A 4.4 nW Lossless Sensor Data Compression Accelerator for 2.9x System Power Reduction in Wireless Body Sensors

Jacob Breiholz*, Farah Yahya*, Christopher J. Lukas*, Xing Chen†,
Kevin Leach*, David Wentzloff†, and Benton H. Calhoun*
*University of Virginia, Charlottesville, VA USA
†University of Michigan, Ann Arbor, MI USA
Email: jsb4ns@virginia.edu

***Abstract*—This paper presents a lossless sensor data compression accelerator for power reduction in wireless body sensors. First, a low complexity compression algorithm is demonstrated for the first time on electrocardiogram (ECG) and acceleration sensor data. Second, the algorithm is implemented as a custom hardware accelerator on a health monitoring driven System on Chip (SoC) in a 130 nm process. The accelerator is closely integrated with the transmitter interface to minimize its contribution to system power and reduce user overhead. The accelerator adds only 4.4 nW processing overhead and reduces the required transmitter duty cycle by 3.7x, reducing the system power by 2.9x, and allowing the entire system to consume just 2.62 µW when transmitting ECG data at a 360 Hz sampling rate.**

***Index Terms*—Lossless data compression, electrocardiogram, acceleration, wireless sensors, wearable sensors.**

.

I. Introduction

Global health care expenditures are increasing at an alarming rate [1]. There is a strong desire for wearable wireless sensor systems to enable continuous long-term health monitoring. Continuous long-term monitoring allows for earlier detection of health problems, leading to earlier intervention and treatment; it also allows for known chronic health problems to be managed proactively. Both cases ultimately lead to reduced health care costs, improved quality of life, and in some cases saved lives. The primary challenge in designing such systems is lowering the power consumption while still retaining the ability to sense, process, and transmit the required data. Typically, the transmitter dominates system power consumption, as wireless links are inherently high power compared to other system components. This work addresses transmitter power consumption through the use of on-node sensor data compression, which reduces the amount of data that must be transmitted, and thus the power consumption of the transmitter. Although compression does incur some processing overhead, it results in system level power reduction if it is carefully implemented to ensure that the savings in transmission power is greater than the additional power consumed by compression.

This work extends the state-of-the-art by first assessing the performance of a low complexity compression algorithm that has previously been demonstrated only on environmental sensor data [2] on ECG and acceleration data. Second, the algorithm is implemented as a custom hardware accelerator on a health monitoring driven SoC [3] and closely integrated with the transmitter interface to minimize its contribution to system power and reduce user overhead. The accelerator adds only 4.4 nW processing overhead and achieves a compression ratio (CR) of 2.39x, the lowest power and highest CR ratio reported to date for lossless on-node compression of ECG data. The transmitter duty cycle is reduced by 3.7x and the overall system power is reduced by 2.9x, thereby extending the lifetime of the sensor node. The remainder of the paper is organized as follows. Section II presents the algorithm selection process and defines the selected algorithm. Section III assesses the algorithm's performance across biomedical sensor data. Section IV presents the hardware implementation and the measured results. Section V concludes the paper.

II. Algorithm selection and definition

To minimize the system power, the algorithm must first be computationally efficient to minimize the processing overhead. Second, the algorithm must maximize the CR to minimize the amount of data that must be transmitted. Third, the algorithm must account for specific sensor data, as the CR of a compression algorithm is highly dependent on the data being compressed. The CR is formally defined as the ratio between the number of uncompressed bits and the number of compressed bits. Algorithms that rely on a linear prediction step followed by an entropy encoding step have shown promise for low overhead compression as they only require elementary integer operations [4], [5]. They are also lossless, and therefore appropriate for compressing sensitive health data. Recently, an algorithm named GAS-LEC that adds an additional adaptation step was proposed and shown to further improve the CR and robustness of this type of algorithm [2], making it the best existing candidate for low power on-node sensor data compression.

The linear prediction step of the algorithm is implemented as a simple delay stage, as more sophisticated prediction stages do not show very significant performance improvements and add additional computational overhead [2]. The differences

between the actual and predicted samples are then computed, as the dynamic range of the differences is typically smaller than that of the actual samples. Thus, these differences are more amenable to entropy encoding than the actual samples. The actual samples can be reconstructed from the differences during the decompression process as long as no data loss occurs and there is a single known sample in the data. Formally, the predicted sample is defined as the previous sample, and the difference between the actual and predicted sample is defined as $d_i = r_i - r_{i-1}$. In the entropy encoding step, the differences are appended with prefix codes that represent the size of the differences, thus eliminating the need for framing and allowing the outputs to be variable length. Prefix codes have the property that no shorter codeword may be the prefix of a longer codeword, so they are uniquely decipherable, and thus allow the output to be decoded. The first step in the entropy encoding process involves computing the minimum number of bits needed to represent d_i, which is formally defined as:

$$n_i = \begin{cases} 0, & \text{if } d_i = 0 \\ log_2(|d_i|) + 1, & \text{otherwise} \end{cases} \quad (1)$$

Once n_i has been determined, it is then necessary to calculate a_i, which is simply the n_i lower bits of d_i if d_i is positive. If d_i is negative, then the two's complement of d_i is used. If d_i is 0, a_i is not needed.

$$a_i = \begin{cases} \text{not needed} & \text{if } d_i = 0 \\ (d_i)^L_{n_i}, & \text{if } d_i > 0 \\ (d_i - 1)^L_{n_i}, & \text{if } d_i < 0 \end{cases} \quad (2)$$

The prefix code s_i can then be determined from n_i. Each n_i has a corresponding s_i according to Tables I and II.

