HAL
archives-ouvertes.fr

Design Exploration for next Generation High-Performance Manycore On-chip Systems: Application to big.LITTLE Architectures

Anastasiia Butko, Abdoulaye Gamatié, Gilles Sassatelli, Lionel Torres, Michel Robert

▶ **To cite this version:**

Anastasiia Butko, Abdoulaye Gamatié, Gilles Sassatelli, Lionel Torres, Michel Robert. Design Exploration for next Generation High-Performance Manycore On-chip Systems: Application to big.LITTLE Architectures. ISVLSI: IEEE Computer Society Annual Symposium on VLSI, Jul 2015, Montpellier, France. pp.551-556, 2015, <10.1109/ISVLSI.2015.28>. <lirmm-01255927>

HAL Id: lirmm-01255927
https://hal-lirmm.ccsd.cnrs.fr/lirmm-01255927

Submitted on 14 Jan 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Design Exploration For Next Generation High-Performance Manycore On-chip Systems: Application To big.LITTLE Architectures

Anastasiia Butko, Abdoulaye Gamatié, Gilles Sassatelli, Lionel Torres and Michel Robert
LIRMM (CNRS and University of Montpellier), Montpellier, France
Email: {firstname.lastname}@lirmm.fr

***Abstract*—Next generation embedded systems will massively adopt on-chip manycore architectures to provide both performance and energy-efficiency. This trend will definitely establish the convergence of embedded computing and high-performance computing. In such a context, one major design challenge will concern the choice of adequate architecture parameters given system requirements. Moreover, it will affect the way applications can suitably exploit architecture resources for an efficient execution. This paper deals with manycore on-chip system design exploration by using via simulation. It presents an approach enabling one to study central design parameters in an accurate and cost-effective manner. This approach is illustrated through the design exploration for ARM big.LITTLE heterogeneous multicore technology in the gem5 framework.**

Keywords*—*High-performance computing, energy-efficiency, manycore, heterogeneous, big.LITTLE, modeling, gem5, trace-driven.

I. Introduction

The number of cores in future on-chip architectures will keep in increasing as already observed in state-of-the-art systems such as the Many Integrated Core Architecture (MIC) of Intel [1], TILE-Gx of Tilera [2], Multi-Purpose Processor Array (MPPA) of Kalray [3] and ARM big.LITTLE architecture [4]. MIC is the architecture adopted by Intel Xeon Phi coprocessors used as compute accelerators in the world's fastest supercomputer (Tianhe-2 [5]). It is composed of 61 cores interconnected by a bi-directional ring network. The TILE-Gx architecture is composed of 72 cores interconnected by a 2D mesh NoC using wormhole routing packets. MPPA is a manycore architecture integrates 256 cores, where cores are distributed across 16 compute clusters. The big.LITTLE technology promotes heterogeneous and adaptive architectures as illustrated in Samsung Exynos Octa 5410 and 5422 chips. It enables to dynamically migrate applications between two different clusters of ARM cores: a low-energy consumption cluster ("LITTLE") composed of four Cortex-A7 cores versus a high-performance cluster ("big") composed of four Cortex-A15 cores.

To illustrate the potential of on-chip big.LITTLE architecture, Table I reports energy-efficiency numbers measured on the Odroid XU3 board (see Figure 1) integrating the Exynos Octa 5422 chip. These numbers have been obtained by executing the high-performance Linpack benchmark to determine the number of floating-point operations per second (flops) and the corresponding power consumption, for different frequencies of the two clusters. Here, the configuration composed of four Cortex-A15 cores running at 800MHz appears as the most energy-efficient.

Thanks to their energy-efficiency, the above on-chip systems are significantly contributing to draw the current convergence of embedded computing and high-performance computing. Considering this trend, an important challenge concerns the design of adequate energy-efficient manycore systems that will successfully fill the requirements of both computing domains.

To address this challenge, relevant design exploration frameworks are required for cost-effectiveness reason. Candidate frameworks must be accurate enough to enable designers to address architecture features with respect to the way applications can suitably exploit resources for an efficient execution.

