Power Optimization of Variable Voltage Core-Based Systems

Inki Hong†, Darko Kirovski†, Gang Qu†, Miodrag Potkonjak†, and Mani B. Srivastava‡
†Computer Science Department, University of California, Los Angeles, CA 90095-1596 USA
‡Electrical Engineering Department, University of California, Los Angeles, CA 90095-1596 USA

Abstract

The growing class of portable systems, such as personal computing and communication devices, has resulted in a new set of system design requirements, mainly characterized by dominant importance of power minimization and design reuse. We develop the design methodology for the low power core-based real-time system-on-chip based on dynamically variable voltage hardware. The key challenge is to develop effective scheduling techniques that treat voltage as a variable to be determined, in addition to the conventional task scheduling and allocation. Our synthesis technique also addresses the selection of the processor core and the determination of the instruction and data cache size and configuration so as to fully exploit dynamically variable voltage hardware, which result in significantly lower power consumption for a set of target applications than existing techniques. The highlight of the proposed approach is the non-preemptive scheduling heuristic which results in solutions very close to optimal ones for many test cases. The effectiveness of the approach is demonstrated on a variety of modern industrial-strength multimedia and communication applications.

1 Introduction

1.1 Motivation

The growing class of portable systems, such as personal computing and communication devices, demands data- and computation-intensive functionalities with low power consumption. For such systems, power consumption is the primary design goal since the battery life is a primary constraint on power, due to the fact that the battery technology has not followed the progress pace of the semiconductor industry. Recent advances in power supply technology along with custom and commercial CMOS chips that are capable of operating reliably over a range of supply voltages make it possible to create processor cores with supply voltage that can be varied at run time according to application timing constraints. The variable voltage processor core can be made to operate at different optimal points along its power vs. speed curve in order to achieve much higher energy efficiency than existing techniques for a wider class of applications. Such systems also require design flexibility which result in the need for implementation on programmable processor platform. In fact, embedded software running on RISC and DSP processor cores has emerged as a leading implementation methodology for such applications as speech coding, modem functionality, video compression and communication protocol processing [15]. Current semiconductor technology allows the integration of programmable processors and memory structures on a single die, which enables the implementation of a system on a single chip. Similarly, the exponential growth of both applications and implementation technology has outpaced the design productivity of the traditional synthesis process. The shrinked time-to-market window has exacerbated the situation. There is a wide consensus that only through reuse of highly optimized cores the demands of the pending applications and the potential ultra large scale integration may be matched. Therefore, low power core-based system-on-chip, consisting of a variable voltage programmable processor core and a memory hierarchy, attracted much attention of virtually all silicon vendors.

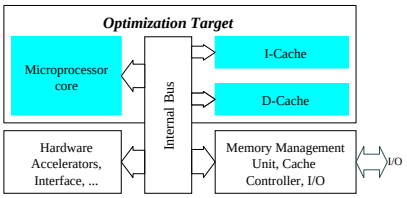

Figure 1: A typical core-based application-specific system-on-chip.

Our synthesis technique targets typical modern application specific system-on-chip, consisting of a variable voltage processor core, instruction and data cache, and a number of optional hardware accelerators and control blocks as depicted in Figure 1. The distribution of power dissipation by the components of application-specific system-on-chip depends on the actual applications running on the system. However, extensive studies indicate that the power consumption of the processor and cache cores accounts for significant portion of the overall power consumption of the described system-on-chip [6]. Therefore, in this paper we focus on the power optimization of the processor and cache cores.

The most effective way to reduce power consumption of a processor core in CMOS technology is to lower the supply voltage level, which exploits the quadratic dependence of power on voltage [2]. Reducing the supply voltage however increases circuit delay and decreases clock speed. The resulting processor core consumes lower average power while meeting the deadlines. This technique is ineffective when tight deadlines are present in systems. Another power optimization technique for processor cores is the system shutdown [15]. The system shutdown technique, though usable even in the presence of tight deadlines, is inferior to the supply voltage reduction technique for the cases when both techniques can be applied. The limitations of the techniques arise due to the fact that systems are designed with a fixed supply voltage. The supply voltage reduction technique attempts to find a single optimal voltage level for the entire processor operation, while the system shutdown technique makes a binary runtime decision whether to turn on or off the power supply.

The goal of the research presented in the paper is to develop the design methodology for the low power core-based real-time system-on-chip based on dynamically variable voltage hardware. The key challenge is to develop effective scheduling techniques that treat voltage as a variable to be determined, in addition to the conventional task scheduling and allocation. Our synthesis technique also addresses the selection of the processor core and the determination of the instruction and data cache size and configuration so as to fully exploit dynamically variable voltage hardware, which will

1-58113-049-x-98/0006/$3.50

result in significantly lower power consumption for a set of target applications than existing techniques.

