Determining Optimal Coherency Interface for Many-Accelerator SoCs Using Bayesian Optimization

Kshitij Bhardwaj, Marton Havasi, Yuan Yao, David M. Brooks, José Miguel Hernández Lobato, and Gu-Yeon Wei

Abstract—The modern system-on-chip (SoC) of the current exascale computing era is complex. These SoCs not only consist of several general-purpose processing cores but also integrate many specialized hardware accelerators. Three common coherency interfaces are used to integrate the accelerators with the memory hierarchy: *non-coherent, coherent with the last-level cache (LLC)*, and *fully-coherent*. However, using a single coherence interface for all the accelerators in an SoC can lead to significant overheads: in the non-coherent model, accelerators directly access the main memory, which can have considerable performance penalty; whereas in the LLC-coherent model, the accelerators access the LLC but may suffer from performance bottleneck due to contention between several accelerators; and the fully-coherent model, that relies on private caches, can incur non-trivial power/area overheads. Given the limitations of each of these interfaces, this paper proposes a novel *performance-aware hybrid coherency interface*, where different accelerators use different coherency models, decided at design time based on the target applications so as to optimize the overall system performance. A new *Bayesian optimization based framework* is also proposed to determine the optimal hybrid coherency interface, i.e., use machine learning to select the best coherency model for each of the accelerators in the SoC in terms of performance. For image processing and classification workloads, the proposed framework determined that a hybrid interface achieves up to 23 percent better performance compared to the other 'homogeneous' interfaces, where all the accelerators use a single coherency model.

Index Terms—System-on-chip (SoC), hardware accelerators, coherence protocols, Bayesian optimization

1 Introduction

THE modern system-on-chip (SoC) comprises of many general-purpose processors and specialized hardware accelerators. To efficiently support several highly-demanding workloads simultaneously, such as image classification, speech recognition, and biometric security technology, the SoCs consist of many heterogeneous accelerators, designed for different applications. Some of the recent examples of high-performance and low-power many-accelerator SoCs include: Nvidia Tegra, Apple 'A' series, and Samsung Exynos. These accelerators are often *loosely coupled*, located outside the processor core, rather than *tightly coupled*, which are integrated into the CPU core's pipeline [4].

Three common coherency interfaces are used to integrate accelerators with the memory hierarchy in a loosely-coupled architecture: *non-coherent*, *coherent with the last-level cache (LLC-coherent)*, and *fully-coherent* [6]. In a non-coherent interfacing model, an accelerator typically uses a private scratchpad memory for local storage and uses software-managed direct memory access (DMA) to request data from the main memory. In LLC-coherent, the DMA is performed to the last-level cache rather than the main memory. Finally, a fully-coherent model involves each accelerator using its own private cache, which implements a cache coherence protocol such as MESI or MOESI, similar to a processor's cache.

There are interesting cost trade-offs involved with each of the three coherence models for accelerator-based SoCs. While the non-coherent model is simple and involves minimal hardware overhead, it also requires significant software management and can suffer from performance/power overheads due to the costly main memory accesses that can span over several cycles. LLC-coherent, on the other hand, can be more power and performance-efficient than non-coherent in case of workloads with a high LLC hit rate. However, the performance of the LLC-coherent model can suffer with increased contention when several accelerators are operating concurrently and during irregular data access patterns that can increase LLC misses. The fully-coherent model requires little software management but can incur significant hardware area/power overhead.

Given the modern SoCs integrate several heterogeneous accelerators to support a multitude of applications, a single coherence interface used by all the accelerators for different applications may not be the most optimal in terms of power and performance [6], [7]. As an alternative, the coherency interface for each accelerator in such SoCs can be selected based on the workloads, with different accelerators using different coherency models. To determine such optimal coherency models for these accelerators, this paper makes the following contributions.

As a first contribution, the paper proposes a novel *performance-aware hybrid coherency interface* for heterogeneous many-accelerator SoCs. In this hybrid interface, a subset of the accelerators can use one type of coherency model, such as non-coherent, while the others use a different model, for example LLC-coherent, depending on which options optimize the overall system performance. The best interface for each accelerator, in terms of performance, is decided at design time based on the targeted applications, and therefore avoids any overheads that can come with an alternative approach of run-time selection of the coherency models: both added hardware costs and performance penalties.

