Energy Efficient Software Through Dynamic Voltage Scheduling

Gangadhar Konduri, James Goodman, Anantha Chandrakasan
Department of EECS,
Massachusetts Institute of Technology, Cambridge

ABSTRACT

The energy usage of computer systems is becoming important, especially for portable battery-operated applications and embedded systems. A significant reduction in the energy consumption of a program can be achieved via code optimizations that transform the code to take advantage of the Instruction Set Architecture (ISA) of the target processor. These code optimizations result in significantly shorter execution times of the software which in turn allow the operating frequency of the processor to be reduced, while maintaining the same throughput as the conventionally coded application. This reduction in operating frequency allows dynamic voltage scheduling to be applied to the processor, which results in energy reduction. This work describes how the operating frequency and the supply voltage can be changed on a low power microprocessor. The examples used show that over an order of magnitude reduction in the energy consumption is possible by using the aforementioned techniques.

1. Introduction

With the proliferation of portable, battery-operated devices such as digital cellular phones and personal digital assistants, the energy consumption of such systems is gaining more importance. For longer battery life, the programs running on these systems have to consume lower energy.

For CMOS design, the switching energy-per-operation of a microprocessor follows EQ 1, where C is the switched capacitance and V_{DD} is the voltage at which the circuit is operating.

$$E_{op} \propto C V_{DD}^{2} \quad (1)$$

The energy consumption of the program is the sum total of the energy consumed by all the operations performed in the program. Hence, there are three ways in which the energy consumption of a program can be reduced: by reducing the number of operations performed, by reducing the switched capacitance of each operation, or by reducing the voltage at which these operations are performed. Several techniques [1] have been developed to reduce the switched capacitance of operations to enable low energy computation. In this paper, we only explore the other two approaches.

The number of operations required to perform a particular task can be reduced by optimizing the code for a given processor. By making full use of the processor's Instruction Set Architecture (ISA), the number of operations (and hence the energy consumption) of the program can be reduced. However, some instructions consume more energy than other instructions either because they draw more supply current(i.e., due to more switching) or because they take more clock cycles for execution. So, care should be taken while reducing the number of operations during the optimization phase.

Reducing the operating voltage results in a quadratic reduction in the energy consumption. The propagation delay for a logic gate is, to the first order, inversely proportional to the voltage. Hence, at lower voltages the cycle time must be increased (i.e., the clock frequency must be lowered). The program takes the same number of cycles, but a longer time to execute. However, the energy consumption is lower because the operating voltage can be reduced. It is important to note that decreasing frequency alone does not reduce the energy consumption, although it does reduce the power dissipation. The energy consumption decreases only when the operating voltage is reduced.

In Section 2, the effect of code optimization on the energy consumption of programs is examined using an example application. In Section 3, the effect of dynamic voltage scheduling on the energy consumption of programs is studied.

2. Code Optimization

Traditionally, code optimization refers to the transformation of the code to reduce the execution time of the programs. These transformations can be used to reduce the energy consumption of programs as they use fewer instructions to perform the task. In addition, by profiling the energy consumption of individual

0-7803-5471-0/99/$10.00©1999 IEEE

instructions, new code transformations can be performed to use equivalent operations that are more energy efficient. For example, efficient utilization of the data caches and register banks results in fewer external memory operations. Code optimizations that perform these transformations result in programs with lower overall energy consumption both within the processor and at the system level (e.g., fewer energy-intensive external memory accesses).

Further optimizations are possible by adapting the algorithm to exploit special features of the target processor such as multi-precision multiply-accumulate (MAC) structures, and multi-media specific instructions such as those in MMX-enhanced processors. Also, the assembly code produced after compiling the source code in a high level language (e.g., C) can be optimized in order to take advantage of such features.

