scheme is generic in the sense that it can work with other global scheduling algorithms as well.

Table I
VOLTAGE-FREQUENCY LEVELS OF PXA270 PROCESSOR

Parameter	Level1	Level2	Level3	Level4	Level5	Level6
Voltage	1.55	1.45	1.35	1.25	1.15	0.90
Frequency	624	520	416	312	128	104
Active Power	925	675	468	301	208	52
Idle Power	260	222	186	154	129	064

A. Simulation Results

1) Effect of variations in bcet/wcet ratio: Simulation settings for this scenario are presented in table-III. We have varied bcet/wcet ratio between 50% and 100% of WCET of tasks such that AET of all tasks has a uniform probability distribution function as suggested in [12].

Table II
POWER-EFFICIENT STATES OF PXA270 PROCESSOR

States	Power(mWatts)	Recovery Time(ms)
Running	925	0
Idle	260	0.001
Standby	1.722	11.43
Sleep	0.163	136.65
Deep sleep	0.101	261.77

We make the following observations on these results. For bcet/wcet $ratio = 1$, Fig.2 depicts no change in total energy consumption due to constant dynamic and static power consumption. Since AET remains constant, energy consumed by processors under non-optimized case and under DVFS expert alone remains unchanged. DPM expert alone, however, saves energy by exploiting the presence of inherent (static) idle intervals. HyPowMan scheme, in this case, converges to DPM expert in a straightforward manner as shown in Fig.2. As bcet/wcet ratio decreases (< 1), opportunities for DVFS expert to save energy are created as well. Fig.2 shows that both DVFS and DPM experts, while working alone, save energy as compared to non-optimized case. For bcet/wcet ratio between 0.5 and 0.9, it can be observed that HyPowMan scheme converges to the best energy savings offered by either expert. This convergence validates our earlier claims that under HyPowMan scheme, (more or less) best possible energy savings can be achieved. Fig.2 shows that under HyPowMan scheme, processors consume slightly more energy than the one offered by best-performing expert alone. This is due to the convergence mechanism in which, initially, experts are frequently substituted amongst them in an attempt to figure out the best-performing expert. We have measured energy savings up to 23.12% for DVFS expert alone, up to 47.94% for DPM expert alone, and up to 47.22% for HyPowMan scheme by interplaying DVFS and DPM experts.

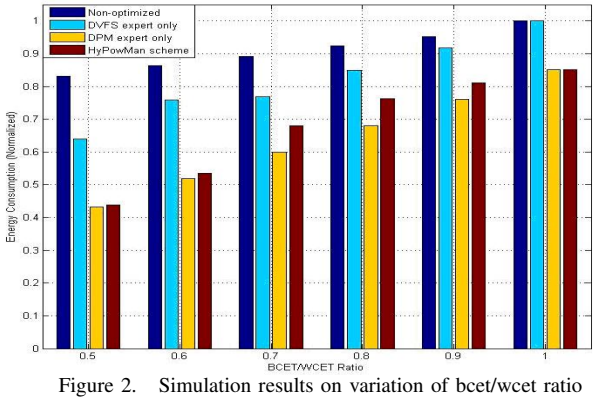

Figure 2. Simulation results on variation of bcet/wcet ratio

2) Effect of variations in number of tasks: Simulation settings for this scenario are presented in table-IV. We have performed simulations by doubling and tripling the number of tasks. Results in Fig.3 depict that, increasing the number of tasks increases the energy savings in all cases (up to 18.05% for DVFS expert alone, 10.23% for DPM expert alone, and 24.71% for HyPowMan scheme). Based on simulation results, we make two very interesting observations. Firstly, in case when both experts are applied as stand-alone techniques, DVFS expert saves more energy than DPM expert does. This is because the main determinant of variations in energy consumption is actual workload and with increased number of tasks, there are potentially more opportunities for tasks to generate dynamic slack and therefore, more possibilities for DVFS expert to reclaim energy. Secondly, HyPowMan scheme can even result in energy savings more than any stand-alone technique in some cases. This is because, over entire simulation time, a single expert cannot always change a processor's power consumption profile (often due to transition costs). HyPowMan scheme, on the other hand, substitutes an expert with the other if it is not performing well under such conditions and eventually results is better energy savings.

