Synthesis Techniques for Low-Power Hard Real-Time Systems on Variable Voltage Processors

Inki Hong†, 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 energy efficiency of systems-on-a-chip can be much improved if one were to vary the supply voltage dynamically at run time. In this paper we describe the synthesis of systems-on-a-chip based on core processors, while treating voltage (and correspondingly, the clock frequency) as a variable to be scheduled along with the computation tasks during the static scheduling step. In addition to describing the complete synthesis design flow for these variable voltage systems, we focus on the problem of doing the voltage scheduling while taking into account the inherent limitation on the rates at which the voltage and clock frequency can be changed by the power supply controllers and clock generators. Taking these limits on rate of change into account is crucial since changing the voltage by even a volt may take time equivalent to 100s to 10,000s of instructions on modern processors. We present both an exact but impractical formulation of this scheduling problem as a set of non-linear equations, as well as a heuristic approach based on reduction to an optimally solvable restricted ordered scheduling problem. Using various task mixes drawn from a set of nine real-life applications, our results show that we are able to reduce power consumption to within 7% of the lower bound obtained by imposing no limit at the rate of change of voltage and clock frequencies.

1 Introduction

In recent years the demand for portable battery-operated computing and communication devices has made low power consumption an essential design attribute. The power reduction approaches that have emerged so far include reduction of switched capacitance, activity based system shutdown, and aggressive supply voltage reduction via exploitation of quadratic dependence of power on voltage together with parallelism and pipelining to recoup lost throughput. However, aggressive supply voltage reduction [1],the most powerful of these techniques, is usable only if throughput (data sample rate) is the sole metric of speed. A single tight latency constraint, as is often present in embedded systems, renders the technique ineffective.

The problem outlined above really arises because conventional systems are designed with a fixed supply voltage. However, there is no fundamental reason that the supply voltage has to be fixed. Instead, it can in principle be varied dynamically at run time. Indeed, advances in power supply technology makes it possible to vary the generated supply voltage dynamically under external control. While many CMOS circuits have always been capable of operating over a range of supply voltages, it is the recent progress in power supply circuits [21, 9] that has made feasible systems with dynamically variable supply voltages. Since both the power consumed and the speed (maximal clock frequency) are a function of the supply voltage, such *variable voltage* systems can be made to operate at different points along their power *vs.* speed curves in a controlled fashion.

In particular a static or dynamic scheduler, in addition to its conventional task of scheduling the computation operations on hardware resources, may also schedule changes in voltage as a function of timing constraints and changing system state. Such voltage scheduling would allow for much higher energy efficiency (lower power) for a wider class of applications than is possible by operating the system at one or two fixed points on the power-speed curve, as is done by conventional approaches of supply voltage reduction to a fixed value [1] and system shutdown [25]. The benefits of quadratic dependence of power on voltage thus become available even in event driven systems as well as in the presence of joint latency and throughput constraints

In this paper, we develop the first approach for power minimization of scheduling, instruction and data cache size determination, and processor core selection. We establish the theoretical framework for designing variable voltage systems by treating voltage as an optimization degree of freedom for the applications with real-time constraints and providing the optimal variable voltage scheduling algorithm for some special cases, where the speed overhead for changing voltages is explicitly accounted for. We develop an effective scheduling heuristic for a general case based on the

optimal algorithm for the restricted problem. By selecting the most efficient voltage profile in the presence of multiple timing constraints under the realistic variable voltage hardware model, our algorithms result in significant savings in energy.

The rest of the paper is organized in the following way. Section 2 presents the related work. Section 3 explains the necessary background. Section 4 discusses the variable voltage scheduling problem and provides an optimal solution to some special cases. The global design flow of the novel synthesis approach is presented in Section 5. In Section 6 we establish the computational complexities of the variable voltage scheduling problem and propose an effective heuristic based on the optimal algorithm for some special cases. Section 7 presents experimental data. Section 8 concludes.

2 Related work

Low power system synthesis has emerged as an important area of research in last 5 years. Several good reviews on power estimation and minimization techniques exist [20, 23]. We review the research results relevant to low power systems based on dynamically variable voltage hardware.

2.1 Variable voltage system and design issues

Of direct relevance to our research is the prior research activity on technology, circuits, and techniques for variable voltage systems. At the technology level, efficient DC-DC converters that allow the output voltage to be rapidly changed under external control have recently been developed [21, 26].

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. Dynamically adapting voltage (and the clock frequency), to operate at the point of lowest power consumption for given temperature and process parameters was first suggested by Macken et. al. [19]. Nielsen et al. [22] extended the dynamic voltage adaptation idea to take into account data dependent computation times in self-timed circuits. Recently, researchers at MIT [9] have extended the idea of voltage adaptation based on data dependent computation time from [22] to synchronously clocked circuits. Because these approaches rely on run-time reactive approaches to dynamic voltage adaptation, they work fine only where *average throughput*. is the metric of performance. Therefore, these approaches are inapplicable to hard real time systems such as embedded control applications with strict timing requirements.

