Parallelized Benchmark-Driven Performance Evaluation of SMPs and Tiled Multi-Core Architectures for Embedded Systems

Arslan Munir
Department of Electrical and Computer Engineering
Rice University, Houston, Texas
Email: arslan@rice.edu

Ann Gordon-Ross*, and Sanjay Ranka+
*Department of Electrical and Computer Engineering
+Department of Computer and Information Science and Engineering
University of Florida, Gainesville, Florida, USA
Email: ann@ece.ufl.edu, ranka@cise.ufl.edu

***Abstract*—With Moore's law supplying billions of transistors on-chip, embedded systems are undergoing a transition from single-core to multi-core to exploit this high transistor density for high performance. However, there exists a plethora of multi-core architectures and the suitability of these multi-core architectures for different embedded domains (e.g., distributed, real-time, reliability-constrained) requires investigation. Despite the diversity of embedded domains, one of the critical applications in many embedded domains (especially distributed embedded domains) is information fusion. Furthermore, many other applications consist of various kernels, such as Gaussian elimination (used in network coding), that dominate the execution time. In this paper, we evaluate two embedded systems multi-core architectural paradigms: symmetric multiprocessors (SMPs) and tiled multi-core architectures (TMAs). We base our evaluation on a parallelized information fusion application and benchmarks that are used as building blocks in applications for SMPs and TMAs. We compare and analyze the performance of an Intel-based SMP and Tilera's TILEPro64 TMA based on our parallelized benchmarks for the following performance metrics: runtime, speedup, efficiency, cost, scalability, and performance per watt. Results reveal that TMAs are more suitable for applications requiring integer manipulation of data with little communication between the parallelized tasks (e.g., information fusion) whereas SMPs are more suitable for applications with floating point computations and a large amount of communication between processor cores.**

***Keywords*-multi-core, embedded systems, performance evaluation**

I. Introduction

As chip transistor counts increase, embedded system design has shifted from single-core to multi- and many-core architectures. A primary reason for this architecture reformation is that performance speedups are becoming more difficult to achieve by simply increasing the clock frequency of traditional single-core architectures because of limitations in power dissipation. This single-core to multi-core paradigm shift in embedded systems has introduced parallel computing to the embedded domain, which was previously predominantly used in supercomputing only. Furthermore, with respect to the computing industry, this paradigm has led to the proliferation of diverse multi-core architectures, which necessitates comparison and evaluation of these disparate architectures for different embedded domains (e.g., distributed, real-time, reliability-constrained).

Contemporary multi-core architectures are not designed to deliver high performance for all embedded domains, but are instead designed to provide high performance for a subset of these domains. The precise evaluation of multi-core architectures for a particular embedded domain requires executing complete applications prevalent in that domain. Despite the diversity of embedded domains, the critical application for many embedded domains (especially distributed embedded domains of which embedded wireless sensor networks (EWSNs) are a prominent example) is information fusion, which fuses/condenses the information from multiple sources. Furthermore, many other applications consist of various kernels, such as Gaussian elimination (GE), which is used in network coding, that dominate the computation time [1]. An embedded domain's parallelized applications and kernels provide an effective way of evaluating multi-core architectures for that embedded domain.

In this paper, we evaluate two multi-core architectures for embedded systems: symmetric multiprocessors (SMPs) and Tilera's tiled multi-core architectures (TMAs). We consider SMPs because SMPs are ubiquitous and pervasive, which provides a standard/fair basis for comparing with other novel architectures, such as TMAs. We consider Tilera's TILEPro64 TMAs because of this architecture's innovative architectural features such as three-way issue superscalar tiles, on-chip mesh interconnect, and dynamic distributed cache (DDC) technology.

In some cases, such as with Tilera's TILEPro64, the multi-core architecture directly dictates the high-level parallel language used as some multi-core architectures support proprietary parallel languages whose benchmarks are not available open source. Tilera provides a proprietary multi-core development environment (MDE) ilib API [2]. Many SMPs are more flexible, such as the Intel-based SMP, which supports OpenMP (Open Multi-processing). These differences in supported languages makes *cross-architectural evaluation* challenging since the results may be affected by the parallel language's efficiency. However, our analysis provides insights into the attainable performance per watt from these two multi-core architectures.

To the best of our knowledge, this paper is the first to evaluate SMPs and TMAs for multi-core parallel embedded systems. We parallelize an information fusion application, a GE kernel, and an embarrassingly parallel (EP) benchmark for SMPs and TMAs to compare and analyze the architectures' performance and performance per watt. This parallelized benchmark-driven evaluation provides deeper insights as compared to a theoretical quantitative approach.

