A Queueing Theoretic Approach for Performance Evaluation of Low-Power Multi-core Embedded Systems

Arslan Munir*, 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: amunir@ufl.edu, 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, the optimal layout of these multiple cores along with the memory subsystem (caches and main memory) to satisfy power, area, and often stringent real-time constraints is a challenging design endeavor. The short *time-to-market* constraint of embedded systems exacerbates this design challenge and necessitates the architectural modeling of embedded systems to reduce the time-to-market by expediting target applications to device/architecture mapping. In this paper, we present a queueing theoretic approach for modeling multi-core embedded systems that provides a quick and inexpensive performance evaluation both in terms of time and resources as compared to the development of multi-core simulators and running benchmarks on these simulators. We also calculate chip area and power consumption for different multi-core embedded architectures with a varying number of processor cores and cache configurations to provide a comparative analysis of multi-core embedded architectures in terms of performance, area, and power consumption. Our performance and power results indicate that multi-core embedded system architectures that leverage shared last-level caches (LLCs) provide the best LLC performance per watt but may introduce main memory response time and throughput bottlenecks for high cache miss rates, whereas architectures leveraging a hybrid of private and shared LLCs alleviate main memory bottlenecks at the expense of reduced performance per watt.**

***Index Terms*—Multi-core; low-power; embedded systems; queueing theory; performance evaluation;**

I. Introduction and Motivation

With Moore's law supplying billions of transistors on-chip, embedded systems are undergoing a paradigm shift from single-core to multi-core to exploit this high transistor density for high performance. This paradigm shift has led to the emergence of diverse multi-core embedded systems in a plethora of application domains (e.g., high-performance computing, dependable computing, mobile computing, etc.). Many modern embedded systems integrate multiple cores (whether homogeneous or heterogeneous) on-chip to satisfy computing demand while maintaining design constraints (e.g., energy, power, performance, etc.). For example, a 3G mobile handset's signal processing requires 35-40 Giga operations per second (GOPS). Considering the limited energy of a mobile handset battery, these performance levels must be met with a power dissipation budget of approximately 1W, which translates to a performance efficiency of 25 mW/GOP or 25 pJ/operation for the 3G receiver [1]. These demanding and competing power-performance requirements make modern embedded system design challenging.

Increasing customer expectations/demands for embedded system functionality has led to an exponential increase in design complexity. While industry focuses on increasing the number of on-chip processor cores to meet customer performance demands, embedded system designers face the new challenge of optimal layout of these processor cores along with the memory subsystem (caches and main memory) to satisfy power, area, and stringent real-time constraints. The short *time-to-market* (time from product conception to market release) of embedded systems further exacerbates design challenges. Architectural modeling of embedded systems helps in reducing the time-to-market by enabling fast application-to-device mapping since identifying an appropriate architecture for a set of target applications significantly reduces the design time of an embedded system. To ensure timely completion of an embedded system's design with sufficient confidence in the product's market release, design engineers have to make tradeoffs between the abstraction level and the accuracy a multi-core architecture model can attain.

Modern multi-core embedded systems allow processor cores to share hardware structures such as last-level caches (LLCs) (e.g., level two (L2) or level three (L3) caches), memory controllers, and interconnection networks [2]. Since the LLC's configuration (e.g., size, line size, associativity) and the layout of the processor cores (on-chip location) has a significant impact on a multi-core embedded system's performance and energy, our work focuses on performance and energy characterization of embedded architectures based on different LLC configurations and the layout of the processor cores. Though there is a general consensus on using private level one (L1) instruction (L1-I) and data (L1-D) caches in embedded systems, there has been no dominant architectural paradigm for private or shared LLCs. Since many embedded systems contain an L2 cache as the LLC, we focus on the L2 cache, however, our study can easily be extended to lower-level caches such as L3 caches and beyond.

Since multi-core benchmark simulation requires significant simulation time and resources, a lightweight modeling technique for multi-core architecture evaluation is crucial

[3]. Previous work presents various multi-core system models, however, these models become increasingly complex with varying degrees of cache sharing [4]. Many of the previous models assumed that sharing amongst processor cores occurred at either the main memory level or the processor cores all shared the same cache hierarchy, however, multi-core embedded systems can have an L2 cache shared by a subset of cores (e.g., Intel's six-core Dunnington processor has L2 caches shared by two processor cores). We leverage for the first time, to the best of our knowledge, queueing network theory as an alternative approach for modeling multi-core embedded systems for performance analysis (though queueing network models have been studied in the context of traditional computer systems [5]). Our queueing network model approach allows modeling the layout of the processor cores (homogeneous or heterogeneous) with caches of different capacities and configurations at different cache levels. Our modeling technique only requires a high-level workload characterization of an application (i.e., whether the application is processor-bound (requiring high processing resources), memory-bound (requiring a large number of memory accesses), or mixed).

Our main contributions in this paper are:

- We present a novel, queueing theory-based modeling technique for evaluating multi-core embedded architectures that does not require architectural-level benchmark simulation. This modeling technique enables quick and inexpensive architectural evaluation both in terms of design time and resources as compared to developing and/or using existing multi-core simulators and running benchmarks on these simulators. Based on a preliminary evaluation using our model, architecture designers can run targeted benchmarks to further verify the performance characteristics of selected multi-core architectures (i.e., our queueing theory-based model facilitates early design space pruning).
- Our queueing theoretic approach quantifies performance metrics (e.g., response time, throughput) for different workload/benchmark characteristics and different cache miss rates. Although general trends for performance metrics could be anticipated for different cache miss rates and workload characteristics, our work for the first time quantifies the percentage increase and decrease in the performance metrics for different cache miss rates and workload/benchmark characteristics for different architectures.
- We calculate chip area and power consumption for different multi-core embedded architectures with a varying number of processor cores and cache configurations to provide a comparative analysis of multi-core embedded architectures in terms of performance (e.g., response time), area, and power consumption.