There has been research on scheduling strategies for adjusting CPU speed so as to reduce power consumption. Most existing work is in the context of non-real-time workstation-like environment. Weiser et al. [29] proposed an approach where time is divided into 10-50 msec intervals, and the CPU clock speed (and voltage) is adjusted by the task-level scheduler based on the processor utilization over the preceding interval. Govil et al. [8] enhanced [29] by proposing and comparing several predictive and non-predictive approaches for voltage changes, and concluded that smoothing helps more than prediction. Finally, Yao et al. [31] described an off-line minimum energy scheduling algorithm and an on-line average rate heuristic for job scheduling with preemption for independent processes with deadlines under the assumption that the supply voltage can be changed arbitrarily without any timing and power overhead.

Hong et al. [10] proposed an approach for the low power core-based real-time system-on-chip based on dynamically variable voltage hardware. While they developed the *non-preemptive* variable voltage scheduling heuristic with the assumption of zero delay in changing voltage levels, in this paper we focus on the *preemptive* variable voltage scheduling while taking into account the inherent limitation on the rates at which voltage and clock frequency can be changed by the power supply controllers and clock generators.

2.2 System level and behavioral level power estimation

Landman and Rabaey have performed the work on the macro and micro level power estimation [16]. The work at Princeton and Fujitsu on the estimation of power consumption for programmable processors [28] is of particular relevance. Evans and Franzon [5] and Su and Despain [27] have proposed power estimation model for cache, specifically for SRAM.

2.3 System level and behavioral level power optimization

One well known technique that results in substantial power reduction is reducing the voltage to reduce the switching power, which is the dominant source of power dissipation in CMOS circuit and is proportional to V_{dd}^2 where V_{dd} is the supply voltage [1]. However, this reduction came with a speed penalty due to increased gate delays. The gate delay is proportional to $\frac{V_{dd}}{(V_{dd}-V_T)^2}$, where V_T is the threshold voltage [1]. Chandrakasan et al. [1] have shown that the strategy of operating at a fixed reduced voltage can be coupled with architectural level parallelism and pipelining to compensate for lower clock rate due to voltage reduction so that the overall throughput remains the same but the overall power is still lowered, although at the cost of increased latency.

Microprocessor core	Clock (MHz)	MIPS	Technology (μm)	Area (mm^2)	Power diss. (mW) (Volt.)
StrongARM	233	266	0.35	4.3	300 (1.65)
ARM, 7	40	36	0.6	5.9	200 (5)
ARM, 7 Low-Power	27	24	0.6	3.8	45 (3.3)
LSI Logic, TR4101	81	30	0.35	2	81 (3.3)
LSI Logic, CW4001	60	53	0.5	3.5	120 (3.3)
LSI Logic, CW4011	80	120	0.5	7	280 (3.3)
DSP Group, Oak	80	80	0.6	8.4	190 (5)
NEC, R4100	40	40	0.35	5.4	120 (3.3)
Toshiba, R3900	50	50	0.6	15	400 (3.3)
Motorola, 68000	33	16	0.5	4.4	35 (3.3)
PowerPC, 403	33	41	0.5	7.5	40 (3.3)
ARM 710 (ARM7 / 8KB)	40	72	0.6	34	424 (5)
SA-110 (StrongARM / 32KB)	200	230	0.35	50	900 (1.65)

Table 1. The performance, area, and power data for a subset of processor cores.

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 [25, 12, 4]. Compilation techniques for low power software have emerged for both general-purpose computation [28] and DSP computations [11]. Numerous behavioral synthesis research efforts have also addressed power minimization [23].

Four research groups have addressed the use of multiple (in their software implementation restricted to two or three) different voltages [2, 13, 18, 24]. They used the term "variable voltage" for a fixed number of simultaneously available voltages.

3 Preliminaries

3.1 Task model

A set $\mathcal{T}$ of independent tasks is to be executed on a system-on-chip. Each task $T_i \in \mathcal{T}$ is associated with the following parameters:

- a_i its arrival time
- d_i its deadline
- p_i its period
- W_i its required number of CPU cycles

We assume, without loss of generality, that all tasks have identical periods. When this is not the case, a simple preprocessing step and application of the least common multiple (LCM) theorem [17] transforms an arbitrary set of periods to this design scenario in polynomial time. As the consequence, when the LCM theorem is applied, there may be a need to run several iterations of a given task within this overall period.

3.2 Variable voltage hardware model

The variable voltage generated by the DC-DC switching regulators in the power supply cannot instantaneously make a transition from one voltage to another. For example, [21] reported transition times of 6 msec/volt for a DC-DC switching converter. In personal communications with the authors, researchers at MIT and Berkeley have reported transitions times of 10 to 100 microsec/volt.

Dual to the supply voltage variation is the accompanying variation of the clock frequency. The time overhead associated with voltage switching is significant in the order of 100s to 10000s of instructions in modern microprocessor. Fortunately, the computation itself can continue during the voltage and frequency change period. However, the speed overhead during the transition period must be explicitly accounted for.

As suggested in [21], we employ a linear model to describe the speed overhead. The maximum rate of voltage change is specified for the power supply regulator, and we can make a transition from a voltage to any other voltages within the maximum rate.

