NOTE: Use these buttons to print or save the form. DO NOT use the browser tool bar. SAVE PRINT

LA-UR-

Approved for public release; distribution is unlimited.

Title:	Energy Efficient Sensor Node Implementations
Author(s):	Jan Frigo, Eric Raby, Sean Brennan, ISR-3 Vinod Kulathumani, West Virginia University Ed Rosten, Cambridge University Christophe Wolinski, Charles Wagner, Francois Charot, IRISA
Intended for:	ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA) 2010, Feb 21-23, 2010, Monterey, California

Los Alamos
NATIONAL LABORATORY
EST. 1943

Form 836 (7/06)

SAVE PRINT CLEAR FORM

Energy Efficient Sensor Node Implementations

Jan Frigo*, Vinod Kulathumani†, Sean Brennan*, Ed Rosten‡, Eric Raby*
Christophe Wolinski§, Charles Wagner§, Francois Charot§
*Distributed Sensor Networks Group, Los Alamos National Labs
†Dept. of Computer Science and Electrical Engineering, West Virginia University
‡Dept. of Engineering, University of Cambridge
§University of Rennes, IRISA

***Abstract*—In this paper, we discuss a low power embedded sensor node architecture we are developing for distributed sensor network systems deployed in a natural environment. In particular, we examine the sensor node for energy efficient processing-at-the-sensor. We analyze the following modes of operation; event detection, sleep(wake-up), data acquisition, data processing modes using low power, high performance embedded technology such as specialized embedded DSP processors and a low power FPGAs at the sensing node. We use compute intensive sensor node applications: an acoustic vehicle classifier (frequency domain analysis) and a video license plate identification application (learning algorithm) as a case study. We report performance and total energy usage for our system implementations and discuss the system architecture design trade offs.**

Keywords: reconfigurable computing, FPGA, DSP, Distributed Sensor Networks (DSN), seismic, acoustic, vehicle classification, license plate detection

I. Introduction

There are growing numbers and varieties of sensor network systems deployed for *monitoring* in a natural environment [23], [3], [25], and [17]. This class of sensor network applications has some common requirements: (1) continuous operation (24/7), (2) low power sleep modes for extended time periods (3) fast wake-up for *event* triggering (4) dynamic range for computationally complex algorithms (5) flexible platform for rapid prototyping and changing system specifications (6) highly reliable, ruggedized, low power components, and (7) sufficient non-volatile memory. In addition, *traffic monitoring* sensor networks usually rely on multiple types of sensors and thus, processing each type requires different system resources. As a result, a general-purpose node architecture suitable for both rapid prototyping and a deployed outdoor environment may require two different implementations, for example, a microprocessor-based implementation may be sufficient for proof-of-concept or rapid prototyping testing, but a more optimized implementation may be required for long-term deployed operation.

Traffic monitoring sensor network research is mainly simulation-based [27], [10], [6] with a few exceptions [14], [4]. Deployed sensor network implementations use a variety of commercial off-the-shelf (COTS) hardware [12], [2]. These COTS systems have some well known draw-backs such as vendor specific Operating Systems (TinyOS)[18] [15], limited I/O and computing capabilities (i.e. no floating point unit (FPU)), small amounts of data storage and limited dynamic range. Field deployments raise new challenges such as issues with the range of network communication, environmental interference, and power management. Deployable hardware is a formidable research challenge and the success and advancement of deployed DSN systems ultimately depends on addressing these challenges.

Our field deployment [8] experiences lead us to explore a more flexible, modular, low-power node architecture. Our purposed node architecture [9] separates the *real-time* sensor data acquisition and data processing from network communications processing. The network communication interface is standard, i.e. the sensor processing modules plug into a common interface to communicate to the network. In addition, it offers the developer a platform that is flexible enough for rapid-prototyping as well as a more optimized implementation. Further, by *processing at-the-sensor* system power resources are conserved. Data is processed immediately with high-performance, energy-efficient technology and power is not wasted passing raw data around on internal buses or across the communication network. In data processing systems where compute- and data-intensive algorithms are used, data transfer is frequently the bottleneck for performance and power utilization. Deployed sensor network systems have to manage these same issues [22]. Often specialized, high performance embedded and/or reconfigurable hardware is used to mitigate this problem. In the same way, we aim to improve the energy efficiency and performance of deployed sensor network systems through the use of specialized hardware to process data at the sensor. Herein we quantify the energy utilization of our proposed implementations for two compute-intensive applications.