We point out that although queueing theory has been used in literature for performance analysis of multi-disk systems [5][6], we for the first time to the best of our knowledge apply queueing theory-based modeling and performance analysis techniques to multi-core embedded systems. Furthermore, we develop a methodology to simulate workloads/benchmarks on our queueing theoretic multi-core model based on probabilities that are assigned according to workload characteristics (e.g., processor-bound, memory-bound, or mixed) and cache miss rates.

Our investigation of performance and energy for different cache miss rates and workloads is important because cache miss rates and workloads can significantly impact the performance and energy of an embedded architecture. Furthermore, cache miss rates also give an indication of the degree of cache contention between different threads' working sets. Our performance, power, and performance per watt results (in terms of floating point operations per second (FLOPS) per watt (FLOPS/W)) indicate that multi-core embedded system architectures that leverage shared LLCs are scalable and provide the best LLC MFLOPS/W. For example, shared LLCs can provide up to 13.8 MFLOPS/W for a four-core architecture. However, shared LLC architectures may introduce main memory response time and throughput bottlenecks for high cache miss rates. Architectures that leverage a hybrid of private and shared LLCs are scalable and alleviate main memory bottlenecks at the expense of reduced MFLOPS/W. For example, hybrid architectures can provide up to 9.2 MFLOPS/W for a four-core architecture. Finally, architectures with private LLCs exhibit less scalability but do not introduce main memory bottlenecks at the expense of reduced MFLOPS/W. For example, private LLCs can provide a throughput of up to 7.6 MFLOPS/W for a four-core architecture.

II. Related Work

Previous work presents evaluation and modeling techniques for multi-core embedded architectures for different applications and varying workload characteristics. Savage et al. [4] proposed a unified memory hierarchy model for multi-core architectures that captured varying degrees of cache sharing at different cache levels. The model, however, only worked for straight-line computations that could be represented by directed acyclic graphs (DAGs) (e.g., matrix multiplication, fast Fourier transform (FFT)). Fedorova et al. [2] studied contention-aware task scheduling for multi-core architectures with shared resources (caches, memory controllers, and interconnection networks). They modeled the contention-aware task scheduler and investigated the scheduler's impact on application performance for multi-core architectures.

Some previous work investigated performance and energy aspects for multi-core systems. Kumar et al. [7] studied power, throughput, and response time metrics for heterogeneous CMPs. They observed that heterogeneous CMPs could improve energy per instruction by 4-6x and throughput by 63% over an equivalent area homogeneous CMP because of closer adaptation to the resource requirements of different application phases. Sabry et al. [8] investigated performance, energy, and area tradeoffs for private and shared L2 caches for multi-core embedded systems. They proposed a SystemC-based platform that could model private, shared, and hybrid L2 cache architectures.

III. Queueing Network Modeling of Multi-core Embedded Architectures

In this section, we define queueing network terminologies and our modeling approach in the context of multi-core embedded architectures. We use the term *jobs* often instead of *tasks* (decomposed workloads resulting from parallelizing a job) to be consistent with queueing network terminology. Our modeling approach is broadly applicable to *multi-programmed workloads* where multiple jobs run on a multi-core embedded architecture as well as for *parallelized applications/jobs* that run different *tasks* on a multi-core embedded architecture.

A *queueing network* consists of *service centers* (e.g., processor core, L1-I cache, L1-D cache, L2 cache, and main memory (MM)) and *customers* (e.g., jobs/tasks). A service center consists of one or more queues to hold jobs waiting for service. Arriving jobs enter the service center's queue and a *scheduling/queueing discipline* (e.g., first-come-first-served (FCFS), priority, round robin (RR), processor sharing (PS), etc.) selects the next job to be served when a service center becomes idle. After being serviced, a job either moves to another service center or leaves the network.

A queueing network is *open* if jobs arrive from an external source, spend time in the network, and then depart. A queueing network is *closed* if there is no external source and no departures (i.e., a fixed number of jobs circulate indefinitely among the service centers). A queueing network is a *single-chain* queueing network if all jobs possess the same characteristics (e.g., arrival rates, required service rates, and routing probabilities for various service centers) and are serviced by the same service centers in the same order. If different jobs can belong to different chains, the network is a *multi-chain* queueing network. An important class of queueing networks is *product-form* where the joint probability of the queue sizes in the network is a product of the probabilities for the individual service centers' queue sizes.

The queueing network performance metrics include response time, throughput, and utilization. The *response time* is the amount of time a job spends at the service center including the queueing delay (the amount of time a job waits in the queue) and the service time. The service time of a job depends on the amount of work (e.g., number of instructions) needed by that job. The *throughput* is defined as the number of jobs served per unit of time. In our multi-core embedded architecture context, throughput measures the number of instructions/data (bits) processed by the architectural element (processor, cache, MM) per second. *Utilization* measures the fraction of time that a service center (processor, cache, MM) is busy. Little's law governs the relationship between the number of jobs in the queueing network N and response time tr (i.e., $N = \lambda \cdot tr$ where λ denotes the average arrival rate of jobs admitted to the queueing network [9]).