Our cross-architectural evaluation results reveal that TMAs outperform SMPs with respect to scalability and performance per watt for applications involving integer operations on data with little communication between processor cores (processor cores are also referred as *tiles* in TMAs). For applications requiring floating point (FP) operations and frequent dependencies between computations, SMPs outperform TMAs with respect to scalability and performance per watt.

II. Related Work

In the area of parallelization of algorithms and performance analysis of SMPs, Brown et al. [3] compared the performance and programmability of the Born calculation (a model used to study the interactions between a protein and surrounding water molecules) using both OpenMP and Message Passing Interface (MPI). The authors observed that the OpenMP version's programmability and performance outperformed the MPI version, however, the scalability of the MPI version was superior to the OpenMP version. Our work differs from previous parallel programming work in that we compare parallel implementations of different benchmarks using OpenMP and Tilera's ilib proprietary API for two multi-core architectures as opposed to comparing OpenMP with MPI, both of which are not proprietary, as in many previous works.

Bikshandi et al. [4] investigated the performance of TMAs and demonstrated that hierarchical tiled arrays yielded increased performance on parallelized benchmarks, such as matrix multiplication (MM) and NASA advanced supercomputing (NAS) benchmarks, by improving the data locality. Zhu et al. [5] presented a performance study of OpenMP language constructs on the IBM Cyclops-64 (C64) architecture that integrated 160 processing cores on a single chip. The authors observed that the overhead of the OpenMP language constructs on the C64 architecture was at least one order of magnitude lower as compared to the previous work on conventional SMP systems.

Some previous work investigated multi-core architectures for distributed embedded systems. Dogan et al. [6] evaluated a single- and multi-core architecture 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 that averaged 50.1 Mega operations per seconds (MOPS). However, the multi-core architecture consumed 10.4% more power than the single-core architecture for relatively light computation workloads that averaged 681 Kilo operations per second (KOPS). Kwok et al. [7] proposed FPGA-based multi-core computing for batch processing of image data in distributed EWSNs. Results revealed that the speedup obtained by FPGA-based acceleration at 20 MHz for edge detection, an image processing technique, was 22x as compared to a 48 MHz MicroBlaze microprocessor. Our work differs from the Kwok's work in that we study the feasibility of two fixed logic multi-core architecture paradigms, SMPs and TMAs, instead of reconfigurable logic.

Although there exists work for independent performance evaluation of SMPs and TMAs, to the best of our knowledge there is no previous work that cross-evaluates these architectures, which is the focus of our work.

III. Multi-Core Architectures and Benchmarks

Many applications require embedded systems to perform various compute-intensive tasks that often exceed the computing capability of traditional single-core embedded systems. In this section, we describe the multi-core architectures along with the applications and/or kernels that we leverage to evaluate these architectures.

A. Multi-Core Architectures

1) Symmetric Multiprocessors: SMPs are the most pervasive and prevalent type of parallel architecture that provides a global physical address space and symmetric access to all of main memory from any processor core. Every processor has a private cache and all of the processors and memory modules attach to a shared interconnect, typically a shared bus [1]. In our evaluations, we study an Intel-based SMP, which is an 8-core SMP consisting of two chips containing 45 nm Intel Xeon E5430 quad-core processors [8] (henceforth we denote the Intel-based SMP as $\text{SMP}^{\text{2xQuadXeon}}$). The Xeon E5430 quad-core processor chip offers a maximum clock frequency of 2.66 GHz, integrates a 32 KB level one instruction (L1-I) and a 32 KB level one data (L1-D) cache per core, a 12 MB level two (L2) unified cache (a dual core option with a 6 MB L2 cache is also available), and a 1333 MHz front side bus (FSB). The Xeon E5430 leverages Intel's enhanced front-side bus running at 1333 MHz, which enables enhanced throughput between each of the processor cores [9].

2) Tiled Multi-Core Architectures: TMAs exploit massive on-chip resources by combining each processor core with a switch to create a modular element called a *tile*, which can be replicated to create a multi-core architecture with any number of tiles. TMAs contain a high-performance interconnection network that constrains interconnection wire length to be no longer than the tile width and a switch (communication router) interconnects neighboring switches. Examples of TMAs include the Raw processor, Intel's Tera-Scale research

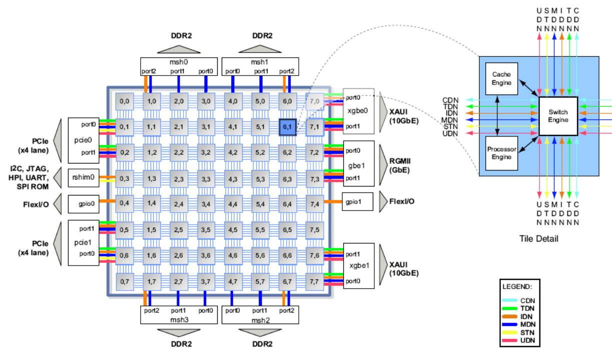
Fig. 1: Tilera TILEPro64 processor [10].

