Supplementary Material for “Multi-core Embedded Wireless Sensor Networks: Architecture and Applications”

Arslan Munir, *Member, IEEE,* Ann Gordon-Ross, *Member, IEEE,* and Sanjay Ranka, *Fellow, IEEE, AAAS*

Abstract—Technological advancements in the silicon industry, as predicted by Moore’s law, have enabled integration of billions of transistors on a single chip. To exploit this high transistor density for high performance, embedded systems are undergoing a transition from single-core to multi-core. Although a majority of embedded wireless sensor networks (EWSNs) consist of single-core embedded sensor nodes, multi-core embedded sensor nodes are envisioned to burgeon in selected application domains that require complex in-network processing of the sensed data. In this paper, we propose an architecture for heterogeneous hierarchical multi-core embedded wireless sensor networks (MCEWSNs) as well as an architecture for multi-core embedded sensor nodes used in MCEWSNs. We elaborate several compute-intensive tasks performed by sensor networks and application domains that would especially benefit from multi-core embedded sensor nodes. This paper also investigates the feasibility of two multi-core architectural paradigms—symmetric multiprocessors (SMPs) and tiled many-core architectures (TMAs)—for MCEWSNs. We compare and analyze the performance of an SMP (an Intel-based SMP) and a TMA (Tilera’s TILEPro64) based on a parallelized information fusion application for various performance metrics (e.g., runtime, speedup, efficiency, cost, and performance per watt). Results reveal that TMAs exploit data locality effectively and are more suitable for MCEWSN applications that require integer manipulation of sensor data, such as information fusion, and have little or no communication between the parallelized tasks. To demonstrate the practical relevance of MCEWSNs, this paper also discusses several state-of-the-art multi-core embedded sensor node prototypes developed in academia and industry. We further discuss research challenges and future research directions for MCEWSNs.

Index Terms—wireless sensor networks, multi-core, embedded systems, symmetric multiprocessors, tiled many-core architecture

1 Introduction

This document presents additional details supplementing our IEEE Transactions on Parallel and Distributed (TPDS) paper with the title *“Multi-core Embedded Wireless Sensor Networks: Architecture and Applications”*.

Advancements in silicon technology, embedded systems, sensors, micro-electro-mechanical systems, and wireless communications have led to the emergence of embedded wireless sensor networks (EWSNs). EWSNs consist of sensor nodes with embedded sensors to sense data about a phenomenon and these sensor nodes communicate with neighboring sensor nodes over wireless links (we refer to wireless sensor networks (WSNs) as EWSNs since sensor nodes are embedded in the physical environment/system). EWSNs have applications in various domains, including surveillance, environment monitoring, traffic monitoring, volcano monitoring, and health care.

• *Arslan Munir is with the Department of Electrical and Computer Engineering, Rice University, Houston, Texas, USA. Ann Gordon-Ross is with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL, USA. Ann Gordon-Ross is also affiliated with the NSF Center for High-Performance Reconfigurable Computing (CHREC) at the University of Florida. Sanjay Ranka is with the Department of Computer and Information Science and Engineering at the University of Florida. e-mail: {arslan@rice.edu, ann@ece.ufl.edu, ranka@cise.ufl.edu}*

Processing and transmission of the large amount of sensed data in emerging applications exceeds the capabilities of traditional EWSNs. For example, consider a military EWSN deployed in a battlefield, which requires various sensors, such as imaging, acoustic, and electromagnetic sensors. In this application, images are appropriate for visually monitoring the battlefield, and electromagnetic and acoustic sensors enable efficient detection and tracking of targets of interest. Once a target is detected, high resolution images and/or video sequences may be required in real-time for detailed study of the target [1]. This application presents various challenges for existing EWSNs since transmission of high-resolution images and video streams over bandwidth-limited wireless links from sensor nodes to the sink node is infeasible. Furthermore, meaningful processing of multimedia data (acoustic, image, and video in this example) in real-time exceeds the capabilities of traditional EWSNs consisting of single-core embedded sensor nodes [2][3], and requires more powerful embedded sensor nodes to realize this application.

Technological advancements in multi-core architectures have made multi-core processors a viable and cost-effective choice for increasing the computational ability of embedded sensor nodes. Preliminary studies have demonstrated the energy-efficiency of multi-core embedded sensor nodes as compared to single-core embedded sensor nodes in an EWSN. For example, Dogan et al. [4] evaluated single-

and multi-core architectures for biomedical signal processing in wireless body sensor networks (WBSNs) where both energy-efficiency and real-time processing are crucial design objectives. Results revealed that the multi-core architecture consumed 66% less power than the single-core architecture for high biosignal computation workloads (i.e., 50.1 Mega operations per seconds (MOPS)) whereas the multi-core architecture consumed 10.4% more power than that of the single-core architecture for relatively light computation workloads (i.e., 681 Kilo operations per second (KOPS)).

This supplementary material document is organized as follows. Section 2 proposes a multi-core embedded sensor node architecture for multi-core embedded wireless sensor networks (MCEWSNs). Section 3 discusses multi-core architectures for multi-core embedded sensor nodes and parallel computing metrics that we use to evaluate these architectures. Section 4 elaborates on several compute-intensive tasks that motivated the emergence of MCEWSNs. Section 5 discusses several prototypes of multi-core embedded sensor nodes. Experimental setup details for the information fusion application are presented in Section 6.

2 Multi-core Embedded Sensor Node Architecture

Fig. 1 depicts the architecture of a multi-core embedded sensor node in our MCEWSN. The multi-core embedded sensor node consists of a sensing unit, a processing unit, a storage unit, a communication unit, a power unit, an optional actuator unit, and an optional location finding unit (optional units are represented by dotted lines in Fig. 1) [2].