Table III
SIMULATION SETTINGS FOR VARIABLE BCET/WCET RATIO

Parameters	Settings
Number of processors(m) in platform	4
Number of tasks (n) in task set	8
Utilization (u_{sum}) of task set	2.50
α for DPM expert	0.80
α for DVFS expert	0.90
β for DPM expert	0.70
β for DVFS expert	0.90
bcet/wcet ratio	$50\% - 100\%$ of WCET

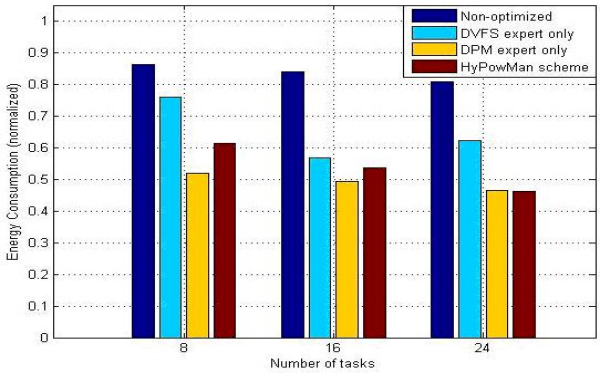

Figure 3. Simulation results on variation in number of tasks

Table IV
SIMULATION SETTINGS FOR VARIABLE NUMBER OF TASKS

Parameters	Settings
Number of processors(m) in platform	4
Number of tasks (n) in task set	8, 16, & 24
Utilization (u_{sum}) of task set	2.50
α for DPM expert	0.80
α for DVFS expert	0.90
β for DPM expert	0.70
β for DVFS expert	0.90
bcet/wcet ratio	60% of WCET

3) Effect of Variations in total utilization: Simulation settings for this scenario are presented in table-V. Multiple task sets with total utilization varying between 50% (lower workload) and 100% (maximum workload) of platform capacity have been generated. Results in Fig.4 depict that the difference in energy savings for a given utilization is more or less the same in all cases. That is, the variations in cummu-

lative utilization do not significantly vary the performance of these techniques and lesser workload naturally favors more energy savings on fixed capacity platforms. Simulation results indicate an average energy savings compared to non-optimized case by up to 22.8% for DVFS expert alone, 41.6% for DVFS expert alone, and 48.2% for HyPowMan scheme for varying utilization values.

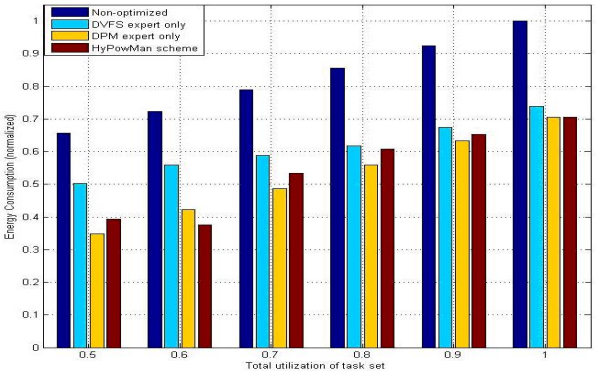

Figure 4. Simulation results on variation in cumulative utilization

Table V
SIMULATION SETTINGS FOR VARIABLE CUMULATIVE UTILIZATION

Parameters	Settings
Number of processors(m) in platform	4
Number of tasks (n) in task set	8
Utilization (u_{sum}) of task set	$60\% - 100\%$
α for DPM expert	0.80
α for DVFS expert	0.90
β for DPM expert	0.70
β for DVFS expert	0.90
bcet/wcet ratio	60% of WCET

4) Effect of Variations in α (Low, Medium, & High): Simulation settings for this scenario are presented in table-VI. Recall from Section 4.1 that value of α indicates the desirable settings of the importance of energy savings compared to the performance degradation. A high value of α indicates a higher preference to energy savings, a low value indicates higher preference to performance while a medium value indicates a reasonable ratio of both. In our experiments, we have varied the value of α ranging from 0.6 (low) to 0.9 (high). We used values of α around 0.7 and 0.75 for the medium values. Results in Fig.5 show that, as the value of α increases, the convergence of HyPowMan scheme is refined with respect to energy savings, i.e. the gains achieved on energy become closer to that of best-performing individual expert. For lower values of α, the relative importance of energy savings is reduced and as a result, HyPowMan scheme converges to the individual expert offering lesser performance degradation. Note that we have limited the value of α between 0.6 and 0.9. For even higher values (close to 1), energy savings closer to ideal policy can be achieved.