In this paper, our node architecture is described in Section II. The case study implementations and related work are discussed in Section III. We report the energy benchmark results for a variety of high performance embedded technologies suitable for *processing-at-the-sensor* in Section IV. Finally, in Section V we make some concluding observations and discuss future work.

II. Node architecture

The application domain for our sensor network systems is where events are infrequent, but significant computational complexity is required. Sensor types and processing are multi-modal. A sensor node may include one or more of the following sensors: (1) audio at 4 KHz sampling, (2) 3-axis seismic at 100 Hz, (3) 3-axis magnetometer at 1k Hz, and (4) GPS for absolute location, (5) low resolution video. Our target power requirements are 300 mW for the system while processing and transmitting and less than 10 mW for the system while acquiring data with the computational processor and RF transceiver asleep. All components must meet an environmental temperature range of -20.C to +70.C, suitable

for long-term, field deployment.

The system implementation goals for this modular architecture are: (1) No central processor is required. (2) Modules are independent, event driven entities with less top-down management than typical sensor motes. (3) Sensing duty cycles are specific to the individual sensor's sampling frequencies. (4) One processor, the sensing or the communication processor is awake at some low-level of processing to activate the advanced processing when an event occurs. (5) Message passing is handled by a separate communication microcontroller (8051).

A. Implementation

The architecture as shown in Fig. 1 combines a low power high performance ARM microcontroller mezzanine board [1], an embedded GPS module, a Texas Instruments CC2431 wireless chip, and four sensor interface connections. The mezzanine carrier board's wireless subsystem consists of a single self-contained COTS wireless system on chip (SoC), a CC2431 containing an embedded 2.4 GHz 802.15.4 compliant radio, a 32 MHz 8051 microcontroller, 8 KBytes RAM, and 128 KBytes flash storage, as well as hardware accelerated encryption, location computation and MAC layer functionality. The GPS module on the carrier board is controlled by the CC2431's embedded 8051 microcontroller. The CC2431 development tools consist of a C compiler and assembler. Each microcontroller core is capable of low power idle and sleep modes controllable by software.

The Phytec ARM mezzanine board is designed around the NXP LPC3180 chip (see Section IV). The ARM mezzanine board can be used for proof-of-concept development and rapid prototyping, or as a high performance co-processor. Further, the mezzanine carrier board can fully operate without the ARM board if a co-processor is not needed, i.e. if the node is a relay or if a processor is placed on the sensor board. Any embedded processor can be added in place of the ARM.

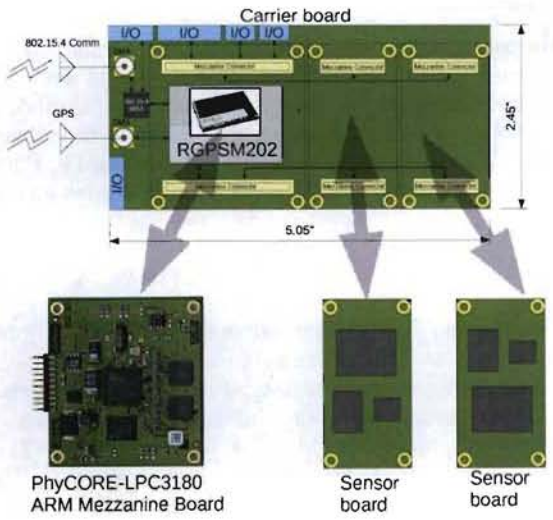

Fig. 1. Proposed modular node architecture

[1] Phytec phyCOREARM9/LPC3180

B. Related work

Reconfigurable technology has not been used in DSN field deployments to perform real-time *processing-at-the-sensor*. Research efforts using FPGAs for DSN applications include implementations to accelerate modeling of mobile agents [7]; address the problem of scalability for mobile ad-hoc networks [19]; and for filtering [26]. In the video processing domain, DSN video processing has been implemented on COTS hardware such as Stargates for many applications such as for low resolution image registration [1], fast image motion computation [16], and face detection [24].

Modular sensor network hardware architectures such as [22] and [21], for example, use a modular stack with any one of a combination of boards depending on the functionality of the node. An microcontroller in each module allows it to operate on its own schedule and power down when not in use. In [22] data acquisition and data transfer issues between modules, and component leakage current consumed a large part of the power budget. These designs utilize embedded processors for compute intensive data processing.