In this paper, we consider the gem5 architecture exploration framework for demonstrating an approach for modeling state-of-the-art multicore on-chip systems and exploring their scalability. We consider the big.LITTLE architecture as a base template by devising a model of the Exynos Octa 5422 clusters in gem5 full-system mode. We evaluate the accuracy of the resulting model via a subset of the Rodinia compute-intensive benchmark suite [6]. We argue that the accuracy of our models, which is around a precision error of 20% is relevant enough to show that the models capture in a consistent way system execution performed on real computer boards including Exynos Octa 5422 chip. In order to accelerate the model-based exploration for large-scale designs, we apply a trace-driven abstraction in gem5 [7] to the big.LITTLE architecture.

The rest of this paper is organized as follows: Section II discusses a few related gem5-oriented ARM core modeling studies. Then, Section III and IV respectively present our big.LITTLE architecture modeling in gem5 and an assessment of the modeling accuracy. Section V explores the design of large-scale system scenarios including more than hundred cores. Finally, concluding remarks are given in Section VI.

II. Related Work

In order to carry out our design exploration for manycore embedded systems, we consider the gem5 simulator system [8]. It is a quasi-cycle accurate simulation platform for computer system architecture research. The gem5 framework provides multiple architecture exploration features: system emulation and complete full-system simulation modes, different CPU models which represent in-order and out-of-order

TABLE I. EVALUATION OF HIGH-PERFORMANCE LINPACK ON THE ODROID XU3 BOARD.

A7@200MHz	A7@800MHz	A7@1.4GHz	A15@200MHz	A15@800MHz	A15@1.4GHz	A15@2GHz
Total board power (W)						
2.2	2.6	3.5	2.9	5.1	7.5	12.5
HPL score (Gflops)						
0.26	1.04	1.68	0.96	3.42	4.96	4.7
Energy-efficiency (Mflops/W)						
118	404	484	347	746	662	376

executions, a large set of ISAs (ALPHA, ARM, x86, SPARC, PowerPC, MIPS) and variety of memory configurations, e.g. cache coherence protocols, interconnects, memory controllers, etc. In the next paragraphs, we mainly discuss a few relevant studies on the use of gem5 for design analysis.

In an early study [9], we evaluated the accuracy of modeling real systems using gem5 simulator. By considering a range of benchmarks from scientific computing and media applications domains, we compared simulation results against a real hardware. The observed accuracy was promising enough to consider gem5 as an interesting architecture design exploration framework. Authors in In [10] design a gem5 model of CoreTile Express system-on-chip (SoC) and estimate the accuracy of Cortex-A15 core, memory system and interconnect. They deeply explore the micro-architectural simulation for the homogeneous dual-core system. The work presented in [11] deals with the modeling and simulation of Cortex-A8 and Cortex-A9 cores in gem5. A comparison in terms of execution time is achieved against a real hardware execution based on ten benchmarks. Authors claim that their core models are more accurate than similar micro-architectural simulators. A similar study has been acieved for Cortex-A7 and Cortex-A15 cores in [12] by focusing on the micro-architectural simulation of these cores. The gem5 and McPAT frameworks have been combined to validate area and energy/performance trade-offs against the published datasheet information. However, this work does not aim to multicore evaluation. It only demonstrates the difference between Cortex-A7 and Cortex-A15 cores running single-threaded applications. The current work rather focus on the multi- and manycore design exploration based on the same cores models.

III. MODELING OF A BIG.LITTLE ARCHITECTURE

We introduce the main features of the computer board integrating the considered big.LITTLE technology. Then, we describe the modifications applied to gem5 so as to infer a corresponding model that will serve later for design exploration.

A. Reference platform main features

To evaluate the accuracy of our big.LITTLE model we used the Odroid XU3 board with the embedded Exynos 5422 chip which is illustrated in Figure 1.

The Exynos 5422 processor includes two clusters known as “LITTLE” cluster with four Cortex-A7 in-order cores and “big” cluster with four Cortex-A15 out-of-order cores. In contrast to the previous chip versions, Exynos 5422 features the heterogeneous multiprocessing (HMP) solution also known as global task scheduling (GTS) thus all eight cores can run simultaneously. The LITTLE cluster supports 200MHz - 1.4GHz frequency range and the big cluster supports 200MHz

