C++ Encapsulated Dynamic Runtime Power Control for Embedded Systems

Orlando J. Hernandez, Godfrey Dande, and John Ofri
Department Electrical and Computer Engineering
The College of New Jersey
hernande@tcnj.edu

Abstract

This paper presents an innovative way to control power consumption in embedded microprocessors. A combination of software profiling, voltage-scaling and frequency throttling, along with run-time speed tables calibration and C++-like constructor and destructor functions is used. It is demonstrated that more power saving can be achieved by this hybrid system than can be achieved using only voltage scaling or frequency throttling.

1. Introduction

Power management is a critical factor in most embedded systems. With more power demanding hardware and software, it has become imperative to include power management schemes in all embedded microprocessor-based systems. Of particular concern are battery run applications that are usually mobile, for example the cell phone.

Power management is also becoming important in non-mobile systems. As more electronic appliances are developed, the demand for electric energy has increased. This tends to drive energy consumption up, so that it is necessary for even the household embedded system to be power smart.

In the approach described in this paper, software is profiled in a real-time environment to evaluate the minimum frequency needed to run a particular function. The intent is to program the processor's clock generation subsystem to provide this frequency at the lowest voltage possible. The power sensitive parameters are tracked and controlled by the software. Running levels of power consumption are initiated by constructor functions for each function. Once the system needs to make a function transition, the parameters are reevaluated and an idle power state is programmed by a function's destructor function. The constructor for the next function will then create a new power control object for that function. The system then assumes a new power level. It is demonstrated that more power saving can be achieved by

this hybrid system than can be achieved using only voltage scaling or frequency throttling.

2. Recent related work

There are various methods of power management that have been realized before. Communication Architecture Based Power Management (CBPM) is a technique that uses system level communication to control power consumption. In this methodology power intensive states of a system's functional units are identified by dividing the known power profiles into macro-states [1]. System level communication is used to prevent the overlap of power intensive states.

Profile driven code execution optimizes power consumption with regard to battery capacity [2]. In this scheme parallelism of functional units is optimized for energy consumption rather than for performance. Code annotation allows the microcomputer to vary fetch and execution rate.

A hybrid of dynamic voltage scaling and compiler supported power management has also been developed. In this system power management hints are inserted in the code [3]. These approaches do not take into account the simultaneous scaling of voltage and frequency, or the real-time encapsulation of the methodology, so it lends itself to runtime calibration of the parameters, as the temperature of the processor chip changes.

3. Design scheme

As a test platform, a Freescale microcontroller is used. The design block diagram is shown in Figure 1.

The design scheme was derived from evaluation of the fundamental power dissipation equation for digital semiconductor products:

$$P = C_{pd}\ f\ V^2 \qquad (1)$$

Where:

P   : power dissipation
C_{pd} : Semiconductor constant

f   : Frequency
V   : Supply voltage

C_{pd} is a characteristic of the chip, therefore it is necessarily a constant. That leaves the options of controlling either or both f and V.
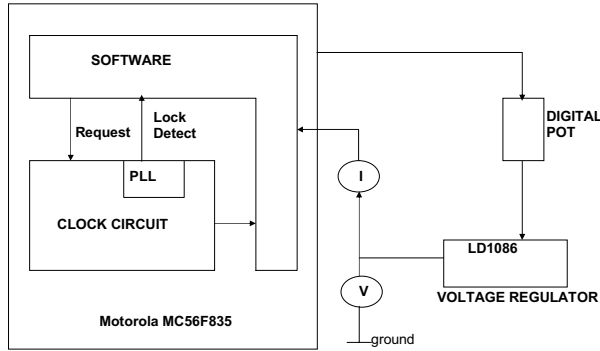

Figure 1. Design block diagram

For certain ranges depending on the microprocessor the frequency can be throttled at a constant voltage. For every frequency request f, V would be minimized and f maximized. Therefore, the system would not operate at a certain voltage level for a given frequency unless that frequency could not be obtained at a lower voltage level. The governing equation for locating our operating point is:

$$f_{max} = f(V_{min}) \tag{2}$$

Essentially the frequency is mapped into a one to one correspondence with the voltage, although generally the same frequency can be obtained at different V.