Other research suggests that to get the best power profile, older, less dense micron technology (with lower leakage current characteristics) should control data acquisition, message passing, and event monitoring– and the processor should be turned off or not used at all due to high leakage current [11]. We propose to use an ultra-low power FPGA as the *processor at the sensor*. It will be the master controller for power management of system components, taking advantage of fast switching times for wake-up and ultra-low power sleep modes in the device. In addition, data acquisition and processing is handled by the FPGA with lots of I/Os to accommodate high resolution bit precision and high performance computation as required by the application. At present, we anticipate using the 8051 micro-controller on the carrier mezzanine board for message passing.

III. Case Study Implementations

In this section, we describe a real time vehicle classifier system and a license plate identification system for traffic monitoring and discuss implementation details on our proposed modular architecture. In the following section we give benchmark results for energy usage and performance for the implementations.

A. Vehicle classifier

The vehicle classification system was developed on Crossbow's Stargate and Mica2 mote hardware [9], [8] using seismic and acoustic sensor data. The goal of this system is to classify vehicles as they approach a specific region into 3 categories: (1) a light-weight vehicle such as a compact car, (2) a moderately heavy vehicle and (3) a very heavy vehicle.

Seismic sensor data is sampled at 100 Hz. A Haar Wavelet computes the energy estimate of the 12-24 Hz band by averaging the coefficients of this band. The variance of the energy estimate is then computed and a variance threshold is used for *vehicle event detection*. Once an event is detected, acoustic data is sampled (4 KHz) and *acoustic processing* (512-point FFT) is initiated. The acoustic data is processed

in real-time until a classification has been determined. For the classification, we use Fisher Linear Discriminant Vector analysis to identify the best projection vector given the training data. We obtain a projection vector to distinguish between vehicle classes. These projection vectors are computed offline and stored locally. The dot product of this projection vector with the feature vector is computed to obtain a classification. Finally, a one byte result indicating the class of the vehicle (one of three classes) is sent to the network.

Implementation Our purposed vehicle classifier system implementation is shown in Figure 2. The seismic sensor is connected to a low pass filter, an Analog-to-Digital converter and then directly to the FPGA. The acoustic sensor connects directly to the FPGA. The FPGA controls event detection (seismic) and acoustic data acquisition and processing.

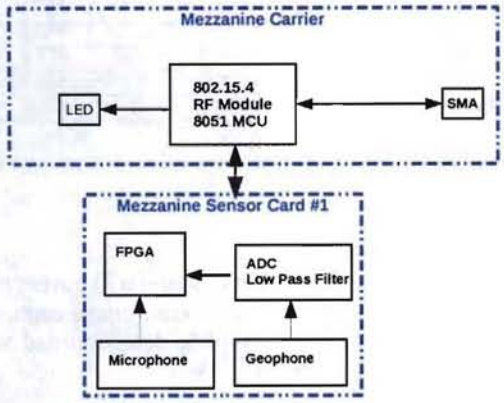

Fig. 2. Vehicle classifier implementation

Figure 3 shows the pipelined architecture corresponding to the seismic detection algorithm. It is composed of four processing modules $PM0..PM3$ and one control module $CM0$. The $PM0$ module saves and scales every 10th sample from the input data stream. The $PM1$ module executes the level-two wavelet transformations. The $PM2$ and $PM3$ modules compute the mean and variance of the wavelet coefficients according to equations 1 and 2. The latency, or first output is accessible after 16.4μs with a run-time system frequency of 90 Mhz. The remaining output results are obtained every 1.1μs. The input data stream rate supported by this system is 164 Msample/sec (MSPS). Every 10th sample from the input data stream is used. This design was implemented with VHSIC hardware description language (VHDL).

