SenseBench: Toward an Accurate Evaluation of Sensor Network Processors

Leyla Nazhandali, Michael Minuth, Todd Austin
The University of Michigan
1301 Beal Ave, Ann Arbor, MI 48109
subliminal@eecs.umich.edu

***Abstract*— Sensor network processors introduce an unprecedented level of compact and portable computing. These small processing systems reside in the environment which they monitor, combining sensing, computation, storage, communication, and power supplies into small form factors. Sensor processors have a wide variety of applications in medical monitoring, environmental sensing, industrial inspection, and military surveillance. Despite efforts to design suitable processors for these systems, there is no well-defined method to evaluate their performance and energy consumption. The historically used MIPS (Millions of Instructions Per Second) and EPI (Energy Per Instruction) metrics cannot provide an accurate comparison because of their dependence on the nature of instructions, which differ across instruction set architectures. On the other hand, the current well-defined benchmarks do not represent typical workloads of sensor network systems, and hence, are not suitable to compare sensor processors. This paper defines a set of stream applications representing the typical real-time workload of a sensor processor. Furthermore, three new metrics, EPB (Energy Per Bundle), xRT (times Real-Time), and CFP (Composition Footprint) are introduced to evaluate and compare such systems.**

I. Introduction

Supercomputers have proceeded to shrink over the years into mini-computers, then desktops, handhelds, and most recently into sensor processors. Size scaling trends in computer design have led to miniaturized systems with decreased cost, lower power consumption, and new computing applications. Sensor processors represent a new level of compact and portable computing. These small processing systems can combine sensing, computation, storage, communication, and power supplies into small form factors. They reside in the environment which they monitor and encompass a vast array of applications, ranging from medical monitoring, to environmental sensing, industrial inspection, and military surveillance.

Currently, network researchers use numerous sensor network testbeds [4], [8], [9], [16], many of which utilize off-the-shelf microcontrollers such as ATMega128L [1] and Intel StrongArm/XScale [2]. The computational power of these microcontrollers is much higher than what most applications require [19]. Moreover, these processors are not energy-efficient enough for the untethered nature of sensor processors which requires them to survive for long periods of time on stored or scavenged energy. Consequently, several projects, such as Clever Dust [22], SNAP/LE [12], Subliminal [19], [18], and the work of Hempstead et. al. [15], have tried to fill this void by designing specialized processors for these systems. However, there has been a lack of accurate methods to compare the performance of these processors since each of them uses different ISAs and applications to evaluate performance and energy consumption. The existence of specialized hardware such as a scheduler (replacing a software operating system) makes it even more difficult to compare the performance level and energy-efficiency of these processors. To address this problem, we propose a set of basic application compositions that represent a typical duty cycle of a sensor processor. In designing these compositions, we pay attention to the stream-based nature of these workloads which process sample data on a regular basis. Furthermore, we present three metrics to evaluate sensor processors: EPB (Energy Per Bundle), xRT (times Real-Time), and CFP (Composition Footprint). We believe these metrics capture three important aspects of sensor processor constraints: EPB represents the energy consumed by the processor core to handle a bundle of samples including overhead for the operating system. xRT characterizes the computational capability of the processor to handle samples in real time. Lastly, CFP exhibits the storage requirement of the processor, which has a direct effect on the size and energy needs of the system.

The remainder of this paper is organized as follows: Section II details the proposed set of application compositions. Section III describes the suggested set of appropriate metrics for sensor network processors along with some comparisons among our designed systems employing these metrics. Section IV presents related work and Section V draws conclusions and suggests future research directions.

II. Proposed Workload

Accurate evaluation of sensor network processors requires two components: A representative workload and a set of suitable metrics. In this section, we make an attempt to provide the first component by proposing a set of application compositions that are built by chaining several lower level applications together. Section II-A describes the building blocks of these compositions and Section II-B presents the application compositions.

A. Application Building Blocks

Application compositions are constructed from basic application building blocks categorized into three groups: commu-

Fig. 1. Computing systems have enjoyed a fast track of size scaling in the past decades.

nication, data processing, and basic library functions as shown in Table I.