We consider the closed product-form queueing network for modeling multi-core embedded architectures because a typical embedded system executes a fixed number of jobs (e.g., a mobile phone has only a few applications to run such as instant messaging, audio coding/decoding, calculator, graphics interface, etc.). Furthermore, closed product-form queueing networks assume that a job leaving the network is replaced instantaneously by a statistically identical new job [5]. Table I describes the multi-core embedded architectures that we evaluate in this paper. We focus on embedded architectures ranging from two (2P) to four (4P) processor cores to reflect current architectures [10], however, our model is applicable to any number of cores. Our modeled embedded architectures contain processor cores, L1-I and L1-D private caches, L2 caches (private or shared), and MM (embedded systems are typically equipped with DRAM/NAND/NOR Flash memory [11][12]).

Consider a closed product-form queueing network with I service centers where each service center $i \in I$ has a service rate μ_i. Let p_{ij} be the probability of a job leaving service center i and entering another service center j. The relative visit count ϑ_j to service center j is:

$$\vartheta_j = \sum_{i=1}^{I} \vartheta_i p_{ij} \tag{1}$$

The performance metrics for a closed product-form queueing network can be calculated using a *mean value analysis* (MVA) iterative algorithm [13]. The basis of MVA is a theorem stating that when a job arrives at a service center in a closed network with N jobs, the distribution of the number of jobs already queued is the same as the steady state distribution of N - 1 jobs in the queue [14]. Solving (1) using MVA recursively gives the following performance metric values: the mean response time $\overline{r}_i(k)$ at service center i, the mean queueing network throughput $\overline{\mathcal{T}}(k)$, the mean throughput of jobs $\overline{t}_i(k)$ at service center i, and the mean queue length $\overline{l}_i(k)$ at service center i when there are k jobs in the network. The initial recursive conditions are $i = 0$ such that $\overline{r}_i(0)$ = $\overline{\mathcal{T}}(0)$ = $\overline{t}_i(0)$ = $\overline{l}_i(0)$ = 0. The values for these performance metrics can be calculated for k jobs based on the computed values for $k - 1$ jobs as [5]:

$$\overline{r}_i(k) = \frac{1}{\mu_i}(1 + \overline{l}_i(k-1)) \tag{2}$$

$$\overline{\mathcal{T}}(k) = \frac{k}{\overline{R}} = \frac{k}{\sum_{i=1}^{I} \vartheta_i \overline{r}_i(k)} \tag{3}$$

$$\overline{t}_i(k) = \vartheta_i \overline{(T)}(k) \tag{4}$$

$$\overline{l}_i(k) = \overline{t_i}(k)\overline{r_i}(k) \tag{5}$$

To explain our modeling approach for multi-core embedded architectures, we describe a sample queueing model for the 2P-2L1ID-2L2-1M architecture in detail (other architecture models follow a similar explanation). Fig. 1 depicts the

TABLE I

Multi-core embedded architectures with varying processor cores and cache configurations (P denotes a processor core, L1ID denotes L1-I and L1-D caches, M denotes the main memory, and the integer constants preceding P, LIID, L2, and M denote the number of these architectural components in the embedded architecture).

Architecture	Description
2P-2L1ID-2L2-1M	Multi-core embedded architecture with 2 processor cores, private L1 I/D caches, private L2 caches, and a shared M
2P-2L1ID-1L2-1M	Multi-core embedded architecture with 2 processor cores, private L1 I/D caches, a shared L2 cache, and a shared M
4P-4L1ID-4L2-1M	Multi-core embedded architecture with 4 processor cores, private L1 I/D caches, private L2 caches, and a shared M
4P-4L1ID-1L2-1M	Multi-core embedded architecture with 4 processor cores, private L1 I/D caches, a shared L2 cache, and a shared M
4P-4L1ID-2L2-1M	Multi-core embedded architecture with 4 processor cores, private L1 I/D caches, 2 shared L2 caches, and a shared M

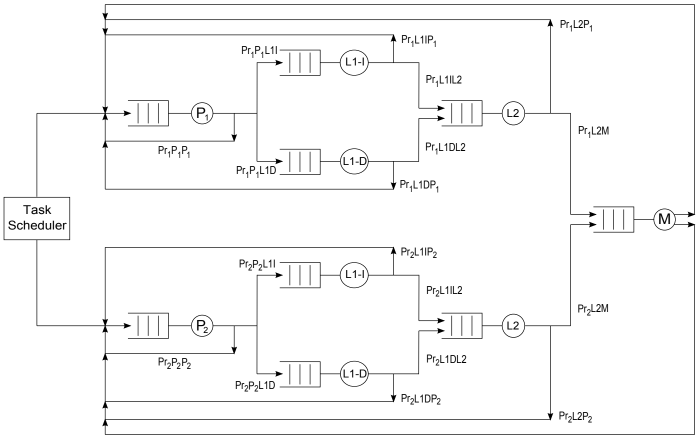

Fig. 1. Queueing network model for the 2P-2L1ID-2L2-1M multi-core embedded architecture.

