MODEL-DRIVEN CLOCK FREQUENCY SCALING FOR CONTROL-DOMINATED EMBEDDED SYSTEMS

Zdravko Karakehayov

Technical University of Sofia,
Kliment Ohridski 8, Sofia-1000, Bulgaria,
e-mail: zgk@computer.org

Abstract: This paper introduces a combination of models and proofs for optimal power management via clock frequency scaling. The approach is suitable for systems on a chip or microcontrollers where a processor runs in parallel with embedded peripherals. Since the methodology is based on clock rate control, it is very easy to implement. A hardware model, a computational model and an energy model underlie the procedure. We proved that the combination of models is sufficient to determine an optimal clock rate for the CPU. Furthermore, we expand the application space taking into account preemption of tasks. Also, we discuss the role of embedded peripherals when selecting the clock frequency in both active and power-saving modes. Simulation results manifest the benefits of clock rate control under the proposed methodology. An example shows a 56% increase of the battery lifetime when the clock rate is changed from the lowest possible level to the optimal value.

Keywords: Embedded systems, power management, dynamic clock scaling, dynamic voltage scaling.

1. INTRODUCTION

Power consumption is an important design metric for battery-powered embedded systems. Along with the battery lifetime, the power consumption affects performance and reliability. Currently, CMOS technology dominates the embedded computing. Static power and dynamic power are the two principle sources of power consumption for CMOS chips. Static power is associated with the current that leaks through transistors even when they are turned off. Dynamic power is proportional to the clock frequency and the square of the supply voltage.

Several methods have been developed to reduce the power consumption. The supply voltage islands method allows different chip areas to use the minimum voltage required to achieve their performance goals [1]. The voltage islands partitioning comes as a natural approach when voltage optimized functional blocks are reused in a SoC design. Threshold voltage paths is a fine-grain technique which assigns low threshold voltage to the transistors on the critical paths to guarantee high performance [2]. The method assigns higher threshold voltage to the transistors on noncritical paths to reduce the static power. Likewise, clock islands is a solution that employs different clock frequencies for different domains of the SoC [3, 4]. While each domain is internally synchronous, the interplay between domains is asynchronous. The systems are labeled Globally Asynchronous Locally Synchronous (GALS). Since the multiple islands are clocked at different rates, the power consumption can be reduced significantly without compromising the system performance.

Power optimization techniques can be applied at run time as well. An efficient method is to scale the clock frequency according to the current work load [5, 6, 7]. Modern processors allow the application to dynamically vary the clock rate to balance performance versus power consumption. Dynamic frequency scaling (DFS) is characterized by very low overhead. Dynamic supply voltage scaling (DVS) is another power reduction technique [8, 9]. DVS exploits slack time by reducing simultaneously clock frequency and supply voltage. Consequently, DVS adapts the performance to the actual requirements of the system. In this way, substantial savings are achieved since the power consumption is proportional to the square of the supply voltage.

Clock frequency scaling and DVS are independent of the previously discussed methods and can be applied at a higher level of abstraction to further improve the energy efficiency. The capability to control the clock rate and supply voltage under variable work load brings intelligence to embedded computers.

Clock frequency scaling is essential for any
power management. Along with its independent application, it is an integrated part of the supply voltage scaling as well.

2. RELATED WORK

Different aspects of the clock frequency scaling are addressed by research. Data-dominated applications, such as multimedia systems, demand constant output rates. The opportunities for clock frequency scaling under requirements for constant output rates are discussed in [10]. Characterizing tasks by the number of required clock cycles is applied in [6, 7, 11, 12, 13]. In related research we investigated optimal clock rates for energy-aware embedded systems [11, 12, 14].

The goal of this paper is to provide optimal clock rates for different behavioral intervals of real-time embedded systems. Examples show that a shift from the lowest possible clock rate to the optimal clock frequency may increase the battery lifetime by 56%.

We discuss two system architectures to evaluate the opportunities for DFS. Actual microcontrollers are referred to illustrate both cases.