For the acoustic processing algorithm, we examine 512-pt integer FFT cores generated from the Xilinx ISE, and ACTEL Libero IDE FPGA development tools. The FFT implementation can support a 12 MSPS throughput with run-time frequency of 60 to 100 MHz.

$$mean = \frac{\sum_{i=0}^{31} \mid amp[i] \mid}{32} \quad (1)$$

$$result[t] = \frac{\sum_{i=0}^{31}(mean - amp[i])^2}{32} * 7 + result[t-1] \quad (2)$$

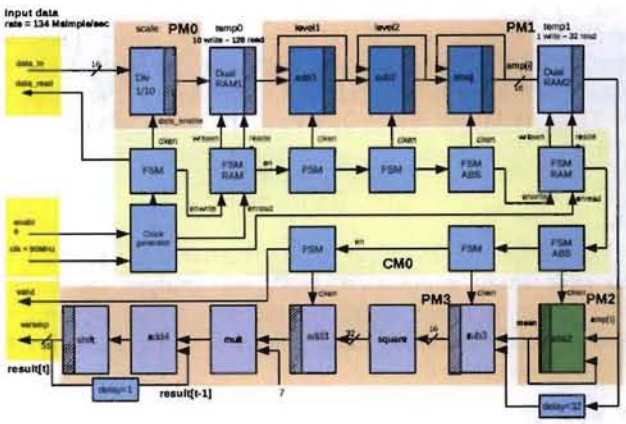

Fig. 3. Wavelet FPGA implementation

B. License plate identification

The license plate identification system [9], [8] extracts license plate information from a moving vehicle on a roadway using video and magnetometer data. The system aims to capture images of the aft end of a vehicle at anticipated vehicle speeds of 10 to 60 mph; extract license plate pixels from the original image (reducing the original image by 60-90%); and to convert the license plate image to text using optical character recognition (OCR). Previously, the license plate identification system was developed and tested on Crossbow's Stargate (400 MHz, XScale ARM-based) processor [9].

The license plate identification algorithm works by applying a classifier to every pixel in an image to create a rough segmentation of the license place, if it exists. From this, the bounding box of the license plate is found, and that section of the image is then resampled to a fixed size. The resampled image is then PNG compressed and sent over the network to a base station computer. The OCR is computed on the base station computer. These steps are a trade off between the amount of network bandwidth used, the latency of the operation and the amount of computing power used locally.

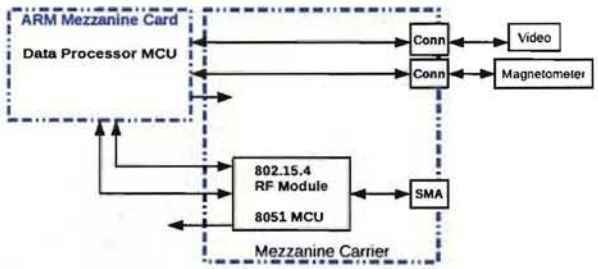

Fig. 4. License plate identification node implementation

Implementation Our purposed video license plate identification system implementation is shown in Figure 4, the camera and magnetometer connect directly to the ARM mezzanine board interface. *Vehicle detection* requires sensing and processing of magnetometer data. Here we are acquiring 16-bit data at 1k Hz and computing the magnetometer error between the raw data and the filtered data to detect a vehicle edge. When a vehicle detection occurs, the image capture routine collects

frames over a window of a few seconds. One frame is selected to be processed by the learning algorithm. In Section IV, the license plate identification algorithm results for the ChipCon LPC3180 ARM and the ADI BF537 Blackfin processors are given.

IV. Results

In this section we examine total system energy utilized for each case study application. First, we analyze the highest power operating mode, computational processing. Second, we examine event detection, data acquisition, sleep modes for each implementation.

For data processing mode, we benchmark our case study applications on the following reconfigurable and embedded architectures: Xilinx, Altera and ACTEL FPGAs[2], a DSP processor [3], two embedded processors. [4]

The FPGA energy results are derived from the Altera (Quartus II), Xilinx (Xpower), and ACTEL (SmartPower) development tools and our field experiment data per the frequency of the routed hardware design. Both quiescent power and dynamic power consumed during processing are calculated. The results in Tables I and II show the total power as a combination of both quiescent and dynamic power. The embedded processor energy estimated herein is taken from benchmarking the algorithms on the actual hardware.