4. Hardware setup and considerations

A Freescale microcontroller was used as the experimental system. This processor was chosen because it could take a variable voltage. The MC56F8357 could also allow for the frequency to be altered. It was important to choose a popular component from a popular manufacturer so that the work we did would be directly applicable to large number of applications.

Once the processor was decided upon, a voltage regulator for the core voltage was chosen. The specifics for the regulator rippled from the choice of microcontroller. Figure 2 shows the circuit diagram for the ST Microelectronics LD1086 programmable voltage regulator configuration. A digital potentiometer was used to vary the regulator output voltage.

The voltage range needed to calibrate the voltage regulator and the digital pot were determined by the microprocessor. The core power supply was accessed, the onboard regulator was disengaged and LD1086 engaged.

The graph below in Figure 3 shows the characterization of voltage output versus the digital pot resistance against for the LD1086. This relationship was used to create a look up table to be referenced in setting voltage levels. The relationship is logarithmic.
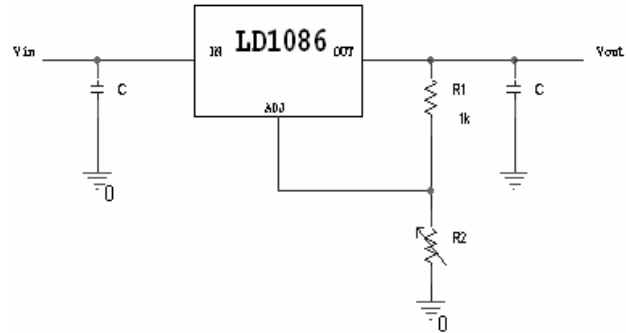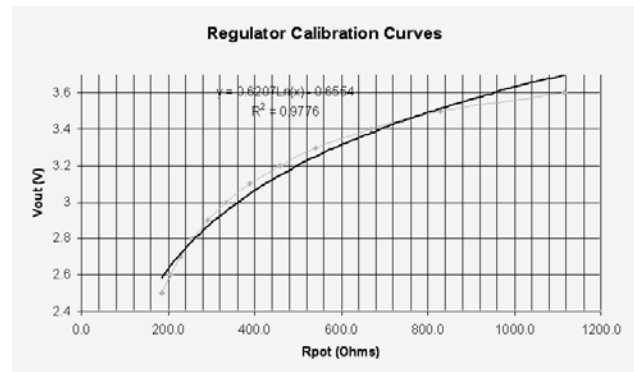

Figure 2. Voltage regulator configuration

Figure 3.   Graph of Vout vs. potentiomenter resistance (R2) for LD1086

5. Software setup and considerations

Three different benchmark programs were used to test out our scheme. Runtime constraints were imposed on these programs to simulate real time running conditions. These were a matrix multiplication benchmark program, Matrix Multiply. The second benchmark is an implementation of the Fast Fourier Transform algorithm. The third benchmark is a C updated version of the classic FORTRAN benchmark program Whetstone. As the name suggests, it does whetstone calculations on an array a set number of times to benchmark a system.

5.1. Software characterization and programming parameters

Table 1 below shows the parameters that had to be evaluated to be encapsulated into the software for this power management scheme. First, the required frequency for each function was determined via real-time profiling. Then, for every frequency the minimum voltage was determined. And finally, the resistance value for the digital potentiometer corresponding to the required voltage was determined. The digital potentiometer could then be directly varied in response to a frequency request.

Table 1. Characterization parameters

Software Function	MIPS	CPI	Fmax	Vout	Rpot
A	32	1	32 MHz	2.97 V	350 Ω
B	56	1	56 MHz	3.51 V	830 Ω
C	16	1	16 MHz	2.61 V	200 Ω

The frequency is maximized at any specific operating voltage, and therefore the efficiency attained by the processor is maximized as well. The objective of profiling the code is to get information about how fast the processor needs to run to execute the different functions in real-time. The Metrowerks Integrated Development Environment (IDE) that is used in conjunction with the board contains a profiling tool that allows the user to analyze the code it runs.