processor, and Tilera's TILE64, TILE*Pro*64, and TILE-Gx processor family [11][12][13]. In our evaluations, we study the TILEPro64 processor depicted in Fig. 1. The TILEPro64 processor features an 8 x 8 grid of 64 90nm tiles (cores) where each tile consists of a three-way very long instruction word (VLIW) pipelined processor capable of delivering up to three instructions per cycle, integrated L1 and L2 caches, and a non-blocking switch that integrates the tile into a power-efficient 31 Tbps on-chip interconnect mesh. Each tile has a 16 KB L1 cache (8 KB instruction cache and 8 KB data cache) and a 64 KB L2 cache, resulting in a total of 5 MB of on-chip cache with Tilera's *dynamic distributed cache (DDC)* technology. Each tile can independently run a complete operating system or multiple tiles can be grouped together to run a multi-processing operating system, such as SMP Linux [14].

B. Benchmark Applications and Kernels

1) Information Fusion: A crucial processing task in distributed embedded systems is information fusion. Distributed embedded systems, such as EWSNs, produce a large amount of data that must be processed, delivered, and assessed according to application objectives. Since the transmission bandwidth is oftentimes limited, information fusion condenses the sensed data and transmits only the selected, fused information to a base station node for further processing and/or evaluation by an operator. Information fusion is also used to reduce redundancy in the received data since the data gathered from neighboring sources/embedded nodes is typically highly correlated or redundant.

For our evaluations, we parallelize an information fusion application both for SMPs and TMAs to investigate the suitability of the two architectures for distributed embedded systems. We consider a hierarchical distributed embedded system consisting of embedded sensor nodes where each cluster head receives sensing measurements from ten single-core embedded sensor nodes equipped with temperature, pressure, humidity, acoustic, magnetometer, accelerometer, gyroscope, proximity, and orientation sensors [15]. The cluster head implements a moving average filter, which computes the arithmetic mean of a number of input measurements to produce each output measurement, to reduce random white noise from sensor measurements. Given an input sensor measurement vector $\boldsymbol{x} = (x(1), x(2), \ldots)$, the moving average filter estimates the true senor measurement vector after noise removal $\boldsymbol{y} = (\hat{y}(1), \hat{y}(2), \ldots)$ as:

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

where M is the filter's window, which dictates the number of input sensor measurements to fuse for noise reduction. When the sensor measurements have random white noise, the moving average filter reduces the noise variance by a factor of $\sqrt{M}$. For practical distributed embedded systems, M can be chosen as the smallest value that can reduce the noise to meet the application requirements. For each of the filtered sensor measurements for each of the embedded sensor nodes in the cluster, the cluster head calculates the minimum, maximum, and average of the sensed measurements. This information fusion application requires $100 \cdot N(3+M)$ operations with complexity $\mathcal{O}(NM)$ where N denotes the number of sensor samples.

2) Gaussian Elimination: The GE kernel solves a system of linear equations and is used in many scientific applications, including the Linpack benchmark used for ranking supercomputers in the TOP500 list of the world's fastest computers [16][17], and in distributed embedded systems. For example, the decoding algorithm for network coding uses a variant of GE (network coding is a coding technique to enhance network throughput in distributed embedded systems) [18]. The sequential runtime of the GE algorithm is $\mathcal{O}(n^3)$. Our GE kernel computes an upper-triangularization of matrices and requires $(2/3) \cdot n^3 + (7/4) \cdot n^2 + (7/2) \cdot n$ FP operations, which includes the extra operations required to make the GE algorithm numerically stable.

3) Embarrassingly Parallel Benchmark: The EP benchmark is typically used to quantify the peak attainable performance of a parallel computer architecture. Our EP benchmark generates normally distributed random variates that are used in simulation of stochastic applications [19]. We leverage Box-Muller's algorithm, which requires $99n$ FP operations assuming that square root requires 15 FP operations and logarithm, cosine, and sine each require 20 FP operations [20].

IV. Parallel Computing Device Metrics

Parallel computing device metrics provide a means to compare different parallel architectures, and the most appropriate device metrics depends upon the targeted application domain. For example, runtime (performance) may be an appropriate metric for comparing high-performance data servers whereas performance per watt is a more appropriate metric for embedded systems that have a limited power budget. In this section, we characterize the metrics that we leverage in our study to compare parallel architectures.