The 'throughput' is calculated as:

$$throughput = n(samples)/executiontime(s)$$

denoted as mega samples per second(MSPS). The 'throughput' refers to the rate of outputs per second the implementation can deliver. The energy is calculated by

$$Energy(J) = measurepower(J/s) * executiontime(s)$$

where the execution time is determined by

$$executiontime(s) = n(cycles)/clockfrequency(cycles/s)$$

A. Processing

Vehicle classifier Estimated energy utilization for the seismic algorithm (the wavelet transformations, mean and variance calculations) is shown in Figure 5 and Table I. The acoustic classifier algorithm (512-pt integer FFT) energy utilization is given in Table II and Figure 6. The total computational energy for the vehicle classifier application is approximately 0.0255 μJ for seismic computation and 1.77 μJ for the acoustic classification if we consider the most efficient devices (see Table I and Table II). These benchmark results show an expected trend, the optimized, low power, embedded, specialized architectures show more energy efficiency–in this case, the FPGAs have the lowest energy utilization for compute intensive data processing.

License plate identification The three processing modes for the license plate identification application are magnetometer vehicle detection and sensing, image capture and license plate pixel identification processing. This section will covers the two most compute-intensive processing modes: image capture, and license plate identification. The vehicle detection and sensing mode will be discussed in Section IV-B.

The license plate identification processing routine performs the following functions: preprocessing, detection of license plate pixels, centroid, resampling, and compression. The energy estimates in Table III below are the total energy for these functions. In this application, the algorithm is a decision tree. The tree is applied independently at every pixel of the image. The decision tree was used in this implementation for computational speed on the CPU. Thus, the benchmark results are for the Stargate (XScale) and two embedded processors, the 3180 and the Blackfin. The most efficient processor was the 3180 at 0.186 J (Table III).

To port this algorithm to an FGPA, the system could learn a detector which is more amenable to processing on an FPGA:

	Time	Pwr	Freq	Thru	Energy
	μs	mW	MHz	MSPS	μJ
Igloo	4.3	5.95	23.16	23.16	0.0255
CycloneII	1.1	164	100	91	0.18
Stratix II	1.1	731	100	91	0.804
Virtex4	0.95	288	105	91	0.274
Spartan3	1.78	120	56	56	0.213
Mica2	1077	60	4	0.093	64.62
DSP	145	262	200	0.689	37.99
LPC3180	42.8	330	208	2.3	14.12
Blackfin	21	1056	500	4.7	22.17

TABLE I
Seismic Processing Energy Utilization

	Time	Pwr	Freq	Thru	Energy
	μs	mW	MHz	MSPS	μJ
Igloo	31.23	185	16.39	16.39	5.77
Proasic3	6.53	272	78.3	78.3	1.77
Stargate	867	1850	400	0.59	1603.9
DSP	3000	227	200	0.17	681
DSP opt	60	339	200	8.5	20.34
LPC3180	877	400	208	0.584	350.8
Blackfin	563	1175	500	0.90	661.5
Blackfin [10]	139	1175	500	3.7	163.3
Blackfin [11]	18.8	1006	500	27	18.91

TABLE II
Acoustic Processing Energy Utilization (512-pt int FFT)

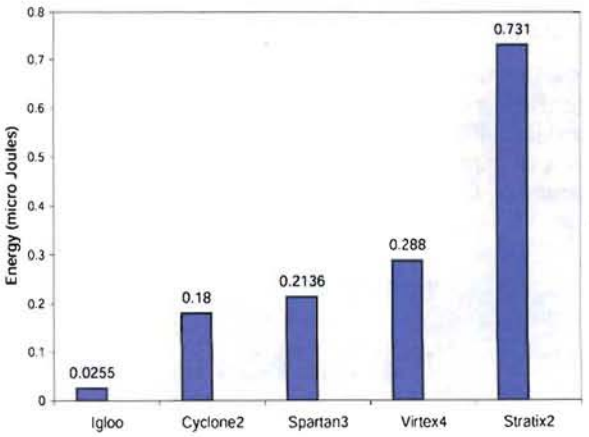

Fig. 5. Seismic Algorithm Energy Trends

[2]Virtex4 XC4VLX15, Spartan3 XC3S400, Stratix II EP2S60, CycloneII EP2C35F, Igloo AGL1000V5)

[3]Texas Instrument's TMS320C5510

[4]ChipCon LPC3180 and Analog Devices Blackfin ADI BF537 processor.)

