In Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, volume 2, pages 45-48, Honolulu, Hawaii, April 2007.

ENERGY-AWARE DATA COMPRESSION FOR WIRELESS SENSOR NETWORKS

Sebastian Puthenpurayil, Ruirui Gu, Shuvra S Bhattacharyya

Department of Electrical and Computer Engineering, and
Institute for Advanced Computer Studies
University of Maryland
College Park, MD, 20742, USA
Email: purayil@umd.edu, rgu@umd.edu, ssb@umd.edu

ABSTRACT

Data compression techniques have extensive applications in power-constrained digital communication systems, such as in the rapidly-developing domain of wireless sensor network applications. This paper explores energy consumption trade-offs associated with data compression, particularly in the context of lossless compression for acoustic signals. Such signal processing is relevant in a variety of sensor network applications, including surveillance and monitoring. Applying data compression in a sensor node generally reduces the energy consumption of the transceiver at the expense of additional energy expended in the embedded processor due to the computational cost of compression. This paper introduces a methodology for comparing data compression algorithms in sensor networks based on the figure of merit D/E, where D is the amount of data (before compression) that can be transmitted under a given energy budget E for computation and communication. We develop experiments to evaluate, using this figure of merit, different variants of linear predictive coding. We also demonstrate how different models of computation applied to the embedded software design lead to different degrees of processing efficiency, and thereby have significant effect on the targeted figure of merit.

Index Terms— DSP software, lossless data compression, linear predictive coding, low power design, wireless sensor networks.

1. INTRODUCTION

Acoustic sensors or seismic recorders acquire signals from acoustic sources and transmit those signals to end users or servers. Signal processing techniques are extensively used in such sensor node applications and also in telecommunications equipment such as gateway station controllers for wireless communications. These applications, and the subsystems to which they interface, involve energy consumption through computation, transmission, and reception. The major source of energy consumption is in the transceiver portions and some form of energy savings can be attained by keeping the transceiver in a standby mode and switching to the operational mode only when there is data for transmission. We can reduce the time the transceiver is in the operational mode if the volume of data to be transmitted is reduced by compression. Reduction in data size for this purpose can give considerable savings in power, which is required in energy-limited applications.

In this paper, we model lossless data compression algorithms and compare their energy efficiency trade-offs between computation and communication. Our comparison is in terms of a metric, defined in Section 3, that is geared towards considering the integrated effect of data compression on computation and communication. In implementation approach, we apply dataflow models to represent candidate compression algorithms, and map them into efficient embedded software realizations. This provides a formal link between algorithm representation and hardware/software optimization that is useful in maximizing the overall impact of compression.

We evaluate different variants of *linear predictive coding* (*LPC*) using our proposed figure of merit, and different forms of dataflow modeling — based on the degree of dynamics in the LPC variants — for implementing the embedded software realizations. Specifically, we experiment with a static version that of LPC that is implemented based on *synchronous dataflow* (*SDF*) modeling [1]; a dynamic version, which is more flexible in its operation, and is implemented based on *parameterized synchronous dataflow* (PSDF) modeling [2]; and a dynamic cyclo-static version that is implemented based on *parameterized cyclo-static dataflow* (*PCSDF*) modeling [2, 3]. We refer to these variants respectively as static-version LPC (*SVLPC*), dynamic-version LPC (*DVLPC*), and dynamic cyclo-static LPC (*DCLPC*).

2. BACKGROUND AND RELATED WORK

Data compression shrinks raw data to smaller volumes, which is desirable for data communication since less data requires less time and less energy for transmission and reception. Previous research on data compression for communication mainly

focuses on how to decrease the communication delay or the required transmission bandwidth. Hans and Schafer [4] present an overview of lossless data compression in the context of audio data. With the emergence of many severely energy-constrained application domains, such as various forms of sensor networks, the topic of energy efficiency is becoming increasingly important. Zhang and Li [5] discuss the implementation of compression algorithms for seismic data. This work estimates the energy reduction after compression that is due to data reduction, and considers the energy costs of communication alone or in isolation from the costs of computation.