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

Speedup: Speedup measures the performance gain achieved by parallelizing a given application/algorithm over the best sequential implementation of that application/algorithm. Speedup S is defined as T_s/T_p, which is the ratio of the serial run time T_s of the best sequential algorithm for solving a problem to the time taken by the parallel algorithm T_p to solve the same problem on p processors. The speedup is ideal when the speedup is proportional to the number of processors used to solve a problem in parallel (i.e., $S = p$).

Efficiency: Efficiency measures the fraction of the time that a processor is usefully employed. Efficiency E is defined as S/p, which is the ratio of the speedup S to the number of processors p. An efficiency of one corresponds to the ideal speedup and implies good scalability.

Cost: Cost measures the sum of the time that each processor spends solving the problem. The cost C of solving a problem on a parallel system is defined as $T_p \cdot p$, which is the product of the parallel run time T_p and the number of processors p used. A parallel computing system is *cost optimal* if the cost of solving a problem on a parallel computer is proportional to the execution time of the best known sequential algorithm on a single processor [21].

Scalability: Scalability of a parallel system measures the performance gain achieved by parallelizing as the problem size and the number of processors varies. Formally, scalability of a parallel system is a measure of the system's capacity to increase speedup in proportion to the number of processors. A scalable parallel system maintains a fixed efficiency as the number of processors and the problem size increases [21].

Computational Density: The computational density (CD) metric measures the computational performance of a device (parallel system). The CD for double precision FP (DPFP) operations can be given as [22]:

$$\text{CD}_{\text{DPFP}} = f \times \sum_i \frac{N_i}{\text{CPI}_i} \tag{2}$$

where f denotes the operating frequency of the device, N_i denotes the number of instructions of type i requiring FP computations that can be issued simultaneously, and CPI_i denotes the average number of cycles per instruction of type i.

Computational Density per Watt: The computational density per watt (CD/W) metric takes into account the power consumption of a device while quantifying performance. We propose a system-level power model to estimate the power consumption of multi-core architectures that can be used in estimating the CD/W. Our power model considers both the active and idle modes' power consumptions. Given a multi-core architecture with a total of N processor cores, the power consumption of the system with p active processor cores can be given as:

$$P^p = p \cdot \frac{P^{active}_{max}}{N} + (N-p) \cdot \frac{P^{idle}_{max}}{N} \tag{3}$$

where P^{active}_{max} and P^{idle}_{max} denote the maximum active and idle modes' power consumptions, respectively. P^{active}_{max}/N and P^{idle}_{max}/N give the active and idle modes' power, respectively, per processor core and associated switching and interconnection network circuitry. Our power model incorporates the power saving features of state-of-art multi-core architectures. Contemporary multi-core architectures provide instructions to switch the processor cores and associated circuitry (switches, clock, interconnection network) not used in a computation to a low-power idle state. For example, a software-usable NAP instruction can be executed on a tile in the Tilera's TMAs to put the tile into a low-power IDLE mode [10][23]. Similarly, Xeon 5400 processors provide an extended HALT state and Opteron processors provide a HALT mode, which are entered by executing the HLT instruction, to reduce power consumption by stopping the clock to internal sections of the processor. Other low-power processor modes are also available [9]. Investigation of a comprehensive power model for TMAs is the focus of our future work.

V. Results

The cross-architectural evaluation of SMPs and TMAs in terms of performance and performance per watt requires parallelization of benchmarks in two parallel languages: OpenMP for SMP and Tilera's MDE ilib API for TILEPro64 TMA. Performance per watt calculations leverage our power model (3) and we obtain the power consumption values for the SMPs and TMAs from the devices' respective datasheets. For example, the TILEPro64 has a maximum active and idle mode power consumption of 28 W and 5 W, respectively [22][24]. Intel's Xeon E5430 has a maximum power consumption of 80 W and a minimum power consumption of 16 W in an extended HALT state [8][9].

In this section, we present device metrics for the $\text{SMP}^{\text{2xQuadXeon}}$ and the TILEPro64 and compare these architectures' metrics for our parallelized benchmarks. All results are obtained with the compiler optimization flag -O3 since our experiments showed that this optimization flag resulted in shorter execution times as compared to lower compiler optimization levels, such as -O2.

A. Benchmark-Driven Results for SMPs

Table I depicts the increase in performance (throughput) in MOPS and performance per watt in MOPS/W for multi-core SMP processors as compared to a single-core processor for the information fusion application for $\text{SMP}^{\text{2xQuadXeon}}$ when the number of fused samples N = 3,000,000, and the moving average filter's window M = 40. For example, an eight-core processor increases the information fusion application's throughput by 4.85x as compared to a single-core processor. The performance per watt results reveal that

TABLE I: Performance results for the information fusion application for SMP$^{\text{2xQuadXeon}}$ when M = 40.