1.2 Motivational Example

To illustrate the key point of the proposed dynamically variable voltage approach, we consider a set of tasks as a motivational example, shown in Table 1. Two independent computations $Task_A$ and $Task_B$ need to be executed on an embedded processor core in the time interval [0, 20]. Each task can be executed immediately after its arrival and is required to be finished by its deadline time. Preemption is not allowed due to the high context-switching cost.

task	arrival	deadline	execution time at 3.3 V
A	0	6	5
B	3	20	5

Table 1: The characteristics of the 2 tasks used to illustrate the motivation for dynamically variable voltage approach.

Assume the maximum supply voltage to be $(V_{dd})_{ref} = 3.3$ volts. Power is normalized to its value at the reference point, i.e., $P(3.3\ volts) = 1$ Watt. Reducing supply voltage results in increased circuit delay and to a good accuracy, the circuit delay is given by $k \times \frac{V_{dd}}{(V_{dd}-V_t)^2}$, where V_t is the threshold voltage, and k is a constant [2]. We assume a typical value of $0.8\ volts$ for the threshold voltage. The power consumption is given by $P = \alpha C_L V_{dd}^2 f$, where f is the system clock frequency, V_{dd} is the supply voltage, C_L is the load capacitance and α is the switching activity [2]. We now consider the application of shutdown, supply voltage reduction, both shutdown and supply voltage reduction, and dynamically variable voltage approach, respectively.

With the shutdown technique, the system will operate at $V_{dd} = 3.3$ volts. The $Task_A$ is executed in the interval [0, 5]. The $Task_B$ is executed in the interval [5,10]. The processor can be shut down for the interval [10, 20] and then be resumed for the next task. The duty cycle of the processor is 50 %, so the average power consumption is 0.5 Watts.

With the supply voltage reduction technique, the system will operate at a lower but fixed supply voltage. The tight deadline on $Task_A$ means that supply voltage can not be lowered less than $V_{dd} = 2.97$ volts. Thus, the system operates at $V_{dd} = 2.97$ volts with $P(2.97\ volts) = 0.67$ Watts. $Task_A$ is executed in the interval [0, 6]. $Task_B$ is executed in the interval [6,12]. The average power consumption is 0.67 Watts. Since the system can be shut down during the interval [12, 20], the average power consumption can be lowered to $0.67 \times \frac{12}{20} = 0.40$ Watts, which results in 20 % power reduction, compared to the shutdown technique.

With the variable voltage hardware, one can schedule the two tasks such that the $Task_A$ is executed in the interval [0, 5] at 2.97 volts with $P(2.97\ volts) = 0.67$ Watts and the $Task_B$ is executed in the interval [6,20] at 1.95 volts with $P(1.95\ volts) = 0.11$ Watts. The average power consumption is $\frac{0.67 \times 6 + 0.11 \times 14}{20} = 0.28$ Watts, which is 44 % lower than the shutdown technique and 30 % lower than the supply voltage reduction in combination with the shutdown technique.

The preceding example illustrates that it is always better to reduce voltage than to shutdown, and that dynamically variable supply voltage helps unleash this potential.

1.3 What is New?

In this paper, we develop the first approach for power minimization of variable voltage core-based application specific systems. The power minimization is addressed at several synthesis tasks including task scheduling, instruction and data cache size determination, and processor core selection. We explore static scheduling algorithms that treat voltage as an optimization degree of freedom for the applications with real-time constraints. By selecting the most efficient voltage profile in the presence of multiple timing constraints, our algorithms result in much larger savings in energy than other techniques such as the system shutdown and the supply voltage reduction.

1.4 Paper Organization

The rest of the paper is organized in the following way. Section 2 presents the related work. Section 3 explains the necessary background material on variable voltage systems. The design flow of the novel synthesis approach is presented in Section 4. In Section 5 the optimization problems are recognized and their computational complexities are established. In the same section the synthesis algorithms are proposed and explained in detail. Section 6 presents experimental data and discussion to evaluate the effectiveness of the approach. The paper is concluded in Section 7.

2 Related Work

We review the research results relevant to low power systems based on dynamically variable voltage hardware.

Supply voltage reduction coupled with architecture level parallelism and pipelining to compensate for lower clock rate due to voltage reduction [2] works well for applications such as signal processing where throughput is the sole metric of speed. Several researchers have addressed the issue of power in event-driven systems, and proposed various techniques for shutting down the system or parts of the system [8, 15].

At the technology level, efficient DC-DC converters that allow the output voltage to be rapidly changed under external control have recently been developed [13]. At the hardware design level, research work has been performed on chips with dynamically variable supply voltage that can be adjusted based on (i) process and temperature variations, and (ii) processing load as measured by the number of data samples queued in the input (or output) buffer [1].