TABLE I
APPLICATION BUILDING BLOCK CATEGORIES.

Category	Application Building Blocks
Communication	aodv, crc, levhop, tea
Data processing	comp_diff, comp_rle, consensus, fir_filt, hist4, thold
Basic library	buf8, intavg, max, min, search, sort_insert, top100

aodv - Ad-hoc On Demand Distance Vector (AODV) Routing [10]: A routing algorithm is essential for networking with neighboring sensor processors. This application handles inter-chip networking. It builds and maintains a routing table according to the base AODV algorithm (RREQ, RREP, RERR messages).

crc8 - 8-bit Cyclic Redundancy Code (CRC) Encoder / Decoder [24]: Output data from a composition may require reliable off-chip transfer. The CRC algorithm detects transmission errors and increases the higher-level reliability of the output data stream. It encodes or decodes a 24-bit piece of data using an 8-bit CRC checksum.

levhop - Level-hop Routing Algorithm [20]: Another routing option is the unidirectional level-hop routing algorithm. This gives each node a level that is representative of the number of hops to a sink node. Messages are forwarded to a higher level until they reach the sink node.

tea - Tiny Encryption Algorithm (TEA) [23]: Data from a composition may require secure off-chip transfer (or in some cases, secure on-chip storage). The Tiny Encryption Algorithm encrypts and decrypts data to accommodate security demands using four 32-bit keys.

comp_diff - Compression Using Nibble Difference: Some data streams, such as temperature and pressure phenomena, have very little fluctuation, making them well-suited for difference compression. It builds a stream of data (using integer or logarithmic representation) by computing the difference between the input (current) and previous datum. If the difference is greater than a nibble in width, an escape value and the raw input datum are stored.

comp_rle - Compression Using Run Length Encoding (RLE) [7]: Some data streams, such as the output of a threshold detector, change very infrequently, making them a good candidate for RLE compression. This application builds a compressed stream of data based on the number of repetitions of samples in the raw data stream. In general, compression helps to minimize energy usage by reducing the size of the data flow for a composition. This is important in a sensor network processor where energy-efficiency is a primary concern.

consensus - Majority Consensus Using Neighbor Node Reports: Input signal noise could trigger a false event while neighboring chips would not trigger the same event. This application examines event reports from neighboring chips and determines, using majority rule, if a *true* event should be thrown.

fir_filt - Integer Finite Impulse Response Filter: Input data streams may require filtering processes, such as high/low pass filters or correlation, to extract meaningful data. This application performs an FIR filter on the input data in either static or running format. It is currently set to a 2-tap filter and can be expanded.

hist4 - Four Entry Histogram: Histograms can be used to characterize the occurrence behavior of a data stream. This application maintains a four-bin histogram with three definable thresholds. The input datum is placed into a bin and the respective tally is updated.

thold - Threshold Detection: Input data streams may require threshold monitoring to detect and report significant events. This application maintains a binary high/low state. The state is controlled using a Schmitt trigger-based mechanism,

comparing current state, input datum, and high and low thresholds.

buf8 - Buffering: Input data for some applications such as TEA and CRC is only meaningful in data chunks greater than 8 bits. Moreover, in the case of off-chip transmission, it is not energy-efficient to transmit a single datum at a time. The buffering application resolves these issues by building a larger chunk of data. When the buffer is full, a signal is asserted, allowing other applications to grab the buffer for processing. Any input data sent to a full buffer are ignored.

intavg - Integer Average: Averaging is common for many data compositions and is included in multiple formats, such as static versus running and integer versus logarithmic representation. It maintains a running average (with single input datum at a time) or calculates the average of an array of values.

min/max - Minimum or Maximum Value: Some input data streams may need their extremes recorded for signaling an important event or for simply keeping the statistics. This application maintains a running minimum or maximum value of input data by comparing the current stored minimum or maximum with the input datum.

search - Binary Search: Some compositions require maintaining a database of past recorded values, which may need to be searched for the occurrence of a specific event. This application looks for the input datum in a sorted array using the binary traversal method and asserts a signal if the datum is found.

top100 - Top 100 Values: This application maintains the top 100 values received from a data stream. Values arrive one at a time and are dropped if not among the top 100.

sort_insert - In-place Insertion Sort: Some applications, such as search and top100, require sorted arrays. This application sorts an input data stream using the insertion method either statically or dynamically. The former operates on an unsorted array of data concurrently while the latter updates a sorted array by placing the new input datum at the right position. The sort is in-place to reduce the total data memory footprint.