Problem Size N	# of Cores p	Execution Time (s) T_p	Speedup $S = T_s/T_p$	Efficiency $E = S/p$	Cost $C = T_p \cdot p$	Perf. (MOPS)	Perf. per watt (MOPS/W)
3,000,000	1	12.02	1	1	12.02	1073.2	22.36
3,000,000	2	7.87	1.53	0.76	15.74	1639.14	25.61
3,000,000	4	4.03	2.98	0.74	16.12	3201	33.34
3,000,000	6	2.89	4.2	0.7	17.34	4463.67	34.87
3,000,000	8	2.48	4.85	0.61	19.84	5201.6	32.51

TABLE II: Performance results for the Gaussian elimination benchmark for SMP$^{\text{2xQuadXeon}}$.

Problem Size (m, n)	# of Cores p	Execution Time (s) T_p	Speedup $S = T_s/T_p$	Efficiency $E = S/p$	Cost $C = T_p \cdot p$	Perf. (MFLOPS)	Perf. per watt (MFLOPS/W)
(2000, 2000)	1	8.05	1	1	8.05	663.35	13.82
(2000, 2000)	2	3.76	2.14	1.07	7.52	1420.21	22.2
(2000, 2000)	4	2.08	3.87	0.97	8.32	2567.31	26.74
(2000, 2000)	6	1.42	5.67	0.94	8.52	3760.56	29.38
(2000, 2000)	8	1.08	7.45	0.93	8.64	4944.44	30.9

multiple cores execute the information fusion application more power efficiently as compared to a single-core processor. For example, a four-core processor attains a 49% better performance per watt than a single-core processor.

Table II depicts the performance and performance per watt results in MFLOPS and MFLOPS/W, respectively, for the GE benchmark for SMP$^{\text{2xQuadXeon}}$ when (m, n) = $(2000, 2000)$ where m is the number of linear equations and n is the number of variables in the linear equation. Results show that the multi-core processor speedups, as compared to the single-core processor, are proportional to the number of cores. For example, an eight-core processor increases the performance and performance per watt by 7.45x and 2.2x, respectively, as compared to a single-core processor.

Finally, the performance results for the EP benchmark for SMP$^{\text{2xQuadXeon}}$ when the number of random variates generated n is equal to 100,000,000 (the result details are omitted for brevity) reveals that the SMP architecture delivers higher MFLOPS/W as the number of cores increases and the attained speedups are close to the ideal speedup.

These results verify that embedded systems using an SMP-based multi-core processor are more performance- and power-efficient as compared to embedded systems using a single-core processor.

B. Benchmark-Driven Results for TMAs

Table III depicts the performance results for TMA-based multi-core processors (TILEPro64) as compared to a single-core processor for the information fusion application when N = 3,000,000 and M = 40. Results indicate that the TMA-based multi-core processor achieves ideal speedups, an efficiency of close to one, and nearly constant cost as the number of tiles increases indicating ideal scalability. For example, a TMA-based multi-core processor with 50 tiles increases performance and performance per watt by 48.4x and 11.3x, respectively, as compared to a single TMA tile.

Table IV depicts the performance results for the GE benchmark for the TILEPro64 when (m, n) = $(2000, 2000)$. Results show that the TMA-based multi-core processor achieves less than ideal speedups and that the efficiency decreases and the cost increases as p increases indicating poor scalability for the GE benchmark. The main reasons for poor scalability are excessive memory operations, dependency between the computations, and the core synchronization operations required by the GE benchmark. However, the TMA-based multi-core processor still attains better performance and performance per watt than a single-core processor. For example, a TMA-based multi-core processor with 56 tiles increases performance and performance per watt by 14x and 3x, respectively, as compared to a single TMA tile.

Finally, the performance results for the EP benchmark running on the TILEPro64 when n = 100,000,000 indicate that the TMA-based multi-core processor delivers higher performance and performance per watt as the number of tiles increases. For example, a TMA-based multi-core processor with eight tiles increases performance and performance per watt by 7.9x and 5.4x, respectively, and with 56 tiles increases performance and performance per watt by 42.6x and 9.1x, respectively, as compared to a single TMA tile.

Comparing the performance and performance per watt results for the information fusion application and EP benchmark reveals that TMAs deliver higher performance and performance per watt for benchmarks with integer operations as compared to the benchmarks with FP operations. For example, the increase in performance and performance per watt for integer operations as compared to FP operations is 13x on average for a TMA with eight tiles. This better performance and performance per watt for integer operations is because Tilera's TMAs do not contain dedicated FP units.

TABLE III: Performance results for the information fusion application for the TILEPro64 when M = 40.