5) Effect of Variations in β (Low, Medium, & High): Simulation settings for this scenario are presented in table-VII. Recall from Section 4.1 that the value of β determines

Table VI
SIMULATION SETTINGS FOR VARIABLE α

Parameters	Settings
Number of processors(m) in platform	4
Number of tasks (n) in task set	8
Utilization (u_{sum}) of task set	2.50
α	$0.60 - 0.90$
β for DPM expert	0.70
β for DVFS expert	0.90
bcet/wcet ratio	60% of WCET

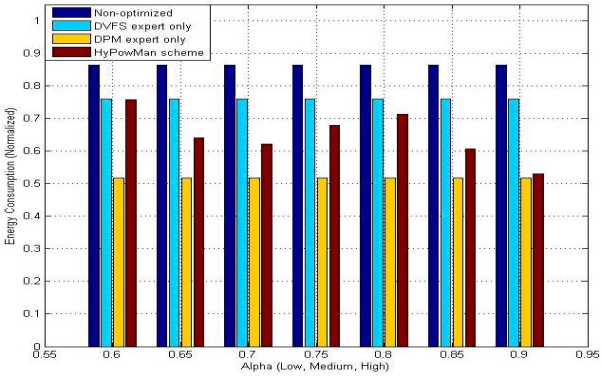

Figure 5. Simulation results on variation in α

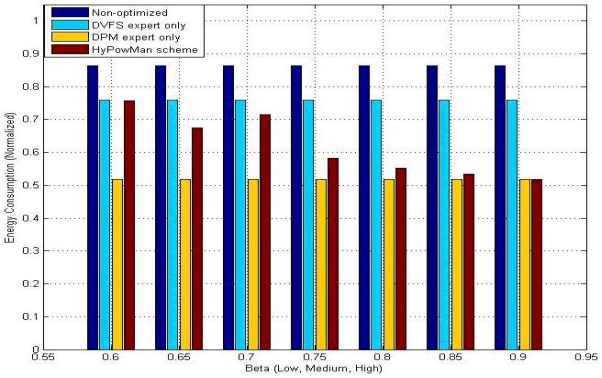

Figure 6. Simulation results on variation in β

Table VII
SIMULATION SETTINGS FOR VARIABLE β

Parameters	Settings
Number of processors(m) in platform	4
Number of tasks (n) in task set	8
Utilization (u_{sum}) of task set	2.50
α for DPM expert	0.80
α for DVFS expert	0.90
β	$0.60 - 0.90$
bcet/wcet ratio	60% of WCET

the granularity of weight factors, i.e. the higher the value of β is set, the lower the variation in weight of each expert occurs for a given input. From results in Fig.6, we observe that as the value of β increases, the granularity of weight update (and hence the probability factor) associated with individual experts with respect to their performance is refined. This refinement in weight update permits HyPowMan scheme to precisely and more frequently evaluate the performance of working expert as well as dormant experts which eventually leads to a better convergence to best-performing ex-

pert. Conversely, when the value of β decreases, HyPowMan scheme has lesser precision in weight updates and therefore, frequently substitutes the working expert, which eventually leads to lesser gains on energy.

VI. Conclusion

In this paper, we have proposed a novel and generic scheme for power management in multiprocessor systems. HyPowMan scheme, instead of designing new power management policies (whether DPM or DVFS) for specific operating conditions, takes a set of well-known existing policies, each of which performs well for a given set of conditions, and adapts at runtime to the best-performing technique for any given workload. A machine-learning algorithm is implemented to evaluate the performance of all techniques and the decision to select best-performing technique at any point in time is taken online and adaptive to the varying workload. Experiments validate that our proposed scheme adapts well to the changing workload and quickly converges to the best-performing technique within the selected policy set by a margin of 1.5% to 11% on energy savings. Results also show that HyPowMan scheme is robust to changing operating conditions. Our proposed scheme enhances the ability of portable embedded systems to adapt with changing workload and work with a larger set of operating conditions to give an overall performance and energy savings that is better than any single policy can offer. HyPowMan evaluates the performance of all experts (working as well as dormant) at every input which may increase the computational overhead if larger expert sets are used. Hence, this aspect requires improvements for an efficient implementation on hardware platforms.

References

[1] A. KARLIN, M. MANESSE, L. MCGEOCH, and S. OWICKI, "Competitive randomized algorithms for nonuniform problems," in *procs. of Algorithmica'99*, 1994.