The second contribution is a new framework to determine the optimal coherency interface for a given SoC and a set of targeted applications. This framework integrates *Bayesian optimization*, which is a machine learning based approach to optimize black-box functions efficiently [5], with the *gem5-Aladdin* architectural simulator that supports modeling of complex heterogeneous many-accelerator SoCs [12]. The proposed framework is called *BayesOpt-gem5-Aladdin*. For a given SoC with different accelerators and a target workload, the Bayesian optimization algorithm intelligently selects the appropriate coherency model for each accelerator so as to optimize its performance, which is obtained by running the workload on the modeled SoC in the gem5-Aladdin simulator.

Finally, to show the effectiveness of the new BayesOpt-gem5-Aladdin approach, this framework was used to determine the optimal coherency interface for a complex SoC with accelerators for image processing. For each accelerator, two possible coherence models are considered: non-coherent or LLC-coherent (fully-coherent is not considered due to its hardware/performance overheads). For this SoC, the BayesOpt-gem5-Aladdin framework determined that *a hybrid coherency interface is the most optimal*, showing up to 23 percent better accelerator performance than the other 'homogeneous' interfaces where all the accelerators are either non-coherent or LLC-coherent.

2 Background

There are two important threads relevant to this paper: the gem5-Aladdin architectural simulator and the Bayesian optimization approach.

- *K. Bhardwaj, Y. Yao, D.M. Brooks, and G.-Y. Wei are with Harvard University, Cambridge, MA 02138. E-mail: kbhardwaj@g.harvard.edu, yuanyao@seas.harvard.edu, {dbrooks, guyeon}@eecs.harvard.edu.*
- *M. Havasi and J.M. Hernández Lobato are with the University of Cambridge, Cambridge CB2 1TN, United Kingdom. E-mail: {mh740, jmh233}@cam.ac.uk.*

Manuscript received 2 Feb. 2019; revised 28 Feb. 2019; accepted 16 Mar. 2019. Date of current version 17 Sept. 2019.
(Corresponding author: Kshitij Bhardwaj.)
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/LCA.2019.2910521

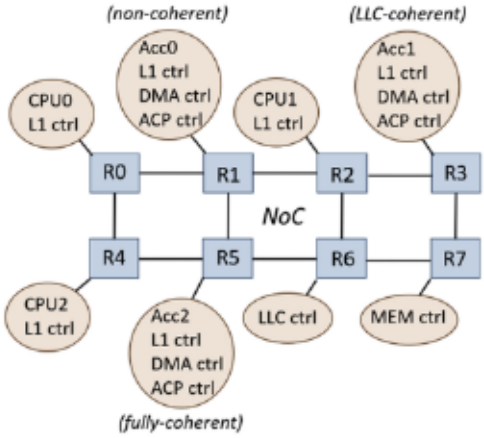

Fig. 1. An example SoC tiled architecture in gem5-Aladdin.

2.1 gem5-Aladdin Architectural Simulator

gem5-Aladdin models interactions within an SoC: between different accelerators, processors and the memory hierarchy [12]. This framework integrates the well-known gem5 architectural simulator [3] with the Aladdin accelerator simulator [11].

In gem5-Aladdin, a user program running on a CPU (in gem5) can invoke an accelerator through an *ioctl* system call. The accelerators are accurately modeled in Aladdin without the need to generate RTL. Aladdin is a trace-based simulator that profiles the dynamic execution of a program and constructs a data dependence graph. The vertices of this graph are LLVM IR instructions and the edges mark the true dependencies between the instructions. Various design optimizations, such as pipelining and loop unrolling, can then be applied, followed by scheduling the graph for execution. When the invoked accelerator finishes, it writes to a shared pointer between the CPU and the accelerator to signal the end of execution. All three coherence models for the accelerators can be implemented in gem5-Aladdin: (i) non-coherent: gem5's DMA engine is used by an accelerator's local scratchpad to access the main memory; (ii) LLC-coherent: an accelerator coherency port (ACP) [2] is implemented, where the accelerator's scratchpad can directly access the LLC through an ACP controller. To model this interface, gem5's MESI protocol is augmented with the ACP controller; and (iii) fully-coherent: each accelerator can use its private cache instead of a scratchpad memory and implements coherence using the gem5's MESI protocol.