2.1 Sensing Unit

The sensing unit senses the phenomenon of interest and is composed of two subunits: sensors (e.g., camera/image, audio, and scalar sensors (e.g., temperature, pressure)) and analog-to-digital converters (ADCs). Image sensors can either leverage traditional charge-coupled device (CCD) technology or complementary metal-oxide-semiconductor (CMOS) imaging technology. The CCD sensor accumulates the incident light energy as the charge accumulated on a pixel, which is then converted into an analog voltage signal. In CMOS imaging technology, each pixel has its own charge-to-voltage conversion and other processing components, such as amplifiers, noise correction, and digitization circuits. The CMOS imaging technology enables integration of the lens, an image sensor, and image compression and processing technology on a single chip. ADCs convert the analog signals produced by sensors to digital signals, which serve as input to the processing unit.

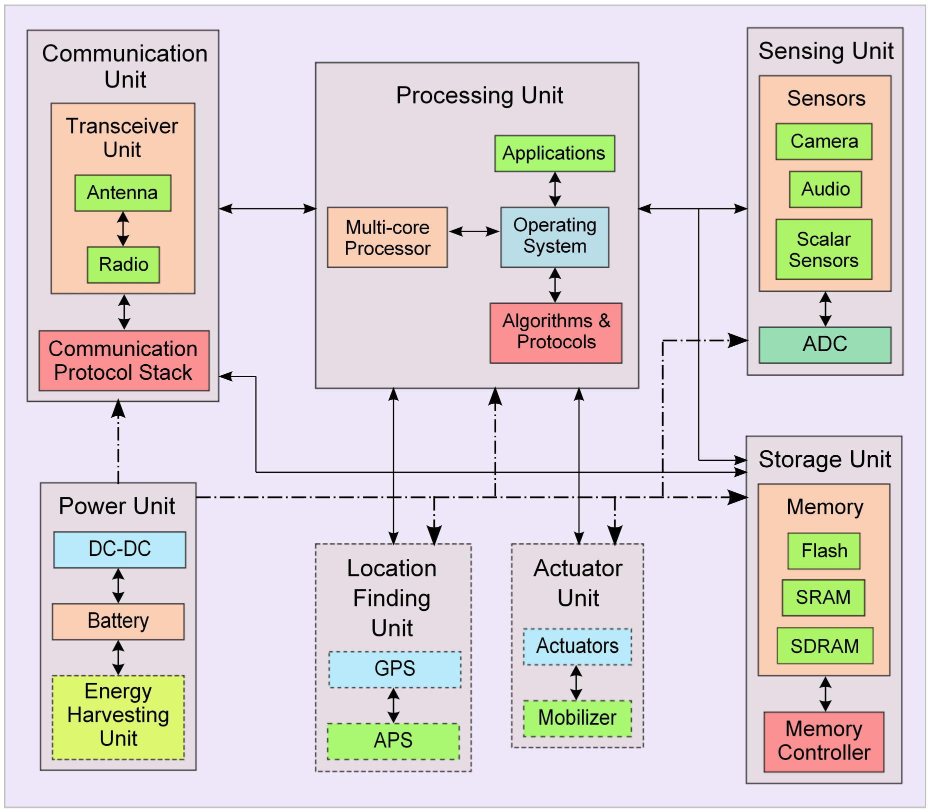
2.2 Processing Unit

The processing unit consists of a multi-core processor and is responsible for controlling sensors, gathering and processing sensed data, executing the system software that coordinates sensing, communication tasks, and interfacing with the storage unit. The processing unit for traditional sensor nodes consists of a single-core processor for general-purpose applications, such as periodic sensing of scalar data (e.g., temperature, humidity). High-performance single-core processors would be infeasible to meet computational requirements since these single-core processors would require operation at high processor voltage and frequency. A processor operating at a high voltage and frequency consumes an enormous amount of power since power increases proportionally to the operating processor frequency and square of the operating processor voltage. Furthermore, even if these energy issues are ignored, a single high-performance processor core may not be able to meet the computational requirements of emerging applications, such as multimedia sensor networks, in real-time.

Multi-core processors distribute the computations across the available cores, which speeds up the computations as well as conserves energy by allowing each processor core to operate at a lower processor voltage and frequency. Multi-core processors are suitable for streaming and complex, event-based monitoring applications, such as in smart camera sensor networks, that require data to be processed and compressed as well as require extraction of key information features. For example, the IC3D/Xetal single-instruction multiple-data (SIMD) processor, which consists of a linear processor array (LPA) with 320 reduced instruction set computers (RISC)/processors, is being used in smart camera sensor networks [5].

2.3 Storage Unit

The storage unit consists of the memory subsystem, which can be classified as *user memory* and *program memory*, and a memory controller, which coordinates memory accesses between different processor cores. The user memory stores sensed data when immediate data transmission is not possible due to hardware failures, environmental conditions, physical layer jamming, limited energy reserves, or when the data requires processing. The program memory is used for programming the embedded sensor node and using flash memory for the program memory provides persistent storage of application code and text segments. Static random-access memory (SRAM), which does not need periodic refreshing but is expensive in terms of area and power consumption, is used as dedicated processor memory. Synchronous dynamic random-access memory (SDRAM) is typically used as user memory. For example, the Imote2 embedded sensor node, which contains a Marvell PXA271 XScale processor operating



DC-DC: Direct Current to Direct Current Converter
SDRAM: Synchronous Dynamic Random-Access Memory
SRAM: Static Random-Access Memory
ADC: Analog to Digital Converter
GPS: Global Positioning System
APS: Ad hoc Positioning System

Fig. 1: Multi-core embedded sensor node architecture.

between 13 and 416 Mhz, has 256 KB SRAM, 32 MB Flash, and 32 MB SDRAM [6].