3.3 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. The switching power, which dominates power consumption, is given by $P = \alpha C_L V_{dd}^2 f_{clock}$, as indicated earlier. αC_L is defined to be effective switched capacitance. It is also known that reduced voltage operation comes at the cost of reduced throughput [1]. The gate delay T follows the following formula: $T = k\frac{V_{dd}}{(V_{dd}-V_T)^2}$ where k is a constant [1] and V_T is the threshold voltage. 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. In particular, the normalized power P_n (= 1 at $V_{dd} = V_{ref}$) as a function of the normalized speed S_n (= 1 at $V_{dd} = V_{ref}$). For example, assuming V_{ref} = 3.3 volts and V_T = 0.8 volts, the power *vs.* speed curve follows the following equation:

$$P_n = 0.164 \cdot S_n^3 + \sqrt{0.893 \cdot S_n^2 + 1.512 \cdot S_n} \cdot (0.173 \cdot S_n^2 + 0.147 \cdot S_n) + 0.277 \cdot S_n^2 + 0.059 \cdot S_n$$

By varying V_{dd}, the system can be made to operate at different points along this curve. From the equation, it is easy to see that the power is a convex function of speed.

3.4 Target architecture

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.

Data on microprocessor cores have been extracted from manufacturer's datasheets as well as from the CPU Center Info web site [3]. A sample of the collected data is presented in Table 1, The last two rows of the table show two integrated microprocessor products with on-chip caches and their system performance data. We assume that the processor cores follow the power model described in the preceding subsection.

Cache size	Line size						
	8B	16B	32B	64B	128B	256B	512B
512B	6.07	5.99	6.02	-	-	-	-
1KB	6.44	6.13	6.07	6.23	-	-	-
2KB	6.88	6.52	6.36	6.34	6.51	-	-
4KB	7.67	7.02	6.66	6.49	6.56	7.35	-
8KB	8.34	7.81	7.21	6.99	6.91	7.65	9.40
16KB	9.30	8.58	8.00	7.62	7.54	8.14	9.80
32KB	1.04e-08	9.45	8.91	8.59	8.69	9.30	10.04

Table 2. A subset of the cache latency model: minimal cycle time (ns) for various direct-mapped caches with variable line sizes.

We use CACTI [30] as a cache delay estimation tool with respect to the main cache design choices: size, associativity, and line size. The energy model has been adopted from [5] and [27]. The overall cache model has been scaled with respect to actual industrial implementations. Caches typically found in current embedded systems range from 128B to 32KB. Although larger caches correspond to higher hit rates, their power consumption is proportionally higher, resulting in an interesting design trade-off. Higher cache associativity results in significantly higher access time. We use a recently developed compiler strategy which efficiently minimizes the number of cache conflicts that considers direct-mapped caches [14]. We have experimented with 2-way set associative caches which did not dominate comparable direct-mapped caches in a single case. Cache line size was also variable in our experimentations. Its variations corresponded to the following trade-off: larger line size results in higher cache miss penalty delay and higher power consumption by the sense amplifiers and column decoders, while smaller line size results in large cache decoder power consumption. Extreme values result in significantly increased access time. 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 was adopted in opposed to write-through, since it is proven

	No optimizations			Block buffering, sub-banking and Gray code addressing [27]		
Cache size	8B	16B	32B	8B	16B	32B
512B	0.330	0.378	0.468	0.291	0.322	0.401
1KB	0.356	0.394	0.463	0.295	0.299	0.345
2KB	0.422	0.444	0.489	0.317	0.271	0.271
4KB	0.651	0.666	0.694	0.456	0.334	0.273
8KB	1.146	1.156	1.175	0.769	0.504	0.347
16KB	2.158	2.164	2.174	1.412	0.869	0.530
32KB	4.198	4.202	4.209	2.702	1.608	0.922

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

to provide superior performance and especially power savings in uniprocessor systems at increased hardware cost [15]. Each of the processors considered is able to issue at most a single instruction per clock period. Thus, caches were designed to have a single access port. A subset of the cache model data is given in Tables 2 and 3. Cache access delay and power consumption model were computed for a number of organizations and sizes, assuming the feature size of 0.5 μm and typical six transistors per CMOS SRAM cell implementation. The nominal energy consumption per single off-chip memory access, $98nJ$, is adopted from [6].

4 Theory of variable voltage scheduling: special solvable cases

Yao, Demers and Shenker [31] have provided the optimal preemptive static scheduling algorithm for a set of independent tasks with arbitrary arrival times and deadlines. The algorithm is based on the concept of critical regions and the tasks in critical regions are scheduled by the earliest deadline first (EDF) algorithm. The running time of the algorithm is $O(nlog^2 n)$ for n tasks. When there is a limit on the maximum voltage change rates, then the problem becomes NP-complete, which is shown by the reduction from the SEQUENCING WITH DEADLINES AND SETUP TIMES problem in page 238 of [7]. In this Section, we discuss some special cases which can be solved optimally.

We find the speed function $S(t)$ of the processor to minimize the energy consumption during the time period $[t_1, t_2]$. The voltage function $V(t)$ can be easily obtained from $S(t)$ [1]. We assume that the speed of the processor can be changed between S_{min} and S_{max} with a maximum change rate K, i.e., at any time t, the speed $S(t)$ satisfies:

$$S_{min} \leq S(t) \leq S_{max} \quad (1)$$

$$|S'(t)| \leq K \quad (2)$$

The *amount of work* that the processor completes during time interval $[t_1, t_2]$ is given by:

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

The *power*, or energy consumed per unit time, is a convex function $P(S)$ of the processor's speed. The function