queueing network model for 2P-2L1ID-2L2-1M. The *task scheduler* schedules the tasks/jobs on the two processor cores P_1 and P_2. We assume that the task scheduler is contention-aware and schedules tasks with minimal or no contention on cores sharing LLCs [2]. The queueing network consists of two chains: chain one corresponds to processor core P_1 and chain two corresponds to processor core P_2. The jobs serviced by P_1 either reenter P_1 with probability $Pr_1P_1P_1$ or enter the L1-I cache with probability Pr_1P_1L1I or L1-D cache with probability Pr_1P_1L1D. The job arrival probabilities into the service centers depends on the workload characteristics. The data from the L1-I and L1-D caches returns to P_1 with probabilities Pr_1L1IP_1 and Pr_1L1DP_1, respectively, after L1-I and L1-D cache hits. The requests from the L1-I and L1-D caches are directed to the L2 cache with probabilities Pr_1L1IL2 and Pr_1L1DL2, respectively, after L1-I and L1-D cache misses. The probability of requests entering P_1 or the L2 cache from the L1-I and L1-D caches depends on the miss rates of the L1-I and L1-D caches. After an L2 cache hit, the requested data is transferred to P_1 with probability Pr_1L2P_1 or enters the MM with probability Pr_1L2M after an L2 cache miss. The requests from the MM always return to P_1 with probability Pr_1MP_1 = 1. The queueing network chain and path for chain two corresponding to P_2 follows the same pattern as chain one corresponding to P_1. For example, requests from the L2 cache in chain two either return to P_2 with probability Pr_2L2P_2 after an L2 cache hit or enter the MM with probability Pr_2L2M after an L2 cache miss.

The queueing network model probabilities for the 2P-2L1ID-2L2-1M multi-core architecture for memory-bound workloads (processor to processor probability P_{pp} = 0.1, processor to memory probability P_{pm} = 0.9) assuming that L1-I, L1-D, and L2 cache miss rates are 25%, 50%, and 30%, respectively, are set as: $Pr_1P_1P_1$ = 0.1, Pr_1P_1L1I = 0.45, Pr_1P_1L1D = 0.45, Pr_1L1IP_1 = 0.75, Pr_1L1DP_1 = 0.5, Pr_1L1IL2 = 0.25, Pr_1L1DL2 = 0.5, Pr_1L2P_1 = 0.7, Pr_1L2M = 0.3, Pr_1MP_1 = 1 (different probabilities are assigned for processor-bound or mixed workloads).

Our queueing network modeling provides a faster alternative for performance evaluation of multi-core architectures as compared to running complete benchmarks on multi-core simulators (and/or trace simulators) though at the expense of accuracy. Our queueing network models only require simulating a subset of the benchmark's instructions (by specifying the job size) such that the queueing network reaches the steady state/equilibrium (the precise minimum number of instructions required depends on the workload behavior) with respect to the workload behavioral characteristics captured by the processor-to-processor and processor-to-memory probabilities (as shown in Fig. 1).

IV. Results

In this section, we present the performance evaluation and power consumption results for the five different multi-core embedded architectures depicted in Table I along with the validation of our queueing network models (for brevity, we present a subset of the results, however, our analysis and derived conclusions are based on our complete set of experimental results). We implement our queueing network models for the multi-core embedded architectures using the SHARPE modeling tool/simulator [5]. We consider the ARM7TDMI processor core, which is a 32-bit low-power processor with 32-bit instruction and data bus widths [15][16]. We consider the following cache parameters [17]: cache sizes of 8 KB, 8 KB, and 64 KB for the L1-I, L1-D, and L2 caches, respectively; associativities of direct-mapped, 2-way, and 2-way for the L1-I, L1-D, and L2 caches, respectively; and block/line sizes of 64 B, 16 B, and 64 B for the L1-I, L1-D, and L2 caches, respectively. We assume a 32 MB MM for all architectures, which is typical for mobile embedded systems (e.g., Sharp Zaurus SL-5600 personal digital assistant (PDA)) [18]. To provide a fair comparison between architectures, we ensure that the total L2 cache size for shared L2 cache architectures and private L2 cache architectures remains the

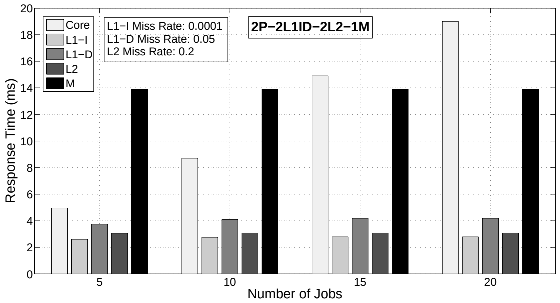

Fig. 2. Queueing network model validation demonstration — response time (ms) for mixed workloads for 2P-2L1ID-2L2-1M for a varying number of jobs N.

same.

We first calculate the service rates for the service centers used in our multi-core queueing models. We assume that the processor core delivers 15 MIPS at 16.8 MHz [15] (cycle time = 1/(16.8×10^6) = 59.524 ns), which for 32-bit instructions corresponds to a service rate of 480 Mbps. We assume L1-I, L1-D, and L2 cache, and MM access latencies of 2, 2, 10, and 100 cycles, respectively [15][19]. With an L1-I cache line size of 64 B, an access latency of 2 cycles, and a 32-bit (4 B) bus, transferring 64 B requires 64/4 = 16 cycles, which results in a total L1-I time (cycles) = access time + transfer time = 2 + 16 = 18 cycles, with a corresponding L1-I service rate = (64 × 8)/$(18 \times 59.524 \times 10^{-9})$ = 477.86 Mbps. The service rates for the L1-D and L2 caches and MM are calculated similarly. We assume that each individual job/task requires processing 1 Mb of instruction and data, which ensures that the steady state/equilibrium behavior of the queueing network for our simulated workloads is reached.