B. Application Compositions

Sensor network processors have several characteristics that set them apart from general purpose processors. First, they have fairly regular duty cycles comprising sample collection, data processing, package preparation and transmission onto the network. Second, they need to frequently communicate with peripherals, such as A/D converters, timers, and radios. Third, they have very limited energy budgets because of their untethered nature. Fourth, in most cases, they have very low performance requirements because they monitor phenomenon with low sample rates.

The combination of these characteristics promotes a top-down approach to the design of the system including usage of specialized hardware, increasing the efficiency of interrupt handling, and lowering overhead of the operating system. Consequently, it is very important to evaluate these processors not just by looking at a limited window on the duty cycle, but by looking at the full picture from the timer interrupt, to interaction with A/D, to processing the data, etc. This also includes the higher level scheduler or operating system and interrupt handler that orchestrates the control flow. In order to capture the full picture, we introduce the notion of application compositions. An application composition is a combination of application building blocks that represents a duty cycle of sensor network processors. In the rest of this section, we present several examples of application compositions.

Secure: This composition produces an encrypted data stream. It collects, filters, and buffers a bundle. It then encrypts the bundle using TEA. This composition can be used for transferring sensitive data, such as the output of a metal detector that determines the presence of troops in the field.

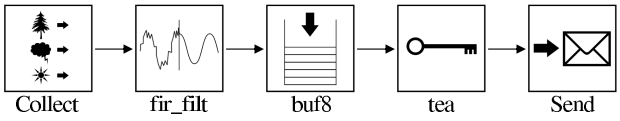

Fig. 2. Secure composition.

Reliable: This composition produces a data stream with CRC checksums. It collects and buffers a bundle, and then appends the CRC checksum. On the receiving end, the checksum can be used to determine transmission errors and force a retransmit of the bundle if needed. This produces a reliable data stream from a higher-level perspective. This composition can be applied to any data stream where bundle-level accuracy is a high priority, including medical applications like bloodstream monitoring.

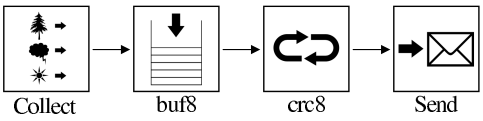

Fig. 3. Reliable composition.

Stats: This composition collects simple statistics for an input data stream. It performs running average, minimum, maximum, and maintains the top 100 values. It can be used for a simple characterization of a data stream from temperature or similar sensors.

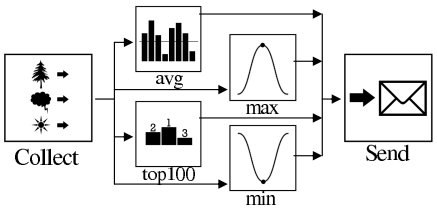

Fig. 4. Stats composition.

Catalog: This composition catalogs a filtered data stream. It collects, filters, and then tallies a bundle into a 4-bin histogram. This is well-suited to characterize the occurrence content of a data stream, such as engine temperature or wind speed.

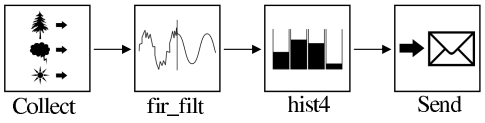

Fig. 5. Catalog composition.

Detect: This composition detects an event by forming a threshold detection consensus with other chips whose threshold detection outputs come from off-chip. This composition is useful for confidently detecting significant events such as seismic anomalies.