Fig. 1 shows an example SoC that can be modeled using gem5-Aladdin. The system is a tiled architecture, with tiles corresponding to the processors, accelerators, and the LLC/memory controllers, connected using a network-on-chip (NoC). Each processor tile consists of a processing element and a private L1 cache controller. The accelerator tile comprises the accelerator compute unit and the L1/ACP/DMA controllers.

2.2 Bayesian Optimization: An Overview

Bayesian optimization (Bayes-Opt) is a machine learning based method that has been shown to be highly-effective for optimizing black-box functions [5]. These functions are expensive to evaluate, lack simple structures like concavity or linearity, and cannot be expressed as closed-form expressions. Bayes-Opt has also been shown to be more effective than other heuristic optimization approaches such as genetic algorithms, especially for accelerator design space exploration, in terms of faster convergence and quality of solutions [10].

Bayes-Opt starts with some initial sampling by evaluating the objective function at random inputs, followed by intelligently selecting those values that will optimize the objective. The algorithm first builds a Bayesian statistical model of the objective function using the initial sampling. Typically, a Gaussian process (GP) is used for this modeling that provides a probability distribution describing the potential values for the objective function ($f(x)$) at some candidate input x [13]. This Gaussian process is updated as the algorithm proceeds and samples new values. The selection of the new input samples is determined by the evaluation of an *acquisition function* [5]: the principle behind using acquisition function is to replace the original complex optimization problem by a simpler optimization problem, using a function which is much cheaper to evaluate than the original objective. The acquisition function evaluates the desirability of evaluating f at a candidate input x, and therefore, the selected input is the one that optimizes the acquisition function.

Algorithm 1. Bayesian Optimization Pseudo-Code

Step 1: Construct a Gaussian process prior on f
Step 2: Evaluate f at random n_0 points
while $n \leq N$ do
  Step 3: Update the posterior probability distribution on f
  Step 4: Select x_n that optimizes the acquisition function, computed using the current posterior distribution
  Step 5: Evaluate $f(x_n)$
  Step 6: Increment n
Step 7: Return the solution with the best $f(x)$

Algorithm 1 presents the pseudo-code for the Bayesian optimization method. The algorithm starts by constructing an initial GP model of the objective function f (i.e., construct a GP prior). In step 2, the function f is evaluated for n_0 random input values. The next steps of the technique are iterative and are performed $N - n_0$ times, where N is the budget on the number of function evaluations. In step 3, using the available data, the posterior probability distribution on f is updated. In step 4, a new input sample x_n is selected, which optimizes the acquisition function, evaluated using the current posterior distribution. In step 5, the original objective is evaluated at this point x_n, followed by incrementing the iteration counter. Once all the iterations are complete, the solution with the best (minimum or maximum) f is returned.

In this paper, *lower confidence bound (LCB)* is used as the acquisition function to minimize the main objective function f [13]. This acquisition function prefers not only the points that lead to small f but also those points, where f is uncertain. The former is called *exploitation* and the latter is referred to as *exploration*. The balance between exploitation and exploration avoids getting stuck in local minima or maxima, and hence improves the quality of the optimal solution. The LCB function is formulated as below

$$\alpha_{LCB}(x, \beta) = \mu(x) - \beta\sigma(x). \tag{1}$$

In Equation (1), μ and σ are the mean and variance of the GP distribution, and β is a tunable trade-off parameter to balance exploitation and exploration. To achieve a global minimum, the points preferred are those that not only minimize f (small μ) but also where f is more uncertain (larger σ).

3 Optimal Coherence Interface Selection

The proposed hybrid coherency interface model is presented in this section, along with a novel Bayesian optimization based framework to determine the optimal coherency interface for each accelerator of an SoC.

