The Influence of Microprocessor Instructions on the Energy Consumption of Wireless Sensor Networks

Nicholas D. Lane and Andrew T. Campbell
Computer Science Department, Dartmouth College
Hanover, New Hampshire, USA
niclane,campbell@cs.dartmouth.edu

***Abstract*— The field of low power compilation could be applied to sensor networks to yield significant savings for the sensing, computation, and communications processes in sensor networks. Such savings could come via simple low power savings flags for future compilers used by sensor network developers. In this paper, we instrument the Moteiv Tmote Sky as a representative sensor, and conduct a set of experiments to study the impact of instruction types, circuit state effects, instruction operand ordering, memory addressing modes, and existing GCC compiler optimization flags on energy consumption. We apply a simple instruction exchange technique to an existing sensor application for a modest gain in energy savings.**

I. Introduction

Energy conservation has been a driving factor in the design of new hardware and software for wireless sensor networks over the last several years. Many energy efficient systems have been proposed by the community, spanning architectures [2], operating systems [3], sensor usage [11], and algorithm and protocol design [16] [5] [4]. One technique not applied to sensor networks to date, which might deliver more energy efficiency savings across the board, is low power compilation. Researchers in the low power compilation community have developed a number of techniques [19] [20] [9] that can optimize the compilation process for energy consumption when targeting specific systems design. These researchers observed that the standard optimization techniques and heuristics offered by today's compilers do not result in binaries that are optimized for energy efficiency. Rather, compilers are designed to optimize the binary for execution performance and to a lesser degree binary size. While there has been work on low power compilation we are not aware of it being applied to sensor networks. The simple question that we start to answer in this paper is can we leverage or adapt the body of work in the field of low power compilation and apply it to sensor networks to gain additional energy savings?

Existing low power compilation techniques are based on an understanding of the relationship between the execution of computational workloads on a specific targeted processor and the associated energy consumption observed. The contribution of our paper only represents a first step at studying this relationship when considering the Tmote Sky [14] sensor device. We conduct a number of experiments using the Tmote Sky in order to characterize this relationship. We conjecture that our results and findings can assist in the future designs of compilers that can specifically exploit sensor hardware design and application workload to reduce the energy consumption. For example, one very simple optimization, derived experimentally, results in a modest but measurable reduction in an existing sensor network application's energy consumption of 4.3%. We conjecture more work on low power compilation techniques can lead to more significant savings, but may require accepting tradeoffs. We discuss these tradeoffs and use the results from a set of experiments to argue for more work in the area of low power compilation for sensor networks.

The structure of the paper is as follows. Section II discusses existing techniques and related work in the field of low power compilation. Following this in Section III, we describe a set of experiments and results that characterize the energy consumption for different workloads for the Tmote Sky. The results from these experiments lead to a number of possible optimization scenarios and recommendations for low power compilation in sensor networks that we discuss in Section IV. We present some concluding comments in Section V.

II. Prior Work on Low Power Compilation

A number of papers have considered the concept of low power compilation typically considering more capable devices such as desktop class processors [20] or even more sophisticated processors [21]. Perhaps more related to sensor hardware is work that has considered digital signal processors (DSPs), and forms of mobile and embedded computing [20]. The authors of [20] observe that the energy consumption of the multiplication instruction can be lowered by 30% simply by switching the order of the input operands under certain circumstances. The authors in [20] study other forms of instruction-level power optimizations across a group of processors finding that some processors are responsive to the rescheduling of instructions while others did not show much advantage. For example, the Fujitsu DSP [20] under study achieves reductions that range between 7.4% and 24.0% based on the specific workload. These were the largest energy reductions seen in the group of processors considered. In contrast, the Intel 486DX2 experiences the smallest levels of energy reduction in response to the techniques tested. Results of low power optimizations for DSPs suggest that simple hardware, such as that found in sensor hardware design, might be responsive to simple techniques such as instruction rescheduling to gain energy savings.

On the other hand, more sophisticated techniques discussed in the low power compilation literature are not appropriate at this time, given the state of sensor hardware. For example, in [7] the authors examine the relationship between energy consumption and standard compiler loop based optimizations such as scalar expansion, loop fission, loop unrolling and forms of linear loop transformations. Assumptions made by the low power compilation community such as the presence of caches, deep pipelines and multiple functional units are simply not present in current sensor device hardware, such as the MSP430 processor used in the Tmote Sky.