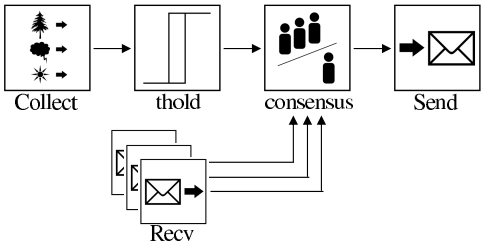

Fig. 6. Detect composition.

Query: This composition searches a buffer for an input datum and transmits the results. It can be used to confirm the occurrence of an event on the chip (searching the buffer for an event number) or a pattern match (searching the buffer for a spectrogram pattern).

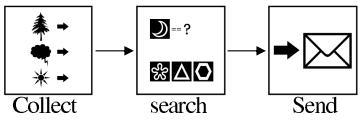

Fig. 7. Query composition.

RLE_stream: This composition compresses the output of a threshold detector using RLE. It collects a bundle, passes it through a threshold detector, and then compresses it. RLE compression is well-suited for this composition's front-end because many threshold output streams have large runs of either 1 or 0.

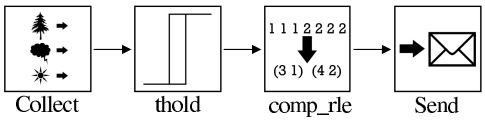

Fig. 8. RLE_stream composition.

Diff_stream: This composition compresses a data stream using difference compression and then buffers the data for transmission. This composition is well-suited for efficient transfer of a data stream with very little fluctuation, such as air or body temperature.

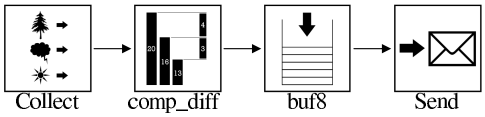

Fig. 9. Diff_stream composition.

AODV_route: This composition implements the AODV routing algorithm for inter-chip communication. It receives an AODV packet and either forwards the packet as an RREQ message or replies to the sending chip with an RREP message. This is useful for establishing routes for workloads, such as Consensus, that require interacting with other chips.

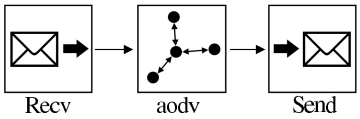

Fig. 10. AODV_route composition.

Simple_route: This composition implements the levhop application and propagates bundles to sink nodes. This is well-suited for data gathering phase, where all sensor processors transmit their data to a sink node for further and more complex analysis.

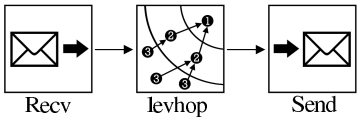

Fig. 11. Simple_route composition.

III. Metrics

EPB (Energy Per Bundle): EPB represents the energy consumed by the processor core to handle a bundle of samples from collecting and processing to sending and storing. A bundle is the smallest unit of data processed in a chain of an application composition and can contain a single or several samples depending on composition and sample precision. For example, in Secure composition, if sample precision is eight bits, a bundle contains eight samples. In this case, the EPB represents the energy consumed by the processor in collecting eight samples, packaging them into two 32-bit pieces of data, encrypting them using TEA, and finally transmitting them off-chip via radio or other communication medium.

xRT: times Real-Time: One important observation regarding sensor processors is that, unlike conventional processors, higher performance is not necessarily advantageous. It is mainly attributed to the fact that they perform most of their operations in real time for applications with slow changes, in which there is no benefit to running faster. Therefore, the performance characteristics of a sensor network processor transform into a binary function which decides whether it is able or unable to service the composition in real time. This depends on the sample rate of the phenomenon. The normalized xRT can be employed as an effective tool to evaluate the performance level of a sensor network processor. It indicates how many times faster the processor can perform the composition with respect to a 1Hz sample rate. The xRT value determines the highest sample rate the processor can handle when performing the specific composition. As an example, we have tabulated the data sample rates of a variety of phenomena in Table II gathered from various sources, including [6], [5], [11]. If a processor has an xRT value of 100 for Secure composition, it can process up to 100Hz sample rate when performing Secure composition, i.e. it will be able

to handle blood pressure but not engine temperature. As long as the sample rate is below that margin, it is not beneficial to increase the performance of the processor.