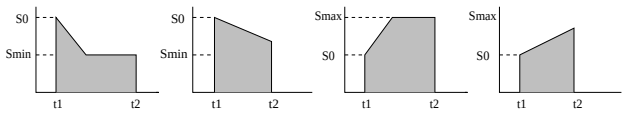
P(S) depends on the technology considered. The total energy consumed during the time interval $[t_1, t_2]$ is:

$$E = \int_{t_1}^{t_2} P(S(t))\, dt \quad (4)$$

Lemma 1. Starting with speed S_0, the work W that the processor can complete, in the time interval $[t_1, t_2]$, falls into the range $[W_{min}, W_{max}]$, with

$$W_{min} = S_c(t_2 - t_1) + \frac{(S_0 - S_c)^2}{2K}$$
$$W_{max} = S_d(t_2 - t_1) - \frac{(S_0 - S_d)^2}{2K}$$

where $S_c = max(S_{min}, S_0 - K(t_2 - t_1))$ and
$S_d = min(S_{max}, S_0 + K(t_2 - t_1))$.



(a) (b) (c) (d)

Figure 1. Illustrated W_{min}, W_{max} for Lemma 1.
(a) W_{min} if $S_0 - K(t_2 - t_1) < S_{min}$
(b) W_{min} if $S_0 - K(t_2 - t_1) \geq S_{min}$
(c) W_{max} if $S_0 + K(t_2 - t_1) \geq S_{max}$
(d) W_{max} if $S_0 + K(t_2 - t_1) < S_{max}$

Since power is a convex function of the processor's speed, it is obvious that if we can start from any speed, to complete a given workload W in any time interval of fixed length Δt, we should run our processor at the constant speed $\frac{W}{\Delta t}$. When we do not have the freedom to choose the starting speed, the following theorem shows that the best we can do is to change the speed as fast and early as possible until we reach a critical point, then keep the constant speed.

Theorem 1. With the same constraints as in Lemma 1, for any workload $W \in [W_{min}, W_{max}]$, there is a unique speed function $S : [t_1, t_2] \rightarrow [S_{min}, S_{max}]$ such that (1), (2), and (3) are all satisfied and the energy (4) is minimized. Moreover, $S(t)$ is defined as:

if $W_{min} \leq W \leq S_0(t_2 - t_1)$,

$$S(t) = \begin{cases} S_0 - K(t - t_1), & \text{if } t_1 \leq t \leq t_c; \\ S_0 - K(t_c - t_1), & \text{if } t_c < t \leq t_2. \end{cases}$$

where $t_c = t_2 - \sqrt{(t_2 - t_1)^2 + \frac{2}{K}[W - S_0(t_2 - t_1)]}$.

and if $S_0(t_2 - t_1) \leq W \leq W_{max}$,

$$S(t) = \begin{cases} S_0 + K(t - t_1), & \text{if } t_1 \leq t \leq t_c; \\ S_0 + K(t_c - t_1), & \text{if } t_c < t \leq t_2. \end{cases}$$

where $t_c = t_2 - \sqrt{(t_2 - t_1)^2 - \frac{2}{K}[W - S_0(t_2 - t_1)]}$.

Next we consider the case when there is a finishing speed constraint. Similarly, we have the following results:

Lemma 2. Starting with speed S_0, to reach the finishing speed S_1 at the end of the time interval $[t_1, t_2]$, the work W that the processor has to complete falls into the range $[W_{min}, W_{max}]$, with

$$W_{min} = S_c(t_2 - t_1) + \frac{(S_0 - S_c)^2}{2K} + \frac{(S_1 - S_c)^2}{2K}$$
$$W_{max} = S_d(t_2 - t_1) - \frac{(S_0 - S_d)^2}{2K} - \frac{(S_1 - S_d)^2}{2K}$$

where $S_c = max(S_{min}, \frac{(S_0 + S_1)}{2} - K\frac{(t_2 - t_1)}{2})$ and
$S_d = min(S_{max}, \frac{(S_0 + S_1)}{2} + K\frac{(t_2 - t_1)}{2})$.

Theorem 2. With the same constraints as in Lemma 2, for any workload $W \in [W_{min}, W_{max}]$, there is a unique speed function $S : [t_1, t_2] \rightarrow [S_{min}, S_{max}]$ such that (1), (2), and (3) are satisfied and the energy (4) is minimized. The speed function will be a step function that satisfies the following:

if $W_{min} \leq W \leq W_1$,

$$S(t) = \begin{cases} S_0 - K(t - t_1), & \text{if } t_1 \leq t \leq x_1; \\ S_0 - K(x_1 - t_1), & \text{if } x_1 < t \leq x_2; \\ S_1 - K(t_2 - t), & \text{if } x_2 < t \leq t_2. \end{cases}$$

if $W_1 \leq W \leq W_2$,

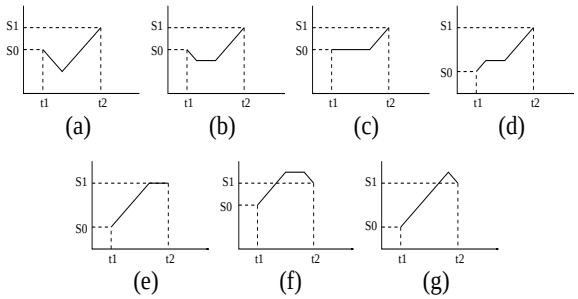
$$S(t) = \begin{cases} S_0 + K(t - t_1), & \text{if } t_1 \leq t \leq x_1; \\ S_0 + K(x_1 - t_1), & \text{if } x_1 < t \leq x_2; \\ S_1 - K(t_2 - t), & \text{if } x_2 < t \leq t_2. \end{cases}$$

and if $W_2 \leq W \leq W_{max}$,

$$S(t) = \begin{cases} S_0 + K(t - t_1), & \text{if } t_1 \leq t \leq x_1; \\ S_0 + K(x_1 - t_1), & \text{if } x_1 < t \leq x_2; \\ S_1 + K(t_2 - t), & \text{if } x_2 < t \leq t_2. \end{cases}$$

where W_1, W_2, x_1, x_2 are constants.