There has been research on task-level scheduling strategies for adjusting CPU speed so as to reduce power consumption. Most of the existing work is in the context of non-real-time workstation-like environment [7, 17]. Yao et al. [19] described an minimum energy schedule for scheduling with **preemption** for independent processes with deadlines.

A number of research groups have addressed the use of multiple (in their software implementation restricted to two or three) different voltages [3, 9, 12, 14]. Interestingly, although they used the term "variable voltage", they actually addressed scheduling when a fixed number of simultaneously available voltages are used. Therefore, there is no similarity between the proposed research and these efforts beyond accidental syntax similarity.

3 Preliminaries

In this Section we describe the task model and hardware and power models for processor cores, caches and variable voltage hardware.

3.1 Computational and Timing Models

A set J of independent tasks is to be executed on a system-on-chip. Each task $j \in J$ is associated with the following parameters: a_j its arrival time, d_j its deadline, p_j its period, and r_j its execution time at the nominal(maximum possible) voltage level V_{ref}. Without loss of generality we assume that all tasks have identical periods. Since context switching overhead is usually high for modern systems, we assume no task preemption.

3.2 Power Model

It is well known that there are three principal components of power consumption in CMOS integrated circuits: switching power, short-circuit power, and leakage power. In CMOS technology, switching power dominates power consumption. It is also known that reduced voltage operation comes at the cost of reduced throughput [2]. The maximum rate at which a circuit is clocked monotonically decreases as the voltage is reduced. From these equations together with the observation that the speed is proportional to f

and inversely proportional to the gate delay, the power vs. speed curve can be derived. By varying V_{dd}, the system can be made to operate at different points along this curve. Due to the convexity of the power vs. speed function, it is always better to run the computation at a constant speed if there is a single computation that needs to be executed by some deadline. In the proposed scheduling heuristic, we use this observation so that each task is executed at a single voltage.

Processor core	Clock (MHz)	MIPS	Technology (μm)	Area (mm^2)	Power diss. (mW)
ARM7 LPower	27	24	0.6	3.8	45 (3.3V)
LSI TR4101	81	30	0.35	2	81 (3.3V)
LSI CW4001	60	53	0.5	3.5	120 (3.3V)
LSI CW4011	80	120	0.5	7	280 (3.3V)
Motorola 68000	33	16	0.5	4.4	35 (3.3V)
PowerPC 403	33	41	0.5	7.5	40 (3.3V)

Table 2: The performance, area, and power data for a subset of processor cores

3.3 Architecture and Hardware Model

Several factors combine to influence system performance: instruction and data cache miss rates and penalty, processor performance, and system clock speed. Power dissipation of the system is estimated using processor power dissipation per instruction and the number of executed instructions per task, supply voltage, energy required for cache read, write, and off-chip data access as well as the profiling information about the number of cache and off-chip accesses. The approach that we use to realistically address the factors leverages on existing cache and processor models.

Data on microprocessor cores have been extracted from manufacturer's datasheets as well as from the CPU Center Info web site [4]. A sample of the collected data is presented in Table 2.

We use CACTI [18] as a cache delay estimation tool with respect to the main cache design choices: size, associativity, and line size. The energy model for cache is adopted from [16]. We use a recently developed compiler strategy which efficiently minimizes the number of cache conflicts that considers direct-mapped caches [10]. We consider only direct-mapped caches based on the results of our experimentation that 2-way set associative caches do not dominate comparable direct-mapped caches in a single case. We estimated the cache miss penalty based on the operating frequency of the system and external bus width and clock for each system investigated. This penalty ranged between 4 and 20 system clock cycles. Write-back is used instead of write through due to its superior performance and power savings. We consider caches with single access port. The nominal energy consumption per single off-chip memory access, $98nJ$, is assumed [11]. A subset of the cache model data is given in 3.

	No optimizations			Block buffering, sub-banking and Gray code addressing [16]		
Cache size	8B	16B	32B	8B	16B	32B
512B	0.3301	0.3777	0.4683	0.2913	0.3219	0.4012
1KB	0.3561	0.3943	0.4626	0.2953	0.2993	0.3452
2KB	0.4223	0.4442	0.4894	0.3171	0.2706	0.2710
4KB	0.6513	0.6662	0.6942	0.4557	0.3336	0.2729
8KB	1.146	1.15.6	1.175	0.7686	0.5044	0.3474
16KB	2.158	2.164	2.174	1.412	0.8693	0.5298
32KB	4.198	4.202	4.209	2.702	1.608	0.9223

Table 3: A subset of the cache power consumption model: power consumption (nJ) estimation for various direct-mapped caches with variable line sizes at 5V

3.4 Variable Voltage Hardware

The variable voltage is generated by the DC-DC switching regulators in the power supply which are essentially a feedback control system. The time for the voltage to reach a steady state at the new voltage is a strong function of the feedback loop parameters and the LC output filter in the switching regulator. [13] reported efficient DC-DC switching regulators with fast transition times. DC-DC converters with faster transition times may be obtained with the cost of increased power dissipation within the converter. Dual to the supply voltage variation is the accompanying variation of the clock frequency. The clock frequency also takes time to stabilize at the new value. The time and power overhead associated with voltage switching is negligible [13] and not considered in the analysis.