TABLE II
SENSOR NETWORK DATA SOURCES.

Phenomena	**Sample Rate** (in Hz)
Atmospheric temperature	0.017 - 1
Barometric pressure	0.017 - 1
Body temperature	0.1 - 1
Natural seismic vibration	0.2 - 100
Blood pressure	50 - 100
Engine temperature & pressure	100 - 150
ECG (heart electrical activity)	100 - 250

CFP (Composition Footprint): CFP represents the storage requirement of the processor, which has a direct effect on the size and energy needs of the system. The relative energy consumption of a sensor processor system is illustrated in Figure 12 [19]. The graph shows the core and memory energy consumption for varied memory sizes in a subthreshold energy-efficient sensor network processor, where the memory architectures are composed of 1/2 RAM and 1/2 ROM. It is clear that an increase in memory demands significantly increases the overall energy consumption of the system. Hence, having small CFP is a critical factor in designing a sensor network processor. As the definition of the composition contains all the software overhead that links different components of the composition, it accounts for the footprint of the operating system, if present.

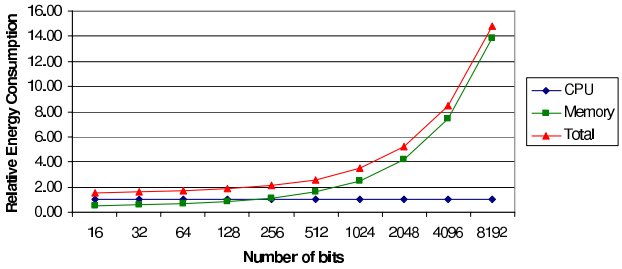

Fig. 12. Relative energy consumption of the core and memory for varied memory sizes.

The metrics EPB and CFP together represent the overall energy consumption of the system. It is important to have two separate metrics for these as most researchers report on core energy consumption only. Moreover, if the results are reported on actual fabricated design, it is helpful to report on energy consumption of the core separately if the core design has been the focus of the work or the size of the memory is larger than the CFP for test purposes.

Case Study: In our previous work we have presented the first [19] and second [18] generations of Subliminal processors. These processors are designed to minimize energy at the minimum-energy voltage, which falls in the subthreshold region. In the first generation, we focused on achieving a high code density using a CISC architecture and an accumulator-based design. We designed several variations of the processor to explore the design space in the subthreshold voltage region and its characteristics. Based on our findings, we designed the second generation of Subliminal processors using a RISC, LOAD-STOR architecture with special modifications to increase code density. These two generations have a different instruction set architecture and, hence, it is not trivial to compare them. However, employing the metrics proposed in this section makes their comparison straightforward.

Figure 13 presents the Composition footprint for several application compositions for the first and second generation of Subliminal processors. It is clear from this chart that the second generation has a significantly lower memory requirement. The explanation for the reduction of CFP in second generation ISA and detailed code density analysis for these processors is presented in [18], [19].

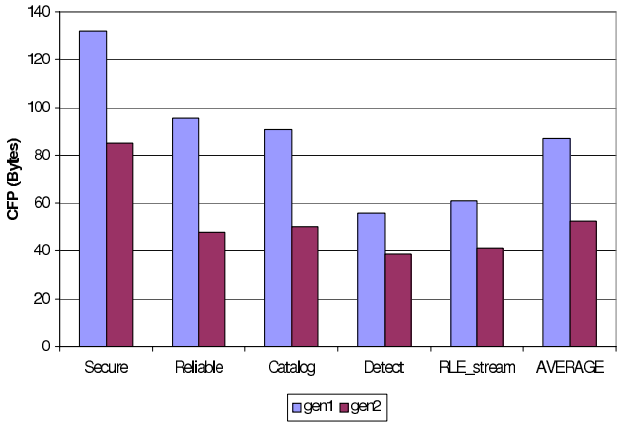

Fig. 13. Composition Footprint comparison between generation 1 and 2 of Subliminal processors.

