2012 Sixth IEEE/ACM International Symposium on Networks-on-Chip

Analytical Performance Modeling of Hierarchical Interconnect Fabrics

Nikita Nikitin, Javier de San Pedro, Josep Carmona and Jordi Cortadella
Universitat Politècnica de Catalunya
Barcelona, Spain

***Abstract*—The continuous scaling of nanoelectronics is increasing the complexity of chip multiprocessors (CMPs) and exacerbating the memory wall problem. As CMPs become more complex, the memory subsystem is organized into more hierarchical structures to better exploit locality. During the exploration and design of CMP architectures, it is essential to efficiently analyze their performance. However, performance is highly determined by the latency of the memory subsystem, which in turn has a cyclic dependency with the memory traffic generated by the cores. This paper proposes a scalable analytical method to estimate the performance of highly parallel CMPs (hundreds of cores) with hierarchical interconnect fabrics. The method can use customizable probabilistic models and solves the cyclic dependencies by using a fixed-point strategy. The technique is shown to be a very accurate and efficient strategy when compared to the results obtained by simulation.**

I. Introduction

The continuous shrinking of CMOS technology has enabled the integration of multiple cores and distributed memory in one chip. Parallelism has also been one of the paradigms to make computations more power efficient. In the last few years, multicore systems have evolved from having few cores [1], [2] to single-chip processors with tens or hundreds of computing units [3]–[5].

Tiled CMPs are an effective approach to architect general-purpose processors under the intense time-to-market pressure [6], [7]. The replication of tiles provides a rapid way of floorplanning many computing units in one chip and communicating them with scalable interconnect fabrics. Figure 1(a) shows an example of a CMP with 16 tiles, each one including a computing core (C), two levels of private on-chip caches (L1, L2), and a router (R) that communicates with the on-chip interconnection network (a mesh). Two memory controllers (MC) provide access to the off-chip memory.

To exploit the locality of memory references, hierarchical interconnects have been proposed [7], [8]. Several cores can be grouped into one cluster to share the on-chip cache, accessible through a local interconnect (e.g., bus, crossbar, ring, etc). Hierarchy increases the intra-cluster hit-ratio and reduces the traffic in the top-level interconnect. Figure 1(b) shows an implementation of a CMP with 4 clusters. Each cluster has two cores with private caches, a shared cache (L3) with tag directory (DIR), a local interconnect (IC), a router and a network interface (NI).

Given the vast space of design parameters, CMP designers are faced with the complex problem of selecting the best architecture subject to a set of constraints. Many design options must be explored, such as the variety of core implementations, interconnect types, topologies, cache hierarchies and memory management policies. Moreover, the amount of configurations increases drastically as the technology advances, allowing more cores and memory to fit into the chip area.

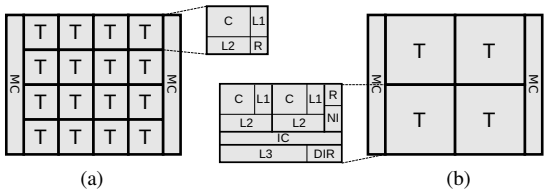


Fig. 1: CMP layouts: (a) flat, (b) hierarchical.

Evaluating the performance of a CMP architecture is essential to take the correct decisions during design. Unfortunately, simulation imposes a prohibitive computational cost when the space of design points grows significantly. In this scenario, analytical modeling becomes an effective alternative for rapidly pruning the design space during early exploration and selecting a small set of promising configurations. Along this line, several analytical models for CMP exploration have been recently proposed (e.g., [9], [10]).

The fundamental problem in evaluating the performance of a CMP is the calculation of the latency for memory requests, given the parameters of the interconnect fabrics and memory hierarchy. A key phenomenon that is underestimated by the existing models is the contention effect of the interconnection fabric. This paper will show that *contention has a major significance in the analysis of CMP performance*. Ignoring contention leads to optimistic latency and throughput measurements, and may overestimate the architectures with saturated interconnects. As a result, exploration may select inefficient architectures and, even worse, discard the promising ones.

Accounting for contention is particularly important when exploring hierarchical CMPs. Interconnects at different levels of hierarchy may deliver different throughput characteristics (e.g., a bus at the cluster level and a mesh at the top level). It is then essential to verify that the required bandwidth between the cores and the memory is delivered at all levels. This verification discards architectures in which one level of the interconnect is saturated, while another remains underutilized and consumes resources unnecessarily.

The purpose of this work is to emphasize the importance and provide analytical methods for modeling contention in

978-0-7695-4677-3/12 $26.00 © 2012 IEEE
DOI 10.1109/NOCS.2012.20

CMP exploration frameworks. The contributions of the paper can be summarized as follows. First, we formulate the cyclic dependency between the latency and the rate of memory requests as a system of non-linear equations that models the contention in the CMP interconnect. Second, we propose three methods to resolve this model: using a general-purpose solver, a *fixed-point iteration* and a *bisection* method. The last two methods show significant runtime savings, trading-off accuracy and convergence. More importantly, these methods can be parametrized with any black-box analytical model for the latency. This makes our strategy flexible to incorporate novel models for on-chip interconnects. Third, we experimentally show the application of the methods for CMP exploration and confirm the importance of evaluating contention.

Next section describes a simple example to emphasize the importance of contention in performance evaluation of hierarchical CMPs. Section III reports related work. The models for CMP throughput, memory latency, traffic and their interdependency are discussed in Section IV. In Section V, we propose three methods to resolve this dependency. The experimental evaluation of the methods is described in Section VI.