III. Experimental Analysis

Toward providing low power compilation for the Tmote Sky sensor device, our goal is to start to quantify the relationship between workloads executed on the sensor and the resulting energy consumption. We are interested in ascertaining the difference between energy consumed while the processor is executing different workloads to help determine (i) the extent of energy consumption savings possible using compiler based techniques and (ii) the types of optimizations that will increase or decrease energy consumption. We first discuss the experimental setup and then the experimental observations. At this stage, it is not clear whether these results would be generally applicable to other sensor hardware without repeating the analysis discussed below.

A. Experimental Testbed

For our experiments we use the Moteiv Tmote Sky sensor [14]. This device includes 10KB of RAM, 48KB of ROM, 1 MB of flash storage, and uses a variant of the MSP430 [6] processor manufactured by Texas Instruments. The MSP430 is a 16-bit RISC based processor that uses a 3 stage pipeline with 16 general purpose registers. The instruction set comprises 27 instructions with 7 memory addressing modes available [6].

Our experimental setup is as follows. An Agilent E3610A Power Supply provides power to a circuit comprising the Tmote Sky and a 101 ohm resistor R. An Agilent 54621D Mixed Signal Oscilloscope measures the voltage drop across the resistor R using 10:1 probes. The Tmote Sky is modified by soldering terminals to the ADC0 and ADC ground connectors that are part of the 10 pin expansion connector [13]. An additional set of 10:1 probes are connected from these soldered terminals to the secondary input on the oscilloscope to monitor voltage changes. We determine the power consumption of Tmote Sky using time series measurements of the voltage drop across R together with the supply voltage. To allow the measurements made during experiments to be correlated to the execution of instructions in the experiment workload we follow a similar methodology to that used in [17]. Instructions to set and clear the ADC0 pin are inserted to mark the beginning and end of the experiment workload. The power consumed during the execution of the experiment workload is then isolated by considering only the voltage dropped across R in between voltage changes at the ADC0 pin.

For experiments in which we measure a small number of processor instructions we modify our methodology. In such cases, we use workloads that perform the same sequences of processor instructions multiple times within a loop. This loop body iterates the instructions a number of times to allow measurements to be made in aggregate, minimizing the effect of overhead and other variants unrelated to the instruction sequences themselves. The setting and clearing of voltages across the two terminals is used but this time to identify the start end end points of the loop. The energy consumption of an instruction sequence is determined by comparison to a baseline measurement made with an empty loop body. The energy consumption of the empty loop body is subtracted from the energy consumption of the loop body containing the workload. This difference is then divided by the quantity of loop iterations to calculate the cost per sequence.

B. Baseline Observations

We perform experiments that include a number of different workload types. These experiments study the impact of instruction type, circuit state effects, instruction operand ordering, memory addressing modes within instruction operands, different sensor network applications, and GCC compiler optimization flags.

Two simple techniques used in low power compiler instruction-level optimization [9] are instruction scheduling and register assignment. These techniques require few processor architectural components and are thus appropriate for the simpler processors found on typical sensors. One factor that determines the effectiveness of techniques based on register assignment is the variation in energy consumption observed when accessing operand values using resisters versus different forms of memory addressing. The larger the variance in energy consumption that is observed the higher the potential gains from register assignment.

Similarly, one factor that determines the effectiveness of techniques based on instruction scheduling is the logical configuration of the processor and it's subsystems. This logical configuration represents the *circuit state*. Each processor instruction requires a particular circuit state. Changing from one state to another has an associated energy cost that depends on the extent of the required configuration change from one instruction to the next. The variance in power consumption due to the circuit state is called the circuit state effect. The larger the variance in energy consumption as a result of the circuit state effect the higher potential for instruction scheduling to be effective.