A. Queueing Network Models Validation

We validate our queueing network models for different cache miss rates and workloads and find that the model's simulation results conform with expected queueing theoretical results. For example, Fig. 2 depicts the response time for mixed workloads (P_{pp} = 0.5, P_{pm} = 0.5) for 2P-2L1ID-2L2-1M as the number of jobs/tasks N varies. The figure shows that as N increases, the response time for the processor core, L1-I, L1-D, L2, and MM increases for all of the cache miss rates. We point out that cache miss rates could increase as N increases due to inter-task address conflicts and increasing cache pressure (increased number of working sets in the cache), but we assume that the cache sizes are sufficiently large enough so that capacity misses remain the same for the considered number of jobs. We present the average response time individually for the processor cores and the L1-I, L1-D, and L2 caches. For smaller L1-I, L1-D, and L2 cache miss rates, the processor core response time increases drastically as N increases because most of the time jobs are serviced by the processor core whereas for larger L1-I, L1-D, and L2 cache miss rates, the MM response time increases drastically because of a large number of MM accesses. These results along with our other observed results conform with the expected queueing theoretical results and validate the correctness of our queueing network models for multi-core architectures. We point out that small variations in results could be due to inaccuracies in the SHARPE simulator, but do not change the overall trends.

B. The Effects of Cache Miss Rates on Performance

In this subsection, we present results describing the effects of different L1-I, L1-D, and L2 cache miss rates on the architecture response time and throughput performance metrics for mixed, processor-bound, and memory-bound workloads. Considering the effects of different cache miss rates is an important aspect of performance evaluation for multi-core embedded architectures with shared resources because cache miss rates give an indication whether the threads (corresponding to tasks) are likely to experience cache contention. Threads with higher LLC miss rates are more likely to have large working sets since each miss results in the allocation of a new cache line. These working sets may suffer from contention because threads may repeatedly evict the other threads' data (i.e., *cache thrashing*) [2]. We obtained results for cache miss rates of 0.0001, 0.05, and 0.2, up to 0.5, 0.7, and 0.7 for the L1-I, L1-D, and L2 caches, respectively. These cache miss rate ranges represent typical multi-core embedded systems for a wide diversity of workloads [20][21][22].

We observed that for mixed workloads (P_{pp} = 0.5, P_{pm} = 0.5), the response times for the processor core, L1-I, L1-D, and MM for 2P-2L1ID-1L2-1M are very close to the response times for 2P-2L1ID-2L2-1M, however, the L2 response time presents interesting differences. The L2 response time for 2P-2L1ID-1L2-1M is 22.3% less than the L2 response time for 2P-2L1ID-2L2-1M when the L1-1, L1-D, and L2 cache miss rates are 0.0001, 0.05, and 0.2, respectively, and N = 5 (similar percentage differences were observed for other values of N) whereas the L2 response time for 2P-2L1ID-1L2-1M is only 6.5% less than the L2 response time when the L1-1, L1-D, and L2 cache miss rates are 0.5, 0.7, and 0.7, respectively. This result shows that the shared L2 cache (of comparable area as the sum of the private L2 caches) performs better than the private L2 caches in terms of response time for small cache miss rates, however, the performance improvement decreases as the cache miss rate increases. Similar trends were observed for processor-bound workloads (P_{pp} = 0.9, P_{pm} = 0.1) and memory-bound workloads (P_{pp} = 0.1, P_{pm} = 0.9).

For mixed workloads, the response time for the processor core, L1-I, L1-D, and MM for 4P-4L1ID-1L2-1M is 1.2x, 1x, 1.1x, and 2.4x greater than the corresponding architectural elements – processor core, L1-I, L1-D, and MM – for 4P-4L1ID-4L2-1M whereas the L2 response time for 4P-4L1ID-1L2-1M is 1.1x less than the L2 response time for 4P-4L1ID-4L2-1M when the L1-1, L1-D, and L2 cache miss rates are 0.5, 0.7, and 0.7, respectively, and $N = 5$. This observation in conjunction with our other experimental results reveal that the architectures with private LLCs provide improved response time for processor cores and L1 caches as compared to the architectures with shared LLCs, however, the response time of the LLC alone can be slightly better for architectures with shared LLCs because of the larger effective size for each

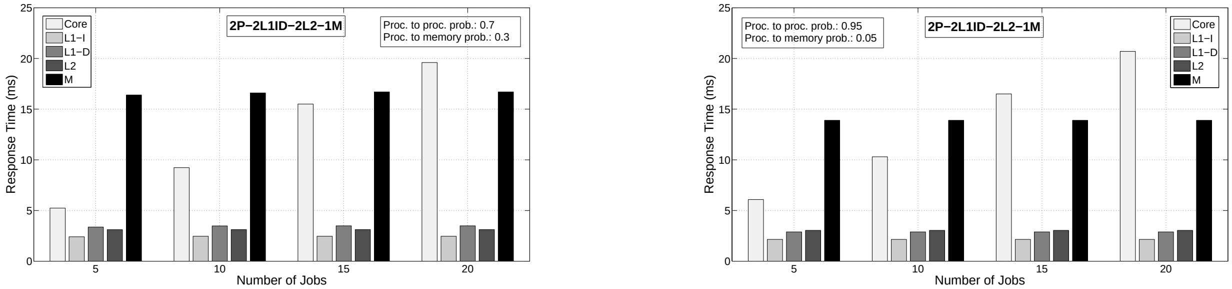

Fig. 3. The effects of processor-bound workloads on response time (ms) for 2P-2L1ID-2L2-1M for a varying number of jobs N for cache miss rates: L1-I = 0.01, L1-D = 0.13, and L2 = 0.3

core. The results also indicate that the MM response time could become a bottleneck for architectures with shared LLCs, especially when the cache miss rates become high. Another interesting observation is that shared LLCs could lead to increased response time for processor cores as compared to the private LLCs because of stalling or idle waiting of processor cores for bottlenecks caused by the MM. Similar trends were observed for processor-bound and memory-bound workloads.