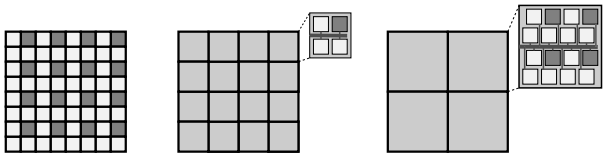
II. The importance of contention: an example

Consider a CMP with 48 cores and 16 shared on-chip cache modules. Figure 2 presents three (of the many) possible architectures with such parameters. One of the architectures has an 8×8 structure of regular tiles connected with a mesh (Figure 2(a)). The cores and caches are shown as light and dark squares, respectively. Solid lines represent the mesh links.

To take advantage of the locality of memory accesses, several cores and caches can be grouped in a cluster and communicate via the local interconnect. For instance, clusters with bus interconnects were shown to notably improve the average communication latency [8]. This fact encourages the exploration of hierarchical interconnects. Figure 2(b) describes the CMP organization with 16 clusters, each one having three cores, one cache and a shared bus. The clusters communicate via the top-level 4×4 mesh. Another option is to increase the cluster size up to 16 components (12 cores and 4 caches) and decrease the dimensions of the top-level mesh (Figure 2(c)).

One of the problems of architectural exploration is to select the configuration with the best performance. We first estimated the throughput of each configuration using only the static (hop-count) latency of the network, i.e., assuming no contention. In this experiment we assumed the ideal throughput of cores (under the assumption of zero-latency memory) to be 2.0 `IPC` (instructions per cycle), and the number of memory references per instruction to be 0.5. The values of static latency (in cycles) and the estimated throughput (in `IPC`) are displayed in the columns L_{est} and θ_{est} of Table I. Hierarchical architectures show a higher performance due to the exploited locality: configuration (c) has the largest size of local cache (per cluster), hence the increased local hit ratio. Therefore, (c) shows the highest estimated throughput.

However, this conclusion is incorrect when network contention is taken into account. Simulation reveals rather distinct



(a) 8×8 mesh (b) 4×4 mesh with bus clusters (c) 2×2 mesh with bus clusters

Fig. 2: Possible architectures for a 48-core CMP.

TABLE I: Performance of architectures in Figure 2.

Architecture	L_{est}	θ_{est}	L_{sim}	θ_{sim}
(a)	11.17	8.23	11.26	8.16
(b)	10.12	9.04	10.40	8.81
(c)	9.95	9.19	16.69	5.58

performance numbers, reported in the L_{sim} and θ_{sim} columns of Table I. For configurations (a) and (b), the estimated throughput with no contention is close to the one reported by simulation. However, the performance of configuration (c) drops by about 40%. In fact, simulation concludes that (c) is the worst in terms of performance.

The reason of this significant discrepancy is the fabric contention. It occurs because of the competition between memory requests for the shared resources of the interconnect. This results into longer latencies, decreasing the overall performance of the system. In this example configuration (b), which incorporates hierarchy at some extent, is the one with the highest throughput and represents the best architectural trade-off between cache locality and communication parallelism.

III. Related work

The topic of CMP design space exploration has been widely studied in the last years. Many simulation-based frameworks (e.g. [11], [12]) appeared to extensively investigate the parameters of core architectures, memory hierarchies and emphasize the importance of their joint optimization for improving the power, performance and thermal characteristics.

Analytical models aim at replacing costly simulations and provide instead a quick insight on the architectures. However, the modeling techniques in the literature significantly underestimate the performance degradation caused by the contention in the communication fabric. The model in [9] studies the trade-off between the number of cores and the on-chip memory size for throughput optimization. The latency model used includes a contention penalty with linear dependency on the number of cores. Still, apart from being inaccurate, this approximation does not allow to compare interconnects with various parameters and topologies. In [10] the authors introduce an energy-performance analytical model for CMP architectures, however they only consider bus interconnects with a simplified contention model. The work in [13] analyzes finite cache penalties in memory hierarchies, but the interconnects are also restricted to buses.

In this paper we propose a generic method for analytical modeling of contention in hierarchical interconnect fabrics. The advantages of hierarchical topologies for many-

core CMPs have been demonstrated in [7], [8]. Our method can be parametrized by an arbitrary latency model for on-chip interconnect. This paper discusses the application of the latency model in [14] due to its flexibility. Other models can be considered, such as [15]. It introduces an accurate model for heterogeneous NoCs that can be useful for modeling variable number of virtual channels and link capacities on the different levels of the hierarchical interconnect. In [16], an approach similar to [14] is proposed, offering an accurate backpressure analysis at the cost of the model efficiency. To overcome the limitations of queueing approaches, alternative latency models (e.g. using non-stationary traffic analysis [17]) can be used.

IV. CMP Performance modeling

This section introduces the models for the evaluation of CMP performance. First, we explain the assumptions and input parameters of the model. Next, the equation to model static latency is presented. This equation is then extended to consider the contention component of communication. Finally, the throughput model is discussed and the formula for memory request rate is derived. The section concludes by emphasizing the cyclic dependency between memory traffic and latency.

A. Assumptions and input parameters of the analytical model

In this paper we focus on systems with two-level hierarchical interconnect fabrics. However, the approach can be applied for an arbitrary number of hierarchical levels, including the particular case of flat interconnects. Several components are grouped into a cluster: cores, components of the memory subsystem and the local interconnect. The top-level interconnect provides communication between the clusters and access to the off-chip memory (Figure 1(b)).

The system has in total N cores, each one with two user-defined parameters. $\texttt{IPC}_0$ is the ideal core throughput, i.e., the amount of instructions executed by the core in one cycle, assuming zero-latency memory. `MPI` is the average number of memory references generated per instruction. These parameters characterize both the core and the workload.