3.5 Summary of Theoretical Results

The *non-preemptive* scheduling of a set of independent tasks with arbitrary arrival times and deadlines on a *fixed voltage* processor is an NP-complete task [5]. Therefore, the *non-preemptive* scheduling of a set of independent tasks with arbitrary arrival times and deadlines on a *variable voltage* processor is also an NP-complete task. Yao, Demers and Shenker [19] have provided the optimal *preemptive* static scheduling algorithm for a set of independent tasks with arbitrary arrival times and deadlines. This solution serves as a lower bound for *non-preemptive* scheduling solutions. This lower bound plays a crucial role in speeding up our branch and bound algorithm for the *non-preemptive* scheduling problem.

4 Design Methodology for Variable Voltage Systems: Global Design Flow

In this Section we describe the global flow of the proposed synthesis system and explain the function of each subtask and how these subtasks are combined into a synthesis system.

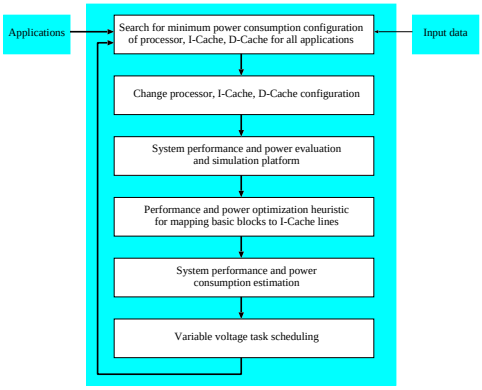

Figure 2: The global flow of the synthesis approach.

Sort processor cores in a list L in an increasing order of $\frac{EnergyPerInstruction}{SystemClockFrequency}$ at the nominal voltage 5V.
Delete the dominated processor cores from L
For each processor core in L in the order of appearance
  For I-cache = 512B..32KB and CacheLineSize = 8B..512B
    For D-cache = 512B..32KB and CacheLineSize = 8B..512B
      Check bounds; if exceeded **break**;
      If (current I- and D-cache configuration has never been evaluated)
        Evaluate performance and power consumption of the cache structure
      Using the cache system analysis, evaluate the system power consumption using variable voltage task scheduling
      Memorize Configuration C if power consumption is minimal

Figure 3: Pseudo code for the resource allocation procedure.

Figure 2 illustrates the synthesis system. The goal is to choose the configuration of processor, I-cache, and D-cache and the vari-

able voltage task schedule with minimum power consumption which satisfy the requirements of multiple non-preemptive tasks. To accurately predict the system performance and power consumption for target applications, we employ the approach which integrates the optimization, simulation, modeling, and profiling tools. The synthesis technique considers each non-dominated microprocessor core and competitive cache configuration, and selects the hardware setup which requires minimal power consumption and satisfies the individual performance requirements of all target applications. The application-driven search for a low-power core and cache system requires usage of trace-driven cache simulation for each promising point considered in the design space. We attack this problem by carefully scanning the design space using search algorithms with sharp bounds and by providing powerful algorithmic performance and power estimation techniques. We use the system performance and power evaluation and simulation platform based on SHADE, DINEROIII and a custom analyser [10, 11].

5 Synthesis Algorithms

The optimization problems encountered in the synthesis of a low power variable voltage system on a chip and competitive optimization algorithms are described in this Section.

5.1 Resource Allocation

In this phase of the synthesis approach, a search is conducted to find an energy-efficient system configuration. The search algorithm is described using the pseudo-code shown in Figure 3. Since performance and power evaluation of a single processor, I- and D-cache configuration requires a trace-driven simulation, the goal of our search technique is to reduce the number of evaluated cache systems using sharp bounds for cache system performance and power estimations. However, a particular cache system is evaluated using trace-driven simulation only once since the data retrieved from such simulation can be used for overall system power consumption estimation for different embedded processor cores with minor additional computational expenses.

The algorithm excludes from further consideration processor cores dominated by other processor cores. One processor type dominates another if it consumes less power at higher frequency and results in higher MIPS performance at the same nominal power supply. The competitive processors are then sorted in ascending order with respect to their power consumption per instruction and frequency ratio. Microprocessors which seem to be more power-efficient are, therefore, given priority in the search process. Later on this step provides sharper bounds for search termination.