3.1 A Performance-Aware Hybrid Coherency Interface

This paper proposes a novel *performance-aware hybrid coherency interface* for heterogeneous many-accelerator SoCs. In this interface, different accelerators use different coherency models, determined

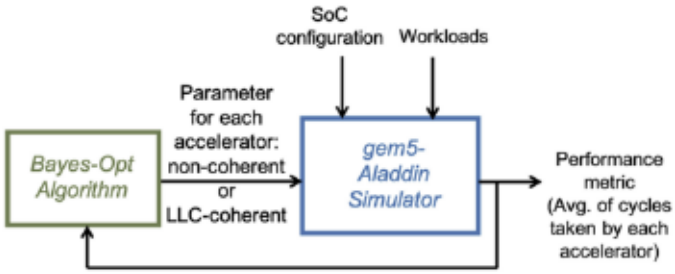

Fig. 2. Proposed BayesOpt-gem5-Aladdin framework.

TABLE 1
SoC Micro-Architecture Configuration

Component	Parameters
CPU Core	Out-of-order X86 @2.5 GHz 8-μop issue width, 192-entry ROB
L2 Cache (LLC)	2 MB, 16-way, LRU
DRAM	LP-DDR4, @1600 MHz, 4 GB, 4 channels, 25.6 GB/s
NoC	4×4 2D-mesh topology, link bandwidth: 128 bits, single-cycle routers and links, 4 VCs per input port

at design time so as to optimize the overall performance. For example, a subset of the accelerators can be non-coherent, while others LLC-coherent or fully-coherent, depending on which options lead to the best performance for the accelerators. In such a system, the expensive main memory accesses are minimized, while also reducing the contention at the LLC, maximizing the overall performance of the accelerators. A system with different coherence interfaces works correctly as follows: (i) in case of non-coherent, CPU flushes the caches before invoking the accelerator, and also after the accelerator writes the data to the memory so as not to read any stale data; (ii) in case of fully-coherent, the accelerators follow the same coherence protocol as the CPUs (MESI in this case); and (iii) in case of LLC-coherent, the request will be sent to the LLC and hence the LLC controller will ensure correctness on behalf of the accelerator. More details can be found in [12].

As a relevant related work, Spandex is a wrapper coherence protocol that has been introduced to integrate heterogeneous devices (CPUs, accelerators, GPUs), where each uses a different coherence protocol, in a correct and efficient way [1]. Our work, on the other hand, is proposing a hybrid coherence interface where the best coherence interface for each accelerator is selected so as to achieve the optimal overall system performance. Currently, if a fully-coherent interface is selected for an accelerator, it is assumed to use the same coherence protocol as the CPUs use (i.e., MESI coherence). However, the proposed approach is general, and can be extended to devices with different coherence protocols. In this case, a Spandex-like wrapper will be required for the correct operation. Also, while Spandex has been shown to achieve better performance than MESI-based protocols for heterogeneous SoCs, it can still incur some hardware overheads due to the additional custom translation units corresponding to each device. For simplicity, MESI protocol is assumed in this paper.

3.2 BayesOpt-gem5-Aladdin Framework

Determining the best coherency interface for each accelerator in a complex SoC, with several heterogeneous accelerators, is non-intuitive and a challenging problem. The performance of each accelerator is affected not only by its computation time and memory access patterns but also due to contention with other accelerators for shared resources. Such characteristics are governed by the coherency models used by the accelerators and are hard to predict at design time.

As shown in Fig. 2, a framework combining Bayesian optimization with gem5-Aladdin is proposed to determine the optimal coherence interface for each accelerator in an SoC. The gem5-Aladdin simulator, as discussed in Section 2.1, can model complex SoCs with several accelerators that can use different coherence models, and run various workloads concurrently. The complete SoC can be described using a 'config file' that is input to the simulator, which includes the configurations of processors, accelerators, memory/caches, and the interconnect. Bayesian optimization, as discussed in Section 2.2, is used to select the best coherence model for each accelerator: non-coherent or LLC-coherent. Fully-coherent (FC) model is not considered due to its significant hardware and performance overheads. In terms of performance, the FC model may not be suitable for some accelerators such as for DNN, as these accelerators access large amount of data (for weights and inputs/outputs), which cannot fit in the L1 caches and can therefore lead to significant cache misses, penalizing the overall performance. In addition, DNN accelerators benefit from using scratchpad memory as opposed to private caches due to their regular, streaming data access patterns. Therefore, to prune the search space and faster convergence, the FC model was not included. Finally, the objective function is to minimize the average (geometric mean) of the cycles taken by these accelerators to execute their respective kernels, obtained from gem5-Aladdin for a given application.