Without loss of generality, we assume that the memory subsystem has four hierarchy levels. Every core has a private *L1* cache and possibly, a private *L2* cache of larger size but higher latency. The clusters incorporate modules of a distributed *L3* cache, shared by all cores. The off-chip memory is accessible via a set of memory controllers. The latencies of the caches and the off-chip memory are provided as parameters.

We use the term *memory flow* to denote a feasible communication between a core and a component of the memory subsystem. For example, each core may access its own *L1* or *L2* caches, or any of the *L3* modules or *MC*s. The set of all possible memory flows for core c is denoted as $F(c)$.

Every flow $f \in F(c)$ is realizable with probability p_f, that defines the probability for c to request data from a certain memory component. These probabilities can be user-defined or calculated with some analytical model. In our work we calculate these probabilities using a *model of cache miss behavior* based on a *power law* that represents the dependency between miss ratio and cache size. This model was proven to be a good approximation [18]:

$$\mathit{Miss}(S) = \kappa S^{-\alpha}, \tag{1}$$

where S is the cache size, and κ, α are the model parameters.

Since *L3* is a distributed cache, its access latency depends on the cluster where the requested data is stored. In this work we assume the probability to find the data in a particular cluster to be inversely proportional to the distance between the requesting core and the cluster. However, our method can be parameterized with any other model for distributed cache.

B. Static latency

In this section we describe how to calculate the average static latency of memory accesses for a core c in the presence of memory hierarchy. Given the probability p_f for each flow $f \in F(c)$ and its latency L_f, the static latency L_c^{st} is:

$$L_c^{st} = \sum_{f \in F(c)} p_f L_f. \tag{2}$$

Since requests to *L3* and *MC* are sent via the communication fabric, its delay must also be considered. This delay represents the latency of the on-chip network traversal and is defined using the routing function $\mathcal{R} : f \rightarrow \pi(f)$, that for any flow f returns its routing path $\pi(f)$. In this work we consider the *XY-routing* function [19], however any deterministic or even adaptive routing can be used, specifying the probabilities for certain paths. The total latency to access an *L3* instance is the sum of the network traversal latency along the path $\pi(f)$ and the *L3* latency. The total latency of the off-chip memory accesses is calculated likewise.

The flow probabilities p_f are obtained using the dependency of miss ratio on the cache size, *Miss(S)*, given by (1). Assuming the sizes S_{L1}, S_{L2} of the two low-level caches, the probabilities to access them are:

$$\begin{aligned} p_{L1} &= 1 - \mathit{Miss}(S_{L1}), \\ p_{L2} &= (1 - p_{L1})(1 - \mathit{Miss}(S_{L2})). \end{aligned}$$

As *L3* is shared, the miss ratio is defined by the effective *L3* size, S_{L3}^{eff}, seen by each core [20]. To estimate S_{L3}^{eff} we use the concept of the average number of cores, sharing each line, as proposed by [20]. The probability to access *L3* is then:

$$p_{L3} = (1 - p_{L1})(1 - p_{L2})(1 - \mathit{Miss}(S_{L3}^{eff})).$$

Finally, p_{L3} should be multiplied by the probability to find the data in a particular *L3* instance (cluster). A similar strategy is used to calculate the probabilities of flows to every memory controller.

C. Queueing model for the on-chip interconnect

Equation (2) describes the static latency of memory accesses. Another important part of the communication delay is the dynamic or contention latency [19]. Contention happens in the interconnect fabrics when several packets compete for the same shared resource, such as a bus or an NoC link. This results into additional delays experienced by packets in

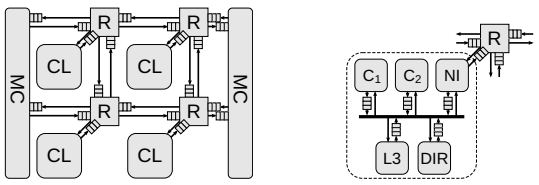


Fig. 3: Queueing representation for (a) mesh NoC and (b) bus-based cluster.

the buffers, distributed over the on-chip interconnect. One of the approaches to estimate the contention delays is to model the CMP as a system of queues and apply queuing theory to calculate the buffer delays.

Figure 3(a) shows the queueing representation of the top-level mesh interconnect. The mesh routers (R) have up to five input-buffered ports to store incoming flits while the router is busy. The primary ports of the routers are connected to the clusters of devices (*CL*), which in case of a flat CMP organization may consist of one device (e.g. a core with private caches in Figure 1(a)). Figure 3(b) presents the queueing model of a cluster, corresponding to one tile of the hierarchical CMP depicted in Figure 1(b). The cluster consists of five devices, communicating via a shared bus: two cores with private caches, an instance of an *L3* shared cache, a directory and a network interface. Every device has a buffer to store the requests to the bus. To distribute the off-chip memory traffic uniformly over the mesh and avoid high contention of certain routers, we assume that memory controllers have multiple connections to the mesh, as shown in Figure 3(a).

D. Total latency

The average total latency for core c, L_c, is obtained by adding the queue delays along the communication paths, denoted as w_q, to the static latency. Hence, given the paths $\pi(f)$ for every flow f, we extend equation (2) accordingly:

$$L_c = L_c^{st} + \sum_{f \in F(c)} p_f \sum_{q \in \pi(f)} w_q. \tag{3}$$

To find the values for w_q, an analytical model for the on-chip interconnects can be used. In this work we apply the model from [14], that permits calculating the delays for a variety of topologies. Given the vector of injection rates into the interconnect, $\bar{\lambda} \in \mathbb{R}^N$, the model proposes to express queue delays in the form of a system of equations with a matrix W:

$$\bar{w}_q = W(\bar{\lambda}). \tag{4}$$

The exact form of the matrix W is given by the expressions (5) and (18) in [14]. What only remains is to compute the injection rates $\bar{\lambda}$, which is covered in the next sections.