2.4 Communication Unit

The communication unit interfaces the embedded sensor node to the wireless network and consists of a transceiver unit (transceiver and antenna) and the communication unit software. The communication unit software mainly consists of the communication protocol stack, and the physical layer software in the case of software defined radio (SDR). The transceiver unit consists of either a wireless local area network (WLAN) card, such as an IEEE 802.11b compliant card, or an IEEE 802.15.4 compatible card, such as a Texas Instrument/Chipcon CC2420 chipset. The choice of a transceiver unit card depends on the application requirements such as desired range and allowable power. The maximum transmit power of IEEE 802.11b cards is higher as compared to IEEE 802.15.4 cards, which results in a higher communication range but consumes more power. For example, the Intel PRO/Wireless 2011 card has a data rate of 11 Mbps and a typical transmit power of 18 dBm, but draws 300 mA and 170 mA for sending and receiving, respectively. The CC2420 802.15.4 radio has a maximum data rate of 250 kbps and a transmit power of 0 dBm, but draws 17.4 mA and 19.7 mA for sending and receiving, respectively.

2.5 Power Unit

The power unit supplies power to various components/units on the embedded sensor node and dictates the sensor node's lifetime. The power unit consists of a battery and a DC-DC converter. The DC-DC converter provides a constant supply voltage to the sensor node. The power unit may be augmented by an optional energy-harvesting unit that derives energy from external sources, such as solar cells. Although multi-core embedded sensor nodes are more power efficient as compared to single-core embedded sensor nodes, energy-harvesting units in multi-core cluster heads and the sink node would prolong the MCEWSN's lifetime. Energy-harvesting units are more suitable for cluster heads and the sink node as these nodes perform more computations as compared to the single-core leaf sensor nodes. Furthermore, incorporating energy-harvesting units in only a few embedded sensor nodes (i.e., cluster heads and sink nodes) would not substantially increase the cost of EWSN deployment. Without an energy-harvesting unit, MCEWSNs would only be suitable for applications with relatively small lifetime requirements.

2.6 Actuator Unit

The optional actuator unit consists of actuators (e.g., motors, servos, linear actuators, air muscles, muscle wire, camera pan tilt, etc.) and an optional mobilizer unit for sensor node mobility. Actuators enhance the sensing task by opening/closing a switch/relay to control functions, such as a camera or antenna orientation and repositioning sensors. Actuators, in contrast to sensors that only sense a phenomenon, typically affect the operating environment by opening a valve, emitting sound, or physically moving the sensor node.

2.7 Location Finding Unit

The optional location finding unit determines a sensor node's location. Depending on the application requirements and available resources, the location finding unit can either be global positioning system (GPS)-based or ad hoc positioning system (APS)-based. Even though GPS is highly accurate, the GPS components are expensive and require direct line of sight between the sensor node and satellites. APS determines a sensor node's position with respect to defined *landmarks*, which may be other GPS-based sensor nodes [7]. A sensor node estimates the distance from itself to the landmark based on direct communication and the received communication signal strength. A sensor node that is two hops away from a landmark estimates its distance based on the distance estimate of a sensor node one hop away from a landmark via the message propagation. A sensor node with distance estimates to three or more landmarks can compute its own position via triangulation.

3 Multi-core Architectures and Parallel Computing Metrics

In this section, we describe the multi-core architectures that we evaluate in our study as well as parallel computing metrics that we leverage for this evaluation.

3.1 Multi-core Architectures

In this subsection, we give an overview of the two multi-core architectures that can be used as processing units in multi-core embedded sensor nodes (Fig. 1). We note that the operating frequency of the studied multi-core architectures is much higher than the ones that can be used for multi-core embedded sensor nodes. However, our purpose in this paper is to evaluate the architectural paradigms' feasibility for multi-core embedded sensor nodes and a lower operating frequency of the studied architectures in real multi-core embedded sensor nodes would only scale down the presented results without any significant changes to the performance trends. Hence, leveraging high computing power SMPs and TMAs will not affect the feasibility insights obtained from benchmark-driven cross-architectural evaluation, which is the intent of this work.

3.1.1 Symmetric Multiprocessors (SMPs)

In the parallel architecture domain, SMPs are the most pervasive and prevalent type, and are therefore an ideal processor candidate for MCEWSNs. SMPs offer a global physical address space, provide symmetric access to main memory, and have private caches. The processors and memory modules communicate over a shared interconnect, the most common being a shared bus [8]. We evaluate an eight-core Intel-based SMP consisting of two Intel Xeon E5430 quad-core processors fabricated at 45nm CMOS lithography [9] with a maximum clock frequency of 2.66 GHz. Each core contains 32 KB of level one instruction (L1-I) cache, 32 KB of level one data (L1-D) cache, and 12 MB of level two (L2) unified cache. Intel's enhanced front-side bus (FSB) running at 1333 MHz provides enhanced inter-core communication throughput [10]. For conciseness, we will refer to this SMP as $\text{SMP}^{\text{2xQuadXeon}}$ in the remainder of this paper.