In this framework, the Bayesian optimization algorithm first constructs a Gaussian distribution model for the average performance metric (f in Algorithm 1), following steps 1-2. Sets of randomly selected coherence models for different accelerators (i.e., n_0, typically 10 sets) and the corresponding evaluated performances are used to generate this initial GP model. Next, steps 3-6 of Algorithm 1 are followed: during the given budget of N iterations, Bayes-Opt selects those sets of coherence models for the accelerators (x_n) that minimize the LCB acquisition function (Equation (1), evaluated using the current posterior distribution); these coherence models are then used by gem5-Aladdin to evaluate performances, which in turn are used to update the GP model used by Bayes-Opt. After all the iterations are complete, the set of optimal coherence models for different accelerators that achieves the minimum average cycles is selected (step 7).

4 Experimental Results

An interesting set of experiments is performed to show the effectiveness of the BayesOpt-gem5-Aladdin framework.

4.1 Experimental Setup

For an exhaustive evaluation, two different homogeneous interfaces, where all the accelerators in a SoC use a single coherency model, are compared with two different hybrid interfaces where the accelerators use varying coherency models. The homogeneous coherency interfaces are: (i) *All-LLC-Coherent* and (ii) *All-Non-Coherent*. The hybrid coherency interfaces are: (a) *Alternate-Hybrid:* one possible hybrid interface solution that achieves high performance but may not be the best, selected manually through a set of random evaluations, where each accelerator is assigned LLC-coherent or non-coherent randomly, and (b) *Bayes-Opt-Hybrid:* the hybrid interface selected by the BayesOpt-gem5-Aladdin framework, i.e., each accelerator can be either non-coherent or LLC-coherent, depending on which option best optimizes its performance.

The SoC configuration used for evaluation, modeled in gem5-Aladdin, is shown in Table 1. Similar to Fig. 1, the system is a tiled architecture comprising: 7 CPU tiles and 7 accelerator tiles, along with the L2 cache controller and main memory controller tiles, connected using a 4×4 NoC. The processors execute different workloads concurrently, offloading various kernels to the appropriate accelerators, effectively leading to all the accelerators operating in parallel.

As shown in Table 2, the image processing and classification workloads are used for evaluation. For image processing

TABLE 2
Targeted Workloads

Workload	Description
fft-transpose	512-point 1D complex FFT
stencil-3D	48 chained 3D second-order stencil, $32 \times 32 \times 32$ input
spmv-crs	Sparse matrix-vector multiply, 5 copies of 2048×512 matrix
Lenet5-CNN	5 layers: 2 convolutional (3×3), 1 pooling, 2 fully-connected, 1.2 MB weights, 98% accuracy on MNIST

workloads, three MachSuite benchmarks are used: *fft-transpose, stencil-3D*, and *sparse matrix vector multiply (spmv)* [9]. For image classification task, a *LeNet5 convolutional neural network (CNN)* is used on MNIST dataset [8]. The modeled SoC in gem5-Aladdin consists of one Lenet5-CNN accelerator, and two instances of each of the image processing accelerators.

Since there are 7 accelerators, the input space for Bayesian optimization has 7 dimensions. In this study, each accelerator can be either non-coherent or LLC-coherent, hence a total of 128 possible configurations. A single-objective function is used for optimization: geometric mean of the cycles taken by different accelerators to execute their respective workloads. While a design space of 128 configurations is considered, the proposed framework can handle a larger design space with 1000s of possible configurations.