E. Throughput model

The throughput of a CMP and the traffic of its interconnect are closely related. To derive the exact dependencies, we start with the performance model for a single core, given in [21]. For a core with the average rate of accesses to remote memory (*RemRate*), and the cost of an access (*RemCost*), the average number of cycles for executing an instruction, CPI, is:

$$\texttt{CPI} = \texttt{CPI}_0 + \textit{RemRate} \cdot \textit{RemCost}, \tag{5}$$

where $\texttt{CPI}_0 = 1/\texttt{IPC}_0$ is the ideal CPI, derived under the assumption of zero-latency memory. For a single-threaded in-order core, the cost of a remote access is the average latency, given by (3), and the remote rate is given by the MPI value. As throughput is typically measured in IPC, the reciprocal of CPI, from (5) we obtain:

$$\theta_c = \frac{1}{\texttt{CPI}} = \frac{1}{\frac{1}{\texttt{IPC}_0} + \texttt{MPI} \cdot L_c}. \tag{6}$$

The throughput of a CMP, θ_{cmp}, is then calculated as the total performance of individual cores:

$$\theta_{cmp} = \sum_c \theta_c. \tag{7}$$

The rate of memory accesses, λ_c, is the probability for a core to issue a remote memory request per cycle. λ_c is proportional to the core throughput and the MPI:

$$\lambda_c = \theta_c \cdot \texttt{MPI} = \frac{\texttt{MPI}}{\frac{1}{\texttt{IPC}_0} + \texttt{MPI} \cdot L_c}. \tag{8}$$

This equation can be extended for the case of more complex out-of-order and multithreaded cores. The difference in modeling an out-of-order core is that the remote memory access does not force the core to stall, hence the effective remote latency L_c decreases [21]. Techniques discussing throughput modeling for out-of-order implementations can be found in [22]. A multithreaded core can be modeled as a group of single-threaded cores. The latency for each thread remains L_c, but the total memory access rate becomes $\lambda_c^{mt} = M\lambda_c$, where M is the number of threads.

F. The cyclic dependency between memory latency and traffic

In order to calculate the buffer delays, equation (4) requires the injection rates at every input (source) of the interconnect, while equation (8) gives the rates of request generation per core. Note that the injection rates in a *flat* interconnect are directly defined by the core rates: for a CMP with N cores, $\bar{\lambda} = \{\lambda_1, .., \lambda_N\}$. In case of a *hierarchical* interconnect fabric, the core rates will correspond to the injection rates at the sources of the cluster-level interconnects, such as the bus in Figure 3(b). The injection rates to the top-level mesh can be calculated, given the fraction of inter-cluster traffic. The latter is defined by the probabilities of access to the *L3* and the off-chip memory, discussed in section IV-B. Below we directly consider the dependency of memory latency on the core rates.

From (3), (4) and (8) we observe the following system of dependencies:

$$\forall c = 1, .., N: \quad \begin{cases} L_c = L(\lambda_1, .., \lambda_N) \\ \lambda_c = \lambda(L_c). \end{cases} \tag{9}$$

This result is quite intuitive: the latency of the memory requests traversing the interconnect depends on the rate of sent

requests, due to the network contention. In turn, the request rate is determined by the latency, as no new memory requests are issued if the execution of cores stalls due to the absence of data. System (9) emphasizes the *cyclic dependency between the latency and rate of memory requests*. In the following section we describe the methods to resolve this dependency.

V. Analytical methods for latency estimation

In this section we propose three methods that can be used to resolve the dependency (9). Apart from the straightforward way to solve the system of equations, the *fixed-point iteration* and *bisection* methods are discussed. The benefit of the fixed-point method is that it delivers the exact solution in case of convergence and can be applied to arbitrary configurations. The bisection always converges for our problem, but finds an approximate solution. However, it was found to be a good approximation for tiled homogeneous CMPs (see Section V-C).

A. Solving the system of nonlinear equations

Given an analytical model for the interconnect latency as a closed form, the straightforward way to find L_c is to solve the system of nonlinear equations. We apply the model from [14], which offers a convenient definition of queue delays via the injection rates in the closed form (4). Hence, *the dependencies* (3)*,* (4) *and* (8) *create a system of equations* with respect to the vectors of variables $\bar{L}$, $\bar{\lambda}$, and $\bar{w}_q$.

The system will always contain nonlinear equations because of (8). The solution to a nonlinear system can be found using a solver, such as MATLAB [23]. Although hard to be proved analytically, our conjecture is that the system has a unique solution. We verified this by running MATLAB with different initial vectors and observing convergence to the same solution.

While this method is straightforward, it has two important drawbacks. First, it only works for the analytical models that provide closed-form equations for latency. Second, unless properly tuned, the methods for solving general systems of nonlinear equations may exhibit poor performance. Since our objective is to apply the method for exploration of a large design space, this limitation is critical. In conclusion, this method is useful to validate the techniques described below.