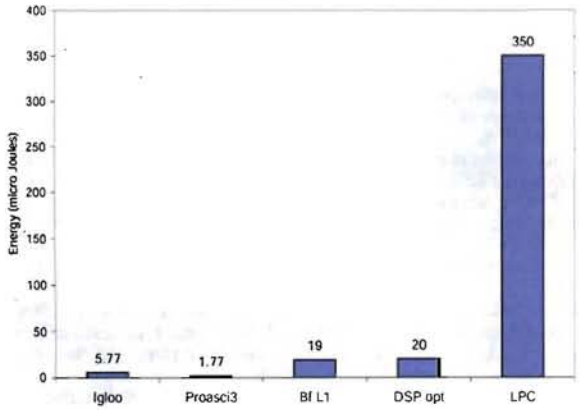

Fig. 6. Acoustic Processing Energy Trends

a weighted sum of thresholded convolutions. In practise, we would build a cascade classifier so that most pixels were discarded by the FPGA, then the few remaining would be processed with a decision tree or something similar.

The image capture routine has not been ported to the 3180, this is a topic for future work. We can estimate the energy usage, by assuming capture takes as least 5 to 10 seconds based on our previous implementation on the Stargate [9] with maximum active MPU mode at 80 mA, 1.2 V. The energy is then 0.480 J to 0.960 J.

	Time	Pwr	Freq	Thru	Energy
	s	W	MHz	KSPS	J
Stargate	0.675	1.575	400	455	1.06
3180	0.61	0.305	208	503	0.186
Blackfin	0.535	1.1484	500	574	0.614

TABLE III
LICENSE PLATE IDENTIFICATION ENERGY UTILIZATION

B. Data Acquisition and Event Detection

Vehicle classifier For the vehicle classifier system, the seismic algorithm signals an event by processing seismic data (100 Hz) and this triggers acoustic and video data acquisition and processing. Here the system is not *on* continuously, it wakes up at periodic intervals and performs a few seconds of computation to signal a vehicle event if it exists. The question arises, if an FPGA is most energy efficient for data processing, is it comparable to a separate microcontroller for wake-up timing, sleep mode and leakage current to control event detection and data acquisition? In addition, should power management be performed by the FGPA for energy efficiency?

Igloo AGL600V5 FPGA The Igloo FPGA devices has Flash Freeze technology which enables power on/off from ultra-low power modes. For this seismic detection using the AGL600V5, the quiescent Flash Freeze leakage power is 114 μW with all voltages (including the core voltage, Vcci = 1.5 V) on and all clocks and I/Os off. For sleep mode with only the core voltage on and all other voltages, clocks and I/Os off, power usage is 10.8 μW The FGPA power management does not require extra components to turn off I/Os or clocks and retains design, SRAM content and registers. Wake up timing is reported as 1 μs [5]. Run time operating frequency for this application is 23.16 MHz.

The Igloo AGL600V5 device has 108 kbits RAM (1024 bits), 24 RAM Blocks (4608-Bit), 1 Kbits user nonvolatile FlashROM and 600k system gates. Eight blocks of I/O with a total of 270 user I/Os.

Our anticipated event detection rate on a remote roadway is approximately 80 vehicle events/hour on average (5-10 seconds "on" time as the vehicle approaches and passes the sensor). So the system is "sensing" at 100 Hz continuously and computing 1 event every 1.35 minutes worst case or approximately 22.2% active, 77.7% sensing mode which utilizes 916.7μW of power–that is over 2 months of continuous use. Best case is 11.1% active mode 88.9% sensing or sleep and the total power usage goes down to 607.5 μW–a little less than 3 and a half months of continuous operation on assuming a 1000 mAH battery. See Figure Vfor a summary of total power results. Determinng a schedule for sleep and/or Flash Freeze mode in the operating plan for this application is a topic for future experiments.

	wakeup time	sleep	idle	active
	μs	μW	mW	mW
AGL600V5	1	11,114*	0.134	5.95
8051	2,54	888,2.7,1.8†	1.476	0.621 - 21
LPC3180	5000	450	6.3(13 MHz)	0.24μW

TABLE IV
TOTAL POWER FOR VEHICLE EVENT DETECTION
(*FLASHFREEZE,†PM1,PM2,PM3)

CC2430 SoC/8051 microcontroller The CC2430 SoC has some useful features for seismic event detection such as 21 digital I/Os, 8-14 bit ADC, and both a watchdog and sleep timer. The 8051 microcontroller has 8 KB RAM, with the upper 4 KB retaining data in all power modes, 16-bit read/write access to memory, and flash memory. In addition, a memory arbiter to handle CPU and DMA access. There are three low power modes (PM1,PM2,PM3) depending on the expected wait time between events–all modes have RAM retension. The low power modes use 1.8 μW to 888 μW depending on the power mode. The wake up timing for the low power modes is reported as 2 μs for PM1 and 54 μs for PM2 and PM3 [13]. The highest operating speed is 32 MHz. Best and worst case estimates for the 8051 per the expected vehicle event rate are 1.866μW and 2.452μW of power respectively.