4.2 Results

Fig. 3 shows the performance results for various accelerators in the SoC using different coherency interfaces. All-Non-Coherent interface performs the worst due to a large number of cycles spent to fetch data from the main memory. All-LLC-Coherent achieves up to 23.6 percent better performance than all non-coherent case as the memory requests are first sent to the LLC, and in case of the LLC hits, results in much shorter access latency. The hybrid solutions, on the other hand, achieve better performance than these homogeneous cases. One possible hybrid interface, manually selected through some random evaluations (alternate-hybrid), with FFT-1/SPMV-0/Stencil-0 as non-coherent and rest as LLC coherent, is able to achieve up to 23.2 percent better performance than All-Non-Coherent and up to 12.4 percent than All-LLC-Coherent. However, this hybrid case also shows a degradation of 3.8 percent for Stencil-0 over the all LLC-coherent case: while using some accelerators as non-coherent reduces contention at the LLC and can achieve better performance for the LLC-coherent accelerators, it can also incur latency overheads for the non-coherent accelerators due to the contention at the memory controller and the long access latency. The Bayes-Opt-Hybrid selected by the proposed framework, with only SPMV-1 as non-coherent, achieves the best performance with up to 22.7 percent (CNN-Lenet5) and 14.5 percent (SPMV-0) improvements over All-Non-Coherent and All-LLC-Coherent, respectively, without any degradations. Note that in this hybrid, selecting one of the SPMVs as non-coherent and the other as LLC-coherent minimizes contention and improves the overall performance.

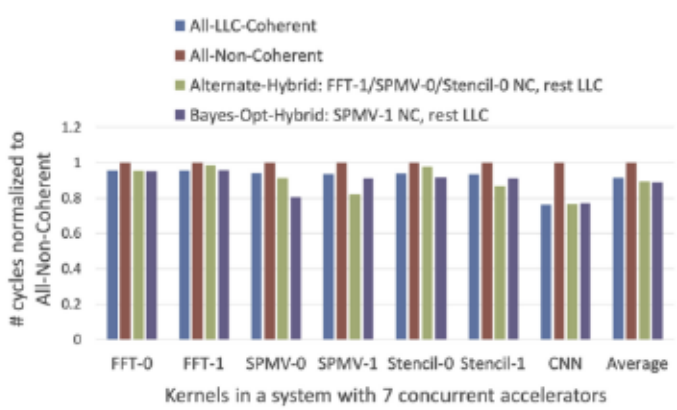

Fig. 3. Performance results for different coherency interfaces.

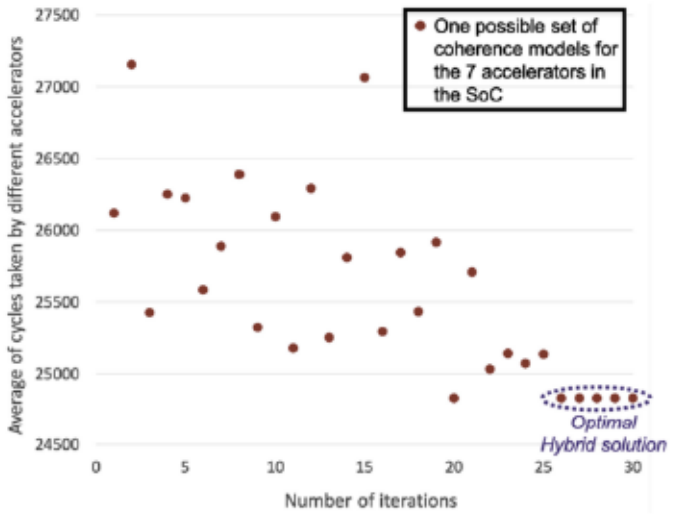

Fig. 4. Convergence of Bayesian optimization.

As shown in Fig. 4, the BayesOpt-gem5-Aladdin framework used only 20 iterations (in addition to the 10 random samples) to converge on this optimal solution. Therefore, while there are 128 possible configurations, only 30 evaluations were performed. Each iteration took ~ 1 hour, where majority of the time was dominated by the gem5-Aladdin run. Note that Bayesian optimization balances exploitation (i.e., evaluating points that minimize the objective) with exploration, evaluating uncertain points, sometimes with significantly worse performance, to avoid any local minima. Finally, as part of the validation, the most optimal solution from Bayes-Opt was confirmed against the best coherence interface obtained from a manual run of all possible 128 configurations.