B. Fixed-point iteration

The algorithm proposed in this section is a popular numerical method for solving the systems of nonlinear equations [24]. While the theoretical speed of convergence of this method is relatively slow, it performs well in practice due to its low cost for a single iteration. Given a system of equations in the form:

$$\bar{x} = F(\bar{x}), \tag{10}$$

where $\bar{x}$ is the vector of unknowns and F is the system matrix, and an initial guess $\bar{x}_0$, the following iterative procedure can be used to find a solution $\bar{x}^*$ (fixed point) of the system:

$$\bar{x}_{n+1} = F(\bar{x}_n),\ n \geq 0.$$

In our setting, $\bar{x}$ is composed of the variables $\{\bar{L},\ \bar{\lambda},\ \bar{w}_q\}$ and matrix F is defined by the right-hand terms of the equations (3), (4) and (8). For the initial guess, $\bar{x}_0$, we use static latencies (2) and compute other values using the same equations: $\bar{L}_0 = \bar{L}^{st}, \bar{\lambda}_0 = \lambda(\bar{L}_0), \bar{w}_{q,0} = W(\bar{\lambda}_0)$.

The benefit of the proposed method is that it does not require closed-form analytical expressions for latencies. Furthermore, any black-box model for the dependency of NoC latency on injection rate can be used. The method hence maintains the modular structure of hierarchical interconnects and permits plugging independent models for different topologies, such as bus (cluster-level) and mesh (top-level). This makes the approach a valid tool for future interconnect modeling.

As a numerical method, fixed-point iteration is subject to convergence issues. For a system in the form (10), the sufficient condition for convergence is [24]:

$$\sum_i \left| \frac{\partial F}{\partial x_i} \right| < 1.$$

In our case, this requires the latency to grow slowly with the injection rate, and vice versa. This condition holds for the communication fabrics that perform far from their saturation throughput (for instance, see chapter 23 in [19]). Although this condition is quite strong, it is not necessary for convergence. In practice we observe that for the majority of configurations the iterative procedure converges.

A second issue of the fixed-point iteration is due to the analytical models based on queueing theory: the queueing models work under the assumption of system being in the steady-state [25]. This means, that for any router with service time T and the sum of arrival rates to its inputs λ, the following condition must hold: $\lambda T < 1$. In other words, there should be no packet accumulation in the input queues of the router. Unfortunately, this requirement may be not satisfied by the initial solution. From (8) we know that the latency L_c and the memory access rate λ_c are inversely proportional. Since static latency is taken for the initial value of L_c, it may be highly underestimated for the configurations with high contention. As a result, the initial value of λ_c will be overestimated and may violate the steady-state condition.

To handle this situation as well as the configurations for which the fixed-point iteration diverges, we propose the method based on bisection search of λ_c, to find a reasonable and fast approximation to the solution.

C. Bisection search for traffic rate

The advantage of the bisection method is that it always converges for our model (due to the intermediate value theorem [24]). Since every core generates traffic at certain rate, λ_c, *multidimensional bisection* [26] can be applied to find the exact rates. However, a good approximation to the exact rates can be obtained by using the less complex *unidimensional bisection*: by simulation we observed that tiled CMPs with uniform clusters tend to have traffic rates that change proportionally to their estimates, obtained by the static latency. Hence, we initialize the vector of injection rates $\bar{\lambda}$ with the values estimated by static latency, and on every bisection step adjust all rates in the same proportion.

To introduce the bisection more formally, let us rewrite equation (8) by isolating L_c, and using the star symbol to distinguish from the latency in (3):

$$L_c^*(\lambda_c) = \frac{1}{\lambda_c} - \frac{1}{\texttt{MPI} \cdot \texttt{IPC}_0}. \quad (11)$$

From (3) and (11) we define the average latencies $L(\bar{\lambda})$ and $L^*(\bar{\lambda})$ as the functions of the vector $\bar{\lambda}$:

$$L(\bar{\lambda}) = \frac{1}{N}\sum_{c=1}^{N} L_c(\bar{\lambda}), \quad L^*(\bar{\lambda}) = \frac{1}{N}\sum_{c=1}^{N} L_c^*(\lambda_c).$$

Finally we introduce the *latency difference function*, F($\bar{\lambda}$):

$$F(\bar{\lambda}) = L(\bar{\lambda}) - L^*(\bar{\lambda}).$$

Figure 4 shows the typical behavior of these functions, emphasizing the cyclic dependency (9). To depict a 2D view of this behavior, we plot $L(\bar{\lambda})$ and $L^*(\bar{\lambda})$ as a function of the average rate $\Lambda = \frac{1}{N}\sum_{c=1}^{N}\lambda_c$. The curve $L^*(\bar{\lambda})$ shows that the average rate of memory requests increases as the latency decreases. In turn, $L(\bar{\lambda})$ shows that the average latency of requests grows with the injection rate. The real values for latency and traffic are defined by the intersection point A of these curves, that can be found as a root of $F(\bar{\lambda})$. Hence, we use the bisection as a root-finding method, that does not require the exact knowledge of the function $F(\bar{\lambda})$ and can be used with any black-box analytical model for latency.

Bisection searches for $\bar{\lambda}$ that satisfies the condition $|F(\bar{\lambda})| < \epsilon$, where ϵ is the solution tolerance. The initial range for $\bar{\lambda}$ is limited by the traffic, obtained with static latency: $\bar{\lambda}_{min} = \bar{0}$, $\bar{\lambda}_{max} = \bar{\lambda}(L_c^{st})$. Assuming the proportionality in variation of the individual components of $\bar{\lambda}$, all components are updated simultaneously. For any pair of consecutive iterations i and $i+1$, either $\bar{\lambda}_{min}^{i+1} = \bar{\lambda}^i$ when $F(\bar{\lambda}^i) < 0$, or $\bar{\lambda}_{max}^{i+1} = \bar{\lambda}^i$ when $F(\bar{\lambda}^i) > 0$. The iteration is continued until the required tolerance for $F(\bar{\lambda})$ or $\bar{\lambda}$ is met [24].