The search for the most efficient cache configuration is bounded with sharp bounds. A bound is determined by measuring the number of conflict misses and comparing the energy required to fetch the data from off-chip memory due to measured conflict misses and the power that would have been consumed by twice larger cache for the same number of cache accesses assuming zero cache conflicts. We terminate further increase of the cache structure when the power consumption due to larger cache would be larger than the energy consumed by the current best solution. Similarly, another bound is defined at the point when the energy required to fetch the data from off-chip memory due to conflict cache misses for twice smaller cache with the assumption of zero-energy consumption per cache access, is larger than the energy required for both fetching data from cache and off-chip memory in the case of the current cache structure. We abort further decrease of the cache structure if the amount of energy required to bring the data due to additional cache misses from off-chip memory is larger than the energy consumed by the current best solution.

When evaluating competitive hardware configurations, the target applications are scheduled with the variable voltage task scheduler using the predicted system performance and power on the configuration considered. Before the non-preemptive task scheduling is performed, more efficient optimal preemptive scheduler is applied to get a lower bound. If the result is worse than the current best solution, the configuration is worse than the current best one.

5.2 Task Scheduling

As described in Section 3, the non-preemptive scheduling of a set of independent tasks with arbitrary arrival times and deadlines on a *variable voltage* processor is an NP-complete task. We have developed an efficient and effective heuristic for the general variable voltage task scheduling problem, which leverages least-constraining most-constrained heuristic paradigm in order to obtain competitive solutions. The algorithm is described using the pseudo-code shown in Figure 4.

The algorithm consists of two phases. In the first phase all tasks are scheduled at the nominal voltage. If a feasible schedule is found in the first phase, in the second phase the supply voltages are adjusted for low-power. These two phases are repeated $LOOPS$ times and the best solution after the iterations is chosen as the final solution. Each iteration of the two phases generates different solutions since the objective functions used in the first phase to guide the search strategy are randomized by a random offset.

In the first phase, the task scheduling at the nominal voltage is done in the following way. Initially, the time period in which the tasks appear is divided into time regions such that each time region has its start or end time equal to a start or end time of some task and there does not exist a task which has either an arrival or deadline time within the time region. Then, for each time region, an objective function $OBJ(TimeRegion)$ is computed as:

$$OBJ(TimeRegiontr) = maxVoltage(tr) \cdot aveVoltage(tr)$$

where the functions $maxVoltage(tr)$ and $aveVoltage(tr)$ assume that all tasks are scheduled from their arrival until their deadline regardless of time overlaps and return the maximum and average voltage for all tasks alive at time region tr, respectively. The objective functions (or constraints) of time regions are used to compute the constraints of tasks. For each task an objective function $OBJ(Task)$ is computed as

$$OBJ(Taskt) = \sum_{tr \in [a_t, d_t]} OBJ(tr) \cdot \frac{r_t^2}{(d_t - a_t)^2}.$$

The most-constrained task t, which does not include in its life-time another tasks' life-time, is then scheduled in the interval equal to its run-time at the nominal voltage which spans over a subset of time regions with the lowest sum of objective functions among any feasible subset of time regions. When comparing the subset of time regions, we only consider the cases that the start time of the task is only at the start time of each time region in the task's life-time or the end time of the task is only at the end time of each time region. Upon scheduling t, other tasks' deadlines and arrival times are updated in the following way. If the scheduled task's life-time is included in the life-time of some other task s and the task s can be scheduled at both partitions of its life-time upon deleting the time regions that contains t, then both partitions of the task's life-time are evaluated for scheduling. Objective functions are computed for all the time-frames with its length equal to the task's nominal voltage run-time such that the start time of the time frame is only at the start time of each time region or the end time of the time frame is only at the end time of each time region, shown in the lower part of Figure 5. The partition which includes the time frame with the minimal sum of objective functions of containing time regions is selected. Otherwise, each remaining task's arrival time and deadline is updated if its lifetime intersects with the lifetime of the task t. This process is repeated until all tasks are scheduled at the system nominal voltage.

We explain the process using the example shown in Figure 5. Six tasks T[0], T[1], T[2], T[3], T[4] and T[5] are shown with their arrival, deadline, and nominal voltage run-times. The set of tasks

Given: a set of tasks T, where each task $t \in T$ is characterized by its arrival time a_t, its deadline d_t, and its execution time r_t at 5V

Repeat $LOOPS$ times

Scheduling tasks at the nominal voltage.

Repeat

Create(TR) >> Create a set TR of time regions $tr_i[start_i, end_i]$ where $start_i$ and end_i correspond to an arrival or deadline time of any two not necessarily distinct tasks and there does not exist another task t_o which has $start_i < a_{t_o}, d_{t_o} < end_i$.