TABLE I
LOWER PREFIX TABLE

n_i	s_i	d_i
0	00	0
1	010	-1,+1
2	011	-3,-2,+2,+3
3	1110	-7,...,-4,+4,...,+7
4	11110	-15,...,-8,+8,...,+15
5	110	-31,...,-16,+16,...,+31
6	100	-63,...,-32,+32,...,+63
7	101	-127,...,-64,+64,...,+127

TABLE II
UPPER PREFIX TABLE

n_i	s_i	d_i
8	111110	-255,...,-128,+128,...,+255
9	11111110	-511,...,-256,+256,...,+511
10	1111111110	-1023,...,-512,+512,...,+1023
11	111111110	-2047,...,-1024,+1024,...,+2047
12	1111110	-4095,...,-2048,+2048,...,+4095

The final compressed output can then be computed, and is defined as s_i concatenated with a_i, or $s_i \mid a_i$. Next, the adaptation step of the algorithm is performed. The adaptation step was added to mitigate the primary drawback of this type of algorithm, which is that they only perform well when the distribution of the differences is centered on zero and the variance is small. The adaptation step involves handling the prefix tables as circular buffers and rotating them after each difference is encoded. This rotation strives to ensure that the most frequently occurring differences are encoded with the shortest prefix codes, and it is also reversible, allowing the output to be decoded. Several different rotation protocols have been proposed, but the lowest complexity and perhaps most universally applicable is a simple greedy adaptive approach in which the tables are always rotated to ensure that the center of each table is on the group belonging to the previous sample. Note that the center of each prefix table is defined as the location of the shortest prefix code s_i, and that the length of the prefix codes increases outwards from the center. This ensures that differences closer to the center of the table are assigned the shorter prefix codes, which means that differences that are closer to the previous differences are encoded with shorter prefix codes. Ultimately, this improves the CR of the algorithm and makes it more robust to data types that do not produce difference distributions that are zero mean and low variance.

III. ALGORITHM PERFORMANCE

We implemented the algorithm in software and evaluated its performance using the MIT-BIH Arrhythmia Database [6] for ECG and the UCI Machine Learning Repository [7] for acceleration. We first evaluated the effectiveness of the linear prediction step by looking at the dynamic range of the differences compared to that of the actual samples, shown in Figure 1. We also computed the mean and variance of the resulting difference distributions for one ECG data set and several acceleration data sets for different activities shown in Table III. The mean is not shown in the table, but was reduced to approximately zero for all of the difference distributions.

TABLE III
STATISTICS SHOWING THE EFFECTIVENESS OF THE LINEAR PREDICTION STEP ON ECG AND ACCELERATION DATA.

Data set	Variance of samples	Variance of differences	Improvement
ECG record 100	1388.8	114.3	12.2x
Working at computer	730.4	109.2	6.7x
Talking while standing	738.3	49	15.1x
Standing	1068.3	103.1	10.4x
Walking	4249.2	1062.0	4.0x
Walking and talking	5464.6	1046.4	5.2x
Going up/down stairs	6766.8	861.9	7.9x
Standing, walking, stairs	6846.8	371.0	18.5x

We then executed the complete algorithm on numerous ECG and accelerometer sets, as shown in Figure 2 and Figure 3. Because we achieved average compression ratios of 2.4x and 2.0x for ECG and acceleration respectively, we can conclude that the algorithm effectively compresses such biomedical sensor data.

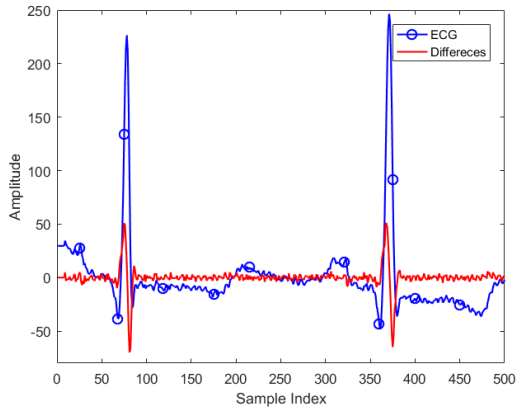

Fig. 1. The dynamic range of the differences is much less than that of the actual waveform. The data is from MIT-BIH record 100 and the DC offset was removed from the ECG waveform to better compare the variability.

