Micro-architectural Adaptation of Codelets using gem5 and CERE

Nicolas Derumigny
ENS de Lyon

Pablo de Oliveira Castro
Université de Versailles Saint-Quentin-en-Yvelines

Abstract

In recent years, many efforts have been deployed in order to increase the speed and efficiency of computing architectures. Another step to increase their efficiency is to adapt themselves to the tasks they do. For example, 64-bit computational power is not as useful as 16-bit one for neuronal network, so specific hardware, faster for 16-bit calculus but slower on 64-bit ones are being developed.

Heterogeneous multicore system, widely spread on embedded computing, already use two CPU cluster (big.LITTLE technology): the big cluster handles heavy tasks, and the little one, more power-efficient but slower, is used the rest of the time. Sometimes the different regions of an application fit better on some architecture, depending of the type of data and the operations they are dealing with. One the big.LITTLE architecture, they can be transferred from one cluster to the other during its running.

The codelet approach, based on the use of the software Codelet Extractor and REplayer (CERE), allows to isolate and tune some sections of the application individually, showing which improvements on the hardware side leads to better performance.

This report presents an overview of the possibility of micro-architectural adaptation using the x86 instruction set simulated by gem5, based on an performance-consumption ratio, calculated with MCPAT. Its impact has been measured on both sequential and parallel applications on one simulated monocore CPU, using the NAS and PARSEC benchmark suites.

1 Introduction

1.1 Generalities

Developing a new CPU architectures is a compromise between several parameters. The common approach is to minimise the manufacturing cost and minimise the average execution time, measured on a suite of benchmark representative of the user's most frequent tasks. But in HPC, the needs depend mostly of the application run: for physicist calculus, a great precision is needed, so developing 128-bit compute units is needed. Whereas in deep learning machines, fast 16-bit computation is enough.

Thus, architectural adaptations could greatly improve the performance of computational unit executing only a fixed task.

The boost technology, increasing the frequency during high loads while the chip is under a given temperature (or does not reach a thermal threshold) can also be taken as an on-the-fly tuning of some chips to adapt to a given task. An other example of this specialisation is GPGPUs and CPUs: GPGPU are efficient in highly-scalable parallel applications, and CPUs are faster on sequential ones.

CERE is a software that extracts fro; an application some pieces of codes corresponding to a selected region, called codelets. Extracted codelets presents themselves as another application that can be run out-of-the-box and recreates the behaviour of the selected region.

The gem5 simulator[4] has been used to quantify micro-architecture impacts on softwares: it allows to fine-tune parameters without using different processors. As it emulates a whole linux system, the measurement can be really slow, that is why running only codelets gives a serious advantage over measuring the entire application. For example, running the IS codelet is four times faster in SE mode that the full application.

1.2 Experimental Protocol

Four x86 CPUs were simulated, one based on the Cortex A-15[8], the i5-3550 (with turbo and non-turbo frequency), the i5-3770U, a low-power mobile CPU and a QX9100. All these CPUs are simulated as one-

core CPU, using the one-core values for non-shared caches and real value for shared caches.

The codelets used were extracted using CERE[7] tool, originally taken from NAS[2] sequential benchmark suite and PARSEC[3] benchmark suite. We chose NAS IS sequential as a simple serial application and pthread-disabled x264 for a more complicated one. Blackscholes and Freqmine, two OpenMP applications from PARSEC suite, were chosen to test multithread performance.

The energy consumption was computed using MCPAT[9] and taken as a measure of the efficiency of each simulated CPU. Indeed, power consumption is the ideal measurement for architecture tuning, as it delimits on one side the maximum computational power of the CPU at fixed architecture (due to frequency limits) and on the other side the cost to run it.

This paper explains in section 2 the related work and its position comparing to the current research. Section 3 describes the compatibility between gem5 and the codelets extracted by CERE, along with the models, the values and the applications chosen for the simulations. Section 4 demonstrates how CERE coupled with gem5 can be used to quickly tune heterogeneus architecture for each codelet.

2 Background

2.1 On the benchmarking of processors

TODO/DELETE : Moreover, benchmarks are a good way to reproduce the usage of a computer[5].

2.2 On the use of codelets

Source code isolation has already been validated as a reliable way to reproduce the comportement of applications.

CERE is a software that creates *codelets*, small parts of an application that can be run separately, from one loop of the application. The loop run inside the application is called the *in-vivo* codelet. CERE targets to create a copy of the original application running only this loop and recreating its execution (cache state especially should be as similar as possible). This process is called *extraction* and leads to the *in-vitro codelet*. The codelet running time is only the time taken by the loop to be executed; it does not include the memory restore stage and the cache warm-up sequence.

CERE is a sofware that extracts codelets from a C/C++/Fortran application using the LLVM compiler. It operates at the Itermediare Represemtation (IR) level, and thus is more flexible than code isolation (isolating the codelet at the source-code level, then compile it) or assembly isolation (isolating the codelet at the binary file level)[7].

At this time, CERE targets only loops and openMP parallel *for* loops. On sequential NAS IS benchmark, the selected codelet covers more than 98% of the total execution time, but is 7.3 to 46.6 times faster than running the full NAS.B suite.