To summarize, a hybrid coherency interface model can achieve significantly better performance than the homogeneous models for many-accelerator SoCs. The results also demonstrate that the proposed BayesOpt-gem5-Aladdin framework is able to converge to the best hybrid solution in a small number of iterations.

5 Conclusion and Future Work

The paper proposes a novel performance-aware hybrid coherency interface for many-accelerator heterogeneous SoCs. A new Bayesian optimization based framework is proposed in conjunction with the gem5-Aladdin simulator to determine the optimal hybrid coherency interface in terms of performance. For image processing workloads, the proposed framework determined that a hybrid interface achieves significantly better performance than the other 'homogeneous' interfaces. As a future work, a more exhaustive design space exploration will be performed considering other parameters as well in the BayesOpt-gem5-Aladdin framework, such as scratchpad size, L2 cache size, NoC parameters, etc.

Acknowledgments

This work was supported in part by the U.S. Government, under the DARPA DSSoC program. This work was also supported in part by the Semiconductor Research Corporation, NSF grant # CNS-1718160, and Intel. We are also thankful to Michael Kishinevsky, Pietro Mercati, and Raid Ayoub at Intel Labs, Hillsboro, OR, for their helpful feedback on this research.

REFERENCES

[1] J. Alsop, M. D. Sinclair, and S. V. Adve, "Spandex: A flexible interface for efficient heterogeneous coherence," in *Proc. 45th Annu. Int. Symp. Comput. Archit.*, 2018, pp. 261–274.

[2] A. Powell and D. Silage, "Statistical performance of the ARM cortex A9 accelerator coherency port in the Xilinx Zynq SoC for real-time applications,"*ReConFig*, pp. 1–6, 2015.

[3] N. L. Binkert, et al., "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, pp. 1–7, 2011.

[4] E. G. Cota, P. Mantovani, G. D. Guglielmo, and L. P. Carloni, "An analysis of accelerator coupling in heterogeneous architectures," in *Proc. 52nd ACM/EDAC/IEEE Des. Autom. Conf.*, 2015, pp. 202:1–202:6.

[5] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proc. IEEE*, vol. 104, no. 1, pp. 148–175, 2016.

[6] D. Giri, P. Mantovani, and L. P. Carloni, "Accelerators and coherence: An SoC perspective," *IEEE Micro*, vol. 38, no. 6, pp. 36–45, Nov./Dec. 2018.

[7] D. Giri, P. Mantovani, and L. P. Carloni, "NoC-based support of heterogeneous cache-coherence models for accelerators," in *Proc. 12th IEEE/ACM Int. Symp. Netw.-Chip*, 2018, pp. 1:1–1:8.

[8] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[9] B. Reagen, R. Adolf, Y. S. Shao, G.-Y. Wei, and D. M. Brooks, "MachSuite: Benchmarks for accelerator design and customized architectures," in *Proc. IEEE Int. Symp. Workload Characterization*, 2014, pp. 110–119.

[10] B. Reagen, J. M. Hernandez-Lobato, R. Adolf, M. Gelbart, P. Whatmough, G.-Y. Wei, and D. M. Brooks, "A case for efficient accelerator design space exploration via Bayesian optimization," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Des.*, 2017, pp. 1–6.

[11] Y. S. Shao, B. Reagen, G.-Y. Wei, and D. M. Brooks, "Aladdin: A pre-RTL, power-performance accelerator simulator enabling large design space exploration of customized architectures," in *Proc. ACM/IEEE 41st Int. Symp. Comput. Archit.*, 2014, pp. 97–108.

[12] Y. S. Shao, S. L. Xi, V. Srinivasan, G.-Y. Wei, and D. M. Brooks, "Co-designing accelerators and SoC interfaces using gem5-Aladdin," in *Proc. 49th Annu. IEEE/ACM Int. Symp. Microarchitecture*, 2016, pp. 48:1–48:12.

[13] N. Srinivas, A. Krause, S. Kakade, and M. Seeger, "Gaussian process optimization in the bandit setting: No regret and experimental design," in *Proc. 27th Int. Conf. Mach. Learn.*, 2010, pp. 1015–1022.