Problem Size N	**# of Tiles** p	**Execution Time (s)** T_p	**Speedup** $S = T_s/T_p$	**Efficiency** $E = S/p$	**Cost** $C = T_p \cdot p$	**Perf. (MOPS)**	**Perf. per watt (MOPS/W)**
3,000,000	1	70.65	1	1	70.65	182.6	34.07
3,000,000	2	35.05	2	1	70.1	368	64.33
3,000,000	4	17.18	4.1	1.02	68.72	750.87	116.6
3,000,000	6	11.48	6.2	1.03	68.9	1123.69	156.94
3,000,000	8	8.9	7.94	0.99	71.2	1449.44	183.94
3,000,000	10	6.79	10.4	1.04	67.9	1899.85	221.17
3,000,000	50	1.46	48.4	0.97	73	8835.62	384.66

TABLE IV: Performance results for the Gaussian elimination benchmark for TILEPro64.

Problem Size (m, n)	**# of Tiles** p	**Execution Time (s)** T_p	**Speedup** $S = T_s/T_p$	**Efficiency** $E = S/p$	**Cost** $C = T_p \cdot p$	**Perf. (MFLOPS)**	**Perf. per watt (MFLOPS/W)**
(2000, 2000)	1	416.71	1	1	416.71	12.81	2.39
(2000, 2000)	2	372.35	1.12	0.56	744.7	14.34	2.51
(2000, 2000)	4	234.11	1.8	0.45	936.44	22.81	3.54
(2000, 2000)	6	181.23	2.3	0.38	1087.38	29.46	4.11
(2000, 2000)	8	145.51	2.86	0.36	1164.08	36.7	4.66
(2000, 2000)	16	84.45	4.9	0.31	1351.2	63.23	5.88
(2000, 2000)	28	52.25	7.98	0.28	1463	102.2	6.79
(2000, 2000)	44	36.26	11.49	0.26	1595.44	147.27	7.08
(2000, 2000)	56	29.72	14	0.25	1664.32	179.68	7.15

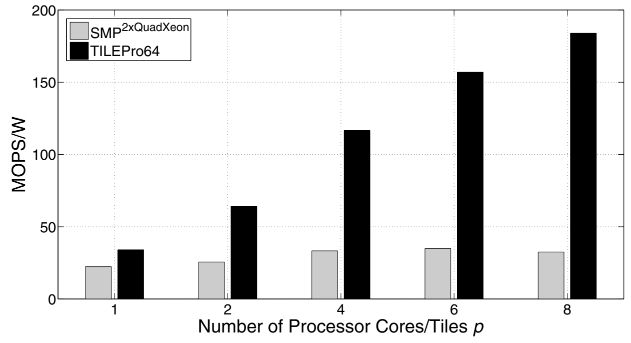

Fig. 2: Performance per watt (MOPS/W) comparison between SMP^{2xQuadXeon} and the TILEPro64 for the information fusion application when N = 3000,000.

These results verify that an embedded system using TMAs as processing units is more performance- and power-efficient as compared to an embedded system using a single tile.

C. Comparison of SMPs and TMAs

To provide cross-architectural evaluation insights for SMPs and TMAs, we compare the performance per watt results based on our parallelized benchmarks for these architectures. These results indicate which of the two architectures is more suitable for particular type of embedded applications.

Fig. 2 compares the performance per watt for the SMP^{2xQuadXeon} and the TILEPro64 for a varying number of cores/tiles for the information fusion application and reveals that the TILEPro64 delivers higher performance per watt as compared to the SMP^{2xQuadXeon}. For example, when the number of cores/tiles is eight, the TILEPro64's performance per watt is 465.8% better than the SMP's. The reason for this better performance per watt for the TILEPro64 is that the information fusion application operates on the private data obtained from various sources, which is easily parallelized on the TILEPro64 using ilib API. This parallelization exploits data locality for enhanced performance for computing moving averages, minimum, maximum, and average of the sensed data. The exploitation of data locality enables fast access to private data, which leads to higher internal memory bandwidth (on-chip bandwidth between tiles and caches) and consequently higher MFLOPS and MFLOPS/W.

The SMP^{2xQuadXeon} attains comparatively lower performance than the TILEPro64 for the information fusion application due to two reasons: the SMP architecture is more suited for shared memory applications and the information fusion application is well suited for architectures that can better exploit data locality; and OpenMP-based parallel programming uses sections and parallel constructs, which requires sensed data to be shared by operating threads even if the data requires independent processing by each thread. While parallelizing the information fusion application for the SMP^{2xQuadXeon}, we first tried using an independent copy of the sensed data (as with the TILEPro64) for each thread to maximize performance. This parallelization resulted in segmentation faults due to the extremely large memory requirements and required us to use shared memory for the sensed data since the current version of OpenMP provides no way of specifying private data for particular threads (although data can be declared private for all of the threads participating in a parallel computation). Therefore, the SMP's comparatively lower performance is partially due to the limitation of OpenMP, which does not allow the declaration of thread-specific private data (i.e., received data from the first source is private to the first thread only whereas other threads have no information of this data, received data from the second source is private to the second thread only, and so on).