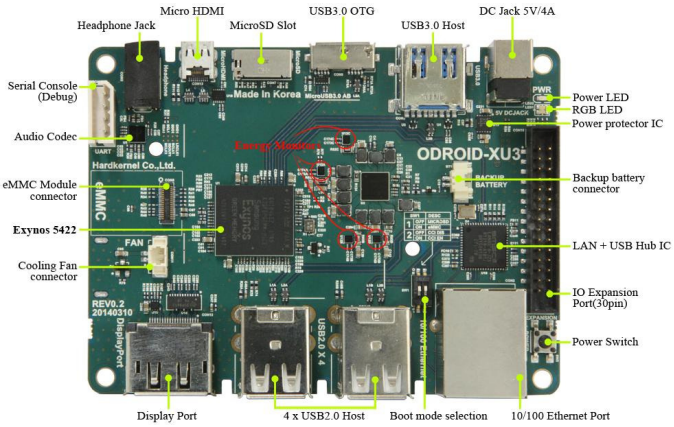

Fig. 1. Odroid XU3 board [13].

- 2GHz. Each core has its 32Kb private L1 data and instruction caches with 2-way associativity and 4ns latency. LITTLE and big clusters contain two 512kB and 2MB L2 caches which are shared among four cluster cores respectively. The cache coherency between them as well as with the memory is maintained by the Cache Coherence Interconnect 400 (CCI) [14]. The main memory is 2GB LPDDR3 RAM running at 933MHz and is integrated by package on package (PoP) method. It has two 32-bit channels and achieves 14.9GB/s memory bandwidth.

B. ARM big.LITTLE model

In order to simulate heterogeneous big.LITTLE architecture in symmetric multiprocessing (SMP), e.g. only big or LITTLE clusters, as well as in HMP modes we calibrated our core model according to the Odroid XU3 reference platform:

- The key feature of the reference big.LITTLE processor is the ability to run eight cores simultaneously. The actual gem5 version does not support ARM full-system simulation with more than four cores out of snoop control unit (SCU) implementation. We bypassed this restriction by modifying the SCU component through not masking the real core number. Another issue relates to the in-order CPU model which is not devised yet to the ARM ISA. This problem is often discussed in the research community and according to gem5 developers there are three solutions: (i) *TimingSimpleCPU* model, (ii) *MinorCPU* model and (iii) *DerivO3CPU* model which can be modified to produce quasi-in-order execution [11] [12]. In our experiments we evaluated all three scenarios. The last modifications related to the heterogeneous nature of the considered architecture are performed in the gem5

TABLE II. RODINIA BENCHMARK DESCRIPTION.

Application/Kernel	Abbreviation	Domain	Problem size
Back Propagation	backprop	Pattern Recognition	65536
Breadth-First Search	bfs	Graph Algorithms	4096
Heart Wall	heartwall	Medical Imaging	test.avi, 1 frame
HotSpot	hotspot	Physics Simulation	64 x 64
K-means openmp/serial	kmeans	Data Mining	100
Lower Upper Decomposition	lud	Linear Algebra	256
k-Nearest Neighbors	nn	Data Mining	42760
Needleman-Wunsch	nw	Bioinformatics	1024
Speckle Reducing Anisotropic Diffusion	srad v1	Image Processing	1 x 502 x 458
	srad v2		512 x 512

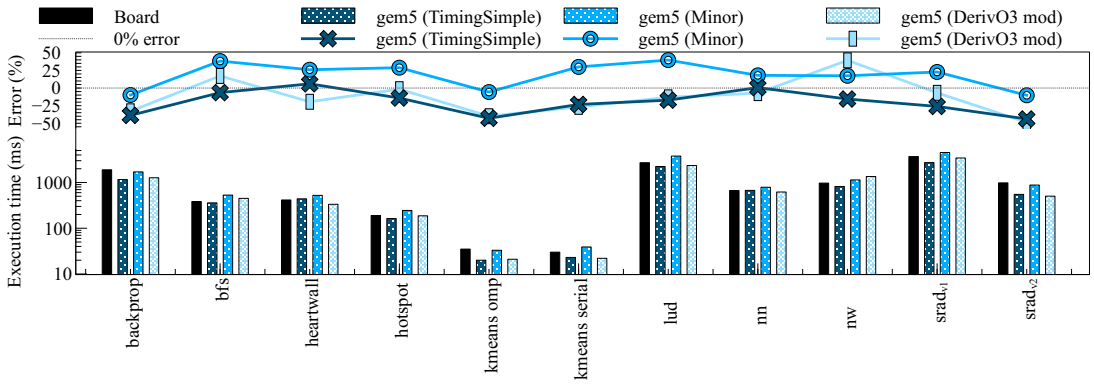

Fig. 2. Execution time comparison for LITTLE Cortex-A7 cluster running at 200MHz.

full-system creation script so as to support multiple CPU models throughout a simulation.