For mixed workloads, the response time of the L2 for 4P-4L1ID-2L2-1M is 1.2x less than 4P-4L1ID-4L2-1M and 1.1x greater than 4P-4L1ID-1L2-1M when the L1-1, L1-D, and L2 cache miss rates are 0.0001, 0.05, and 0.2, respectively, and $N = 5$. The MM response time for 4P-4L1ID-2L2-1M is 2.3x less than 4P-4L1ID-1L2-1M whereas the MM response time for 4P-4L1ID-2L2-1M and 4P-4L1ID-4L2-1M is the same when the L1-1, L1-D, and L2 cache miss rates are 0.5, 0.7, and 0.7, respectively, and $N = 5$. The response times for the processor core and L1-I/D are comparable for the three architectures (4P-4L1ID-4L2-1M, 4P-4L1ID-2L2-1M, and 4P-4L1ID-1L2-1M). These results along with our other results show that having LLCs shared by fewer cores (e.g., the L2 cache shared by two cores in our considered architecture) do not introduce MM as a response time bottleneck whereas the MM becomes the bottleneck as more cores share the LLCs, especially for large cache miss rates.

We evaluated the effects of cache miss rates on throughput for processor-bound workloads (P_{pp} = 0.9, P_{pm} = 0.1) for 2P-2L1ID-2L2-1M as N varies. Results reveal that there is no apparent increase in processor core throughput as N increases from 5 to 20 because processors continue to operate at a utilization close to 1 when the L1-1, L1-D, and L2 cache miss rates are 0.3, 0.3, and 0.3, respectively (similar trends were observed for other cache miss rates). The MM throughput increases by 4.67% (4.67% - 1.64% = 3.03% greater than the mixed workloads) as N increases from 5 to 20 when L1-1, L1-D, and L2 cache miss rates are 0.5, 0.7, and 0.7, respectively. In this case, the MM percentage throughput increase is greater for processor-bound workloads as compared to mixed workloads because the MM is underutilized for processor-bound workloads (e.g., a utilization of 0.519 for processor-bound workloads as compared to a utilization of 0.985 for mixed workloads when N = 5). However, the MM absolute throughput for processor-bound workloads is less than the mixed workloads (e.g., MM throughput of 38.5 Mbps for processor-bound workloads as compared to a throughput of 73 Mbps for mixed workloads when N = 5). Similar trends were observed for memory-bound workloads and mixed workloads for architectures with two or four cores with private and shared LLCs (these throughput trends would continue as the number of cores increases).

C. The Effects of Workloads on Performance

In this subsection, we present results describing the effects of different workloads on the response time and throughput performance metrics when the L1-1, L1-D, and L2 cache miss rates are held constant. We discuss the effects of varying the *computing requirements* of these workloads. The computing requirement of a workload signifies the workload's demand for processor resources, which depends on the percentage of arithmetic, logic, and control instructions in the workload relative to the load and store instructions. The computing requirements of workloads are captured by P_{pp} and P_{pm} in our model.

Fig. 3 depicts the effects of varying computing requirements for processor-bound workloads on response time for 2P-2L1ID-2L2-1M as N varies where the L1-I, L1-D, and L2 cache miss rates are 0.01, 0.13, and 0.3, respectively. The figure depicts that as N increases, the response time for the processor core, L1-I, L1-D, L2, and MM increases for all values of P_{pp} and P_{mm}. The figure shows that as P_{pp} increases, the response time of the processor increases whereas the response time of L1-I, L1-D, L2, and MM is affected negligibly because of the processor-bound nature of the workloads. For example, the processor response time increases by 19.8% as P_{pp} increases from 0.7 to 0.95 when N = 5. The response time of L1-I, L1-D, L2, and MM decreases by 10.8%, 14.2%, 2.2%, and 15.2%, respectively, as P_{pp} increases from 0.7 to 0.95 when N = 5 because an increase in P_{pp} results in a decrease in memory requests, which decreases the response time for the caches and MM.

We observe that the response time for the processor core, L1-I, and L1-D for 2P-2L1ID-1L2-1M is very close (within 7%) to 2P-2L1ID-2L2-1M as the computing requirements of the processor-bound workload varies. However, 2P-2L1ID-1L2-1M provides a 21.5% improvement in L2 response time and a 12.3% improvement in MM response time over 2P-2L1ID-2L2-1M when P_{pp} = 0.7 and a 23.6% improvement in L2 response time and a 1.4% improvement in MM

TABLE II
AREA AND PEAK POWER CONSUMPTION FOR MULTI-CORE ARCHITECTURES.

Architecture	Area (mm^2)	Power (mW)
2P-2L1ID-2L2-1M	0.8528	524.896
4P-4L1ID-4L2-1M	1.7064	1049.79
2P-2L1ID-1L2-1M	0.7823	470.5
4P-4L1ID-1L2-1M	1.4874	788.472
4P-4L1ID-2L2-1M	1.5658	941.232

TABLE III
AREA AND PEAK POWER CONSUMPTION OF THE ARCHITECTURAL ELEMENTS FOR THE 2P-2L1ID-2L2-1M.