From Theorem 2, it is simple to see that the speed functions which minimize energy are of some restricted shapes. For example, if $S_1 > S_0$, then $S(t)$ has to be one of the following seven shapes in Figure 2.



(a) (b) (c) (d) (e) (f) (g)

Figure 2. All possible shapes of an optimal speed function when $S_1 > S_0$.

We consider the following scheduling problem:
Ordered Scheduling Problem (OSP): Given a partition $0 = t_0 < t_1 < t_2 < ... < t_{n-1} < t_n = t$ of the time interval $[0, t]$, there is a workload W_i to be completed during the interval $[t_{i-1}, t_i]$, we want to find the speed function $S : [0, t] \rightarrow [S_{min}, S_{max}]$, such that:

(i) $S(0) = S_0$,
(ii) $S_{min} \leq S(t) \leq S_{max}$,
(ii) $|S'(t)| \leq K$,
(iv) $W_i = \int_{t_{i-1}}^{t_i} S(t)\, dt$, and
(v) $E = \int_0^t P(S(t))\, dt$ is minimized.

For this problem, we can use the following approach. Let $S(t)$ be a speed function that we are looking for and $x_i = S(t_i)$ for $i = 1, 2, \cdots, n$. Using the Theorem 2,

we can express $S(t)$ as a function of unknown variables $x_1, x_2, \cdots, x_n$. Now we can plug this expression into (v), get a system of (non-linear) equations involving x_i's from the first order condition to minimize the energy function (v). The system of (non-linear) equations is solved by using some numerical method, if the closed form solution can not be found. The speed function $S(t)$ is determined by the formulae in Theorem 2 and the solution(s) of the system of (non-linear) equations.

There are a few comments to be made about this approach. First, this approach is impractical since the (non-linear) system of equations is very difficult to be solved in many cases. Secondly, when applying Theorem 2 in this approach, we have to check another condition: $W_{min}(x_{i-1}, x_i) \leq W_i \leq W_{max}(x_{i-1}, x_i)$, which will make the problem even more difficult. Thirdly, since the solution(s) to the (non-linear) system of equations is the potential candidate for our minimization problem, we have to apply (at least) the second order condition to check it. Finally, there are still some more mathematical details that we have to take into consideration, e.g., the continuity and differentiability of the power function. Therefore, some good heuristics will be more useful in real life.

We note that the following special case of the OSP problem can be solved optimally.

Restricted Ordered Scheduling Problem (ROSP): With the same constraints as in the OSP problem, $S(t_i)$ for $i = 1, 2, \cdots, n$ are additionally given.

It is trivial to see that the Theorem 2 provides an optimal speed function for the ROSP problem.

5 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.

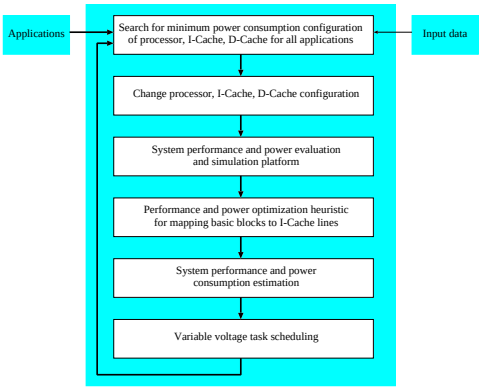


Figure 3. The global flow of the synthesis approach.

Figure 3 illustrates the synthesis system. The goal is to choose the configuration of processor, I-cache, and D-cache and the variable voltage task schedule with minimum power consumption which satisfy the timing requirements of multiple 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, as shown in Figure 5. 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. The search algorithm to find an energy-efficient system configuration is described using the pseudo-code shown in Figure 4.

```
Sort processor cores in a list L in an increasing order of
  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
      Using the cache system analysis, evaluate the power consumption
        of the entire system using variable voltage task scheduling
      Memorize Configuration C if power consumption is minimal
```

Figure 4. Pseudo code for the search of power efficient system configuration.

Since performance and power evaluation of a single processor, I- and D-cache configuration requires a time-consuming 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. This step provides later on 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.