The energy-savings that can be achieved using the aforementioned optimizations are best demonstrated through the use of an example application. All of the measurements were taken using the SA-1100 microprocessor, a low power microprocessor from the StrongARM family [3]. The processor was driven by an external voltage. The power supply current and the execution time of the programs were then measured in order to calculate the total energy consumption of the program. All measurements were taken with the processor operating at a supply voltage of 1.5 Volts. Both the Instruction and the Data caches were enabled when the program was executed.

The application used is an implementation of a cryptographic algorithm known as the Quadratic Residue Generator (QRG [2]). The QRG performs repeated modular squaring operations of the form:

$$x_{i+1} = x_i^2 \bmod n \qquad (2)$$

where n is the product of two large prime numbers. The least significant $loglog\ n$ bits of each result can then be extracted to form a cryptographically-secure pseudo-random sequence that can be used to encrypt/decrypt data streams. The size of the modulus n (e.g., 512 - 1024 bits) requires the use of multi-precision arithmetic which is implemented using both generic-C and highly-optimized assembler code.

Optimization can be performed at both the algorithmic level and at the instruction level. In this example, at the algorithmic level the symmetry of the squaring operation is exploited in order to halve the number of single-precision multiplications that are required to perform a multi-precision multiplication. At the instruction level, the assembly-code implementation was transformed to exploit the processor's built-in 32x32-bit multiply, 64-bit accumulate structure in order to speedup the multiplication/squaring operations and reduce the number of external memory accesses. The generic C implementation only allows access to the least significant 32-bits of the product which, in turn, requires the operands to be processed 16-bits at a time to avoid overflow. Given that multi-precision multiplication is an $O(n^2)$ operation, this leads to significant degradation in performance. A multiply/add pair of instructions consumes 23% more energy than a single multiply-accumulate (MLA) instruction, which can be used for performing the same task. The MLA instruction takes the same number of cycles to complete as the MUL instruction.

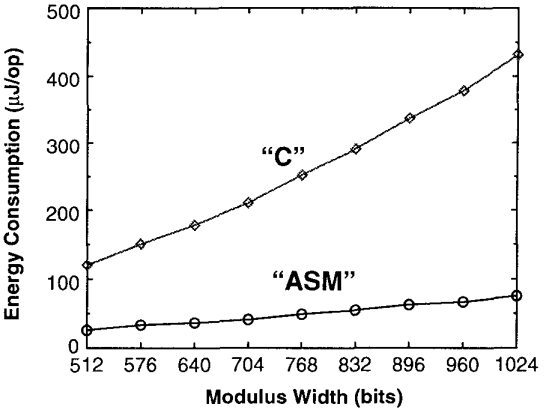

Figure 1. Energy Consumption of QRG Application

The energy consumption of the unoptimized and the optimized code implementing the QRG was measured for a variety of moduli, varying in size from 512 to 1024 bits, and the resulting energy consumption is shown in Figure 1.

The multiplier supports early termination and can complete in 1,2 or 3 cycles depending on the magnitude of the first operand[4]. If bits 31 to 11 of the first operand are all 0 or 1 (sign-extension), the multiplication completes in only 1 cycle. If bits 31 to 23 are all 0 or 1, then the multiplication takes 2 cycles to complete. In all other cases, it takes 3 cycles to complete. Since this implementation treats operands differently, it makes the MUL and MLA operations not commutative in the energy consumption in some cases and this property can be utilized in the programs, where the behavior of operands is known, to reduce the energy consumption. However, it could not be utilized in this example because we used randomly generated input data.

Some other processor-specific features that can be utilized to reduce energy consumption are multiple shifts in a single cycle using the barrel shifter, load and store multiple instructions, and conditional execution on instructions to eliminate some branches and their associated branch penalties. In particular, the load and store multiple instructions can be used to efficiently implement the context switching involved in the method calls that occur in the programs.