Our paper differs from this related work in its emphasis on design and implementation considerations that relate compression algorithms with the underlying embedded software. Also, in our experiments, we focus on compression of acoustic signals, although our overall methodology can be applied equally to other types of sensor signals. In our previous research, we have developed preliminary experimental results [6] on energy consumption trade-offs between computation and communication based on the SVLPC and DVLPC compressions methods mentioned above. In this paper, we introduce a figure of merit that captures this trade-off to evaluate the impact of compression in an integrated manner, and we consider a third variant of LPC implementation, DCLPC, that is geared towards adaptive operation and more streamlined software implementation. DCLPC demonstrates how reformulating a compression algorithm (in this case DVLPC) in terms of a more suitable model of computation (in this case PCSDF) can result in a more efficient software realization. Using our figure of merit for energy-aware compression and the three selected variants of LPC, we compare the performance of these different realizations of lossless compression, and demonstrate our methodology for integration and implementation of compression techniques into energy-constrained sensor node platforms.

The DCLPC implementation that is examined in this paper was presented initially in [3], where it has been used as an example for demonstrating novel techniques for modeling and scheduling dataflow graphs. The topic of this paper and treatment of DCLPC is different in that we focus here on integrated consideration of compression performance and energy efficiency.

3. FIGURE OF MERIT

The data compression ratio, expressed as the percentage of compressed data size with respect to the original data size, is a direct metric for quantifying the reduction in data quantity achieved by a compression algorithm. In the application of energy-limited systems, energy consumption is a metric that needs to be considered explicitly. Various metrics are used to consider energy efficiency. For example, Ye uses energy consumption to evaluate the energy efficiency of different communication protocols for sensor networks [7]. The metric of network lifetime is also used (e.g., see Bhardwaj [8]), where energy efficiency is represented by the total operational time of a network. Ammer and Rabaey [9] propose a metric of energy-per-useful-bit (EPUB) to evaluate and compare sensor network physical layers.

In this paper, we introduce a new performance metric that is geared towards capturing the energy efficiency of data compression algorithms together with their realizations on specific sensor node platforms. We represent this figure of merit as a ratio D/E, which can be evaluated in terms of a given segment of data that is to be compressed and transmitted. Here, D represents size of the data segment (before compression), and E is the total energy consumption for both computation and communication — that is, for the compression of the data, and for communication of the compressed data. Using this metric, we can explore in detail trade-offs involving overall energy efficiency and compression depth when considering different compression algorithms, together with their DSP software realizations, and their targeted embedded processors.

4. SVLPC, DVLPC, AND DCLPC

In the static version of LPC that we consider, the parameters of LPC are constant and known beforehand. Details of the static parameter configuration and SDF modeling used for implementing SVLPC is explained in [6]. In SVLPC, the number of data values (*tokens*) produced and consumed by each software module (dataflow graph *actor*) is constant and known at compile time. This admits a representation in terms of SDF modeling principles [1].

SVLPC has the advantage of being simpler and more predictable due to the underlying SDF structure of the computations. However, the compression ratio of LPC can be improved by dynamically changing its parameters, such as the frame size and model order. For this purpose, we employ a dynamic version of LPC, called DVLPC, where the frame size and model order are determined during run time, as explained in [6]. Our implementation of DVLPC is based on a PSDF representation [2], where the computations are organized in the form of a dynamically reconfigurable SDF graph.

The third form of LPC implementation in our experiments, called DCLPC, can be viewed as a reformulation of the DVLPC-based design in terms of parameterized cyclo-static dataflow (PCSDF) modeling [2]. PCSDF provides for the efficient quasi-static scheduling of PSDF modeling, but also allows for finer grained modeling of dataflow properties, which can lead to more thorough software optimization [3]. Details of the operation and dataflow properties of the DCLPC version are explained in [3].

5. EXPERIMENTAL APPROACH

We have developed embedded software implementations based on the three versions of compression described in the previous section. The computational blocks (dataflow actors) in these realizations are implemented in the C language, and are designed in a modular way so that they can be used conveniently in other applications, such as other compression schemes that we will consider in our future work. The overall C implementation for each application version is derived from the dataflow graph in conjunction with the actor code using methods described in [3].