- To support multiple clocks for LITTLE and big clusters we changed the *fs.py* script.
- Since each cluster has its individually shared among the cores L2 cache we added new option that identifies the number of L2 caches as well as their parameters. As gem5 does not contain the ARM CCI-400 interconnect component we ensured the cache coherency by connecting L2 caches and memory via the coherent crossbar (*CoherentXBar* [15]).
- The recent gem5 version provides multiple DRAM controller models. Thus we chose LPDDR3 with two 32-bit channels. Note that the LPDDR3 timing corresponds to 800MHz frequency and not 933MHz.

IV. ACCURACY EVALUATION

Now we assess the accuracy of the previous big.LITTLE architecture model against the reference Odroid XU3 platform.

A. Rodinia benchmark suite

The reference Odroid XU3 board run Linux Kernel 3.10 which allows GTS. We modified the Linux Kernel 3.10 in order to run it on gem5 simulator.

The Rodinia benchmark suite for heterogeneous computing [6] is used to validate our big.LITTLE model. It contains twenty applications and kernels from different scientific domains which is parallelized with OpenMP for multicore CPUs and with CUDA API for GPUs. We used OpenMP implementation with four threads per each cluster. Also, the `GOMP_CPU_AFFINITY` variable is used to ensure identical thread scheduling on the board and on the gem5 system. The following eleven applications and kernels are chosen: *backprop*, *bfs*, *heartwall*, *hotspot*, *kmeans openmp/serial*, *lud*, *nn*, *nw*, *srad v1/v2*. The complete set of application description and problem size are presented in Table II.

B. Analysis

We explored three available options to model ARM in-order processor and to identify the accuracy of each one:

1) *TimingSimpleCPU* is the simplest purely functional in-order model which uses timing memory accesses.
2) *MinorCPU* is an in-order processor model with a fixed pipeline but configurable data structures and execute behavior. It supports the Fetch (1,2), Decode and Execute pipeline stages.
3) *DerivO3CPU* (modified) is the most complex out-of-order model which has Fetch, Decode, Rename, Issue/Execute/Writeback and Commit pipeline stages.

The comparative results are presented in Figure 2. Note that the scale for the execution time is logarithmic. The figure shows the execution time for eleven Rodinia applications and kernels executed on the Cortex-A7 cluster running at 200MHz on: (i) reference board, (ii) gem5 *TimingSimpleCPU* model, (iii) gem5 *MinorCPU* and (iv) modified gem5 *DerivO3CPU*. As we can see, the absolute error percentage varies between 1% and 50%. The minimum and maximum errors as well as the absolute average error for each scenario are listed in Table III. The results show that the execution time absolute error for