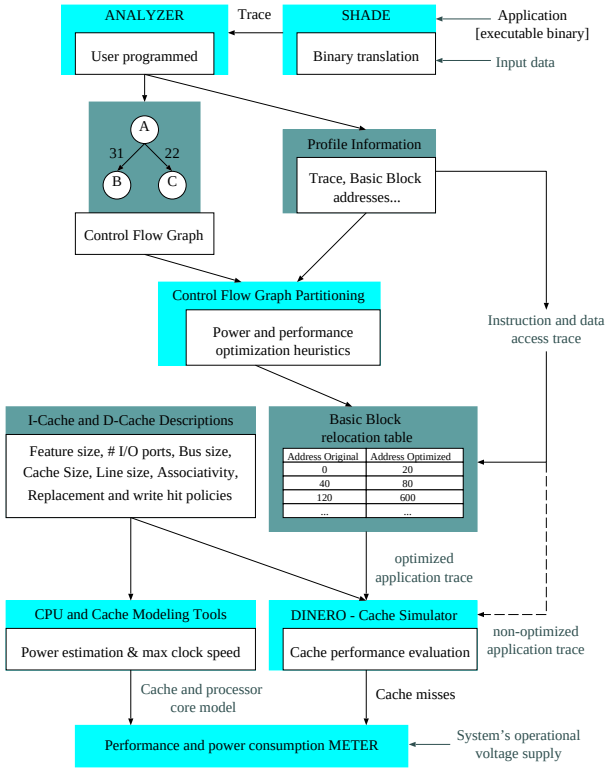


Figure 5. The system performance and power evaluation and simulation platform.

To estimate the system performance and power on the configuration under consideration, we use the system performance and power evaluation and simulation platform based on SHADE and DINEROIII [14, 15]. SHADE is a tracing tool which allows users to define custom trace analyzers and thus collect rich information on runtime events. SHADE currently profiles only SPARC binaries. The executable binary program is dynamically translated into host machine code. The tool also provides a stream of data to the translated code which is directly executed to simulate and trace the original application code. A custom analyzer composed of approximately 2,500 lines of C code, is linked to SHADE to control and analyze the generated trace information. The analyzer sources relevant trace information from SHADE and builds a control flow graph (CFG) corresponding to the dynamically executed code. The analysis consists of two passes. The first pass determines the boundaries of basic blocks, while the second pass constructs a CFG by adding control flow information between basic blocks. We also collect the frequencies of control transfers through each basic block, and branch temporal correlation. Once the CFG is obtained, an algorithm is employed to reposition application basic blocks in such a way that instruction cache misses and cache decoder switching activity are minimized. Our experimentation uses a basic block relocation look up table to simulate the relocation of basic blocks in main memory. An entry in the basic block relocation table consists of two elements: the original and optimized starting address of the basic block. To simulate cache performance of a given application and data stream, we use a trace-driven cache simulator DINEROIII. Cache is described using a number of qualitative and quantitative parameters such as instruction and data cache size, replacement policy, associativity, etc.

The system optimization process is composed of a sequence of activations of these tools. The SHADE analyzer traces program and data memory references as well as the CFG. The CFG is used to drive the code reposition module which produces a new application mapping table. Stream of references are sent to a program that uses the basic block relocation look-up table to map from the original address space into the optimized address space. The re-mapped trace of addresses, along with all unmodified data memory references, is sent to DINEROIII for cache simulation.

6 Variable voltage scheduling

In this Section, we first formulate the *variable voltage scheduling* problem to minimize power consumption and propose an efficient and effective heuristic for the problem.

6.1 Problem formulation

Let $\mathcal{T}$ be a set of tasks and, associated with each task $T_i \in \mathcal{T}$ are a_i its arrival time, d_i its deadline and, W_i its workload, e.g., CPU cycles required. We are using a single variable voltage (variable speed) processor as described in Section 4 to execute the set $\mathcal{T}$. Let $t_0 = min\{a_i : T_i \in \mathcal{T}\}$

and $t_1 = max\{d_i : T_i \in \mathcal{T}\}$. A *scheduler* of $\mathcal{T}$ is a pair of functions defined on $[t_0, t_1]$, such that for any $t \in [t_0, t_1]$, $S(t)$ is the speed of the processor and $T(t)$ is the task being processed at time t. A scheduler is *feasible* if

- $\int_{a_i}^{d_i} S(t)\delta(T(t), T_i)\, dt \geq W_i$, where $\delta(x, y)$ is 1 if $x = y$ and 0 otherwise.
- $S(t) \in [S_{min}, S_{max}]$
- $|S'(t)| \leq K$

The *scheduling problem* is to determine, for any given set of tasks, whether there is a feasible scheduler. The *power minimization problem* is to find a feasible scheduler such that the total energy consumed E is minimized, where $E = \int_{t_0}^{t_1} P(S(t))\, dt$.

6.2 Heuristic scheduler for a single preemptive variable voltage processor

As described in Section 4, the preemptive scheduling of a set of independent tasks with arbitrary arrival times and deadlines on a *variable voltage* processor with a limit on the maximum voltage change rates is an NP-complete task. We have developed an efficient and effective heuristic for the general variable voltage task scheduling problem, which leverages on the least-constraining most-constrained heuristic paradigm as well as the optimal algorithm for the ROSP problem in Section 4 in order to obtain competitive solutions. The algorithm is described using the pseudo-code shown in Figure 6.

Input: a set of m tasks $\mathcal{T} = \{(a_i, d_i, W_i)\}$, the processor's speed range $[S_{min}, S_{max}]$ and the maximal speed change rate K.
Output: a feasible scheduler which minimizes the power consumption to complete $\mathcal{T}$.