3.1.2 Tiled Many-Core Architectures (TMAs)

TMAs are constructed using modular elements—*tiles*—which provides easy scalability to any arbitrary number of tiles. For intra-tile communication, each tile connects to a switch (communication router) within a high-performance interconnection network and each switch connects to a neighboring switch, which constrains the interconnection wire length to be no longer than the tile width. Examples of TMAs include the Intel's Tera-Scale research processor, the Raw processor, and Tilera's TILE*Pro*64 and TILE-Gx processor family [11][12]. Fig. 2 depicts our evaluated TMA, which is Tilera's TILEPro64 processor, fabricated at 90nm CMOS lithography and consists of 64 tiles (cores) in an 8 x 8 grid. Each tile has a three-way very long instruction word (VLIW) pipelined processor, which can execute up to three instructions per cycle (IPC). The switches are non-blocking, which provides a power-efficient on-chip interconnection mesh network operating at 31 Tbps. Each tile has 8 KB of L1-I cache, 8 KB of L1-D cache, and 64 KB of L2 cache, collectively providing 5 MB of on-chip cache with Tilera's *dynamic distributed cache (DDC)* technology. An operating system (OS) can be run independently on each tile or the tiles can be grouped to run a multi-processing OS (e.g., SMP Linux [13]). The TILEPro processor family is suitable for a variety of application domains, such as advanced networking, wireless infrastructure, telecom, digital multimedia, and cloud computing. Our prior work [14] provides further details on TMAs.

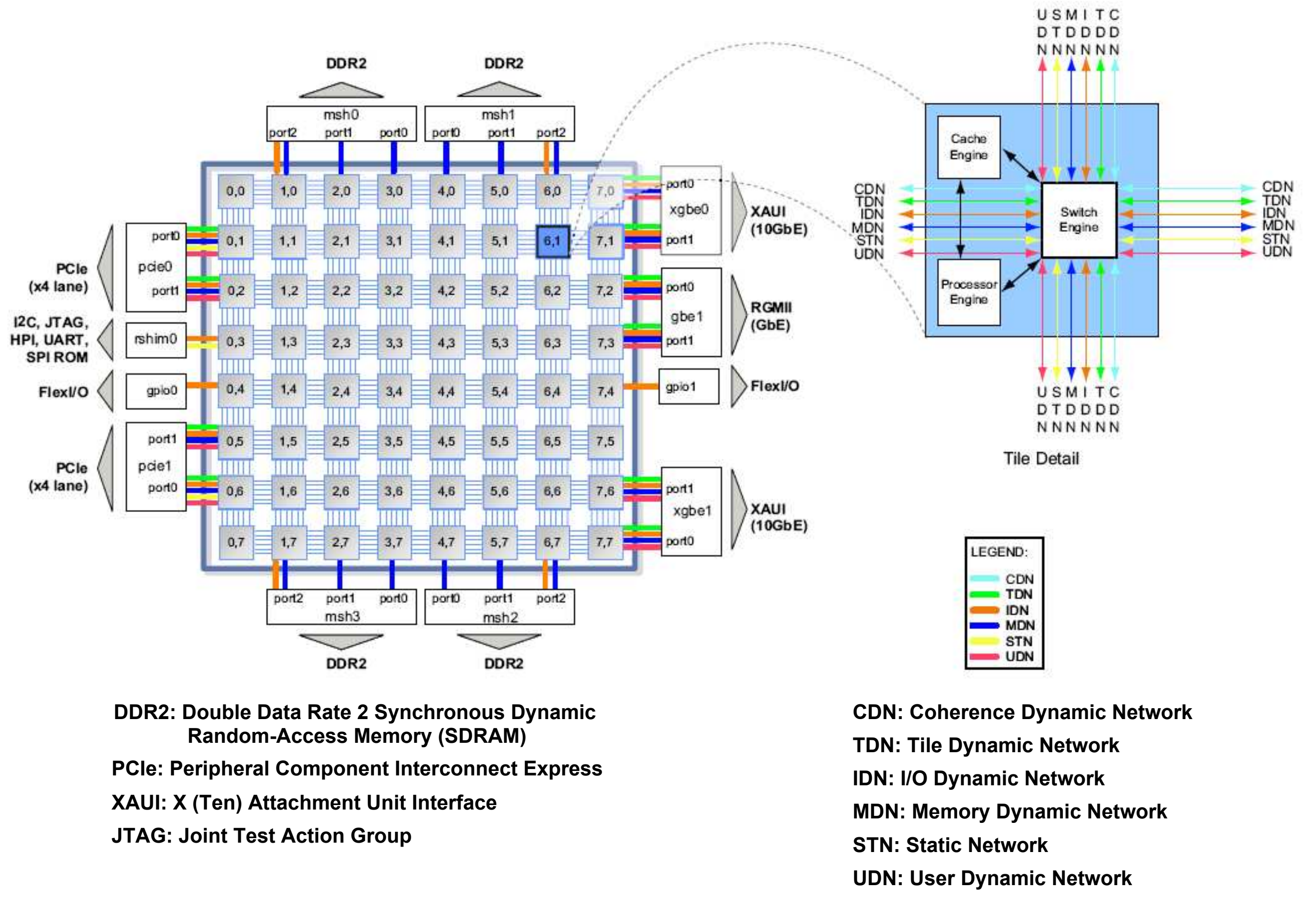

DDR2
msh0
msh1
msh2
msh3
port0
port1
port2
pcie0
pcie1
rshim0
gpio0
gpio1
xgbe0
xgbe1
gbe1
PCIe (x4 lane)
I2C, JTAG, HPI, UART, SPI ROM
FlexI/O
XAUI (10GbE)
RGMII (GbE)
Cache Engine
Switch Engine
Processor Engine
Tile Detail
LEGEND:
CDN
TDN
IDN
MDN
STN
UDN
DDR2: Double Data Rate 2 Synchronous Dynamic Random-Access Memory (SDRAM)
PCIe: Peripheral Component Interconnect Express
XAUI: X (Ten) Attachment Unit Interface
JTAG: Joint Test Action Group
CDN: Coherence Dynamic Network
TDN: Tile Dynamic Network
IDN: I/O Dynamic Network
MDN: Memory Dynamic Network
STN: Static Network
UDN: User Dynamic Network

Fig. 2: Tilera's TILEPro64 processor (adapted from [15]).