For each $tr_i \in TR$

Compute its objective function $OBJ(TimeRegion\ tr_i)$

For each task $t_i \in T$

Compute $OBJ(Task\ t_i)$

Select the task $t \in T$ such that it has the maximum $OBJ(t)$ and there does not exist another task s such that $a_s > t_a$ and $d_s < d_t$

Find the sequence of SEQ time regions $[tr_1, tr_{SEQ}]$ such that $a_t \leq start_{tr_1}$ and $d_t \geq end_{tr_{SEQ}}$ and $[tr_1, tr_{SEQ}]$ has the smallest $\sum_{i=1}^{SEQ} OBJ(tr_i)$

If $end_{tr_{SEQ}} - start_{tr_1} = r_t$

Schedule t from $start_{tr_1}$ to $end_{tr_{SEQ}}$

Else If $OBJ(tr_1) > OBJ(tr_{SEQ})$

Schedule t from $OBJ(tr_{SEQ}) - r_t$ to $OBJ(tr_{SEQ})$

Else Schedule t from $OBJ(tr_1)$ to $OBJ(tr_1) + r_t$

Delete t from list of tasks T

Create(TR)

For each $tr_i \in TR$

Compute its objective function $OBJ(TimeRegion\ tr_i)$

For each task $s \in T$

If scheduling of task t splits the life-time of task s into two time portions p_1 and p_2 both larger than the nominal voltage run-time

For each portion p_i find the sequence of time regions $[tr_1, tr_{SEQ_i}]$ which comprises time period greater or equal to r_s and has minimal $S = \sum_{i=1}^{SEQ_i} OBJ(tr_i)$

Assign the task to the portion which has greater S

Else update the task arrival and deadline times

until all tasks are scheduled.

Voltage adjusting of scheduled tasks for low-power

For each task t

Compute the earliest start st_t and latest finish fi_t times according to the existing scheduling order.

Compute the minimal voltage v_t at which the task can run if expanded from st_t to fi_t

Repeat

For each task t

If there exists task $s \in T$ such that $fi_s = st_t$ and $v_s < v_t$ and $d_s > a_t$

Schedule s from st_s to $max(st_s + r_s(v_t) + \frac{st_t - st_s - r_s(v_t)}{2}, a_t)$

Schedule t from $max(st_s + r_s(v_t) + \frac{st_t - st_s - r_s(v_t)}{2}, a_t)$ to d_t

If there exists task $s \in T$ such that $fi_t = st_s$ and $v_s < v_t$ and $a_s < d_t$

Schedule t from st_t to $min(d_s - r_s(v_t) - \frac{d_s - r_s(v_t) - fi_t}{2}, d_t)$

Schedule s from $min(d_s - r_s(v_t) - \frac{d_s - r_s(v_t) - fi_t}{2}, d_t)$ to d_s

until no further energy consumption improvements

Compute system energy E. **If** $E < BestE$, $BestE = E$

Return (BestE)

Figure 4: Pseudo code for the variable voltage task scheduling.

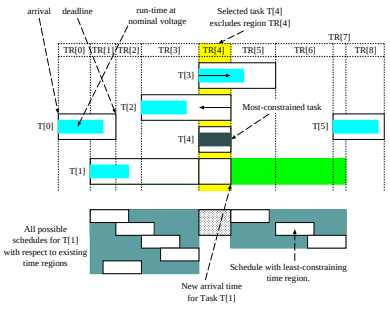

Figure 5: Scheduling tasks at nominal voltage.

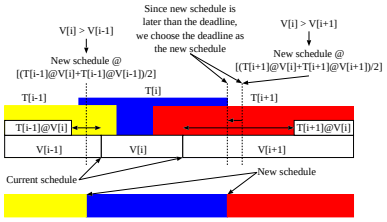

Figure 6: Adjusting the voltage of scheduled tasks for low-power.

creates a set of nine time regions TR[0] through TR[8]. Based on objective functions computed, task T[4] is selected as the most-constrained. Once this task is scheduled the arrival times and deadlines of tasks T[2] and T[3] are updated as shown (see arrows in Figure 5). However, task [0] can be scheduled at both TR[1, 2, 3] and TR[5, 6, 7]. There are four possible schedules with respect to the existing time regions at TR[1, 2, 3] and three (actually five, but only three distinct) at TR[5, 6, 7] as shown in the bottom part of Figure 5. The least-constraining schedule (marked in Figure 5) is in the TR[5, 6, 7] partition since during this time region there are no alive tasks. Therefore, the life-time of task T[1] is reduced to TR[5, 6, 7].