In what follows, we report on the experimental results due to instruction type, circuit state, and memory address mode. Table I reports on the energy consumption variation for a portion of the instruction set, and Table II reports on the energy consumption variation due to the memory address mode of instruction operands. Due to space constraints we only present a representative selection of the complete set of results; see [8] for a more comprehensive set of results. Tables I and II report the energy consumption for individual instructions using two metrics; these are *energy*, reported in nano-joules per instruction and *normalized energy*, reported in nops per instruction. A nop (*no-operation*) instruction

Instruction	Energy (nJ)	Normalized Energy (nops)
mov	4.4	1.01
add	4.6	1.05
sub	4.7	1.08
cmp	4.7	1.12
bit	4.7	1.06
xor	4.8	1.08
and	4.8	1.07
bic	4.9	1.09
bis	4.9	1.12

TABLE I
ENERGY CONSUMPTION VARIATION DUE TO INSTRUCTION TYPE

Operand Memory Modes	Energy (nJ)	Normalized Energy (nops)
indexed to register	13.2	2.99
register to indexed	17.0	3.87
register to absolute	15.8	3.60
indirect to register	9.0	2.04
immediate to register	9.0	2.06
absolute to register	26.1	5.95
register to indirect	17.5	3.97

TABLE II
ENERGY CONSUMPTION VARIATION DUE TO THE MEMORY ADDRESS MODE OF OPERAND

performs no actual operation but is used often as a type of place holder instruction for timing or alignment issues that occur during instruction execution within a processor. The normalized energy metric is the ratio of the energy spent performing a given instruction to the energy spent performing a nop.

Instruction Type. A measurable variance in the amount of energy used per instruction type has been observed in the literature for other hardware (e.g., [19]). We assess the energy consumption due to the execution of different types of the MSP430 processor [6] instructions used by the Tmote Sky. We do this by keeping other factors constant that could induce variance in the energy consumption, such as the type of memory address mode used by the operands. Table I reports measurements made with instructions using only register based operands. A 10.8% spread of energy consumption due to the processor instructions is observed.

Circuit State. Next, we test for variance in energy consumption due to a change in circuit state by executing instruction pairs, holding the second instruction constant while varying the first. In limited testing of 20 instruction pairs, the largest difference in energy consumption attributable to circuit state change is 0.88*nops*, while the smallest difference is 0.18*nops*. Tabulated results of the tested pairs are not shown in this paper due to space constraints but can be found in [8]. We are currently testing the remainder of the 27^2 pairs for completeness.

Memory Address Mode. The operations that occur as a result of particular instructions within the MSP430 vary depending on the memory address mode currently in use. For instance in "indirect register" addressing mode the operand specifies a register used as a pointer to a location in memory, while in "absolute" addressing mode the operand specifies the memory address directly. A subset of our results for the *mov* instruction using different operand addressing modes are presented in Table II. To evaluate the effect of the memory address mode we compare the energy consumption of the *mov* instruction in different memory address modes. We keep other factors that could potentially impact the results, such as the specified memory address, constant. The results indicate that there is a large variation of energy consumption which is dependent on the memory addressing mode. In particular, the *mov* instruction consumes nearly three times more energy in the "absolute to register" case than in the "indirect to register" case.

C. Impact on a Reference Application

To provide a first order assessment of the potential impact of compiler optimizations for low power operation we examine an existing sensor network application. Guided by the simple experimental observations discussed in Section III-B we aim to find a simple instance where the replacement of a sequence of instructions results in a reduction in energy consumption of the application. We examine EccM [12], an implementation of public key encryption for sensor network communication that generates keys based on elliptical curve cryptographic techniques. Using the same experimental setup and methodology as in Section III-B we first instrument the code so we can measure the energy consumption of the generation of a single key. This instrumented binary is generated with the GCC compiler flag -O4 which results in the maximum possible extent of compiler optimizations being performed. This analysis shows that generation of a 163 bit key requires approximately 800 mJ of energy.

Next, we profile the key generation process by instrumenting appropriate function calls in the application and using the TOSSIM simulator [10]. This is done in an effort to find frequently executed code to optimize for low power operation. We examine the assembly instructions for a frequently executing loop identified using TOSSIM. Within the loop body we identify a particular group of instructions with operands using the indexed memory addressing mode, which includes a base plus offset memory access component. However for this group of instructions within the loop the offset is identical. Therefore, indexed mode is not required and can be replaced with indirect mode by changing the initialization of the base address to that of the base address plus the constant memory offset. Such a replacement choice is motivated by observations discussed in Section III-B and in particular shown in Table II. With these changes, the energy consumed during the the generation of a single key drops by 4.3% for the key generation. While such a result is inconclusive in determining the utility of low power compilation it is nonetheless a promising start. Potentially a series of similar optimizations applied throughout an application would provide a more significant energy savings.