LPC3180: The NXP LPC3180 chip runs at 208 MHz with 32 MByte of SDRAM operating at 104 MHz and 32 MByte of NAND flash. The ARM core is an ARM926EJ-S CPU with an IEEE vector floating point (VFP) co-processor. The floating point performance is approximately 5 times faster than the Intel Stargates XScale PXA255 CPU (400 MHz) using a software floating point library. During continuous floating point operation power consumption of the mezzanine board is approximately 330 mW. (The LPC3180 ARM board runs a standard Linux 2.6.10 kernel for rapid prototyping.) The LPC3180 board's low power modes are as follows: direct RUN is 7 mA at 13 MHz (slow clock), and STOP mode is 500 μamps(lowest power mode). For wake-up timing, typical values for ARM CPUs are less than 0.5 ms. Leakage current for the 3180 is reported as 3 μamps [20].

C. Total System Energy

Vehicle Classifer: When we compare the power characteristic for the CC2430/8051 and the Igloo FPGA in Figure V, the

overall power saved using the FGPA is approximately 2.6-3.0x for seismic event detection.

	Time	Pwr	Energy
	μs	mW	μJ
seismic detect	4.3	5.95	0.0255
acoustic processing	31.23	185	5.77
total vc system	35.23	190.95	5.795
mag detect	1.01 μs	96-240	0.24 μJ
image capture	5-10s	96-240	0.48 - 2.4 J
image processing	0.61s	305	0.186 J
total ip id system	5.61-10.61s	497-785	0.666 - 2.586 J

TABLE V
TOTAL RUN-TIME SYSTEM ENERGY

V. CONCLUSION

The central idea of our node architecture is to keep the data transfer, both intra-module and to the network, at a minimum by processing data in real-time on the sensor board. In DSN and conventional processing systems where compute- and data-intensive algorithms are used, data transfer is frequently the bottleneck for performance and power utilization. In our system, we have flexibility to choose the most suitable embedded processor for the task. The master controller for sensing, data acquisition and power control is the processor/microcontroller on the sensor board. In most cases, that will be the FPGA which offer fast wake up times and ultra low power sleep modes. The 802.15.4 wireless link on the mezzanine carrier board is available to partially or fully reconfigure the FGPA in-situ–an advantage for deployed systems with changing system requirements. Reduced need for memory storage since event data is significantly reduced, i.e one byte of data for vehicle classification and 3k bytes for license plate identification.

In this paper, we discuss a low power embedded sensor node architecture we are developing for distributed sensor network systems deployed in a natural environment. In particular, we examine the sensor node for energy efficient processing-at-the-sensor. We analyze the following modes of operation; sensing, sleep, data acquisition, data processing modes using low power, high performance embedded technology for sensing node. We use compute intensive sensor node applications: an acoustic vehicle classifier and a video license plate identification application as a case study. We report performance and total energy usage for our system architecture and discuss the system architecture design trade offs.

Future work Investigate hybrid FPGA with analog, non-volitile memory such as ACTEL's Fusion chip. Analyze and develop an FGPA implementation for the vidoe node. Benchmark magnetometer on FPGA. Build and field test prototype systems to measure energy efficiencies.

VI. ACKNOWLEDGEMENTS

This work was supported by the U.S. Department of Energy/NNSA and Los Alamos National Laboratory funds under LANS, LLC Contract No. W-7405-ENG-36. This document is approved for public release under LAUR-09-.