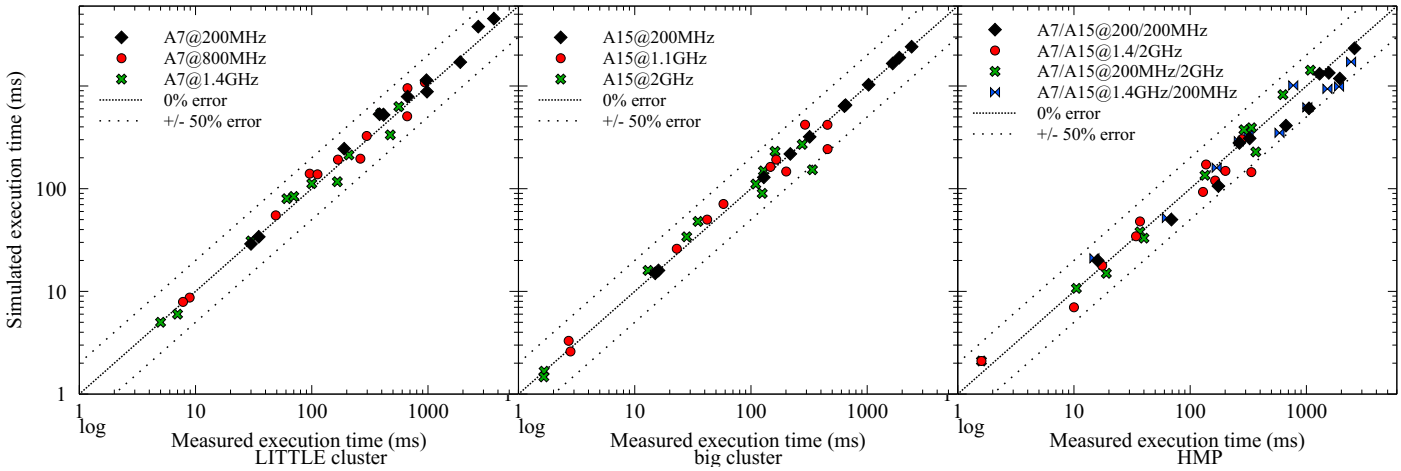

Fig. 3. Execution time comparison between gem5 model and Exynos Octa 5422.

all three models is around 22%. Thus, we conclude that for performance evaluation it is enough to use the *TimingSimpleCPU* model. However, for more detailed studies, such as micro-architectural or power consumption exploration, it is necessary to switch to a more detailed one.

TABLE III. CORTEX-A7 IN-ORDER MODEL EXECUTION TIME ERROR SUMMARY.

CPU model	Minimum error	Maximum error	Absolute average error
TimingSimpleCPU	0.6%	43.9%	21.4%
MinorCPU	5.7%	39.7%	22.5%
DerivO3CPU (mod)	2.2%	48.7%	21.6%

For further accuracy evaluation to simulate the in-order Cortex-A7 cores we selected the modified gem5 *DerivO3CPU* model. The following three scenarios are considered:

- Accuracy evaluation of the LITTLE Cortex-A7 cluster in SMP mode running at 200MHz, 800MHz and 1.4 GHz (LITTLE 1, 2, and 3 respectively).
- Accuracy evaluation of the big Cortex-A15 cluster in SMP mode running at 200MHz, 1.1GHz and 2GHz (big 1, 2, and 3 respectively).
- Accuracy evaluation of the big.LITTLE in HMP mode with Cortex-A7/A15 running at 200MHz/200MHz, 1.4GHz/2GHz, 200MHz/2GHz and 1.4GHz/200MHz (big.LITTLE 1, 2, 3 and 4 respectively).

The correlation results are shown in Figure 3. Each scenario has eleven points which correspond to the chosen Rodinia kernels and applications. Their execution time varies between milliseconds and seconds thus the scale is logarithmic. Two large-dotted lines show the -50% and 50% error edges. The comparative results, such as minimum/maximum and average absolute errors are presented in Table IV.

To summarize the average absolute errors are 19.3%, 20.1% and 22.9% for LITTLE cluster, big cluster and big.LITTLE in HMP mode respectively.

For a more detailed analysis we considered application output information over the time spent in different stages,

TABLE IV. EXECUTION TIME ERROR SUMMARY.

Scenario	Minimum error	Maximum error	Absolute average error
LITTLE 1	2.2%	48.7%	21.6%
LITTLE 2	2%	45.6%	19.7%
LITTLE 3	1%	38.7%	17%
big 1	2.7%	39.6%	17.9%
big 2	5.8%	46.7%	21.1%
big 3	0.9%	54.8%	21.4%
big.LITTLE 1	1.4%	43.2%	22.7%
big.LITTLE 2	0.8%	56.9%	23.3%
big.LITTLE 3	0.9%	37.8%	19.5%
big.LITTLE 4	2.4%	48.3%	26%

e.g. initial setup, I/O, kernel execution, etc. We chose three applications: *bfs*, *lud* and *srad v1*. The comparative results between reference measured and modeled Cortex-A15 cluster running at 1.1GHz are shown in Table V. We noticed that execution time precision error varies dramatically between 5% and 90% according to execution stages. The total execution time is compensated. We observed that throughout presented three examples the computation kernel stage error is low and amounts around 20%. At the same time, stages which are related to memory operations, e.g. *Store results*, *Read image from file*, *Save image into file*, etc., produce high error percentage. Thus, we conclude that the main source of error in our model is memory system. One of the reason is LPDDR3 model difference mentioned in Section III-B. Another possible cause is non-realistic cache coherence protocol used in classical memory model [15] as well as the lack of CCI-400 model. This observation can also explain the slight error raise by switching to the HMP mode as its memory communication become more complex and inaccurate cache coherence system provides a noticeable discrepancy.