1. Use the critical regions [31] to find a partition $0 = t_0 < t_1 < \cdots < t_{n-1} < t_n = t$ of the time interval $[0, t]$
2. Assign the workload W_i for the task T_i in the interval $[t_{j-1}, t_j]$ if T_i appears exactly once in the partition.
3. **REPEAT** {
4. Determine the priority of the remaining tasks.
5. Assign the workload to all time intervals of the task T_i with the highest priority in a least constraining way.
6. } **UNTIL** (All tasks are assigned its workload)
7. Find an optimal solution to the corresponding ROSP problem.
8. Relax constraints on $S(t_i)$'s and improve the solution.

Figure 6. Pseudo code for the variable voltage task scheduling algorithm.

The global flow of the heuristic consists of the following three phases. The heuristic first reduces the problem to the ROSP problem by carefully scheduling the tasks and assigning the workload using the most constrained least constraining heuristic paradigm. Next, the reduced problem is solved optimally. Finally, the solution is further improved by relaxing the starting and finishing speed constraints from the ROSP problem.

Step 1 can be performed in polynomial time using the algorithm given in [31]. In step 2, we first assign the full workload for the tasks with only one time interval. In step 4, the constrainedness of a task T_i is determined by the $COST$ function:

$COST(T_i) = \sum_{j=1}^{p}((S_{L(i_j)} - S_{i_j})^2 + (S_{R(i_j)} - S_{i_j})^2)$.

i_j is the jth time interval for the task T_i with p time intervals. $L(i_j)(R(i_j))$ is the left(right) neighboring time interval to the time interval i_j. $S_{i_j} = \frac{WORK_{i_j}}{length\ of\ time\ interval\ i_j}$, where $WORK_{i_j} = W_i \times \frac{length\ of\ time\ interval\ i_j}{length\ of\ all\ time\ intervals\ of\ T_i}$, if i_j is not already assigned workload and $WORK_{i_j}$ is as assigned otherwise. In step 5, we assign the workload to all time intervals of the task T_i with the highest priority in such a way that it minimizes the $COST(T_i)$.

In step 7, we can get a scheduler which is feasible as well as optimal subject to the condition that *at time t_i, we have to operate the processor at speed $S(t_i)$*. Note that this condition is not necessary for the original problem. Then we may apply the approach described in Section 4 to the OSP problem and find the optimized values for each $S(t_i)$. As discussed in Section 4, however, this approach may not be practical. Thus, in step 8, we relax constraints on $S(t_i)$'s from step 7 and improve the solution. The optimal scheduler for the ROSP problem provides the local optimal speed functions in each interval $[t_{i-1}, t_i]$ which are given by Theorem 2. Recall the shapes of the speed functions given in Theorem 2. For example, if we have a speed function consisting of a shape (a) followed by a shape (b), then it will create a "pike" at the joint point. In this case, if we use a slower speed at the joint point, the speed function can be smoothed. Further calculation shows that this "pike" can be completely eliminated with a slower speed. Based on this observation, in step 8, the local optimal solution obtained from step 7 approaches the global optimal by eliminating all the "pikes".

7 Experimental results

7.1 Design descriptions

We used nine applications to demonstrate the effectiveness of the approach. All nine applications are in public domain and can be obtained via the Internet.

JPEG encoder/decoder from the Independent JPEG Group implements JPEG baseline, extended-sequential, and progressive compression processes. We used integer DCT for decoding a JPEG file to a PPM file. Integer DCT and progressive compression process options were used for compression.

GSM encoder/decoder (Technische Universität, 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

Task Set	Processor Core	I-cache		D-cache		Energy Consumption (J)		
		Cache Size	Block Size	Cache Size	Block Size	Lower Bound	New Approach	[31]+ROSP
10 tasks	LSI CW4001	512B	64B	1KB	128B	**2.39**	**2.48**	**3.05**
	LSI CW4001	512B	64B	1KB	128B	**2.39**	**2.48**	**3.05**
20 tasks	LSI TR4101	512B	64B	4KB	32B	**10.6**	11.8	**13.9**
	Motorola 68000	1KB	128B	2KB	64B	10.9	**11.5**	14.1
30 tasks	PowerPC403	1KB	32B	2KB	64B	**45.3**	49.0	**61.2**
	LSI CW4011	1KB	64B	2KB	64B	46.5	**48.1**	63.9
40 tasks	PowerPC403	1KB	32B	2KB	32B	**72.6**	**78.2**	**87.5**
	PowerPC403	1KB	32B	2KB	32B	**72.6**	**78.2**	**87.5**
50 tasks	LSI CW4011	1KB	64B	2KB	64B	**100.2**	**108.6**	**128.6**
	LSI CW4011	1KB	64B	2KB	64B	**100.2**	**108.6**	**128.6**

Table 4. The most power-efficient configurations obtained by the proposed approach.

term prediction) coding at 13 kbit/sec. A 16 bit linear PCM data file is used to measure execution parameters of the GSM encoder. The measurement of the GSM decoder execution characteristics was done using the output of the GSM encoder.

EPIC (Efficient Pyramid Image Coder), from University of Pennsylvanian, implements lossy image compression and decompression utilities.

Mipmap, Osdemo, and Texgen from University of Wisconsin use a 3-D graphic library called Mesa. 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.

7.2 Experimental data

The processor cores described in Table 1 are used with the assumption that they can be operated at the dynamically variable [0.8, 5.0] V supply voltage. Performance and power consumption data have been scaled according to the performance *vs.* voltage and power *vs.* voltage relationships, respectively. The limit on the voltage change rate is assumed to be 10000 instructions/volt for all processor cores. We consider several different mixes of the above nine applications. The details of the application mixes and deadlines are provided in the technical report version of this paper.