The computation model allows preemption of tasks and we show how to modify the clock rate when the CPU is available again. Also, we investigate the requirements of embedded peripherals for different clock rates in active and power-saving mode. The proof for this case is associated with specific energy models. Numerical simulations illustrate typical behavior intervals.

3. HARDWARE PLATFORM

Since the target architecture is based on a microcontroller or SoC, the CPU runs in parallel with a variable number of embedded peripherals [15, 16]. Fig. 1 shows a hardware platform, model Common Clock (CC). The architecture consists of an oscillator (OSC), divider (D), CPU, divider for peripherals (DP) and peripherals (P). Parallel ports, serial ports, timer/counters and A-D converters are typical peripherals. Most peripherals require a certain clock rate for a proper operation.

Fig. 1. – Hardware platform, model CC.

We assume hardware provides the following mechanisms for power management:

- The CPU and embedded peripherals can be individually enabled and disabled. When the CPU is running, the system is in an active mode. If the CPU is switched off, the system is in a power saving mode.
- The clock rate can be scaled by division of the oscillator frequency.
- A timer counts the clock cycles to keep track of the workload processed. This is important in case of preemption.

Fig. 2 shows another model for the hardware platform – Separate Clock (SC). Under this scheme, scaling of the clock rate does not affect the peripherals speed. Consequently, the SC architecture allows a straightforward implementation of DFS.

Fig. 2. – Hardware platform, model SC.

Hardware does not necessarily provide a mechanism for dynamic voltage scaling. In case of voltage-scalable systems, additional energy savings can be achieved [17, 18]. To enable DVS the system incorporates a DC/DC voltage converter, a serial bus between the processor and the DC/DC converter and a voltage controlled oscillator (VCO). Supply voltage and clock frequency are changed by sending the desired frequency over the serial bus. Upon receiving the desired frequency, the DC/DC converter compares this frequency with the current frequency and either increases or decreases the supply voltage. According to the changed voltage, the VCO adapts the clock rate to a higher or lower level. Typical transition times are in the range of tenths of microseconds.

4. COMPUTATIONAL MODEL

Assume that the system's functionality is partitioned into tasks. A set of tasks,

\[ T = \{ T_1, T_2, \ldots, T_n(T) \} \]

is mapped to a particular processor. Each task is characterized by its workload, deadline and period. The deadline is the time when all computation must finish. The period is the interval between two consecutive executions. If the task's workload is measured in number of clock cycles, task-level power reduction can be applied. Different approaches can be taken to control the clock rate. Based on the current task's deadline, the clock rate can be declined as much as possible. However, if the processor completes the task ahead of the deadline and enters a power-saving mode, the energy could be minimized. Fig. 3 lays out all phases of the real-time computational model.
The period, $T_p$, accommodates the execution time $T_{ACT}$ and the power-saving period $T_{PS}$,

$$T_p = T_{ACT} + T_{PS}$$  \hspace{1cm} (2)

The real-time computational model is a typical approach to specify control-dominated embedded systems when the interaction with the environment dictates a specific timing. At the same time, tasks are executed on physical platforms which impose timing constrains as well. Calculating the energy for a period of time provides more realistic results compared to only task’s execution energy.

### 5. ENERGY MODEL

Assume that the supple current $I_{DD}$ scales linearly with the clock frequency for both active ($I_{DD,ACT}$) and a power saving mode ($I_{DD,PS}$). The energy model is expressed by the following empirical equations:

$$I_{DD,ACT} = k_{ACT}f + n_{ACT}$$  \hspace{1cm} (3)

$$I_{DD,PS} = k_{PS}f + n_{PS}$$  \hspace{1cm} (4)

Fig. 4 shows possible relationships between the supply currents in active and power-saving modes. The plots are task-related and depend on the peripherals and memory currently employed.

Each task utilizes a different set of instructions. As a result, power variation from task to task can be observed. Moreover, for each task a different set of embedded peripherals runs in parallel with the CPU.

An individual, fine-grain energy model should be considered for each timing interval, active and power-saving. We assume that peripherals continue to run when the CPU is in the power-saving mode. Peripherals are enabled and disabled at the beginning and the end of each task according to the current requirements. The task-aware energy model is relevant for SoC or microcontroller-based hardware platforms.