V. ARCHITECTURE EXPLORATION

The presented big.LITTLE gem5 model allows us to explore important parameters as cache size, interconnect width, memory technology. However, due to current limitation of gem5 to simulate easily more than eight ARM cores, exploring large-scale ARM-based system models is not feasible. Thus to evaluate the scalability of the Rodinia benchmark running on

TABLE V. APPLICATION DIFFERENT STAGE COMPARISON.

Stage	Execution time (ms) Board	gem5	Error
	bfs		
Read graph	0.58	0.04	-93.7%
Allocate memory	3.7	4.5	21.5%
Kernel	33.7	41.5	23%
Store results	5.1	3.3	-35.1%
Total	44.0	49.6	12.7%
	lud		
Kernel	79.476	72.384	-8.9%
Verify	216.514	347.586	60.5%
Total	295.99	419.976	41.9%
	srad v1		
Initial setup	0.31	0.06	-79.4%
Read image from file	140.5	259.4	84.6%
Resize image	3.5	3.3	-5.4%
Allocate memory	0.12	0.09	-29.5%
Extract image	90.3	8.8	-90.3%
Compute	14.4	11.1	-22.9%
Compress image	38.8	23.3	-40%
Save image into file	170.8	111.1	-35%
Free memory	2.4	0.9	-62.3%
Total	461.1	418.5	-9.2%

the big.LITTLE heterogeneous manycore we used the trace-driven approach [7]. To demonstrate the exploration flow we chose *hotspot* application with 1024 problem size.

A. Trace-driven simulation

The trace-driven simulation consists on three phases: (i) collection, (ii) reduction and (iii) simulation. The last phase implies the replacement of gem5 full-system cores by trace injectors (TIs) which main goal is to replay the traces obtained in the collection phase. The key advantages of such trace-driven approach is the significant reduction of the simulation time and also the ability to replicate the traces in order to evaluate system scalability [7].

To collect the Cortex-A7 traces we used the *TimingSimpleCPU*. As we showed in Section IV-B this model is suitable for performance evaluation and in addition it provides well-organized traces where each request is always followed by a response. The Cortex-A15 trace-driven simulation is a tedious task. Its out-of-order nature at times complicates trace injections and requires extra micro-dependency analysis. To solve this issue we decided to emulate the Cortex-A15 behavior by using collected Cortex-A7 traces.

In Figure 4 we illustrated the *hotspot* kernel runtime behavior captured on the Odroid XU3 board with Scalasca/Score-p instrumentation [16] and analyzed with Vampir tool [17]. The figure represents execution of four threads under two Cortex-A7 and two Cortex-A15 cores running at the same frequency. Expectedly the Cortex-A15 duration is less than the Cortex-A7 corresponding to 0.16s and 0.23s respectively. Based on these values we calculated an acceleration factor as 1.45x and applied it for big cluster trace-driven simulation. Consequently, the acceleration factor varies from one application to another.

Trace replication technique [7] relies on overlapping trace patterns with the increasing number of TIs. The *hotspot* kernel consists on two stages:

1) Read input data stage is performed by master thread successively and takes 80% of total execution time.
2) Parallel region stage is executed on all available cores evenly and takes the remaining 20% of total execution time.

The percentage values shown above are taken from the Scalasca/Vampir profile and correlate with the published analysis [6]. To obtain the replication pattern we captured the parallel region traces presented in Figure 5. We illustrated the trace pattern collected at the core#0 (Figure 5 a)) and at the core#1 (Figure 5 b)) on the system with four cores and 4 threads. Each kernel iteration is composed on two `pragma omp parallel for`: (i) compute temperature and (ii) store results. We observed that the results storage region has a significant raise of cache miss number. The further exploration is focus on the parallel region evaluation only.