VI. Experimental results

In this section we describe several experiments used to validate the proposed analytical method for efficiency and quality. Validation is performed with respect to simulation. Section VI-A describes our simulation environment. Further sections focus on the experiments for the analytical model.

A. Simulation environment

To verify the analytical model we have developed a flit-level simulator for hierarchical CMP interconnects on top of BookSim 2.0 [19]. In contrast to the analytical model, the simulator performs flit-level modeling of contention in the interconnect fabric and cycle-accurate calculation of throughput.

To simulate a hierarchical CMP, three enhancements were made to BookSim. First, BookSim purely probabilistic traffic injection patterns were replaced with *state machine models for cores, caches and memory controllers*. The cores inject memory requests based on the average workload parameters and stall waiting for the replies. Memories accept requests from cores and send replies after a predefined latency. As another extension, *support for hierarchical topologies* was added. This enables simulation of multi-level interconnect fabrics with arbitrary depth. Finally, we implemented *bus and multibus* topologies.

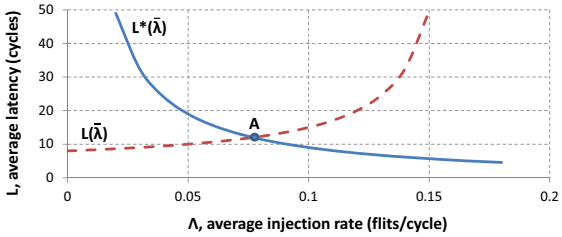


Fig. 4: Behavior of the latency functions $L(\bar{\lambda})$ and $L^*(\bar{\lambda})$.

Each simulation was run long enough to obtain a 2% relative error (the same value used for the analytical method) with a 95% confidence degree. The 95% confidence interval is calculated using the popular batch means method [27].

B. Efficiency of the analytical methods

In this section we compare the efficiency of the three analytical methods for resolving the cyclic dependency presented in Section V: solver (MATLAB), fixed-point iteration (FP) and bisection (BS). We generated a set of CMPs having flat mesh topologies with various dimensions and contention degrees. The reason to select flat interconnects is to demonstrate that even for rather simple architectures the obtained system of equations is hard to be tackled in a straightforward way.

The test cases and the results of modeling are summarized in Table II. The first three columns display the test name, mesh dimension and the ratio of contention latency, with respect to the average total latency. The fourth column represents the number of variables and equations in the obtained system. The fifth column shows the time required to find a solution using the general nonlinear solver provided by MATLAB. The last two columns show the time consumed by the FP and BS methods. For each test case the three methods converged to the same solution, within the given tolerance region of 2%.

Test cases T1 to T5 accentuate how the MATLAB time grows with the mesh size. The solution to T5-T7 could not been found within an hour. Clearly, this straightforward method is not acceptable for efficient exploration of CMPs.

The purpose of test cases T6 and T7 is to compare the FP and BS methods. Configuration T6 has higher contention than T5. As a result, FP method takes more time to converge than BS. Configuration T7 has even more contention, resulting in a violation of the steady-state assumption for the queueing model (see Section V-B). Hence, FP can not be used in this case, so BS is the only option.

We observed that FP typically outperforms BS, when the contention component of latency is moderate, i.e. does not exceed about 30-40% of the total latency. Hence we choose to run FP first and use BS only when the former method fails.

C. Architectural exploration for CMPs

To validate the quality of the analytical methods in performance estimation of hierarchical architectures, we carry out an

TABLE II: Performance comparison of analytical methods.

Test	Mesh	Cont. lat.	Num. of var./eqn.	Runtime (sec)		
				MATLAB	Fixed-point	Bisection
T1	2×2	5%	236	0.023	0.001	0.001
T2	4×4	13%	1224	1.412	0.001	0.002
T3	6×6	8%	3108	30.831	0.002	0.003
T4	8×8	12%	6128	408.539	0.006	0.010
T5	10×10	23%	10620	Timeout (1hr)	0.010	0.012
T6	10×10	46%	10620	Timeout (1hr)	0.022	0.015
T7	10×10	55%	10620	Timeout (1hr)	NA	0.016

experiment for CMP design space exploration. Our framework reads a setup file with the parameters for cores, memories and workloads, generates a multitude of architectures, and for every architecture obtains the throughput, using both modeling and simulation. With this experiment we demonstrate that the modeling selects a very similar set of best-throughput architectures as simulation, but in much shorter time.

Table III shows the exploration setup. The estimates of chip and component area were taken from the Niagara 2 processor [2]. We scaled the core area and memory density down to the 16*nm* to allow hundreds of cores fit into the chip area. The $\texttt{IPC}_0$, $\texttt{MPI}$ of cores and the miss ratio dependency on cache size were estimated from the benchmarks in [28]. The number of cores and cache sizes were varied to explore the trade-off between the computing units and the on-chip memory. All architectures were generated with the mesh-of-buses topology. The exploration of the mesh dimensions compromises the number of clusters and processors per cluster.

Given these parameters, our framework generates 1062 feasible configurations. The simulation of all the configurations took 324 minutes, while performance modeling was done in just 16.8 seconds, delivering more than 1000x speedup. The best architecture by simulation is the configuration #937, with a throughput of 30.81 $\texttt{IPC}$. It consists of 6×6 mesh (36 clusters, 5 cores per cluster), a total of 180 cores with 64Kb L1, 256Kb L2 private caches and 68Mb shared *L3* cache.