The parameters of the energy model for each task can be obtained via power analysis. There are two possibilities: simulations and physical measurements. Physical measurements are easier to implement and the results are closer to the actual energy signature of the system [19, 20]. A task is repeatedly executed to measure the average current drawn by the system for different clock frequencies.
\[ E_p = k_{ps} V_{dd} T_p - (n_{ACT} - n_{ps}) V_{dd} N/f^2 \] (8)

There is one positive critical number,

\[ f = \left(\frac{n_{ACT} - n_{ps}}{k_{ps}}\right)^{1/2} \left(\frac{N}{N/T_p}\right)^{1/2} \] (9)

If \( n_{ACT} > n_{ps} \), \( E_p = 2(n_{ACT} - n_{ps}) V_{dd} N/f^3 > 0 \). In this case, the energy per period has a minimum for

\[ f = \left(\frac{n_{ACT} - n_{ps}}{k_{ps}}\right)^{1/2} \left(\frac{N}{N/T_p}\right)^{1/2} \] (10)

From \( N f^{-1} \leq T_{dl} \), \( T_{dl} \geq (Nk_{ps} T_p (n_{ACT} - n_{ps})^{-1})^{1/2} \).

If \( n_{ACT} \leq n_{ps} \), \( E_p > 0 \) and \( E_p \) increases when we move from \( f_{min} \) towards \( f_{max} \). The clock frequency is determined by the deadline requirement, \( f = N/T_{dl} \).

**Corollary.** If a task was preempted for a period of \( t_p \) when the CPU already executed \( N_1 \) clock cycles, \( n_{ACT} > n_{ps} \) and

\[ T_{dl} \geq \left(\frac{n_{ACT} - n_{ps}}{k_{ps}}\right)^{1/2} \left(\frac{N - N_1}{T_p}\right)^{1/2} - N_1 \left(\left(\frac{n_{ACT} - n_{ps}}{k_{ps}}\right)^{1/2} \left(\frac{N}{T_p}\right)^{1/2}\right) - t_p \]

the energy per period has the smallest value for clock frequency

\[ f = \left(\frac{n_{ACT} - n_{ps}}{k_{ps}}\right)^{1/2} \left(\frac{N - N_1}{T_p}\right)^{1/2} - N_1 \left(\left(\frac{n_{ACT} - n_{ps}}{k_{ps}}\right)^{1/2} \left(\frac{N}{T_p}\right)^{1/2}\right) - t_p \] (11)

If \( n_{ACT} \leq n_{ps} \), the energy per period has the smallest value for clock frequency

\[ f = \frac{N - N_1}{T_{dl}} - N_1 \left(\frac{N}{T_{dl}}\right)^{-1} - t_p \] (12)

Assume that a task has a work load of 200000 cycles and a period of 300 ms. The task's integration is planned for three systems. All of them have \( k_{ACT} = 0.92 \times 10^{-3} \), \( n_{ACT} = 11 \times 10^{-3} \), and \( k_{ps} = 0.5 \times 10^{-9} \). The systems have different \( n_{ps} \), \( 3 \times 10^{-3}, 11 \times 10^{-3} \) and \( 13 \times 10^{-3} \) respectively. The supply voltage is 3V. Fig. 5 shows how the energy per period scales with the clock rate. Based on the workload and deadline, the minimum value for the clock frequency is 2 MHz. This clock rate must be applied for the systems with \( n_{ps} = 3 \times 10^{-3}, 11 \times 10^{-3} \) and \( 13 \times 10^{-3} \). The other system has the smallest value for the energy at 4 MHz and this frequency is selected.

If the optimal clock frequency is higher than the deadline requirement, the task can be optimized for memory via a compiler mode. If the optimal clock rate is lower than the deadline requirement, there is potential for power reduction and the task should be optimized for speed. For example, if the deadline is reduced to 20 ms, the clock frequency must be increased to 10 MHz. The corresponding energy levels are marked by solid squares.
Fig. 7. – Energy for variable clock rate and supply voltage.