[2] L. BENINI, A. BOGLIOLO, and G. D. MICHELI, "A survey of design techniques for system-level dynamic power management," in *IEEE Trans. on VLSI Sys.*, 2000.

[3] L. BENINI, A. BOGLIOLO, G. PALEOLOGO, and G. DEMICHELI, "Policy optimization for dynamic power management," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits & Systems*, 1999, pp. 813–833.

[4] Y. LU and G. DE MICHELI, "Adaptive hard disk power management on personal computers," in *IEEE Procs. of Great Lakes Symp. on VLSI*, 1999.

[5] D. RAMANATHAN, S. IRANI, and R. K. GUPTA, "Latency effects of system level power management algorithms," in *Procs. of IEEE Int'l Conf. on CAD*, 2000.

[6] S. IRANI, S. SHUKLA, and R. GUPTA, "Online strategies for dynamic power management in systems with multiple power-saving states," in *ACM Trans. on Emb. Comp. Sys.*, 2003.

[7] L. BENINI and G. DE MICHELI, *Dynamic Power Management: Design Techniques and CAD Tools*. Springer, 1997.

[8] P. YANG, C. WONG, P. MARCHAL, F. CATTHOOR, D. DESMET, D. VERKEST, and R. LAUWEREINS, "Energy-aware runtime scheduling for embedded-multiprocessor socs," in *procs. of IEEE Design and Test of Comp.*, 2001.

[9] N. NAVET and B. GAUJAL, *Ordonnancement temps reel et minimisation de la consommation de energie*, ser. Chapter-4 in System Temps Reel -Volume 2. Kluwer Publishers, 2006.

[10] M. K. BHATTI, C. BELLEUDY, M. FAROOQ, and M. AUGUIN, "Assertive dynamic power management strategy for globally scheduled rt multiprocessor systems," in *procs. of PATMOS'09*, 2009.

[11] "XScale Microarchitecture," http://www.marvell.com/.

[12] H. AYDIN, R. MELHEM, D. MOSSE, and P. MEJIA-ALVAREZ, "Power-aware scheduling for periodic real-time tasks," in *procs. of IEEE Trans. on Comp.*, 2004.

[13] G. DHIMAN and T. S. ROSING, "Dynamic power management using machine learning," in *procs. of ICCAD'06*, 2006.

[14] Q. QIU and M. PEDRAM, "Dynamic power management based on continuous-time markov decision processes," in *procs. of DAC-99*, 1999, pp. 555–561.

[15] C.-H. HWANG, C. ALLEN, and H. WU, "A predictive system shutdown method for energy saving of event-driven computation," in *procs. of ACM Int'l Conf. on CAD*, 1996.

[16] G. A. PALEOLOGO, L. BENINI, A. BOGLIOLO, and G. D. MICHELI, "Policy optimization for dynamic power management," in *procs. of DAC-98*, 1998, pp. 182–187.

[17] Y. SHIN, K. CHOI, and T. SAKURAI, "Power optimization of real-time embedded systems on variable speed processors," in *procs. of ICCAD'00*, 2000, pp. 365–368.

[18] P. PILLAI and K. SHIN, "Real-time dynamic voltage scaling for low-power embedded operating systems," in *ACM Sym. on OS Principles*, 2001.

[19] D. ZHU, N. ABOUGHAZALEH, D. MOSSE, and R. MELHEM, "Power aware scheduling for and/or graphs in multiprocessor real time systems," in *procs. of ICPP'02*, 2002.

[20] H. AYDIN and Q. YANG, "Energy-aware partitioning for multiprocessor real-time systems," in *procs. of IPDPS'03*, 2003.

[21] M. K. BHATTI, C. BELLEUDY, and M. AUGUIN, "An inter-task real time dvfs scheme for multiprocessor embedded systems," in *procs. of DASIP-2010*, 2010.

[22] H. CHENG and S. GODDARD, "Sys-edf: a system-wide energy-efficient scheduling algorithm for hard real-time systems," in *Int'l Journal of Embedded Systems, Vol.4*, 2009.

[23] V. DEVADAS and A. H., "On the interlay of dynamic voltage scaling and dynamic power management in real-time embedded applications," in *procs. of EMSOFT'08*, 2008.

[24] "ARM," http://www.arm.com/.

[25] "STORM," http://storm.rts-software.org.

[26] T. A. AlEnawy and H. Aydin, "Energy-aware task allocation for rate monotonic scheduling," in *procs. of IEEE Symposium, RTAS-2005*, 2005, pp. 213–223.