In Figure 5 we sorted the configurations by throughput along the horizontal axis, as estimated by simulation. One can see that the modeling estimate follows well the simulation curve. The analytical model for latency underestimates the contention, and therefore deviation increases with its degree. Configurations with similar throughput may have various contention degrees, hence the noisy behavior of the modeling curve. The error in throughput varies up to 19% with the

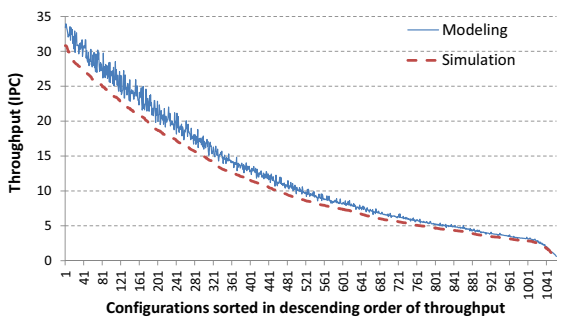


Fig. 5: Throughput comparison for modeling and simulation.

TABLE III: Parameters of the exploration.

Parameter	Value
Chip area	350 mm^2
Core area	1.25 mm^2
Core $\texttt{IPC}_0$	2.0
Core $\texttt{MPI}$	0.5
L1 size	64, 128 Kb
L2 size	64 Kb to 3 Mb
Memory density	1 mm^2 / Mb
Mesh dimensions	2×2 to 16×16
Miss ratio dependency on cache size	$0.05 \cdot CacheSize^{-0.4}$
Cache latency dependency on size	$5.0 \cdot CacheSize^{0.5}$ cycles
Off-chip memory latency	100 cycles

average value being 10%, which corresponds to the error reported by the latency model [14].

However, what really matters for exploration are the relative, rather than the absolute values of throughput. Indeed, when exploring the huge design space we would like to effectively prune suboptimal architectures and leave a moderate subset of promising solutions. These configurations can be simulated further to select the best one. Hence, we are interested in comparing the *order of configurations by the highest throughput*, as delivered by modeling and simulation. And here our technique demonstrates very accurate results: Figure 6 shows the comparison for the best-throughput order. To make the picture illustrative, we limit to consider the 50 best configurations, however the explained tendency is maintained for the whole set. The horizontal axis specifies the number N of best configurations chosen *by simulation*. The vertical axis indicates the minimum number of best configurations chosen *by modeling*, that include all the N best ones by simulation. For example, the point with coordinates (1; 2) means that the best configuration by simulation (#937) has the second place in modeling. Furthermore, the throughput of #937 is 30.81 $\texttt{IPC}$, while the throughput of the best configuration by modeling (#940) is 30.80 $\texttt{IPC}$, and is within our modeling tolerance. The rightmost point on the plot (50; 64) means that the 64 best configurations by modeling include all 50 best by simulation. This is actually a very accurate result for the analytical model, when comparing more than one thousand of configurations.

We also demonstrate that approximation by the static latency delivers poor order. It biases the exploration towards the configurations with large clusters, given the fact that the long contention latency in the buses is not considered. The point (1; 33) in Figure 6 means that the best configuration (#937) is on the 33rd position, when not considering contention.

For comparison, we also checked a configuration similar to #937, having the same number of cores and cache size, but exploiting less locality by using a 12×15 mesh with one core and one cache module per cluster. The throughput of this configuration is 24.23 $\texttt{IPC}$, that is 21% less. This witnesses the importance of the hierarchical fabrics exploration, effectively using the locality of memory accesses.

D. Scalability of the modeling

To investigate the scalability of the analytical model, we generated several CMPs with mesh-of-buses topology and

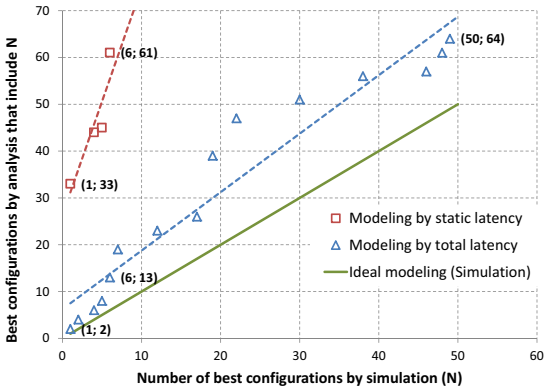


Fig. 6: Order comparison for modeling and simulation.

similar structure. Each cluster contains four components (three cores and one cache module) and a bus interconnect. The top-level mesh dimensions are varied from 2×2 to 16×16, producing CMPs with 16 to 1024 components. For each test case we executed both fixed-point and bisection and compared the average runtime value of these two methods with simulation. Figure 7 shows the results of comparison.

Our probabilistic simulator demonstrates very good performance. Simulation of the 16-component CMP takes just 2.5 seconds, and for the 1024-component CMP about 600 seconds. However, the modeling yet brings about three orders of magnitude improvement in efficiency. For the 16- and 1024-component CMPs modeling took only 0.002 seconds and 3.3 seconds respectively. In one second our method handles a CMP with nearly 700 components. This result justifies high scalability of the proposed method and its ability to efficiently explore architectures with many hundreds of cores.

VII. Conclusions

Analytical models for CMP performance are crucial to make the architectural exploration possible. This paper shows that such models need to incorporate the contention factor in order to adequately estimate performance. We have presented three analytical methods to model the contention of hierarchical interconnects, by resolving the cyclic dependency between the memory latency and traffic. The validity and efficiency of the model were proved through extensive simulation and with an example of architectural exploration.