REFERENCES

[1] e. a. Balasubramanian, Harini. Image registration in low resolution visual sensor netowrks. In *International Conference on Information Processing in Sensor Networks*, 2008.
[2] J. Barton and et al. A miniaturised modular platform for wireless sensor networks. *IEEE Circuit Theory and Design, vol. 3, p. 35-38*, Aug. 2005.
[3] U. Center for Embedded Network Sensing. Leap: Low power energy aware processing. *http://research.cens.ucla.edu/projects/2007/Systems/LEAP/*, 2008.
[4] e. a. Coleri, Sinem. Sensor networks for monitoring traffic. In *IPSN*, 2004.
[5] A. Corporation. Igloo. *http://www.actel.com/products/igloo*, 2009.
[6] e. a. Corredor, Ivan. Design and deployment of industrial sensor networks: Experiences from a semiconductor plant and the north sea. In *Collecter Iberoamerica 2005*, 2008.
[7] e. a. Du, Hongtao. Modeling mobile-agent-based collaborative processing in sensor networks using generalized stochastic petri nets. *IEEE International Conference on Systems, Man and Cybernetics, vol. 1, p. 563-568*, Oct. 2003.
[8] J. e. a. Frigo. Radiation detection and situation management by distributed sensor networks. In *SPIE Proceedings on Defense, Security and Sensing*, 2009.
[9] J. e. a. Frigo. Sensor network based vehicle classification and license plate identification system. In *IEEE INSS*, 2009.
[10] e. a. Ganguly, A. R. Knowledge discovery from sensor data for security applications. *Knowledge Discovery from Sensor Data for Security Applications*, pages 187–204, 2007.
[11] M. e. a. Hempstead. An ultra low power system architecture for sensor network applications. In *IEEE*, 2005.
[12] J. Hill, M. Horton, R. Kling, and L. Krishnamurthy. The platforms enabling wireless sensor networks. *Communications of the ACM, vol 47*, June 2004.
[13] T. Instruments. http://focus.ti.com/docs/prod/folders/print/cc2430.html. In *Chipcon Products CC2430 SoC*, 2009.
[14] e. a. Krishnamurthy, Lakshman. Wireless sensor network-based system for measuring and monitoring road traffic. In *SenSys 2005*, Nov. 2005.
[15] K. Langendoen, A. Baggio, and O. Visser. Murphy loves potatoes: experiences from a pilot sensor network deployment in precision agriculture. *20th International Parallel and Distributed Processing Symposium, vol. 2006*, Apr. 2006.
[16] X. Lu and R. Manduchi. Fast image motion computation on an embedded computer. In *Conference on Computer Vision and Pattern Recognition*, 2006.
[17] K. Martinez, P. Padhy, A. Elsaify, G. Zou, A. Riddoch, and J. K. Hart. Deploying a sensor network in an extreme environment. *Proceedings of the IEEE Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, 2006.
[18] e. a. Nassr, M. S. Development, implementation, and experimentation of parametric routing protocol for sensor networks. *Proceedings of SPIE-the international society for optical enginering, vol. 6394*, 2006.
[19] W. Ong, G. Venkataraman, S. Emmanuel, and A. Das. Fpga implementation of cluster formation algorithms in mobile ad-hoc networks. *Intelligent Sensors, Sensor Networks and Information Processing Conference. p. 19-24*, Dec. 2005.
[20] N. Phillips. http://www.standardics.nxp.com/products/lpc3000/lpc3180/. In *3180 processor*, 2009.
[21] J. Portilla, T. Riesgo, and A. de Castro. A reconfigurable fpga-based architecture for modular nodes in wireless sensor networks. *IEEE Conference on Programing Logic, 2007*, June 2007.
[22] B. Schott, M. Bajura, J. Czarnaski, J. Flidr, T. Tho, and L. Wang. A modular power-aware microsensor with >1000x dynamic power range. In *Information Processing in Sensor Networks SPOTS (IPSN-SPOTS)*, 2007.
[23] R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler. Lessons from a sensor network expedition. *1st European Workshop on Wireless Sensor Networks*, 2004.
[24] Y. Weng and A. Doboli. Smart sensor architecture customized for image processing applications. In *IEEE Real-Time Embedded Technology and Applications Symposium*, Aug. 2004.
[25] G. Werner-Allen, O. Marcillo, J. Johnson, M. Ruiz, and J. Less. Deploying a wireless sensor network on an active volcano. *IEEE Internet Computing*, Mar. 2006.
[26] M. B. Yeary, W. Zhang, J. Q. Trelewicz, Y. Zhai, and B. McGuire. Theory and implementation of computationally efficient decimation filter for power-aware embedded systems. *IEEE Transactions on Instrumentation and Measurement, vol. 55, no. 5*, Oct. 2006.
[27] e. a. Zhang, Mingchen. Three-tiered sensor network architecture for traffic information monitoring and processing. In *IPSN*, 2004.