IV. Toward Low Power Compilation for Sensor Networks

The underlying motivation for this work is to develop compilation tools that produce binaries optimized for low power consumption in sensor networks. These can increase energy efficiency through compilation without any change to the source code or effort by the developer. The existing compilation process of a nesC application involves preprocessing to enable compilation to be performed by a GCC C cross compiler. In the standard TinyOS [3] development environment the MSPGCC cross compiler [1] is used to generate a binary for the Texas Instruments MSP430 microprocessor used by the Tmote Sky. The compilation process performs no optimizations specifically with the purpose of energy optimization. The optimizations that do occur are principally focused on optimizing the execution time. These optimizations are applied without consideration to the specifics of the sensor device hardware or the nature of sensor networks and their applications. In the following, we consider some of the key areas that need to be addressed in future work.

A. Viability of the Approach.

Compilation techniques only directly alter the energy efficiency of the processor. However, the processor is not the only contributor to energy consumption on a sensor device. Components such as the transceiver are only indirectly influenced by the processor's execution yet they influence energy efficiency a great deal. Other components such as the voltage converter in a Mica2 sensor was observed within the GDI [18] deployment to strongly influence battery utilization and thus energy efficiency. Perhaps the most significant contributor to energy conservation is the influence of higher level software system components of the sensor such as the specific characteristics of the operating system and networking protocols. All of these factors limit the ability for compilation-based techniques to reduce overall energy consumption. It is therefore necessary to perform more experiments to determine the extent to which forms of compilation techniques can reduce energy consumption, in the context of overall platform operation.

B. Defining Compiler Behavior.

The implications of existing forms of compilation optimizations on energy conservation need to be also further assessed. Following this there is a need to develop new optimizations specifically tailored toward energy savings. Existing low power compilation literature can assist somewhat in this process. Beyond developing a set of suitable optimizations to apply, the next challenge is to define the compiler heuristics that determine under which conditions such optimizations would be useful. One element that needs to be incorporated into this process is the effect of optimizations on battery utilization. The influence of the battery discharge curve on sensor energy efficiency is already established. As observed in the GDI sensor network deployment [18] substantial influence on the network lifetime can occur due to the difference between the actual battery discharge as opposed to expected battery discharge. Instances were observed during the deployment of as much as 40% of unutilized battery capacity. In [15] the authors find that a 52% increase in data collection could be achieved over the deployment lifetime if such battery effects are considered when selecting transmission power levels. Similar optimizations that incorporate awareness of battery discharge curves into low power compilation for sensor hardware may yield significant savings.

C. Augmenting the Compilation Process.

The optimizations performed by a low power compiler benefit from knowledge of the system hardware specifications and application requirements. For such a compiler to be practical it needs to be able to expose the relevant aspects of it's behavior to developers with simple mechanisms (eg. parameters, configuration files). Developers and not compiler experts need to be able provide such requirements without needing to modify internal compiler components.

Application Requirements. Sensor applications may have performance requirements for the operation of their code. Compiling for energy efficiency may degrade execution speed performance since fewer speed optimizations are applied due to higher energy cost. Providing the application developer the ability to manage this tradeoff would be of value. A finer grained approach maybe more appropriate rather than control via compiler flags that simply alter the degree of low power optimization performed. An example is providing a means to indicate (e.g., using preprocessor directives) critical sections of code that should be optimized for speed rather than energy.

System Hardware Specifications. Knowledge of the characteristics of the hardware platform target is required for many types of low power compilation techniques. For instance aggressive register allocation and instruction ordering are the basis for a number of existing techniques but to be effective these types of optimizations require specific knowledge of the target hardware, such as the register counts, clock speeds, aspects of the architecture design. Typical compilers do not have this level of awareness.

V. Conclusion

In this paper, we proposed applying techniques used in the area of low power compilation to sensor networks in order to potentially extend the lifetime of such networks. We studied the impact of instruction types, circuit state effects, instruction operand ordering, memory addressing modes and GCC compiler optimization flags on energy consumption. We reported a subset of our findings which we used to identify a simple instruction level optimization to an existing sensor application. The application of this optimization resulted in a modest 4.3% reduction in the power consumption of the application. This result in itself is inconclusive as to establishing the utility of this approach but it does suggest further examination is warranted. While this paper presents more questions than it answers our main objective is to raise the issue of developing new low power compiler optimization techniques for sensor networks.