Our experiments are carried out for two different target processors from Texas Instruments — the TI64xx (using the Code Composer Studio simulation tool) and TI5509 (using the DSK hardware emulator).

6. COMPARISON AMONG SVLPC, DVLPC AND DCLPC

Figure 1 shows a comparison of the energy consumption for the three LPC versions with segment sizes ranging from 50 to 500 samples. In this experiment, the target processor is TI 64XX, with the CPU frequency set at 25 MHz. The energy consumed is given by the product of the power consumed and the time taken to convert the segment size to bits. The time taken is determined by simulation, and the power consumption used in our calculation is obtained from the processor data sheet.

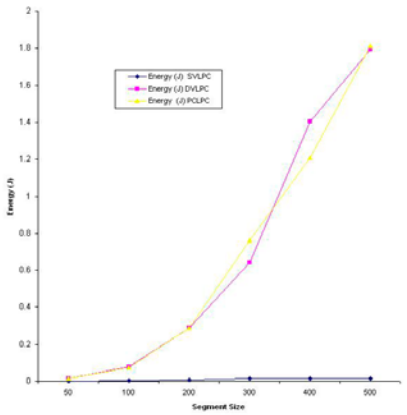

Fig. 1. Energy consumption comparison.

Figure 2 shows a comparison in terms of the figure of merit D/E, which was introduced in Section 3. The comparison here is between the static and dynamic versions, and is based on the number of CPU clock cycles, power consumption, and number of bits after compression. In this experiment, we used the Texas Instruments C5509A DSK hardware emulation board.

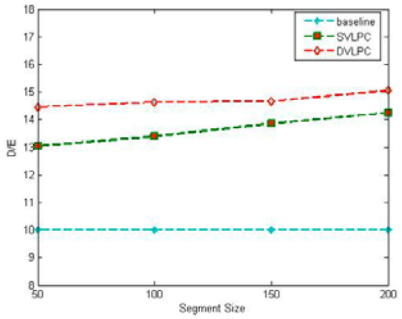

Fig. 2. Comparison of D/E: SVLPC vs. DVLPC.

The results quantify how the additional computational cost of the more intensive DVLPC compression approach is more than compensated by the decreased energy consumption in the transceiver due to reduced transmission volume. The D/E metric also gives a means for evaluating progressively more intensive compression schedules, and determining the point for a particular platform at which the benefits of increased compression achieves saturation or inversion in terms of energy efficiency.

Recall that the DCLPC is an improved version of DVLPC that is reformulated in terms a dataflow format (PCSDF) that provides the same compression behavior but also provides for more thorough software optimization. This results, for example, in more efficient memory usage: the amount of data memory required for the DVLPC implementation is significantly more than for the DCLPC implementation (133,367 bytes vs. 123,332 bytes). If the internal memory is not large to provide for this amount of data storage (along with all other required application functionality) a DVLPC-based implementation will generally have to employ external memory more extensively compared to a DCLPC implementation. This can result in a large performance and energy consumption penalty since off-chip memory requires significantly more energy and time to access.

Table 1 shows the number of clock cycles and figure of merit D/E from DVLPC and DCLPC if compression is run using external memory. These results are obtained using energy consumption monitoring features of the TI C55X power optimization DSK, which monitors DSP current consumption on the processor core, I/O, and board power rails. The results demonstrate significant energy efficiency of DCLPC compared to DVLPC under external memory operation.

7. ADAPTIVE DVLPC

We have integrated SVLPC and DVLPC into a fourth version, which can be viewed as an adaptive form of DVLPC. We thus refer to this version as *adaptive DVLPC*. Adaptive DVLPC employs a feedback approach in which the previ-

Table 1. CPU cycles and D/E from different models

Size	DV(clock)	PC(clock)	DV(D/E)	PC(D/E)
50	12,678,804	11,514,988	32.5972	35.8738
100	28,151,006	26,047,554	29.2992	31.6470
150	43,879,900	40,361,380	28.2039	30.6444
200	62,799,868	60,085,727	26.2760	27.4545

Table 2. Comparison of adaptive DVLPC vs. DVLPC.