B. Results

Based on previous trace-driven design, we present the ARM big.LITTLE heterogeneous manycore scalability analysis. We evaluated three scenarios:

- LITTLE cluster with 4, 8, 16, 32, 64 and 128 cores (injectors),
- big cluster with 4, 8, 16, 32, 64 and 128 cores (injectors),
- big.LITTLE in HMP mode with 4/4, 8/8, 16/16, 32/32 and 64/64 cores (injectors).

The execution time and speedup for each scenario are presented in Figure 6. As we can see, the best execution time, as well as the speedup obviously shows big cluster. The LITTLE cluster provides the worst execution time. The big.LITTLE speedup is normalized by the faster big cluster. We observed that the execution time in HMP mode is worst than in the big cluster and slightly better than in the LITTLE cluster. It explained by the OpenMP programming nature that we observed in Figure 4 where faster Cortex-A15 cores wait when the slower Cortex-A7 cores terminate. For all three scenarios the speedup reaches the plateau around 64 cores (injectors). It explained by the memory/interconnect saturation.

To address this common issue we propose to explore the big.LITTLE architecture with alternative network-based *Ruby* memory subsystem [15]. System includes two-level cache hierarchy. The consistency of the memory is maintained by the MESI coherence protocol. This protocol models inclusion between the L1 and L2 caches and has four stable states, M, E, S and I, hence the name. The interconnection network has the following features: Mesh topology, XY routing algorithm

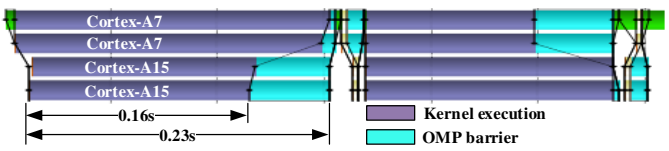

Fig. 4. Hotspot parallel region runtime behavior running on the Odroid XU3 board.

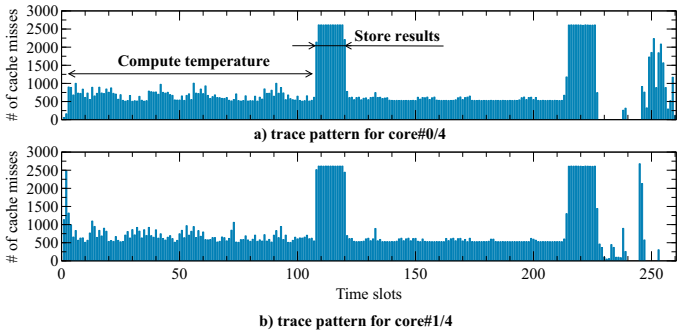

Fig. 5. Hotspot parallel region trace pattern.

and detailed GARNET network micro-architecture model (16-byte links, 10 virtual networks, 4 virtual channels per virtual network, 4 buffers per virtual channel, 1 cycle on-chip link latency). Figure 6 b) shows the achieved speedup for the LITTLE cluster (Ruby) up to 128 cores. Application shows a plateau which originates from saturation of the external memory bandwidth that according to the gem5 statistic file is about 200 million DDR accesses per second. *Hotspot* parallel region investigation shows that system scalability can be improved by efficient network interconnect on around 30% of execution time speedup.

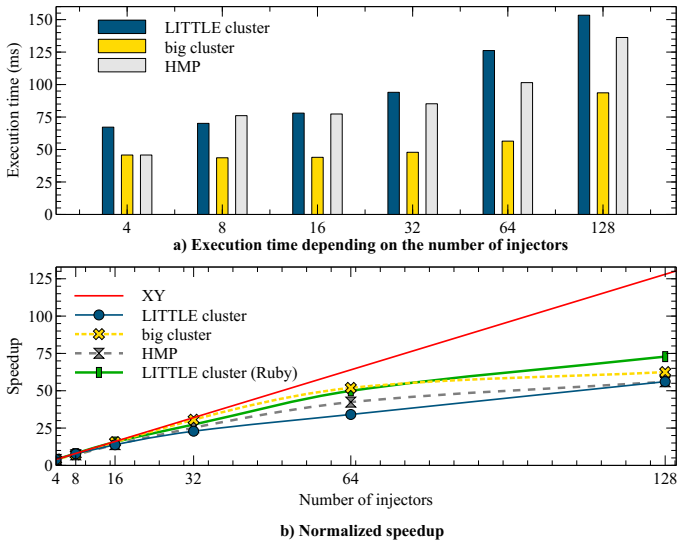

Fig. 6. Execution time and speedup evaluation.