Element	Area (mm^2)	Power (mW)
Core	0.065	2.016
L1-I	0.11	135.44
L1-D	0.0998	79.76
L2	0.578	307.68
MM	34.22	3174.12

response time when P_{pp} = 0.95. Similar trends were observed for memory-bound and mixed workloads for architectures with two or four cores containing private or shared LLCs. These results indicate that the shared LLCs provide more improvement in MM response time for comparatively less compute-intensive processor-bound workloads.

We observed the effects of varying computing requirements for processor-bound workloads on throughput for 2P-2L1ID-2L2-1M as N varies. As N increases, the throughput for the processor core, L1-I, L1-D, L2, and MM increases for all values of P_{pp} and P_{mm}. Furthermore, as P_{pp} increases, the throughput of the processor core increases whereas the throughput of L1-I, L1-D, L2, and MM decreases because of relatively fewer memory requests. For memory-bound workloads, the processor core, L1-I, and L1-D throughput for 2P-2L1ID-2L2-1M and 2P-2L1ID-1L2-1M are comparable, however, 2P-2L1ID-1L2-1M improves the L2 throughput by 106.5% and 111% whereas the MM throughput decreases by 126% and 121.2% when P_{pm} is 0.7 and 0.95, respectively. Similar trends were observed for processor-bound and mixed workloads for the architectures with two and four cores containing private or shared LLCs.

D. Area and Power Consumption

In this subsection, we present area and worst-case (peak) power consumption results for different multi-core embedded architectures obtained using CACTI 6.5 [23] assuming a 45 nm process. The core areas are calculated using Moore's law and the International Technology Roadmap for Semiconductors (ITRS) specifications [24] (i.e., the chip area required for the same number of transistors reduces approximately by 1/2x every technology node (process) generation).

Table II shows the area and peak power consumption for different multi-core embedded architectures. These results do not include MM area and power consumption to isolate area and peak power consumption of the processor cores and caches. This MM isolation from the results enables deeper insights and a fair comparison for the embedded architectures since we assume an off-chip MM that has the same size and characteristics for all evaluated architectures. The results show that 2P-2L1ID-2L2-1M requires 8.3% more on-chip area and consumes 10.4% more power as compared to 2P-2L1ID-1L2-1M. 4P-4L1ID-4L2-1M requires 8.2% and 12.8% more on-chip area and consumes 10.3% and 24.9% more power as compared to 4P-4L1ID-2L2-1M and 4P-4L1ID-1L2-1M, respectively. These results reveal that the architectures with shared LLCs become more area and power efficient as compared to the architectures with private or hybrid LLCs as the number of cores in the architecture increases.

Table III shows the constituent area and peak power consumption for the processor cores, L1-I, L1-D, L2, and MM for 2P-2L1ID-2L2-1M. This area and power consumption breakdown can be given similarly for other multi-core embedded architectures and are omitted for brevity. The results show that the MM consumes the most area and power consumption followed by L2, L1-I, L1-D, and the processor core. We observe that the shared L2 caches for 2P-2L1ID-1L2-1M and 4P-4L1ID-1L2-1M require 14% and 24% less area and consume 21.5% and 74% less power as compared to the private L2 caches for 2P-2L1ID-2L2-1M and 4P-4L1ID-4L2-1M, respectively. The hybrid L2 caches for 4P-4L1ID-2L2-1M require 14% less area and consume 21.4% less power as compared to the private L2 caches for 4P-4L1ID-4L2-1M whereas the shared L2 cache for 4P-4L1ID-1L2-1M requires 8.7% less area and consumes 43% less power as compared to the hybrid L2 caches for 4P-4L1ID-2L2-1M. These results indicate that power-efficiency of shared LLCs improves as the number of cores increases.

We emphasize that the workloads and cache miss rates have a large influence on an architecture's power consumption. For example, higher cache miss rates lead to an increase in power consumption because of more frequent requests to the power hungry MM. The processor-bound workloads are likely to consume less power than the mixed workloads, which in turn consumes less power than the memory-bound workloads. The power efficiency of processor-bound workloads stems from the fact that workloads spend more time in power-efficient processor cores as compared to the power hungry caches and MM (as shown in the power breakdown of architectural elements in Table III). Furthermore, power consumption of an architecture increases as the number of jobs increases because increased utilization restricts prolonged operation of architectural elements (e.g., processor cores) in low-power modes.

E. Performance per Watt

In this subsection, we present performance per watt results for multi-core embedded architectures in terms of FLOPS/W assuming 64-bit floating point operations. We observe that the performance per watt delivered by the processor cores and the L1-I and L1-D caches for these architectures are very close (within 7%), however, L2 caches present interesting results. The peak performance per watt for L2 caches is 11.8 MFLOPS/W and 14.3 MFLOPS/W for 2P-2L1ID-2L2-

1M and 2P-2L1ID-1L2-1M, respectively, when L1-I, L1-D, and L2 cache miss rates are all equal to 0.3, P_{pm} = 0.9, and N = 20. The peak performance per watt for L2 caches is 7.6 MFLOPS/W, 9.2 MFLOPS/W, and 13.8 MFLOPS/W for 4P-4L1ID-4L2-1M, 4P-4L1ID-2L2-1M, and 4P-4L1ID-1L2-1M, respectively, when L1-I, L1-D, and L2 cache miss rates are all equal to 0.2, P_{pm} = 0.9, and N = 20. These results indicate that architectures with shared LLCs provide the highest LLC performance per watt followed by architectures with hybrid LLCs and then private LLCs, however, architectures with shared LLCs may potentially introduce an MM response time bottleneck especially for higher cache miss rates. Similar trends were observed for processor-bound and mixed workloads.