Figure 14 represents the EPB and xRT characteristics of several Subliminal processors from the first and second generations for the RLE_stream composition. Given a phenomenon sample rate and a selected application composition, any processor that has xRT equal or higher than the sample rate of that phenomenon is capable of meeting its real-time demands when being processed by that application composition. For example, in Figure 14, the Base model of the second generation can perform RLE_stream composition for any sample rate equal to or smaller then 4.15 kHz. As for the EPB, the graph shows that the first generation processors have to consume more than four pJ to service a bundle (in this case, a single sample) using the RLE_stream composition. However, the second generation processors consume less than half the energy to perform the same service. It is clear that although these generations have different instruction set architectures, EPB and xRT metrics provide a simple way to compare them.

IV. RELATED WORK

There are a few projects trying to design hardware for sensor processors. However each of them uses a different set of applications to evaluate the performance and energy consumption of their design.

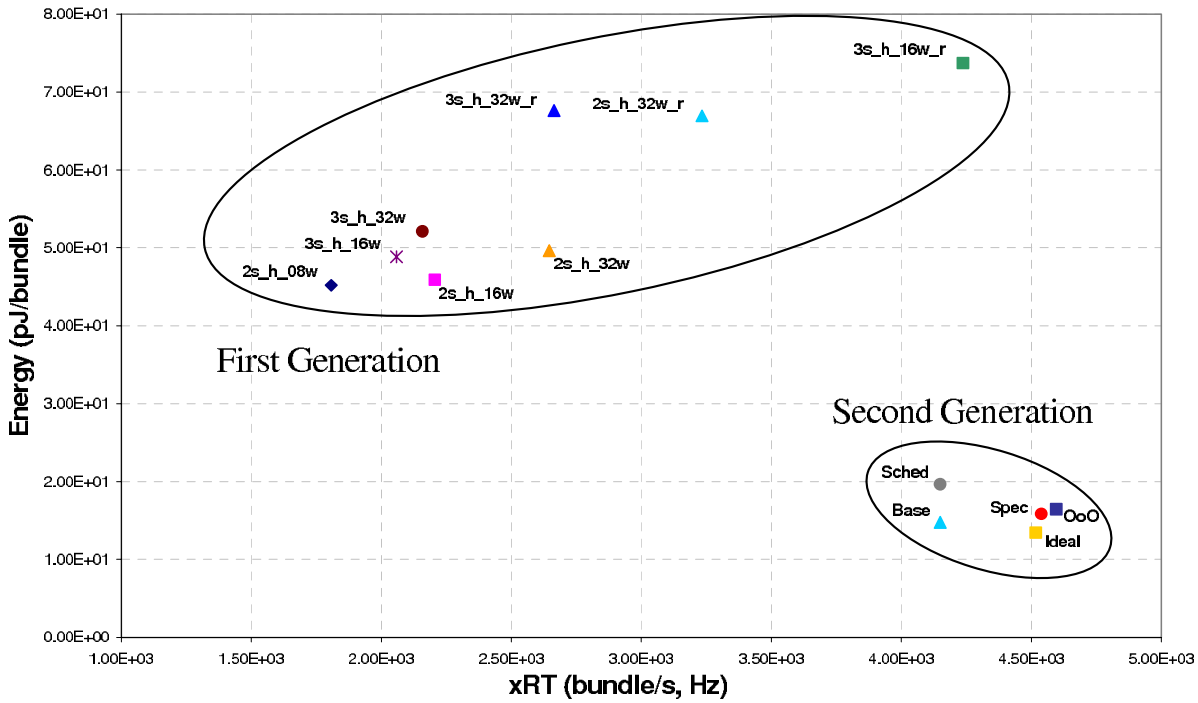

Fig. 14. Pareto analysis of several Subliminal processors from first and second generation with respect to RLE_stream composition.

Clever Dust 1 and 2 [21], developed as part of the Smart Dust project at UC-Berkeley, are two microcontrollers specifically designed for Dust sensor motes with the objective of reducing energy consumption. There is no mention of the applications used to estimate the energy consumption of the processor, neither in the Clever Dust paper [22], nor in the dissertation detailing the design and evaluation of Clever Dust [21]. Therefore, it is not clear what type of instruction the 12pJ per instruction represents.