VI. Conclusion

In this paper, we proposed ARM big.LITTLE cores models in the gem5 architecture exploration framework, which are accurate enough to enable a relevant design exploration of heterogeneous manycore systems. More precisely, models of Cortex-A7 and Cortex-A15 cores have been combined in full-system mode and evaluated against the Exynos Octa 5422 system-on-chip in order to assess the models accuracy. A reasonable precision error of about 20% has been obtained. Due to the limitation of gem5 full-system mode to simulate more than eight ARM cores, we applied our trace-driven approach [7] to explore the heterogeneous nature of big.LITTLE systems including more than one hundred cores. The scalability of such systems has been addressed and compared with homogeneous system configuration. All the study has been achieved by using a subset of the Rodinia compute-intensive benchmark suite [6].

Future work includes a more detailed analysis of out-of-order Cortex-A15 core in gem5. Then, traces could be generated using this model for replication and further architecture parameter exploration, e.g., cache and memory configuration.

Acknowledgment

The research leading to these results has received funding from the European Community Seventh Framework Programme (FP7/2013-2016) under the Mont-Blanc 2 Project: http://www.montblanc-project.eu, grant agreement n° 610402.

References

[1] Intel, "Many Integrated Core Architecture," 2015. [Online]. Available: http://www.intel.com

[2] EZchip, "TILE-Gx Multicore ," 2015. [Online]. Available: http://www.tilera.com

[3] Kalray, "Multi-Purpose Processor Array," 2015. [Online]. Available: http://www.kalrayinc.com

[4] ARM, "big.LITTLE Technology," 2015. [Online]. Available: http://www.arm.com

[5] TOP500, "TOP500 Supercomputer Sites," 2015. [Online]. Available: http://www.top500.org/

[6] S. Che, M. Boyer, J. Meng, D. Tarjan, J. Sheaffer, S.-H. Lee, and K. Skadron, "Rodinia: A benchmark suite for heterogeneous computing," in *Workload Characterization, 2009. IISWC 2009. IEEE International Symposium on*, Oct 2009, pp. 44–54.

[7] A. Butko, R. Garibotti, L. Ost, V. Lapotre, A. Gamatié, G. Sassatelli, and C. Adeniyi-Jones, "A trace-driven approach for fast and accurate simulation of manycore architectures," in *Design Automation Conference (ASP-DAC), 2015 20th Asia and South Pacific*, Jan 2015, pp. 707–712.

[8] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, Aug. 2011. [Online]. Available: http://doi.acm.org/10.1145/2024716.2024718

[9] A. Butko, R. Garibotti, L. Ost, and G. Sassatelli, "Accuracy evaluation of gem5 simulator system," in *Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), 2012 7th International Workshop on*, July 2012, pp. 1–7.

[10] A. Gutierrez, J. Pusdesris, R. Dreslinski, T. Mudge, C. Sudanthi, C. Emmons, M. Hayenga, and N. Paver, "Sources of error in full-system simulation," in *Performance Analysis of Systems and Software (ISPASS), 2014 IEEE International Symposium on*, March 2014, pp. 13–22.

[11] F. Endo, D. Courousse, and H.-P. Charles, "Micro-architectural simulation of in-order and out-of-order arm microprocessors with gem5," in *Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XIV), 2014 International Conference on*, July 2014, pp. 266–273.

[12] F. A. Endo, D. Couroussé, and H.-P. Charles, "Micro-architectural simulation of embedded core heterogeneity with gem5 and mcpat," in *Proceedings of the 2015 Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools*, ser. RAPIDO '15. New York, NY, USA: ACM, 2015, pp. 7:1–7:6. [Online]. Available: http://doi.acm.org/10.1145/2693433.2693440

[13] "Odroid-xu3," 2015. [Online]. Available: http://www.hardkernel.com

[14] *CoreLink CCI-400 Cache Coherent Interconnect Technical Reference Manual*, ARM, November 16 2012, revision r1p1.

[15] gem5, "Classic Memory System," 2015. [Online]. Available: http://www.m5sim.org

[16] "Scalasca," 2015. [Online]. Available: http://www.scalasca.org/

[17] "Vampir - performance optimization," 2015. [Online]. Available: https://www.vampir.eu/