After the functions are characterized the software is used to control the output voltage of the regulator via the programmable digital potentiometer. Communication between the processor and the potentiometer is through a serial port in each component.

5.2. Software design procedure

Since the compiler bundled with the evaluation board did not support C++, C++-like constructor and destructor functions were designed in C to adjust the power parameters (Supply Voltage and Frequency) at the entry and exit points of each function. This approach is 100% compatible and portable to a true C++ implementation. The main usage of the destructor function is to return the supply voltage level to a safe maximum, so that the next function called will begin its process with enough voltage to continue execution.

5.3. Runtime recalibration

For most semiconductor chips, as the temperature raises, the operating frequency degrades. Because of the heat generated by the processor during runtime, the characterization table of Supply Voltage versus Frequency has to be recreated. The maximum frequency that a digital semiconductor chip runs at a given supply voltage is a function of the die temperature.

The software periodically compensates for variations of chip temperature. The system clock circuitry integrated in the processor is used for this task. The calibration of the Supply Voltage versus Frequency table consists on setting the clock generation circuitry integrated in the processor to different frequency and voltage combinations, and registering the lowest supply voltage for which a given frequency was synthesized successfully, as indicated by the assertion of the lock detection indicator of the clock generator. This exercise is performed with the clock generator frequency bypassed and using the crystal frequency directly to ensure that the processor has a stable clock to run this routine.

This ensures that at all times as the processor die heats up because of self heating and/or environmental conditions, the optimum clock frequency is set for any given supply voltage. The code structure was kept as simple as possible to avoid unnecessary overhead.

6. Expected results

It is expected that using this scheme, better power management will be achieved than using frequency scaling or voltage scaling alone as shown in Figure 4. The "No Control" line on the graph shows power consumption using a constant supply voltage and a constant frequency equal to the maximum frequency needed to run all the functions, and it is the most inefficient. Frequency or Voltage scaling saves some energy, which is shown by the "Freq. Scaling" and "Voltage Scaling" lines. Finally, the last data line "V and F Scaling" shows the power savings that is achieved using both voltage and frequency scaling per our scheme. Figure 4 below shows that the most energy savings is achieved by our power management method of both frequency and voltage scaling during runtime tailored to the specific software being executed.

Table 2 Shows a comparison of the power savings obtained by the scheme presented in this paper and either frequency or voltage scaling, or no power control at all. It is seen that this scheme can save up to 53-54% power over no control at al, and an additional 13-14% power savings over frequency scaling, which does better in our setup than voltage scaling at 32-33% power savings. This means that the scheme presented in this paper is 22-23% more efficient than frequency scaling alone, and 31-32% more efficient than voltage scaling alone in our setup.

7. Conclusions

In this paper, a process of analyzing and profiling the functions of an embedded real-time software system to minimize power usage by a processor has been described. It is demonstrated that more power saving can be achieved by this hybrid system than can be achieved using only voltage scaling or frequency throttling. Embedded processors have increased in speed by large factors, thus execution time overhead for this scheme is a minor factor.