Size	D(clock)	A(clock)	D(D/E)	A(D/E)
100	22,999,430	8,556,558	119.1	129.1
200	426,023,069	15,210,157	63.5	177.1
300	941,318,880	17,392,059	50.9	184.5
400	2,065,488,259	155,516,714	32.5	146.8
500	2,637,102,688	198,078,833	32.6	158.2

ously achieved compression depth δ_i (the ratio of compressed data size to original data size) is used to control the model order M of the core LPC compression subsystem. Adaptive DVLPC has an additional computational block that compares δ_i to a pre-determined threshold τ, and provides feedback that is used to guide selection of the model order that will be used for subsequent compression. The model order used in the previous frame can be retained and used for the next frame if $delta_i > tau$. Otherwise, the adaptive operation of the system will produce a new model order M, which will be passed as a parameter value update to the core compression subsystem. In our present approach, the threshold tau is determined experimentally based on the impact on overall D/E.

Table 2 shows a comparison of the CPU cycles and figure of merit D/E between adaptive DVLPC and DVLPC with the Texas Instruments TI64xx as the target processor. We see that the adaptive approach has reduced the CPU clock cycles considerably and in turn obtained more energy efficiency, as indicated by D/E. Using adaptive DVLPC, we can "tighten" or "loosen" the compression configuration based on the energy budget.

8. CONCLUSIONS

Data compression is an effective way to improve energy efficiency in wireless sensor networks. However, to fully understand and optimize the impact of compression in an energy-limited scenario, it is necessary to consider the associated trade-off between computation and communication. In this paper, we have developed a methodology for design and implementation of compression software that is geared toward tuning operation of a compression system with respect to this computation/communication trade-off. The methodology involves evaluating system performance with respect to a figure of merit that is based on the amount of data that can be transmitted for a given energy budget. The methodology also involves careful dataflow modeling of candidate compression algorithms so that the embedded software can be streamlined, thereby enabling use of more intensive compression for a given energy budget and a given processing platform. We have presented experimental results that consider a variety of compression techniques and two different embedded processors. The results demonstrate the effectiveness of our methodology in quantitatively exploring the design space associated with integration of compression techniques into energy-limited sensor node platforms.

9. REFERENCES

[1] E. A. Lee and D. G. Messerschmitt, "Static scheduling of synchronous dataflow programs for digital signal processing," *IEEE Transactions on Computers*, vol. C-36, no.2, February 1987.

[2] B. Bhattacharya and S. S. Bhattacharyya, "Parameterized dataflow modeling for DSP systems," *IEEE Transactions on Signal Processing*, vol. 49, no.10, pp. 2408–2421, October 2001.

[3] S. Saha; S. Puthenpurayil and S. S. Bhattacharyya, "Dataflow transformations in high-level DSP system design," *In Proceedings of the International Symposium on System-on-Chip, Tampere, Finland*, November 2006.

[4] M. Hans and R. W. Schafer, "Lossless compression of digital audio," *IEEE Signal Processing Magazine*, vol. 18, no.4, pp. 21–32, July 2001.

[5] Y. Zhang and J. Li, "Efficient seismic response data storage and transmission using arx model-based sensor data compression algorithm," *Earthquake Engineering and Structural Dynamics*, vol. 35, pp. 781–788, 2006.

[6] S. Puthenpurayil; R. Gu and S. S. Bhattacharyya, "Compression techniques for minimum energy consumption," *25th Army Science Conference*, November 2006.

[7] W. Ye; J. Heidemann and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," *Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies*, November 2002.

[8] M. Bhardwaj; T. Garnett and A.P. Chandrakasan, "Upper bounds on the lifetime of sensor networks," *IEEE International Conference on Communication*, vol. 3, pp. 785–790, June 2001.

[9] J. Ammer and J. Rabaey, "The energy-per-useful-bit metric for evaluating and optimizing sensor network physical layers," *International Workshop on Wireless Network and Sensor Networks*, June 2006.