SNAP/LE [12] is a sensor network processor based on an asynchronous, data-driven, 16-bit RISC core with a low-power idle state and low-wakeup latency. SNAP/LE has an in-order, single-issue core that does not perform any speculation. In order to estimate the energy per instruction of this processor, the developers are using six applications: Packet Transmission, Packet Reception, AODV Route Reply, AODV Packet Forward, Temperature Sense, and Range Comparison. Although these applications represent some of the typical workload seen on these systems, they do not cover a wide range of applications. Moreover, as they use energy per instruction in their evaluation, it is not easy to compare SNAP/LE to others.

Hempstead, et. al. [15] present an application-driven architecture which offloads immediate event handling from the general purpose computing component and leaves it idle in order to lower the power consumption of the system. The developers use four applications to evaluate power consumption of their system: Collect_Transmit, Collect_Compare_Transmit, Packet Forward, and Reconfigure. This system is the most difficult one to compare to others since the notion of conventional instructions is replaced with high-level commands that turn on/off different hardware subcomponents.

Finally, in our previous work, we have designed the first and second generation of Subliminal sensor processors with two instruction set architectures. In order to compare our designs, we had to find the equivalence of each instruction between them. We used a subset of applications presented in this paper to evaluate the energy per instruction.

There has been an attempt to create benchmarks for sensor network systems in [14]. However, this is a preliminary work trying to develop applications for tinyOS-based systems [16] rather than a general approach to applications in sensor network systems. In the benchmark domain, spec [3], mediabench [17], and mibench [13] have been developed and used by several researchers in order to more accurately evaluate hardware systems.

V. Insights and Future Work

In this paper, we proposed a novel approach to benchmarking for sensor network processors. Our approach is based on the following observations about sensor network processing:

- Sensor network processing has a distinctly different workload than that of traditional desktop and workstation class workloads, thus, benchmark suites should exist that address these differences.
- Sensor network processing is a real-time processing activity, thus benchmarks developed for this application space should account for how well a particular platform meets the real-time demands of the application data source.
- In addition to performance metrics, the area, cost, and energy-constraints of sensor network platforms make it imperative to evaluate designs based on energy-consumed-per-task and application memory demands.

We present in this paper a novel sensor network benchmark workload suite that accounts for each of the above observations. We introduce real-time, stream-oriented benchmark applications based on the common activities of low- to medium-bandwidth sensor network processors. Given these basic application tasks, we demonstrate how stream compositions are formed to perform a non-trivial representative task. These tasks are then fed data sources, based on a variety of physical, environmental, mechanical, and biological data sources, to produce a real-time stream processing experiment.

To evaluate these benchmarks and compare their performance across disparate platforms, we propose the use of three new evaluation metrics: EPB (Energy Per Bundle), xRT (times Real-Time), and CFP (Composition Footprint). xRT measures the real-time performance of an application on a sensor network platform, indicating how many times faster than real-time the platform is able to handle the real-time data source. EPB quantifies the amount of energy required (on average) to process a data element for a given application task and data source. Finally, CFP evaluates the memory demands of an application, which are strong determinants of the cost and energy demands of a sensor network processor. In each case, the metrics are directly comparable across sensor network processors. Thus, if designers implement these application tasks and make the described measurements, it will become possible to directly compare these diverse sensor platform designs.

Looking ahead, we see a number of ways to enhance our benchmark suite.

- Incorporating sporadic and periodic events in a single composition. For example, developing a composition that includes periodic data processing as well as sporadic packet routing.
- Forming different data sets for each composition that result in average- or worst- case EPB and xRT characteristics.
- Collaborating with sensor network system-level designers to introduce new compositions and improve the existing ones.

We are in the process of preparing a public distribution of our benchmark suite. The distribution will include C-language implementation of the described application tasks, which evaluators may compile directly or re-implement in assembly on their platform. Also included in the distribution are real-time data sources, driver programs to run experiment and collect results, and a database of measured results. We hope to make our distribution available by Winter 2006.