Fig. 3 shows that the SMP^{2xQuadXeon} achieves higher MFLOPS/W than the TILEPro64 for the GE benchmark. For example, the SMP^{2xQuadXeon} achieves a 563% better performance per watt than the TILEPro64 when the number of cores/tiles is eight. The results also indicate that the SMP^{2xQuadXeon} exhibits better scalability and cost-efficiency than the TILEPro64. For example, the SMP^{2xQuadXeon}'s cost-efficiency is 0.93 and the TILEPro64's cost-efficiency is 0.36 when the number of cores/tiles is eight. The GE benchmark requires excessive memory operations and communication and synchronization between processing cores, which favors the SMP-based shared memory architecture since the communication transforms to read and write operations in shared memory, and hence better performance per watt for the SMP^{2xQuadXeon} as compared to the TILEPro64. In TMAs, communication operations burden the on-chip interconnection network, especially when communicating large amounts of data. Furthermore, the higher memory bandwidth (both on-chip and external memory) for the SMP^{2xQuadXeon} as compared to the TILEPro64 leads to higher memory-sustainable CD and thus enhanced performance for the GE benchmark, which requires frequent memory accesses.

For the EP benchmark, the SMP^{2xQuadXeon} delivers higher MFLOPS/W than the TILEPro64 because the EP benchmark's execution time on the SMP^{2xQuadXeon} is significantly less than the execution time on the TILEPro64 (detailed results are omitted for brevity). For example, the SMP^{2xQuadXeon} achieves 4.4x better performance per watt than the TILEPro64 when the number of cores/tiles is equal to eight. The comparatively larger execution time on the TILEPro64 is due to the complex FP operations (e.g., square root, logarithm) in the EP benchmark, which require many cycles to execute on the integer execution units in the TILEPro64.

We also compare the overall execution time for the benchmarks (detailed results are omitted for brevity) for SMPs and TMAs to provide insights into the computing capability of the processor cores in the two architectures regardless of the power consumption. Results show that the execution time of the benchmarks on a single core of the SMP is significantly less than the execution time of the benchmarks on a single tile of the TMA. For example, for the information fusion application, the execution time on a single core of the SMP^{2xQuadXeon} is 6x less than the execution time on a single tile of the TILEPro64. This execution time difference is primarily due to the lower computing power and operating frequency of each tile and the TILEPro64's lack of FP execution units. Each tile on the TMA has a maximum clock frequency of 866 MHz as compared to the SMP^{2xQuadXeon}'s maximum clock frequency of 2.66 GHz. The better performance of a single core of the SMP^{2xQuadXeon} as compared to a single tile of the TILEPro64 confirms the corollary of Amdahl's law that emphasizes the performance advantage of a powerful single core over multiple less powerful cores [25]. We point out that this execution time difference may be exacerbated for memory-intensive benchmarks because of the larger L2 cache on the SMP^{2xQuadXeon} (12 MB) as compared to the TILEPro64 (5 MB on-chip cache with Tilera's DDC technology).

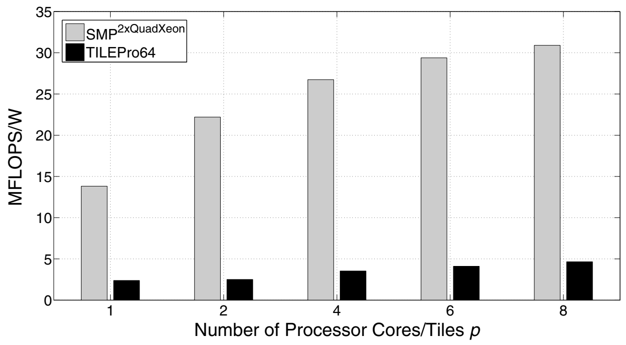

Fig. 3: Performance per watt (MFLOPS/W) comparison between SMP^{2xQuadXeon} and the TILEPro64 for the Gaussian elimination benchmark when $(m, n) = (2000, 2000)$.