V. Conclusions and Future Work

In this paper, we developed closed product-form queueing network models for performance evaluation of multi-core embedded architectures for different workload characteristics. The performance evaluation results indicated that the architectures with shared LLCs provided better cache response time and MFLOPS/W than the private LLCs for all cache miss rates especially as the number of cores increased. The results also revealed the downside of shared LLCs indicating that the shared LLCs are more likely to cause a main memory response time bottleneck for larger cache miss rates as compared to the private LLCs. The memory bottleneck caused by shared LLCs may lead to increased response time for processor cores because of stalling or idle waiting. However, results indicated that the main memory bottleneck created by shared LLCs can be mitigated by using a hybrid of private and shared LLCs (i.e., sharing LLCs by a fewer number of cores though hybrid LLCs consume more power than the shared LLCs and deliver comparatively less MFLOPS/W. The area and power consumption results for the multi-core embedded architectures revealed that the multi-core architectures with shared LLCs became more area and power efficient as compared to the architectures with private LLCs as the number of processor cores in the architectures increased.

In our future work, we plan to characterize the behavior of benchmarks (i.e., whether processor-bound, memory-bound, or mixed) and run these benchmarks on a multi-core simulator with a different number of cores and cache configurations for further verification of our queueing network modeling approach. Future work also includes enhancing our queueing theoretic model for performance evaluation of heterogeneous multi-core embedded 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] J. Balfour, "Efficient Embedded Computing," Ph.D. dissertation, EE Department, Stanford Univ., May 2010.

[2] A. Fedorova, S. Blagodurov, and S. Zhuravlev, "Managing Contention for Shared Resources on Multicore Processors," *Communications of the ACM*, vol. 53, no. 2, pp. 49–57, February 2010.

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

[4] J. Savage and M. Zubair, "A Unified Model for Multicore Architectures," in *Proc. of ACM IFMT*, Cairo, Egypt, November 2008.

[5] R. Sahner, K. Trivedi, and A. Puliafito, *Performance and Reliability Analysis of Computer Systems: An Example-Based Approach Using the SHARPE Software Package*. Kluwer Academic Publishers, 1996.

[6] R. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. Wiley, 1991.

[7] R. Kumar and *et al.*, "Heterogeneous Chip Multiprocessors," *IEEE Computer*, vol. 38, no. 11, pp. 32–38, November 2005.

[8] M. Sabry, M. Ruggiero, and P. Valle, "Performance and Energy Trade-offs Analysis of L2 On-chip Cache Architectures for Embedded MP-SoCs," in *Proc. of IEEE/ACM GLSVLSI*, Providence, Rhode Island, USA, May 2010.

[9] J. Medhi, *Stochastic Models in Queueing Theory*. Academic Press, An imprint of Elsevier Science, 2003.

[10] Intel, "Dual-Core Intel Xeon Processors LV and ULV for Embedded Computing," March 2011. [Online]. Available: ftp://download.intel.com/design/intarch/prodbref/31578602.pdf

[11] O. Kwon, H. Bahn, and K. Koh, "FARS: A Page Replacement Algorithm for NAND Flash Memory Based Embedded Systems," in *Proc. of IEEE CIT*, Sydney, Australia, July 2008.

[12] L. Shi and *et al.*, "Write Activity Reduction on Flash Main Memory via Smart Victim Cache," in *Proc. of ACM GLSVLSI*, Providence, Rhode Island, USA, May 2010.

[13] M. Reiser and S. Lavenberg, "Mean Value Analysis of Closed Multi-chain Queueing Networks," *Journal of ACM*, vol. 27, no. 2, pp. 313–322, April 1980.

[14] K. Sevcik and I. Mitrani, "The Distribution of Queueing Network States at Input and Output Instants," *Journal of ACM*, vol. 28, no. 2, pp. 358–371, April 1981.

[15] ARM7TDMI, "ATMEL Embedded RISC Microcontroller Core: ARM7TDMI," August 2011. [Online]. Available: http://www.atmel.com/

[16] ——, "ARM7TDMI Data Sheet," August 2011. [Online]. Available: http://www.atmel.com/

[17] TILERA, "Tile Processor Architecture Overview," in *TILERA Official Documentation, Copyright 2006-2009 Tilera Corporation*, November 2009.

[18] L. Yang and *et al.*, "Online Memory Compression for Embedded Systems," *ACM TECS*, vol. 9, no. 3, pp. 27:1–27:30, March 2010.

[19] Freescale, "Cache Latencies of the PowerPC MPC7451," August 2011. [Online]. Available: http://cache.freescale.com/files/32bit/doc/app_note/AN2180.pdf

[20] R. Min, W.-B. Jone, and Y. Hu, "Location Cache: A Low-Power L2 Cache System," in *Proc. of ACM ISLPED*, Newport Beach, California, August 2004.

[21] Y. Chen and *et al.*, "Accelerating Video Feature Extractions in CBVIR on Multi-core Systems," *Intel Technology Journal*, vol. 11, no. 4, pp. 349–360, November 2007.

[22] P. Jain, "Software-assisted cache mechanisms for embedded systems," Ph.D. dissertation, EECS Department, MIT, February 2008.

[23] CACTI, "An Integrated Cache and Memory Access Time, Cycle Time, Area, Leakage, and Dynamic Power Model ," August 2011. [Online]. Available: http://www.hpl.hp.com/research/cacti/

[24] ITRS, "International Technology Roadmap for Semiconductors," August 2011. [Online]. Available: http://www.itrs.net/