Once tasks are scheduled, the procedure which tunes the supply voltage of each task is invoked. It iteratively traverses the set of tasks and tries to lower the voltage of each task until no energy dissipation improvement occurs. Whenever a task t has a neighboring task s with a life-time that intersects the life time of t and supply voltage v_s lower than v_t, the border between the tasks is moved according to Figure 6. For example, in Figure 6, task T[i] has been assigned a supply voltage higher than task T[i-1]. The new arrival time of T[i] and deadline time of T[i-1] is moved to the arithmetic mean of the current arrival/deadline time and the arrival/deadline time as if T[i-1] is supplied with T[i]'s voltage V[i]. For a fixed task scheduling this voltage adjustment algorithm results in solutions arbitrarily close to the optimal solution.

6 Experimental Results

We used nine public domain applications to demonstrate the effectiveness of the approach. JPEG software from the Independent JPEG Group implements JPEG baseline, extended-sequential, and progressive compression processes. GSM software, obtained from Technische Universität at Berlin, is an implementation of the European GSM 06.10 provisional standard for full-rate speech transcoding, prI-ETS 300 036, which uses RPE/LTP (residual pulse excitation/long term prediction) coding at 13 kbit/s. EPIC (Efficient Pyramid Image Coder), obtained from University of Pennsylvania, implements lossy image compression and decompression utilities. The remaining benchmarks (Mipmap, Osdemo, and Texgen), obtained from University of Wisconsin, use a 3-D graphic library called Mesa. Mipmap is a simple mipmap texture mapping example. Osdemo is a demo program that draws a set of simple polygons using the Mesa 3-D rendering pipe. Finally, Texgen renders a texture mapped version of the Utah teapot.

In order to evaluate the efficiency of our algorithmic approach for variable voltage system synthesis, we conducted a set of experiments. We experimented our variable voltage scheduling algorithm by scheduling sets of applications with their time constraints on a number of configurations and comparing the obtained results with a lower bound obtained using Yao's preemptive scheduling algorithm with assumed zero preemption cost. The results are presented in Table 4. The low-power processor-cache application-mix champion configurations are described in the first five columns of Table 4. The configurations are assumed to have operational dynamically variable [0.8, 3.3] V supply voltage. The next four pairs of columns present the power consumption lower bound and the result obtained

Core	I-Cache		D-cache		Energy (J) 10 Tasks		15 Tasks	
	Cache	Block	Cache	Block	LB	New	LB	New
M68000	1KB	128B	2KB	128B	2.46	2.50	2.61	2.66
ARM7LP	1KB	128B	2KB	128B	2.42	2.47	2.50	**2.54**
TR4101	512B	64B	4KB	32B	2.43	2.50	2.52	2.58
CW4001	512B	64B	1KB	128B	2.39	**2.44**	2.49	**2.54**
PC403	1KB	32B	2KB	32B	2.48	2.55	2.75	2.78
CW4011	1KB	64B	2KB	64B	2.50	2.59	2.72	2.77
Average					*2.44*	*2.51*	*2.58*	*2.65*
Core	I-cache		D-cache		20 Tasks		25 Tasks	
	Cache	Block	Cache	Block	LB	New	LB	New
M68000	1KB	128B	2KB	128B	11.1	11.9	27.3	**27.1**
ARM7LP	1KB	128B	2KB	128B	12.3	13.1	29.9	31.8
TR4101	512B	64B	4KB	32B	10.6	**11.0**	27.3	27.9
CW4001	512B	64B	1KB	128B	13.8	15.0	35.0	37.4
PC403	1KB	32B	2KB	32B	14.1	14.4	37.2	38.7
CW4011	1KB	64B	2KB	64B	14.8	15.4	39.6	40.7
Average					*12.8*	*13.4*	*32.7*	*33.9*
Core	I-cache		D-cache		35 Tasks		50 Tasks	
	Cache	Block	Cache	Block	LB	New	LB	New
M68000	1KB	128B	2KB	128B	59.9	64.1	141.7	142.0
ARM7LP	1KB	128B	2KB	128B	64.4	64.5	139.9	140.2
TR4101	512B	64B	4KB	32B	63.7	63.9	127.3	127.9
CW4001	512B	64B	1KB	128B	69.4	73.7	135	136.3
PC403	1KB	32B	2KB	32B	53.0	**53.4**	109.7	110.7
CW4011	1KB	64B	2KB	64B	55.2	55.4	100.2	**100.6**
Average					*61.0*	*62.5*	*125.6*	*126.3*

Table 4: Results on a set of architectures and application mixes: LB - Lower Bound, New - Our Approach.

using our algorithm for four different sets of applications extracted from the nine applications.