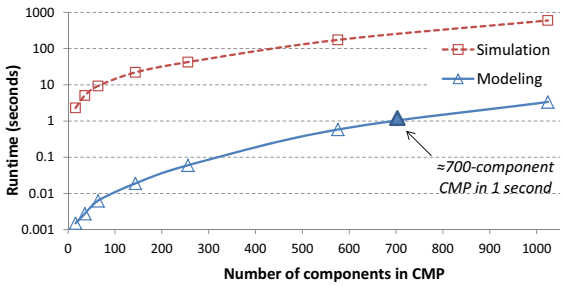


Fig. 7: Performance comparison of modeling and simulation.

VIII. Acknowledgment

This research has been funded by a grant from Intel Corp., project CICYT TIN2007-66523, and FPI grant BES-2008-004612.

References

[1] D. Pham et al., "Overview of the architecture, circuit design, and physical implementation of a first-generation cell processor," *Solid-State Circuits*, vol. 41, pp. 179–196, 2006.
[2] U. Nawathe et al., "An 8-core 64-thread 64b power-efficient SPARC SoC," in *Solid-State Circuits*, feb. 2007, pp. 108 –590.
[3] S. Bell et al., "TILE64 - processor: A 64-core SoC with mesh interconnect," in *Solid-State Circuits*, feb. 2008, pp. 88 –598.
[4] S. Vangal et al., "An 80-tile 1.28TFLOPS network-on-chip in 65nm CMOS," in *Solid-State Circuits*, feb. 2007, pp. 98 –589.
[5] J. Owens et al., "GPU computing," *Proceedings of the IEEE*, vol. 96, pp. 879 –899, may 2008.
[6] M. Taylor et al., "The Raw microprocessor: a computational fabric for software circuits and general-purpose programs," *Micro, IEEE*, vol. 22, no. 2, pp. 25 – 35, mar/apr 2002.
[7] J. Balfour and W. J. Dally, "Design tradeoffs for tiled CMP on-chip networks," in *Proc. Intl. Conf. on Supercomputing*, 2006, pp. 187–198.
[8] R. Das, S. Eachempati, A. Mishra, V. Narayanan, and C. Das, "Design and evaluation of a hierarchical on-chip interconnect for next-generation CMPs," in *High Performance Comp. Arch.*, feb. 2009, pp. 175 –186.
[9] T. Oh, H. Lee, K. Lee, and S. Cho, "An analytical model to study optimal area breakdown between cores and caches in a chip multiprocessor," in *ISVLSI '09*, may 2009, pp. 181 –186.
[10] A. Cassidy, K. Yu, H. Zhou, and A. Andreou, "A high-level analytical model for application specific CMP design exploration," in *Design, Automation Test in Europe*, march 2011, pp. 1 –6.
[11] M. Monchiero, R. Canal, and A. Gonzalez, "Power/performance/thermal design-space exploration for multicore architectures," *Parallel and Distributed Systems*, vol. 19, no. 5, pp. 666 –681, may 2008.
[12] Y. Li, B. Lee, D. Brooks, Z. Hu, and K. Skadron, "CMP design space exploration subject to physical constraints," in *High-Performance Computer Architecture*, feb. 2006, pp. 17 – 28.
[13] R. E. Matick, T. J. Heller, and M. Ignatowski, "Analytical analysis of finite cache penalty and cycles per instruction of a multiprocessor memory hierarchy using miss rates and queuing theory," *IBM J. Res. Dev.*, vol. 45, pp. 819–842, November 2001.
[14] U. Ogras, P. Bogdan, and R. Marculescu, "An analytical approach for network-on-chip performance analysis," *Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, pp. 2001 –2013, dec. 2010.
[15] Y. Ben-Itzhak, I. Cidon, and A. Kolodny, "Delay analysis of wormhole based heterogeneous NoC," in *NOCS'11*, may 2011, pp. 161 –168.
[16] S. Foroutan, Y. Thonnart, R. Hersemeule, and A. Jerraya, "An analytical method for evaluating network-on-chip performance," in *Design, Automation Test in Europe*, march 2010, pp. 1629 –1632.
[17] P. Bogdan and R. Marculescu, "Non-stationary traffic analysis and its implications on multicore platform design," *Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, pp. 508 –519, april 2011.
[18] A. Hartstein, V. Srinivasan, T. Puzak, and P. Emma, "On the nature of cache miss behavior: is it square root of 2," *Journal of Instruction-Level Parallelism*, vol. 10, 2008.
[19] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers, Inc., 2003.
[20] A. R. Alameldeen, "Using compression to improve chip multiprocessor performance," Ph.D. dissertation, 2006.
[21] J. L. Hennessy and D. A. Patterson, *Computer Architecture, 4th Edition: A Quantitative Approach*. Morgan Kaufmann Publishers Inc., 2006.
[22] S. Eyerman, L. Eeckhout, T. Karkhanis, and J. E. Smith, "A mechanistic performance model for superscalar out-of-order processors," *ACM Trans. Comput. Syst.*, vol. 27, pp. 1–37, May 2009.
[23] "MATLAB," http://www.mathworks.com.
[24] R. Burden and D. Faires, *Numerical Analysis*. Brooks Cole, 2010.
[25] L. Kleinrock, *Queueing Systems, Volume 1*. Wiley-Interscience, 1975.
[26] G. Wood, "The bisection method in higher dimensions," *Math. Program.*, vol. 55, June 1992.
[27] G. S. Fishman, "Grouping observations in digital simulation," *Management Science*, vol. 24, pp. 510–521, 1978.
[28] K. Olukotun, *Chip Multiprocessor Architecture: Techniques to Improve Throughput and Latency*. Morgan and Claypool Publishers, 2007.