3. Dynamic Voltage Scheduling

Dynamic Voltage Scheduling (DVS) refers to the technique of scheduling the optimum V_{DD} for the operations to obtain energy savings. Reducing the operating voltage should be accompanied by reduction of the clock frequency to ensure the correct operation at that voltage. DVS results in significant savings in the energy consumption of the overall program because it reduces the energy consumption of operations. This control of clock speed can be done by the Operating System Scheduler. Such dynamic speed setting policies have been evaluated by trace-driven simulations on Unix workstations in [5],[6]. Some DVS algorithms for interval-based voltage schedulers have been analyzed with simulation results in [7].

The energy dissipation of CMOS circuits is given by equation:

$$E = CV_{DD}{}^2 f\left(\frac{N}{f}\right) + I_{Leakage} V_{DD}\left(\frac{N}{f}\right) \qquad (3)$$

where f represents the clock frequency, V_{DD} denotes the operating voltage, $I_{leakage}$ represents the leakage current, N represents the number of cycles for which the program is run and C denotes the switched capacitance. As can be seen from EQ 3, reducing V_{DD} decreases the first term quadratically. As explained in Section 1, the reduction in the voltage should be accompanied by a corresponding decrease in the clock frequency, which increases the second term. The reduction in V_{DD} and the decrease of leakage currents caused by this reduction offset this increase. Typically, the leakage currents are very small and therefore the changes in the second term do not significantly affect the total energy consumption. Thus, although the execution time of the program increases, the programs still require the same number of clock cycles, and because V_{DD} is now reduced, the total energy consumption of the program decreases.

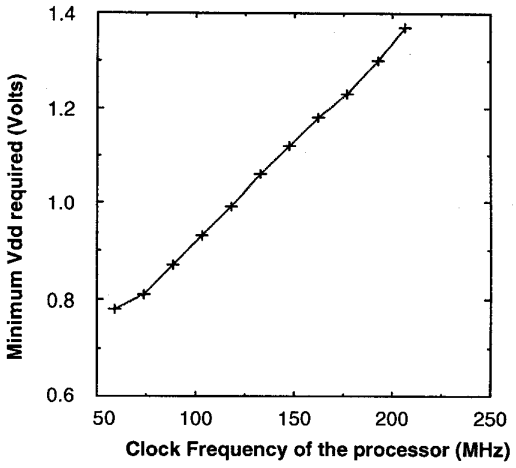

Figure 2. Minimum operating voltage as a function of the clock frequency.

This technique could be utilized when there is a slack in the time required for the job. For example, it could be applied if the time available to do a task is more than the execution time of the program when the processor is running at the maximum frequency. Under such conditions the same task could be accomplished by slowing down the processor, reducing the supply voltage, and using all of the available time in order to reduce energy. This technique could also be used when the energy remaining in the battery is either less than that required to operate the circuits at higher voltages or when the lifetime of the battery for a given function is required to be more than the lifetime achievable by operating the device at the maximum voltage. The increased execution time is the penalty paid in these cases for saving the battery energy through DVS. Energy can also be saved by reducing the internal frequency and the voltage during the idle periods of operation or by reducing the supply voltage to zero in some parts of the circuit during long periods of idleness. An embedded dynamic supply controller [8] would enable the dynamic change of voltage via software control. The latency to change the supply voltage is dependent on the feedback control used. These latencies are typically of the order of 10's - 1000's μs.

Reducing V_{DD} to the minimum required voltage for the operation at that operational frequency is the key to this technique because it leverages the maximum savings in the energy consumption. The minimum V_{DD} values for the various frequencies at which the processor can run has to be calibrated and shown in Figure 2. The voltage of operation scales almost linearly with the frequency.