Our non-preeptive scheduling algorithm resulted in solutions which were only **1.47%** in average higher than the lower bound determined using Yao's preemptive scheduling algorithm over series of more than 10^4 schedulings. An average standard deviation of voltages per schedule is **0.61V** over the same set of scheduling actions. The average maximal voltage per scheduling was $3.12V$, while the average minimal voltage was $1.08V$. The results also indicate the importance of effective allocation for more complex systems such that while for the 10-task mix the best configuration is only 3% better than the average, our allocation technique finds for the case of 50-task mix a configuration with 25% power savings.

7 Conclusion

We developed the design methodology for the low power core-based real-time system-on-chip based on dynamically variable voltage hardware. The key contribution was to develop effective scheduling techniques that treat voltage as a variable to be determined, in addition to the conventional task scheduling and allocation. Our synthesis technique also addressed the selection of the processor core and the determination of the instruction and data cache size and configuration so as to fully exploit dynamically variable voltage hardware, which resulted in significantly lower power consumption for a set of target applications than existing techniques. The highlight of the proposed approach was the non-preemptive scheduling heuristic which resulted in solutions very close to optimal ones for many test cases. The effectiveness of the approach was demonstrated on a variety of modern industrial-strength multimedia and communication applications, such as MPEG and JPEG encoders and decoders.

REFERENCES

[1] A. Chandrakasan, V. Gutnik, and T. Xanthopoulos. Data driven signal processing: an approach for energy efficient computing. In *International Symposium on Low Power Electronics and Design*, pages 344–352, 1996.

[2] A. P. Chandrakasan, S. Sheng, and R.W. Broderson. Low-power CMOS digital design. *IEEE Journal of Solid-State Circuits*, 27(4):473–484, 1992.

[3] J.-M. Chang and M. Pedram. Energy minimization using multiple supply voltages. In *International Symposium on Low Power Electronics and Design*, pages 157–162, 1996.

[4] http://infopad.eecs.berkeley.edu/CIC/.

[5] M.R. Garey and D.S. Johnson. *Computer and Intractability: A Guide to the theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, 1979.

[6] R. Gonzalez and M. Horowitz. Energy dissipation in general purpose microprocessors. *IEEE Journal of Solid-State Circuits*, 31(9):1277–1284, 1996.

[7] K. Govil, E. Chan, and H. Wasserman. Comparing algorithms for dynamic speed-setting of a low-power CPU. In *ACM International Conference on Mobile Computing and Networking*, pages 13–25, 1995.

[8] C. Hwang and A.C.-H. Wu. A predictive system shutdown method for energy saving of event-driven computation. In *IEEE/ACM International Conference on Computer-Aided Design*, pages 28–32, 1997.

[9] M. C. Johnson and K. Roy. Datapath scheduling with multiple supply voltages and level converters. *ACM Transactions on Design Automation of Electronic Systems*, 2(3), 1997.

[10] D. Kirovski, C. Lee, W. Mangione-Smith, and M. Potkonjak. Application-driven synthesis of core-based systems. In *IEEE/ACM International Conference on Computer-Aided Design*, pages 104–107, 1997.

[11] D. Kirovski, C. Lee, W. Mangione-Smith, and M. Potkonjak. Synthesis of power efficient systems-on-silicon. In *Asia and South Pacific Design Automation Conference*, 1998.

[12] Y.-R. Lin, C.-T. Hwang, and A.C.-H Wu. Scheduling techniques for variable voltage low power designs. *ACM Transactions on design Automation of Electronic Systems*, 2(2):81–97, 1997.

[13] W. Namgoong, M. Yu, and T. Meng. A high-efficiency variable-voltage CMOS dynamic DC-DC switching regulator. In *IEEE International Solid-State Circuits Conference*, pages 380–381, 1997.

[14] S. Raje and M. Sarrafzadeh. Variable voltage scheduling. In *International Symposium on Low Power Design*, pages 9–14, 1995.

[15] M. Srivastava, A. P. Chandrakasan, and R. W. Brodersen. Predictive system shutdown and other architectural techniques for energy efficient programmable computation. *IEEE Transactions on VLSI Systems*, 4(1):42–55, 1996.

[16] C.-L. Su and A.M. Despain. Cache design trade-offs for power and performance optimization: a case study. In *International Symposium on Low Power Design*, pages 63–68, 1995.

[17] M. Weiser, B. Welch, A. Demers, and S. Shenker. Scheduling for reduced CPU energy. In *USENIX Symposium on Operating Systems Design and Implementation*, pages 13–23, 1994.

[18] S.J.E. Wilton and N.P. Jouppi. CACTI: an enhanced cache access and cycle time model. *IEEE Journal of Solid-State Circuits*, 31(5):677–688, 1996.

[19] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced CPU energy. In *IEEE Annual Foundations of Computer Science*, pages 374–382, 1995.