VI. Conclusions

In this paper, we compared the performance of symmetric multiprocessors (SMPs) and tiled multi-core architectures (TMAs) (focusing on the TILEPro64) based on a parallelized information fusion application, a Gaussian elimination (GE) kernel, and an embarrassingly parallel (EP) benchmark. Our results revealed that the SMPs outperform the TMAs in terms of overall execution time, however, TMAs can deliver comparable or better performance per watt. Specifically, results indicated that the TILEPro64 exhibited better scalability and attained better performance per watt than the SMPs for applications involving integer operations and for the applications that operate primarily on private data with little communication between operating cores by exploiting the data locality, such as in the information fusion application. The SMPs depicted better scalability and performance for benchmarks requiring excessive communication and synchronization operations between operating cores, such as in the GE benchmark. Results from the EP benchmark revealed that the SMPs provided higher peak floating point performance per watt than the TMAs primarily because the studied TMAs did not have a dedicated floating point unit.

Our future work includes further evaluation of SMPs and TMAs for other benchmarks, such as a block matching kernel for image processing, video encoding and decoding, convolution, and fast Fourier transform (FFT) because these benchmarks would provide insights into the architectures' suitability for other domains, such as signal processing.

We also plan to develop a robust energy model for the SMPs and TMAs as well as expand our evaluation to include field-programmable gate array (FPGA)-based multi-core architectures.

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

REFERENCES

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

[2] TILERA, "Tilera Multicore Development Environment: iLib API Reference Manual," in *Tilera Official Documentation*, April 2009.

[3] R. Brown and I. Sharapov, "Performance and Programmability Comparison Between OpenMP and MPI Implementations of a Molecular Modeling Application," *Springer-Verlag Lecture Notes in Computer Science*, vol. 4315, pp. 349–360, November 2008.

[4] G. Bikshandi, J. Guo, D. Hoeflinger, G. Almasiy, B. Fraguelaz, M. Garzaran, D. Padua, and C. von Prauny, "Programming for Parallelism and Locality with Hierarchically Tiled Arrays," in *Proc. of the ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP)*, Manhattan, New York City, New York, March 2006.

[5] W. Zhu, J. Cuvillo, and G. Gao, "Performance Characteristics of OpenMP Language Constructs on a Many-core-on-a-chip Architecture," in *Proc. of the 2005 and 2006 International Conference on OpenMP Shared Memory Parallel Programming*, ser. IWOMP'05/IWOMP'06. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 230–241. [Online]. Available: http://portal.acm.org/citation.cfm?id=1892830.1892855

[6] 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.

[7] T. T.-O. Kwok and Y.-K. Kwok, "Computation and Energy Efficient Image Processing in Wireless Sensor Networks Based on Reconfigurable Computing," in *Proc. of the International Conference on Parallel Processing Workshops (ICPPW)*, Columbus, Ohio, August 2006.

[8] Intel, "Intel Xeon Processor E5430," February 2012. [Online]. Available: http://ark.intel.com/Product.aspx?id=33081

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

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

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

[12] TILERA, "Manycore without Boundaries: TILE64 Processor," February 2012. [Online]. Available: http://www.tilera.com/products/processors/TILE64

[13] ——, "Manycore without Boundaries: TILEPro64 Processor," February 2012. [Online]. Available: http://www.tilera.com/products/processors/TILEPRO64

[14] IBM, "Linux and Symmetric Multiprocessing," February 2012. [Online]. Available: http://www.ibm.com/developerworks/library/l-linux-smp/

[15] Android, "SensorEvent," February 2012. [Online]. Available: http://developer.android.com/reference/android/hardware/SensorEvent.html

[16] Top500, "Top 500 Supercomputer Sites," February 2012. [Online]. Available: http://www.top500.org/

[17] LINPACK, "LINPACK Benchamarks," February 2012. [Online]. Available: http://en.wikipedia.org/wiki/LINPACK

[18] 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.

[19] NPB, "NASA Advanced Supercomputing (NAS) Parallel Benchmarks," February 2012. [Online]. Available: http://www.nas.nasa.gov/Resources/Software/npb.html

[20] JavaDoc, "Class Flops: Counting floating point operations," February 2012. [Online]. Available: http://ai.stanford.edu/~paskin/slam/javadoc/javaslam/util/Flops.html

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

[22] J. Williams, C. Massie, A. George, J. Richardson, K. Gosrani, and H. Lam, "Characterization of Fixed and Reconfigurable Multi-Core Devices for Application Acceleration," *ACM Trans. on Reconfigurable Technology and Systems*, vol. 3, no. 4, November 2010.

[23] TILERA, "Tile Processor Architecture Overview," in *Tilera Official Documentation*, November 2009.

[24] ——, "TILEmPower Appliance User's Guide," in *Tilera Official Documentation*, January 2010.

[25] K. Asanovic, R. Bodik, J. Demmel, T. Keaveny, K. Keutzer, J. Kubiatowicz, N. Morgan, K. Patterson, David Sen, J. Wawrzynek, D. Wessel, and K. Yelick, "A View of the Parallel Computing Landscape," *Communications of the ACM*, vol. 52, no. 10, October 2009.