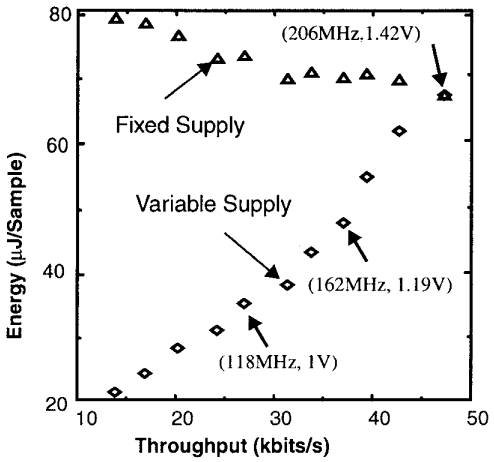

Figure 3. Energy dissipation per sample obtained by optimizing V_{dd} and clock frequency.

The time-energy tradeoff involved in DVS is best illustrated through the use of a simple example.Suppose a particular task has 75% process utilization when the processor is run at the frequency of 200 MHz and 1.5 Volts. By reducing the clock frequency to 150 MHz and the voltage to 1.2V, the energy consumption of the program is reduced by approximately 20% without any performance degradation because of the lower processor utilization in the first case.

The trade-off of how the energy consumption of a program varies with the time required to perform the task (assuming all the time is used to do the same task), using the technique of DVS, has been studied. The frequency and the operating voltage are the factors varied in this experiment. To demonstrate this idea, the optimized assembly code of the QRG algorithm used in the previous section was executed on the microprocessor, whose internal clock frequency can be changed via software control. This chip supports dynamic clock frequency switching for reduced operating power. The code is executed using a 1024 bit modulus and both the energy consumption and the throughput are calculated using the measured power supply current and the execution time. The total energy consumption is plotted against throughput in Figure 3. For comparison purposes, the energy consumed by the same program at a fixed voltage, but at different frequencies of operation, is also plotted. The energy consumption in this case increases, but not significantly, with the reduction in frequency, because of the leakage factors.

Significant energy savings were observed using both dynamic voltage scheduling and code optimization techniques. As a comparison, the energy consumption of the C program (for 1024 bit key per operation) with no dynamic voltage scheduling and at the maximum voltage of operation was 432μJ. Code optimization gave savings of more than 80%, bringing down the energy consumption to 75.6μJ. After voltage scaling, the energy consumption was reduced to 21.3μJ, giving savings of 95% when compared with the energy consumed by the original unoptimized C code.

4. Conclusion

Significant reductions in the energy consumption of programs can be achieved through code optimizations and the technique of dynamic voltage scheduling. The trade-off involved in this technique is the extra time taken to execute the program and this makes it applicable in the cases where the extra time is available. In order to fully exploit the benefit of variable voltage, efficient switching regulators must be developed and incorporated into software systems.

Acknowledgment

This work is funded by DARPA.

References

[1] A. Chandrakasan, R. Brodersen, Low Power CMOS Design, IEEE Press, 1998.

[2] L. Blum, M. Blum, and M. Shub, "A Simple Unpredictable Pseudo-Random Number Generator", SIAM Journal on Computing, v. 15, n. 2, pp. 364-383, 1986.

[3] J. Montanaro et.al., "A 160-MHz, 32-b, 0.5-W CMOS RISC processor", IEEE Journal Of Solid-state Circuits, pp.1703-1714, November 1996.

[4] SA-110 Microprocessor Instruction Timing Application Note, URL: http://www.intel.com/design/strong/techdocs/278194.htm

[5] M. Weiser et.al., "Scheduling for reduced CPU energy", Proc. of first symposium on OS Design and Implementation, pp. 13-23, November 1994.

[6] K. Govil, E. Chan, H. Wasserman, "Comparing algorithms for dynamic Speed-Setting of a Low Power CPU", International Conference on Mobile Computing and Networking, pp. 13-25, November 1995.

[7] T. Pering, T. Burd, R. Broderson, "The simulation and evaluation of dynamic voltage scaling algorithms", ISLPED, pp. 76-81, August 1998.

[8] V. Gutnik, A. Chandrakasan, "Embedded power supply for low power DSP", IEEE Trans. on VLSI Systems, pp. 425-435, December 1997.