Fig. 8 lays out the same energy plot when the clock frequency goes up to 100 MHz. Inspection shows that the greater than linear relationship between supply voltage and energy occurs for high clock rates. DVS is vital in that area. Optimal selection of the clock frequency is a major requirement for low clock rates. The example illustrated in Fig. 5 shows a 13% increase in the energy per period when the optimal clock frequency is replaced with the minimal frequency of 2 MHz. A deadline of 200 ms will allow a clock rate of 1 MHz. If this frequency is selected, the energy will be increased by 56%. Since the increase of energy after the optimal point is small, higher clock rates can be used to decline the vulnerability to preemption.

Fig. 8. – High clock rates require DVS.

Assume that for the current example the task encapsulates the overall functionality of the system. The task is executed 24 times per day and the system has a 3100 mAh battery. While the optimal clock rate of 4 MHz clock rate determines a 218 days lifetime for the battery, a clock frequency of 1 MHz will decline the lifetime to 139 days. Neglecting the energy model and selecting the lowest possible clock rate may reduce the battery lifetime significantly.

7. EMBEDDED PERIPHERALS

As discussed in Section 3, we distinguish between two models for hardware platforms, CC and SC. Under SC it is possible to scale the CPU’s clock rate without affecting the peripherals clock frequency. Microcontrollers, such as MSP430, are capable of scaling the processor's frequency keeping the peripherals speed unchanged [21]. Other microcontrollers, such as Atmel ATmega164P/V, Atmel AT9155800 and TMS470, have a common prescaler for the CPU and peripherals [16, 22, 23, 24]. For tasks running in parallel with these peripherals clock frequency scaling is still feasible if local prescalers can be adjusted to keep the peripherals clock rate constant. Timer/counters are normally capable of dividing the clock frequency. In case of clock frequency scaling the initial coefficients of their prescalers become factors.

We assumed that the system runs at a single clock rate within the Tp period. However, microcontrollers with common clock may impose limitations not only to the clock rate in the active mode, but also to the clock frequency in the power saving mode. For example, if the system's functionality requires an analog-to-digital conversion after a certain task, the processor must select the clock frequency which allows the desired conversion time before going to sleep. Assume that an AT91M55800A microcontroller is allocated for the system and the required conversion time is 10 µs. The microcontroller has to switch the clock frequency to 1.1 MHz and then to gate off the CPU [16]. Theorem 2 deals with the relationship between the energy per period and the clock rate in active mode when the clock rate in the power saving mode is fixed.

Theorem 2. Let the energy model is defined as

\[ I_{DD,ACT} = k_{ACT}f + n_{ACT} \quad \text{and} \quad I_{DD,PS} = k_{PS}f + n_{PS}. \]

Suppose that a task has a workload of N clocks, \( T_{DL} \) deadline and \( T_p \) period of execution. The energy per period is defined on the closed interval \([f_{MIN}, f_{MAX}]\).

When the task completes, the clock frequency is switched to \( f_{PS} \).

If \( k_{PS}f_{PS} + n_{PS} > n_{ACT} \), the energy per period has the smallest value for \( f = N/T_{DL} \).

If \( k_{PS}f_{PS} + n_{PS} < n_{ACT} \), the smallest value for the energy per period occurs at the \( f_{MAX} \) endpoint.

If \( k_{PS}f_{PS} + n_{PS} = n_{ACT} \), the energy per period does
not scale with the clock frequency in active mode.

**Proof.** The energy per period

\[
E_P = \int_0^T P(t) \, dt = V_{DD} N (n_{ACT} - n_{PS})/f - V_{DD} k_{PS} f_{PS} N/f
\]

(14)

+ \(V_{DD} k_{PS} T_f P_{PS} + V_{DD} k_{ACT} N + n_{PS} V_{DD} T_f\)

We get for the first derivative,

\[
E_P' = V_{DD} N (k_{PS} f_{PS} - n_{ACT} + n_{PS})/f^2
\]

(15)

If \(k_{PS} f_{PS} + n_{PS} > n_{ACT}\), \(E_P' > 0\) and \(E_P\) increases when we move from \(f_{MIN}\) towards \(f_{MAX}\). The clock frequency is determined by the deadline requirement.