Table 2. Comparison of power savings

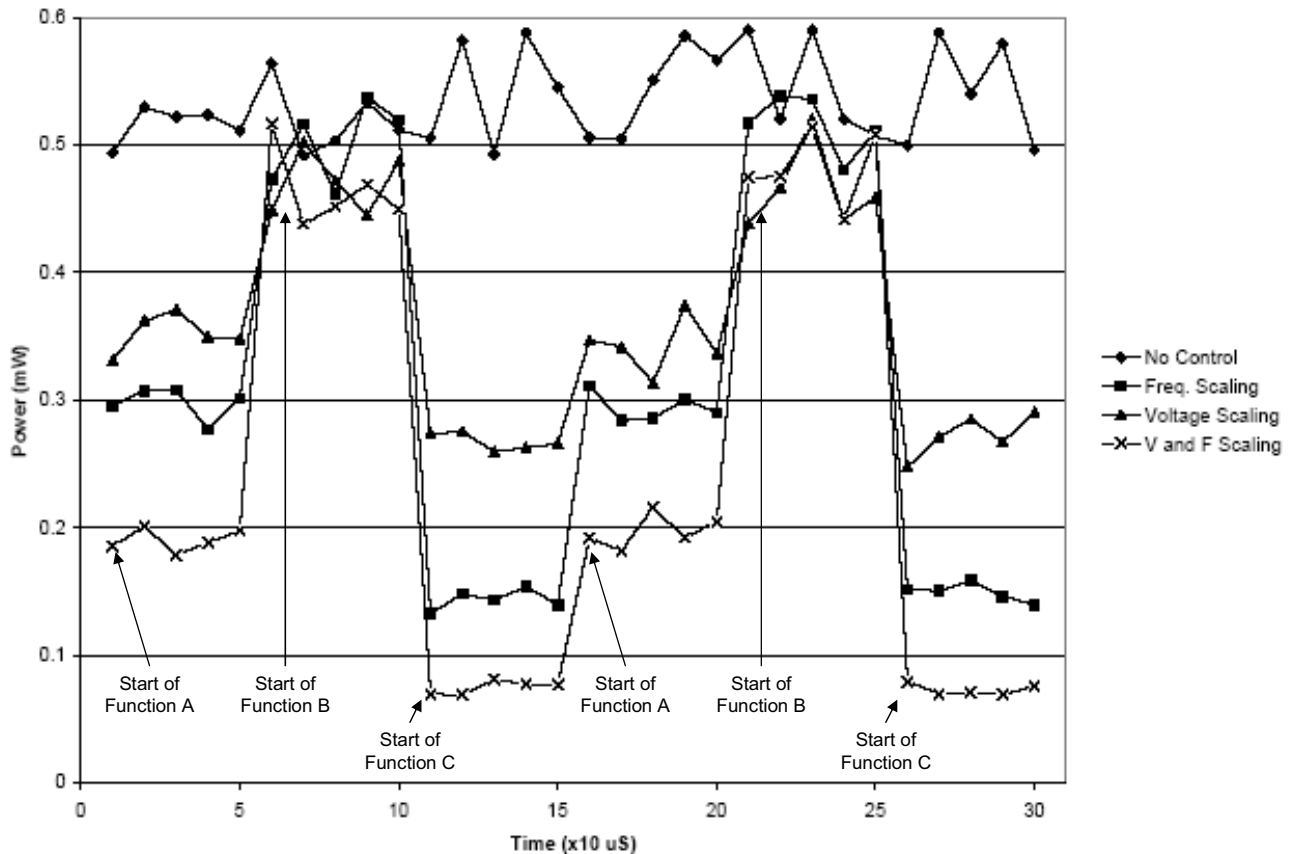|	No Control	Freq. Scaling	Voltage Scaling	V and F Scaling
Average Power (mW)	0.5347	0.3173	0.3619	0.2472
Power/Energy Savings	-	40.67 %	32.33 %	53.78 %

Figure 4. A comparative graph of the expected power profiles

8. References

[1] K. Lahiri et al "Communication Architecture Based Power Management for Battery Efficient Systems." *ACM DAC 2002, New Orleans, Louisiana, USA, June 10-14, 2002.*

[2] D. Marculescu, "Profile-Driven Code Execution for Low Power Dissipation," *ACM ISLEP, Rapallo, Italy, 2000.*

[3] N. Abhoghazaleh et al, "Energy Management for Real Time Embedded Applications with Compiler Support," *ACM LCTES'03, San Diego, California, USA, June 11-13, 2003.*

[4] L. Soengsoo and S. Takayasu, "Runtime Power Control Scheme Using Software Feedback Loop for Low-Power Real Time Applications," *IEEE Transactions ISBN 0-7803-5974-7, 2000.*

[5] G. Magklis, M. L. Scott, G. David, H. Albonesi, and S. Dropsho. "Profile-Based Dynamic Voltage and Frequency

Scaling for a Multiple Clock Domain Microprocessor," *ACM ISLEP'02, August 2002.*

[6] T. Ball and J. R. Larus "Efficient Path Profiling," *Proceedings of the 29th Annual IEEE/ACM International Symposium, 2004.*