The most power-efficient processor, I-cache and D-cache configurations for the target scenarios by the proposed allocation heuristic using Yao's optimal preemptive scheduling algorithm with no limit on voltage change rates, Yao's optimal preemptive scheduling algorithm with the ROSP solution taking into account the rate change limits, and our scheduling heuristic with limit on voltage change rates, respectively, are described in Table 4. For each task set, the first row represents the best configuration for the proposed allocation heuristic using Yao's optimal preemptive scheduling algorithm with no limit on voltage change rates, while the second row provides the best configuration for the proposed allocation heuristic using our scheduling heuristic. Columns 7, 8, and 9 provide the energy consumption of the target scenarios by Yao's optimal scheduling algorithm under ideal condition, our scheduling heuristic, and Yao's optimal scheduling algorithm with the ROSP solution under rate change limits, respectively, where the numbers in bold represent the best energy consumption achieved for each algorithm.

Our approach resulted in only **6.8** % more energy consumption over the power consumption lower bound which was obtained by using Yao's optimal preemptive scheduling algorithm with no limit on voltage change rates, which always results in a solution better than or equal to the optimal preemptive solution with any limit on voltage change rates. Our approach also resulted in **20.2** % reduction in energy consumption from Yao's optimal preemptive scheduling algorithm with the ROSP solution under the rate change limits. Among the five application mixes, three cases resulted in the same configurations for both scheduling algorithms, while their actual task schedules were different.

8 Conclusion

Voltage is an important dimension that, with appropriate CAD tools and schedulers, can be actively manipulated at run time instead of just the static optimization of a fixed voltage as is currently done. However, for this to happen, the CAD tools must incorporate an understanding of the limitations inherent to variable voltage hardware. In this paper we have accomplished this by describing, in the context of a core-processor based system-on-a-chip synthesis design flow, a scheduler that not only schedules a set of computation tasks on hardware resources but also schedules changes in voltage (and correspondingly the clock frequency) while meeting constraints on the rate of voltage change imposed by the voltage regulator. Our results show that the proposed heuristic algorithm is extremely effective and is within 7% of the best possible variable voltage hardware system. Moreover, our research for the first time formalizes how synthesis tools might leverage run-time controlled variation of supply voltage while taking into account inherent limitations of the physical voltage and clock regulation circuits.

References

[1] 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.

[2] 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.

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

[4] F. Douglis, P. Krishnan, and B. Bershad. Adaptive disk spin-down policies for mobile computers. In *USENIX Symposium on Mobile and Location-Independent Computing*, pages 121–137, 1995.

[5] R.J. Evans and P.D. Franzon. Energy consumption modeling and optimization for SRAM's. *IEEE Journal of Solid-State Circuits*, 30(5):571–579, 1995.

[6] R. Fromm, S. Perissakis, N. Cardwell, C. Kozyrakis, B. McGaughy, D. Patterson, T. Anderson, and K. Yelick. The energy efficiency of IRAM architectures. In *International Symposium on Computer Architecture*, pages 327–337, 1997.

[7] 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.

[8] 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.

[9] V. Gutnik and A. Chandrakasan. An efficient controller for variable supply-voltage low power processing. In *Symposium on VLSI Circuits*, pages 158–159, 1996.

[10] I. Hong, D. Kirovski, G. Qu, M. Potkonjak, and M. Srivastava. Power optimization of variable voltage core-based systems. In *Design Automation Conference*, pages 176–181, 1998.

[11] I. Hong, M. Potonjak, and R. Karri. Power optimization using divide-and-conquer techniques for minimization of the number of operations. In *IEEE/ACM International Conference on Computer-Aided Design*, pages 108–113, 1997.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] P. E. Landman and J. M. Rabaey. Activity-sensitive architectural power analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 15(6):571–587, 1996.

[17] E. L. Lawler and C. U. Martel. Scheduling periodically occuring tasks on multiple processors. *Information Processing Letters*, 12(1):9–12, 1981.

[18] 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.

[19] P. Macken, M. Degrauwe, M. Van Paemel, and H. Oguey. A voltage reduction technique for digital systems. In *IEEE International Solid-State Circuits Conference*, pages 238–239, 1990.

[20] F. N. Najm. A survey of power estimation techniques in VLSI circuits. *IEEE Transactions on VLSI Systems*, 2(4):446–455, 1994.

[21] 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.

[22] L. S. Nielsen, C. Niessen, J. Sparso, and K. van Berkel. Low-power operation using self-timed circuits and adaptive scaling of the supply voltage. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2(4):391–397, 1994.

[23] M. Pedram. Power minimization in IC design: principles and applications. *ACM Transactions on Design Automation of Electronic Systems*, 1(1):3–56, 1996.

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

[25] 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.

[26] A. J. Stratakos, S. R. Sanders, and R. W. Brodersen. A low-voltage CMOS DC-DC converter for a portable battery-operated system. In *Power Electronics Specialist Conference*, volume 1, pages 619–626, 1994.

[27] 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.

[28] V. Tiwari, S. Malik, and A. Wolfe. Power analysis of embedded software: a first step towards software power minimization. *IEEE Transactions on VLSI Systems*, 2(4):437–445, 1994.

[29] 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.

[30] 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.

[31] 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.