3.2 Parallel Computing Device Metrics

In this section, we define the metrics used to quantitatively compare our investigated multi-core architectures.

Run Time: The *serial run time* T_s of a program is the time required to execute the program on a sequential computer. The *parallel run time* T_p is the time elapsed from the start of a program to the moment the last processor finishes execution.

Speedup: The speedup S measures the performance gain achieved via application parallelization as compared to the execution time of the best sequential implementation of the application. S is defined as the ratio of the *serial run time* T_s to the *parallel run time* T_p to solve the same problem (i.e., $S = T_s/T_p$).

Efficiency: The fraction of time a processor is actively executing an application is the system's efficiency E. E is computed as the ratio of the speedup S to the number of processors p (i.e., $E = S/p$).

Cost: The collective processor time required to execute an application in a parallel system is the system's cost C. C on a parallel system is computed as the product of the parallel run time T_p and the number of processors p (i.e., $C = T_p \cdot p$). A parallel system is *cost optimal* if the parallel system's cost is proportional to the execution time of the best known sequential algorithm on a single processor [16].

Scalability: A parallel system's scalability evaluates the efficiency of application parallelization as the number of processors increases, wherein an optimally-scalable parallel system maintains a speedup increase proportional to the increase in the number of processors and the problem size [16].

Power: A processor's total (system-level) power consumption comprises both the dynamic and static power consumptions. The dynamic power consumption depends on the supply voltage, clock frequency, capacitance, and the signal activity whereas the static power consumption mainly depends on the supply voltage, temperature, and capacitance [17]. Our system-level power model estimates a multi-core system's power consumption, and considers both the active and the idle mode power consumptions. Our power estimation model can be used to estimate the system's performance per watt. The power consumption of a system with N processor cores and p active processor cores is:

$$P^p = p \cdot \frac{P^{active}_{max}}{N} + (N - p) \cdot \frac{P^{idle}_{max}}{N} \qquad (1)$$

where P^{active}_{max} and P^{idle}_{max} denote the system's maximum active and idle mode power consumptions, respectively. P_{max}^{active}/N and P_{max}^{idle}/N give the active and idle mode power consumptions per core (and the associated switching and interconnection network circuitry), respectively. We consider state-of-the-art power saving mechanisms, such as instructions to switch idle cores and associated circuitry (switches, clock, interconnection network) into a low-power idle state (e.g., Tilera's `NAP` instruction puts a tile into a low-power `IDLE` mode [18]).

Performance per Watt: Performance per watt evaluates a device's delivered/attainable performance while taking the device's power consumption into consideration. We report performance with respect to MOPS or Mega floating point operations per second (MFLOPS), and performance per watt with respect to MOPS per watt (MOPS/W) or MFLOPS per watt (MFLOPS/W).

4 Compute-Intensive Tasks Motivating the Emergence of MCEWSNs

Many applications require embedded sensor nodes to perform various compute-intensive tasks that often exceeds the computing capability of traditional single-core sensor nodes. These tasks include information fusion, encryption, network coding, software defined radio, etc., and motivate the emergence of MCEWSNs. In this section, we discuss these compute-intensive tasks requiring multi-core support in an embedded sensor node.

4.1 Information Fusion

A critical processing task in EWSNs is information fusion, which can benefit from a multi-core processor in an embedded sensor node. EWSNs produce a large amount of data that must be processed, delivered, and assessed according to application objectives. Since the transmission bandwidth is limited, information fusion condenses the sensed data and transmits only the selected fused information to the sink node. Additionally, the data received from neighboring sensor nodes is often redundant and highly correlated, which warrants fusing the sensed data. Formally, information fusion encompasses theory, techniques, and tools created and applied to exploit the synergy in the information acquired from multiple sources (sensors, databases, etc.) such that the resulting fused data/information is considered qualitatively or quantitatively better in terms of accuracy or robustness than the acquired data from any single data source [19]. Data aggregation is an instance of information fusion in which the data from various sources is aggregated using summarization functions (e.g., minimum, maximum, and average) that reduce the volume of data being manipulated. Information fusion can reduce the amount of data traffic, filter noisy measurements, and make predictions and inferences about a monitored entity.

Information fusion can be computationally expensive, especially for video sensing applications. Unlike scalar data, which can be combined using relatively simple mathematical manipulations such as average and summation, video data is vectorial and requires complex computations to fuse (e.g., edge detection, histogram formation, compression, filtering, etc.). Reducing transmission overhead via information fusion in video sensor networks requires a substantial increase in intermediate processing, which warrants the use of multi-core cluster heads in MCEWSNs. Multi-core cluster heads fuse data received from multiple sensor nodes to eliminate redundant transmission and provide fused information to the sink node with minimum data latency. Data latency is the sum of the delay involved in data transmission, routing, and information fusion/data aggregation [20]. Data latency is important in many applications, especially real-time applications, where freshness of data is an important factor. Multi-core cluster heads can fuse data much faster than single-core sensor nodes, which justifies the use of multi-core cluster heads in MCEWSNs with complex real-time computing requirements.