REFERENCES

[1] Atmega128l avr microcontroller datasheet. *http://www.atmel.com*.

[2] Intel pxa255 xscale processor datasheet. *http://www.intel.com/design/pca/prodbref/252780.htm*.

[3] Spec. spec benchmark suite release 1.0. 1989.

[4] Intel mote research project website. *http://www.intel.com/research/exploratory/motes.htm*, 2004.

[5] Monitoring in anaesthesia. *http://www.liv.ac.uk/ãfgt/Phys_Meas_Notes.pdf*, 2004.

[6] Multilog, multilogpro and ecolog weather stations. *www.fourier-sys.com/pdfs/data_loggers/*, 2004.

[7] Online resource for information on the data compression field. *http://www.data-compression.info/Algorithms/RLE/*, 2004.

[8] University of California, Los Angeles, Wireless Integrated Network Sensors website. *http://wins.ucla.edu*, 2004.

[9] Wireless sensing networks project at Rockwell scientific website. *http://wins.rsc.rockwell.com*, 2004.

[10] C.E.Perkins. Ad-hoc on-demand distance vector routing. *MILCOM Panel on Ad Hoc Networks, http://cireseer.ist.psu.edu/article/perkins97adhoc.html*.

[11] C.-Y. Chong. Sensor networks: Evolution, opportunities and challanges. In *Proceedings of the IEEE , Volume: 91 , Issue: 8*, Aug. 2003.

[12] V. Ekanayake, C. Kelly, and R. Manohar. An ultra low-power processor for sensor networks. In *Proceedings of the 11th international conference on Architectural support for programming languages and operating systems*, 2004.

[13] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown. Mibench: A free, commercially representative embedded benchmark suite. In *IEEE 4th Annual Workshop on Workload Characterization*, 2001.

[14] M. Hempstead, D. Brooks, and M. Welsh. Tinybench: The case for a standardized benchmark suite for tinyos based wireless sensor network devices. In *Proceedings of the First IEEE Workshop on Embedded Networked Sensors (EmNets'04)*, Nov. 2004.

[15] M. Hempstead, N. Tripathi, P. Mauro, G.-Y. Wei, and D. Brooks. An ultra low power system architecture for sensor network applications. In *International Symposium on Computer Architecture*, 2005.

[16] 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.

[17] C. Lee, M. Potkonjak, and W. H. Mangione-Smith. Mediabench: A tool for evaluating and synthesizing multimedia and communicatons systems. In *MICRO 1997*, 1997.

[18] L. Nazhandali, M. Minuth, B. Zhai, J. Olson, T. Austin, and D. Blaauw. A second generation sensor network processor with application-driven memory optimizations and out-of-order execution. In *International Conference on Compilers, Architecture, and Synthesis for Embedded Systems*, 2005.

[19] L. Nazhandali, B. Zhai, J. Olson, A. Reeves, M. Minuth, R. Helfand, S. Pant, T. Austin, and D. Blaauw. Energy optimization of subthreshold-voltage sensor network processors. In *International Symposium on Computer Architecture*, 2005.

[20] C. Schurgers and M. B. Srivastava. Energy efficient routing in wireless sensor networks. *MILCOM'01, Vienna, VA*, Oct. 2001.

[21] B. A. Warneke. *Ultra-Low Energy Architecture and Circuits for Cubic Millimeter Distributed Wireless Sensor Networks*. PhD thesis, University of California, Berkeley, 2003.

[22] B. A. Warneke and K. S. Pister. An ultra-low energy microcontroller for smart dust wireless sensor networks. In *International Solid-State Circuits Conference*, 2004.

[23] D. J. Wheeler and R. M. Needham. TEA, a tiny encryption algorithm. *Lecture Notes in Computer Science*, 1008, 1995.

[24] R. A. Williams. A painless guide to crc error detection algorithms. *http://www.repairfaq.org/filipg/LINK/F_crc_v3.html*.