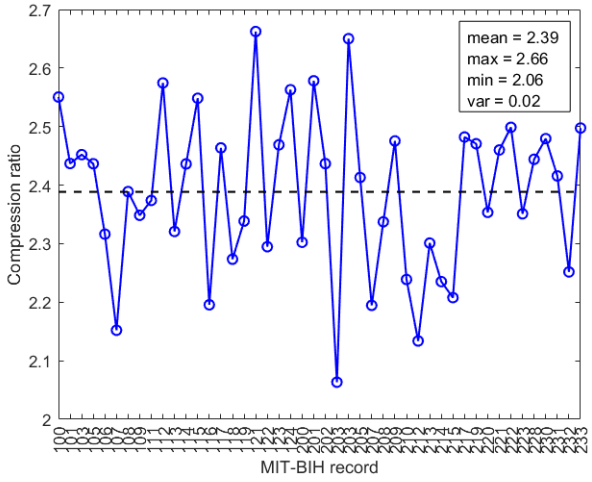

Fig. 2. The proposed algorithm is effective at compressing 46 ECG records from the MIT-BIH database. Modified lead II (MLII) data was used.

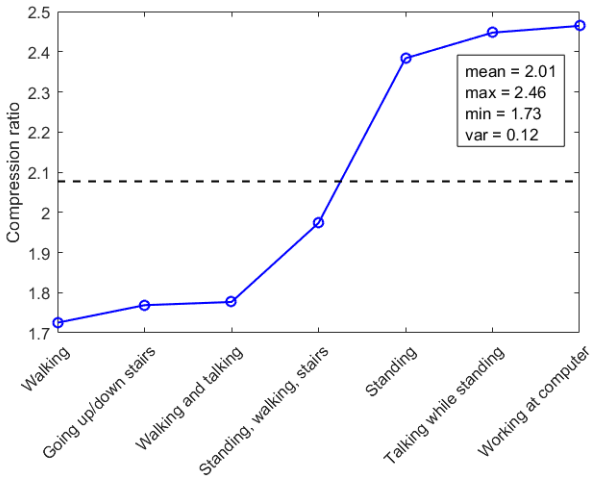

Fig. 3. The proposed algorithm is effective at compressing acceleration data for various activities of daily living from the UCI repository. Data from participant 1 was used.

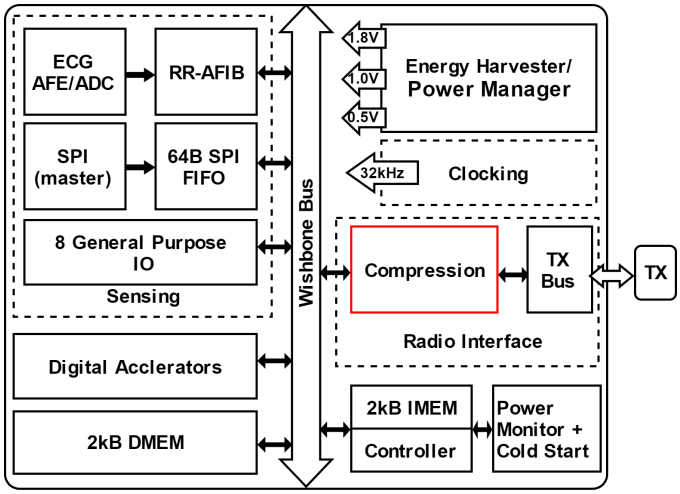

Fig. 4. Block diagram of SoC showing the compression accelerator closely integrated with the transmitter interface.

IV. Hardware Implementation and Results

The accelerator was carefully designed to minimize the processing power increase and user overhead. The accelerator was integrated into the transmitter interface on a health monitoring driven SoC [3], shown in Figure 4, and can be bypassed or enabled with a single configuration bit. The accelerator relies mostly on combinational logic and processes inputs in two clock cycles, adding minimal latency to the system. It also has a special operating mode that allows it to effectively compress multiple sensor data streams concurrently, such as 3-axis acceleration data.

To evaluate the system-level benefits of compression, we measured the system power consumption with and without compression while transmitting ECG and acceleration sensor data. The system uses an FSK modulated transmitter that works at 2.4 GHz and is controlled by the SoC through the transmitter interface. We measured the active power of the transmitter to be 1.05 mW and the sleep power to be 160 nW. For ECG, we used data from MIT-BIH record 100 and maintained a data rate of 360 Hz, which is the original sampling rate and a typical sampling rate for ECG. For acceleration, we used data from UCI participant one for the activity of standing, walking, and going up/down stairs and maintained a data rate of 52 Hz (per axis with 3-axis data, 156 Hz overall), which is the original sampling rate and a typical sampling rate for acceleration. In software, a CR of 2.5x and 2.0x was achieved for the ECG and accelerometer data sets respectively. In hardware, the benefits of compression are compounded due to the system architecture being 16b and the sensor resolutions being 11b and 12b for ECG and acceleration respectively. This is a common occurrence in many sensor systems and has the effect of increasing the uncompressed size of each sensor data sample to the system bit-width unless complex data packing