Omnibus Model for Information Fusion: The Omnibus model [21] guides information fusion for sensor-based devices. Fig. 3 illustrates the Omnibus model with respect to our MCEWSN architecture and we exemplify the model's usage by considering a surveillance application performing target tracking based on acoustic sensors [19]. The *Observe* stage, which can be carried out at single-core sensor nodes and/or multi-core cluster heads, uses a filter (e.g., moving average filter) to reduce noise (*Signal Processing*) from acoustic sensor data provided by the embedded sensor nodes (*Sensing*). The *Orientate* stage, which is carried out at multi-core cluster heads, uses the filtered acoustic data for range estimation (*Feature Extraction*) and estimates the target's location and trajectory (*Pattern Processing*). The *Decide* stage, which is carried out at multi-core cluster heads and/or multi-core sink nodes, classifies the sensed target (*Context Processing*) and determines whether the target represents a threat (*Decision Making*). If the target is a threat, the *Act* stage, which is carried out at the control and analysis center (CAC), intercepts the target (*Control*) (e.g., with a missile) and activates available armaments (*Resource Tasking*).

4.2 Encryption

Security is an important issue in many sensor networking applications since sensors are deployed in open environments and are susceptible to malicious attacks. The sensed and/or aggregated data must be encrypted for secure transmission to the sink node. The two main practical issues involved in encryption are the size of the encrypted message and the encryption execution time. Privacy homomorphisms (PHs) are encryption functions suitable for MCEWSNs that allow

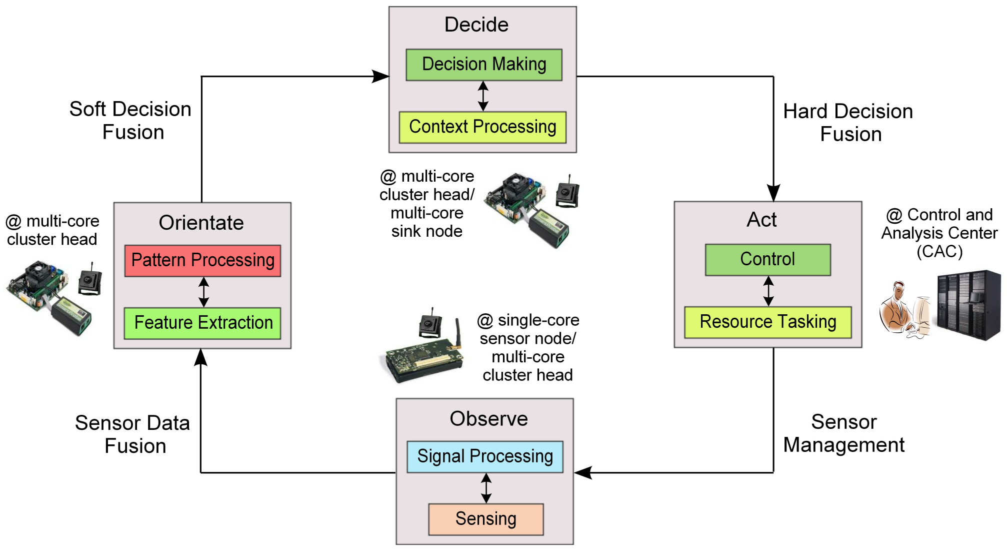


Fig. 3: Omnibus sensor information fusion model for an MCEWSN architecture.

a set of operations to be performed on encrypted data without knowing the decryption functions [20]. PHs use a positive integer $d \geq 2$ for computing the secret key for encryption such that the size of the encrypted data increases by a factor of d as compared to the original data. The security of the encrypted data increases with d as well as the execution time for encryption. For example, the execution time for encryption of one byte of data is 3,481 clock cycles on a MICA2 mote when $d = 2$ and increases to 4,277 clock cycles when $d = 4$. MICA2 motes cannot handle the computations for $d \geq 4$ [20], hence, applications requiring greater security require multi-core sensor nodes and/or cluster heads to perform these computations.

4.3 Network Coding

Network coding is a coding technique to enhance network throughput in multi-nodal environments, such as EWSNs. Despite the effectiveness of network coding for EWSNs, excessive decoding cost associated with network coding hinders the technique's adoption in traditional EWSNs with constrained computing power [22]. Future MCEWSNs will enable adoption of sophisticated coding techniques, such as network coding to increase network throughput.

4.4 Software Defined Radio (SDR)

SDR is a radio in which some or all of the physical layer functions execute as software. The radio in existing EWSNs is hardware-based, which results in higher production costs and minimal flexibility in supporting multiple waveform standards [23]. MCEWSNs can realize SDR-based radio by enabling fast, parallel computation of signal processing operations needed in SDR (e.g., fast Fourier transform (FFT)). SDR-based MCEWSNs would enable multi-mode, multi-band, and multi-functional radios that can be enhanced using software upgrades.

5 Multi-core Embedded Sensor Nodes

Several initiatives towards multi-core embedded sensor nodes have been undertaken by academia and industry for various real-time applications. In this section, we describe several state-of-the-art multi-core embedded sensor node prototypes.

5.1 InstraNode

InstraNode is a dual-core sensor node for real-time health monitoring of civil structures, such as highway bridges and skyscrapers. InstraNode is equipped with a 4000 mAh lithium-ion battery, three accelerometers, a gyroscope, and an IEEE 802.11b (Wi-Fi) card for communication with other nodes. One low-power processor core in InstraNode runs at 3 V and 4 MHz and is dedicated to sampling data from sensors whereas the other faster, high-power processor core runs at 4.3 V and 40 MHz and is responsible for networking tasks, such as transmission/reception of data and execution of a routing algorithm. Furthermore, InstraNode possesses multi-modal operation capabilities such as wired/wireless and battery-powered/AC-adaptor powered options. Experiments indicate that the InstraNode outperforms single-core sensor nodes in terms of power-efficiency and network performance [24].