ACKNOWLEDGMENT

We would like to thank Ronald Peterson, Satish Prabhakaran and Nihal D'Cunha for their invaluable assistance, particularly with regard to the experimental design and the interpretation of results presented. Nicholas Lane is supported by Grant number 2005-DD-BX-1091 awarded by the Bureau of Justice Assistance through the Institute for Security Technology Studies, Dartmouth College.

REFERENCES

[1] Mspgcc. http://mspgcc.sourceforge.net/, 2006.

[2] M. Hempstead, N. Tripathi, P. Mauro, G.-Y. Wei, and D. Brooks. An ultra low power system architecture for sensor network applications. *SIGARCH Comput. Archit. News*, 33(2):208–219, 2005.

[3] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. S. J. Pister. System architecture directions for networked sensors. In *Architectural Support for Programming Languages and Operating Systems*, pages 93–104, 2000.

[4] B. Hohlt, L. Doherty, and E. Brewer. Flexible power scheduling for sensor networks. *Information Processing in Sensor Networks (IPSN 04)*, April 2004.

[5] J. W. Hui and D. Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. *The 2nd ACM Conference on Embedded Networked Sensor Systems*, November 2004.

[6] T. Instruments. Msp430x2xx family user's guide (rev. b). http://www-s.ti.com/sc/psheets/slau144b/slau144b.pdf, 2006.

[7] M. Kandemir, N. Vijaykrishnan, M. J. Irwin, and W. Ye. Influence of compiler optimizations on system power. In *DAC '00: Proceedings of the 37th conference on Design automation*, pages 304–307, New York, NY, USA, 2000. ACM Press.

[8] N. D. Lane. Exploiting microprocessor instructions to manage energy consumption in wireless sensor networks. http://www.cs.dartmouth.edu/ niclane/pubs/emi2meciwsn.pdf, 2005.

[9] C. Lee, J. K. Lee, T. Hwang, and S.-C. Tsai. Compiler optimization on instruction scheduling for low power. In *ISSS '00: Proceedings of the 13th international symposium on System synthesis*, pages 55–60, Washington, DC, USA, 2000. IEEE Computer Society.

[10] P. Levis. Tossim: Accurate and scalable simulation of entire tinyos applications. In *Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003), November 2003.*, 2003.

[11] H. Liu, A. Chandra, and J. Srivastava. dsense: Data-driven stochastic energy management for wireless sensor platforms. http://www.cs.umn.edu/tech_reports_upload/tr2005/05-018.pdf, 2005.

[12] D. Malan, M. Welsh, and M. Smith. A public-key infrastructure for key distribution in tinyos based on elliptic curve cryptography. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, New York, NY, USA, 2004. ACM Press.

[13] Moteiv. Tmote sky datasheet. http://www.moteiv.com/products/docs/tmote-sky-datasheet.pdf, 2005.

[14] Moteiv. Tmote sky. http://www.moteiv.com/products-tmotesky.php, 2006.

[15] C. Park, K. Lahiri, and A. Raghunathan. Battery Discharge Characteristics of Wireless Sensor Nodes: An Experimental Analysis. In *SECON '05: Proceedings of the Second Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, September 2005.

[16] J. Polastre and D. Culler. Versatile low power media access for wireless sensor networks. *The Second ACM Conference on Embedded Networked Sensor Systems*, November 2004.

[17] V. Shnayder, M. Hempstead, B. rong Chen, G. W. Allen, and M. Welsh. Simulating the power consumption of large-scale sensor network applications. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 188–200, New York, NY, USA, 2004. ACM Press.

[18] R. Szewczyk, J. Polastre, A. Mainwaing, and D. Culler. Lessons from a sensor network expedition. *European Workshop on Sensor Networks (EWSN)*, January 2004.

[19] V. Tiwari, S. Malik, and A. Wolfe. Compilation techniques for low energy: An overview. In *Proc. of Symp. Low-Power Electronics, 1994.*, 1994.

[20] V. Tiwari, S. Malik, A. Wolfe, and M. T.-C. Lee. Instruction level power analysis and optimization of software. *J. VLSI Signal Process. Syst.*, 13(2-3):223–238, 1996.

[21] M. Valluri, L. John, and H. Hanson. Exploiting compiler-generated schedules for energy savings in high-performance processors. In *ISLPED '03: Proceedings of the 2003 international symposium on Low power electronics and design*, pages 414–419, New York, NY, USA, 2003. ACM Press.