If \(k_{PS} f_{PS} + n_{PS} < n_{ACT}\), \(E_P' < 0\) and \(E_P\) decreases when we move from \(f_{MIN}\) towards \(f_{MAX}\). The smallest value for \(E_P\) occurs at the \(f_{MAX}\) endpoint.

If \(k_{PS} f_{PS} + n_{PS} = n_{ACT}\), \(E_P = 0\) and \(E_P\) does not scale with the clock frequency in active mode.

Fig. 9 shows the energy per period for a variable clock rate and different frequencies for the peripherals in power saving mode. The task in focus has a workload of 300000 cycles and a period of 300 ms. The task has a deadline of 200 ms. Also, \(k_{ACT} = 0.92 \times 10^{-9}\), \(n_{ACT} = 0.4\), \(k_{PS} = 0.3 \times 10^{-9}\) and \(n_{PS} = 0.013\). The supply voltage is 3V.

**Fig. 9. – Energy per period, predefined \(f_{PS}\).**

Fig. 10 is a related simulation which helps to distinguish between two cases: the smallest value for \(E_P\) occurs at the \(f_{MAX}\) endpoint and the smallest value for \(E_P\) occurs at the \(f_{MIN}\) endpoint.

The second case, the \(f_{MIN}\) endpoint, requires a test whether the task meets the deadline requirement. Also, Fig. 10 manifests how important it is to take into account the clock rate in the power saving mode. For low \(f_{PS}\) the clock frequency must be selected at the \(f_{MAX}\) endpoint. Conversely, for high \(f_{PS}\) the clock rate must be as low as possible. Fig. 11 shows energy models that are relevant for Theorem 2.

**Fig. 10. – Energy per period influenced by \(f_{PS}\).**

**Fig. 11. Compare \(n_{ACT}\) and \(I_{DD,PS}\) for \(f_{PS}\).**

While Theorem 1 is based on the relationship between \(n_{ACT}\) and \(n_{PS}\), Theorem 2 compares \(n_{ACT}\) and \(I_{DD,PS}\) for \(f_{PS}\).
8. CONCLUSION

This paper presents a methodology for optimal power management of real-time embedded systems. The power consumption is controlled via clock frequency scaling. This approach is suitable for systems on a chip or microcontrollers where a processor runs in parallel with embedded peripherals. The combination of a hardware model, a real-time computational model and an energy model is central for the method. The hardware model assumes a power saving mode, individual control over the peripherals and clock rate tuning. These requirements are completely consistent with all modern hardware platforms. The hardware platforms are classified into two types: SC which allows a straightforward implementation of DFS and CC which requires compensation of the clock changes at some peripherals. The computational model is based on a single-task timing and accords well with both static and dynamic scheduling. We have proved that the parameters of the computational and energy models are necessary and sufficient to calculate the optimal clock rate. Moreover, we discussed the influence of preemption and embedded peripherals and outlined the power management application space. We modified the computational model to meet timing requirements from peripherals and provided a proof for this case as well. Finally, we applied numerical simulation to illustrate the proposed methodology. Simulation results showed that a small deviation from the optimal clock frequency may lead to a significant increase in energy.

6. REFERENCES


Zdravko Karakehayov is an Associate Professor in the Department of Computer Systems at the Technical University of Sofia, Bulgaria. Formerly he was with the Technical University of Denmark, Lyngby and the University of Southern Denmark, MCI-CPD, Sønderborg. Dr. Karakehayov received the Ph.D. degree in computer science from Technical University of Sofia, Bulgaria. He is a senior member of the IEEE Computer Society and a Distinguished Visitors Program speaker. Dr. Karakehayov currently chairs the Computer Chapter, IEEE Bulgarian section. He co-authored five books in the field of embedded systems and holds eight patents. His research field includes low-power design for embedded systems, low-power and secure routing for wireless sensor networks. Dr. Karakehayov served as a reviewer for the Journal Transactions on Embedded Computing Systems and several international conferences.