5.2 Mars Rover Prototype Mote

Etchison et al. [25] have proposed a high-performance EWSN for the Mars Rover, which consists of dual-core mobile sensor nodes and a wireless cluster consisting of multiple processors to process image data gathered from the sensor nodes and to make decisions based on gathered information. The prototype mote consists of a Micro ATX motherboard with Intel's dual-core Atom processor, 2 GB of RAM, and is powered by a 12 V/5 A DC power supply for lab testing. Each mote performs data acquisition, processing, and transmission.

5.3 Satellite-Based Sensor Node (SBSN)

Vladimirova et al. [26] have developed a system-on-chip (SoC) satellite-based sensor node (SBSN). The SBSN prototype contains a SPARC V8 LEON3 soft processor core, which allows configuration in an SMP architecture [27]. The LEON3 processor core runs software applications and interfaces with the upper layers of the communication stack using the IEEE 802.11 protocol. The SBSN prototype uses a number of intellectual property (IP) cores, such as a hardware accelerated Wi-Fi MAC, a transceiver core, and a Java co-processor. The Java co-processor enables distributed computing and Internet protocol (IP)-based networking functions in SBWSNs. The inter-satellite communication module (ISCM) in the SBSN prototype adheres to IEEE 802.11 and CubeSat design specifications. The ISCM supports ground communication links and inter-satellite links (ISLs) at variable data rates and configurable waveforms to adapt to channel conditions. The ISCM incorporates S-band (2.4 GHz) and a 434/144 MHz radio frontend interfaced to a single reconfigurable modem. The ISCM uses a high-end AD9861 ADC/digital-to-analog converter (DAC) for the 2.4 GHz radio frontend for a Maxim 2830 radio and a low-end AD7731 for the 434/144 MHz frontend for an Alinco DJC-7E radio. Additionally, ISCM incorporates current and temperature sensors and a 16-bit microcontroller for housekeeping purposes.

5.4 Multi-CPU-based Sensor Node Prototype

Ohara et al. [28] have developed a prototype for an embedded sensor node using three PIC18 central processing units (CPUs). The prototype is supplied by a configurable voltage stabilized power supply, but the same voltage is supplied to all CPUs. The prototype allowed each CPU's frequency to be statically changed by changing a corresponding ceramic resonator. Experiments revealed that the multi-CPU sensor node prototype consumed 76% less power as compared to a single-core sensor node for benchmarks that involved sampling, root mean square calculation, and pre-processing samples for transmission.

5.5 Smart Camera Mote

Kleihorst et al. [29] developed a smart camera mote, which consists of four basic components: color image sensors, an IC3D SIMD processor (a member of the Philips' Xetal family of SIMD processors) for low-level image processing, a general purpose processor for intermediate and high-level processing and control, and a communication module. Both of the processors are coupled with a dual-port random-access memory (RAM) that enables these processors to work in a shared workspace. The IC3D SIMD processor consists of a linear array of 320 RISC processors. The peak pixel performance of the IC3D processor is approximately 50 Giga operations per second (GOPS). Despite high pixel performance, the IC3D processor is an inherently low-power processor, which makes the processor suitable for multi-core embedded sensor nodes. The power consumption of the IC3D processor for typical applications, such as feature finding or face detection, is below 100 mW in active processing modes.

6 Results

In this section, we describe the information fusion application experimental setup details.[1]

We consider a hierarchical MCEWSN for information fusion such that each cluster head receives sensing measurements from ten single-core sensor nodes equipped with temperature, pressure, humidity, acoustic, magnetometer, accelerometer, gyroscope, proximity, and orientation sensors [30]. To reduce the random white noise from sensor measurements, a moving average filter, which computes the arithmetic mean of a number of input measurements to produce each output measurement, is executed on the cluster head. Given an input sensor measurement vector $\boldsymbol{x} = (x(1), x(2), \ldots)$, the moving average filter estimates the true sensor measurement vector after noise removal $\boldsymbol{y} = (\hat{y}(1), \hat{y}(2), \ldots)$ as:

$$\hat{y}(k) = \frac{1}{M} \sum_{i=0}^{M-1} x(k-i), \ \ \forall k \geq M \tag{2}$$

where the filter window size M indicates the number of fused input sensor measurements. Given sensor measurements with random white noise, the moving average filter reduces the noise variance by a factor of $\sqrt{M}$. M should be chosen such that M is the smallest value that reduces the noise in accordance with the application's requirements. After calculating the cluster's nodes' filtered sensor measurements (i.e., after applying moving average filter) for each of the sensor node in the cluster, the cluster head determines the sensed measurements minimum, maximum, and average values. This information fusion requires $100 \cdot N(3+M)$ operations with a runtime complexity of $\mathcal{O}(NM)$ where N is the number of sensor measurements. Our results evaluate a parallelized information fusion application using our parallel performance metrics (Section 3.2) to illustrate the advantages for leveraging multi-core as compared to single core architectures for cluster heads.

Acknowledgments

This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and the National Science Foundation (NSF) (CNS-0953447 and CNS-0905308). Any opinions, findings, and conclusions or recommendations expressed in this

1. Detailed results for the information fusion application are presented in Section 4 of the main paper.

material are those of the author(s) and do not necessarily reflect the views of the NSERC and the NSF.

References

[1] M. Hamdi, N. Boudriga, and M. Obaidat, "Bandwidth-Effecitve Design of a Satellite-Based Hybrid Wireless Sensor Network for Mobile Target Detection and Tracking," *IEEE Systems Journal*, vol. 2, no. 1, pp. 74–82, March 2008.

[2] I. F. Akyildiz, T. Melodia, and K. R. Chowdhury, "Wireless Multimedia Sensor Networks: Applications and Testbeds," *Proceedings of the IEEE*, vol. 96, no. 10, pp. 1588–1605, October 2008.

[3] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks: A Survey," *Elsevier Computer Networks*, vol. 38, no. 4, pp. 393–422, March 2002.

[4] A. Y. Dogan, D. Atienza, A. Burg, I. Loi, and L. Benini, "Power/Performance Exploration of Single-Core and Multi-core Processor Approaches for Biomedical Signal Processing," in *Proc. of the Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, Madrid, Spain, September 2011.

[5] A. Kak, "Parallel Histogram-based Particle Filter for Object Tracking on SIMD-based Smart Cameras," in *Purdue Robot Vision Lab*, Purdue University, West Lafayette, Indiana, October 2011. [Online]. Available: https://engineering.purdue.edu/RVL/Research/SIMD_PF/index.html

[6] MEMSIC, "Imote2 Hardware Bundle for Wireless Sensor Networks," October 2011. [Online]. Available: www.memsic.com

[7] A. Munir and A. Gordon-Ross, "Optimization approaches in wireless sensor networks," in *Sustainable Wireless Sensor Networks*, W. Seah and Y. K. Tan, Eds. InTech, 2010. [Online]. Available: http://www.intechopen.com/articles/show/title/optimization-approaches-in-wireless-sensor-networks

[8] D. Culler, J. Singh, and A. Gupta, *Parallel Computer Architecture: A Hardware/Software Approach*. Morgan Kaufmann Publishers, Inc., 1999.

[9] Intel, "Intel Xeon Processor E5430," April 2011. [Online]. Available: http://ark.intel.com/Product.aspx?id=33081

[10] ——, "Quad-Core Intel Xeon Processor 5400 Series Datasheet," August 2008. [Online]. Available: http://www.intel.com/assets/PDF/datasheet/318589.pdf

[11] S. Keckler, K. Olukotun, and H. Hofstee, *Multicore Processors and Systems*. Springer, 2009.

[12] TILERA, "TILEPro Processor Family," March 2013. [Online]. Available: http://www.tilera.com/products/processors/TILEPro_Family

[13] IBM, "Linux and Symmetric Multiprocessing," May 2011. [Online]. Available: http://www.ibm.com/developerworks/library/l-linux-smp/

[14] A. Munir, F. Koushanfar, A. Gordon-Ross, and S. Ranka, "High-Performance Optimizations on Tiled Many-Core Embedded Systems: A Matrix Multiplication Case Study," *The Journal of Supercomputing*, 2013.

[15] TILERA, "Tile Processor Architecture Overview for the TILEPro Series," in *Tilera Official Documentation*, November 2009.

[16] V. Kumar, A. Grama, A. Gupta, and G. Karypis, *Introduction to Parallel Computing*. The Benjamin/Cummings Publishing Company, Inc., 1994.

[17] E. Musoll, "A Cost-Effective Load-Balancing Policy for Tile-Based, Massive Multi-Core Packet Processors," *ACM Transactions on Embedded Computing Systems*, vol. 9, no. 3, February 2010.

[18] TILERA, "Tile Processor Architecture Overview for the TILEPro Series," in *Tilera Official Documentation*, November 2009.

[19] E. F. Nakamura, A. A. Loureiro, and A. C. Frery, "Information Fusion for Wireless Sensor Networks: Methods, Models, and Classifications," *ACM Computing Surveys*, vol. 39, no. 3, August 2007.

[20] R. Rajagopalan and P. Varshney, "Data-Aggregation Techniques in Sensor Networks: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 8, no. 4, pp. 48–63, 2006.

[21] M. Bedworith and J. O'Brien, "The Omnibus Model: A New Model for Data Fusion?" *IEEE Aerospace and Electronic Systems Magazine*, vol. 15, no. 4, pp. 30–36, April 2000.

[22] D. Kim, K. Park, and W. Ro, "Network Coding on Heterogeneous Multi-Core Processors for Wireless Sensor Networks," *Sensors*, vol. 11, no. 8, pp. 7908–7933, 2011.

[23] G. R. Murthy, "Control, Communication and Computing Units: Converged Architectures," *International Journal of Computer Applications*, vol. 1, no. 4, pp. 49–54, 2010.

[24] C. Park, Q. Xie, and P. Chou, "InstraNode: Dual-Microcontroller Based Sensor Node for Real-Time Structural Health Monitoring," in *Proc. of the IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON)*, Santa Clara, California, September 2005.

[25] J. Etchison, G. Skelton, Q. Pang, and T. Hulitt, "Mobile Intelligent Sensor Network Used for Data Processing," Jackson Sate University, Jackson, Mississippi, 2010. [Online]. Available: http://www.iiis.org/CDs2010/CD2010SCI/SCI_2010/PapersPdf/SA874PZ.pdf

[26] T. Vladimirova, C. Bridges, J. Paul, S. Malik, and M. Sweeting, "Space-based Wireless Sensor Networks: Design Issues," in *Proc. of the IEEE Aerospace Conference*, Big Sky, Montana, March 2009.

[27] Aeroflex, "Leon3 Processor," October 2011. [Online]. Available: http://www.gaisler.com/cms/index.php?option=com_content&task=view&id=13&Itemid=53

[28] S. Ohara, M. Suzuki, S. Saruwatari, and H. Morikawa, "A Prototype of a Multi-Core Wireless Sensor Node for Reducing Power Consumption," in *Proc. of the International Symposium on Applications and the Internet (SAINT)*, Turku, Finland, July 2008.

[29] R. Kleihorst, B. Schueler, A. Danilin, and M. Heijligers, "Smart Camera Mote with High Performance Vision System," in *Proc. of the Workshop on Distributed Smart Cameras (DSC)*, Boulder, Colorado, October 2006.

[30] Android, "SensorEvent," November 2011. [Online]. Available: http://developer.android.com/reference/android/hardware/SensorEvent.html