CERE captures the memory context at a page-granularity level, which is lighter than a full dump and then faster to replace in memory when replaying inside a simulator. Moreover, a cache warm-up can done before replaying the codelet by running one time the selected loop. This step should not be avoided when tuning microarchitecture parameters such as cache size or cache line size, as the warm-up could be slower but the other executions much faster.

2.3 On the simulators

The gem5 simulator

The gem5 simulator is a cycle-accurate simulator. Its accuracy has been demonstrated on ARM simulation on both in-order and out-of-order processor, comparing real and simulated Cortex-A8 and Cortex-A9[8]. This comparisons reveals an average absolute error of only 7%.

Replaying SPLASH benchmark on gem5 shows an error on the execution time from 1.39% to 17.94%, explained by an inaccurate simulation of the DDR memory. Nevertheless, the gem5 simulator now handles different memory types, including modern DDR3, DDR4 and GDDR5, which should be more accurate than the tested DDR memory used in SPLASH tests[6].

The MCPAT simulator

TODO or delete.

3 Simulation framework

3.1 The gem5 simulator

The gem5 simulator can be run in two different modes: syscall emulation (SE) and fullsystem mode (FS). The Syscall emulation mode simulates only the

behaviour of the CPU inside a linux operating system, and therefore cannot efficiently simulate multithreaded application, as no scheduler has been implemented. Besides, SE mode required a static linkage of all the required libraries.

On the contrary, the fullsystem mode emulate a full CPU; as the OS is emulated, the simulation is really slow (about fifteen minutes to boot linux on an x86 AtomicSimpleCPU). Nevertheless, FS mode is more accurate and more flexible. Indeed, FS mode can handle dynamic libraries, assuming that they are well installed in the virtual disk image. Moreover, gem5 featured a checkpoint functionality which avoid booting again when the CPU is changed.

3.1.1 Syscall emulation mode

Two changes on gem5 has been made to allow the use of CERE sequential codelets. First, the *getdents* syscall has been implemented, which is called inside the *readdir* function, used in the codelet memory mapping function. The second change concern a bug occurring when reading EOF with the syscall *read* while providing an invalid pointer: it should work and write nothing, but caused a page fault in gem5. Both patches were submitted to gem5 community.

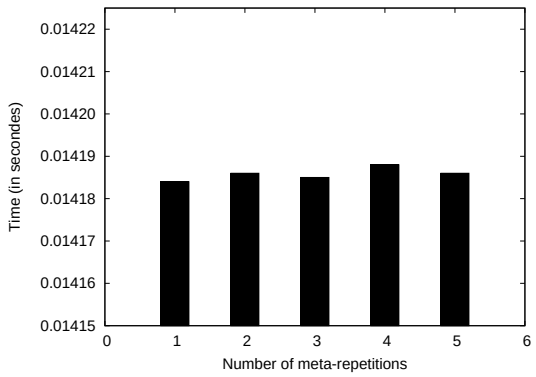

Figure 1: Variations of the codelet loop execution time on NAS IS class W benchmark using the i5-3550 configuration without turbo, with four CPUs.

In order to statically compile codelets with CERE[1], the argument *-static* must be provide at the link time. Noting that this could provoked some relocation error when specifying the entry point of the program: offset 0x60000000 (used as default start point in CERE) is used to place the standard C library in sequential NAS IS. Experimentally, offset 0x40000000 did not cause any trouble when compiling sequential codelets.

OpenMP parallel codelets cannot be run on SE mode due to the lack of real pthread implementation in the SE subsystem. Indeed, statically linking with *libiomp.a* results in a forced exit because of the missing pthread management syscalls.

With these changes, and assuming that all the syscalls have been implemented inside the gem5 simulator, any sequential codelet should work in SE mode. Given that there is no scheduler in gem5 SE mode, and given that there is no proper implementation of pthread implementation in SE mode[2], parallel codelets cannot be realistically replayed on SE mode.

Using the region __cere__is_ranked_475 with five meta-repetitions (six total runs of the loop, as one is used to warm the cache up), we observed 4x speedup, comparing to running the full benchmark[3]. All the data analysis is done on the median of these five meta-repetitions, but as the fluctuation is at most 0,03% (figure 1)[4], we can imagine running a codelet with only two or three meta-repetitions without significant bias.

3.1.2 Fullsystem mode

To run codelets in FS mode, only a few changes have been done: a more recent image than the ubuntu 7.04 available on gem5 site has been used, base on ubuntu-core 14.04. The kernel used is version 3.2.40 with default gem5 configuration.

To run OpenMP applications, just putting the *libiomp5.so* and *libomp.so* in */usr/lib*[5] works. Therefore setting KMP_affinity to **scatter**, which should assign each thread to a different core if available, results in a segmentation fault when starting the codelet. As a consequence, codelets on small inputs shows inexploitable behaviour (figure 2) on four cores. Moreover, gem5 seems to hang when simulating more

[1]CERE version 0.2 was used in this paper.

[2]M5thread has not been tested due the lack of scheduler and the miss of important syscall implementations in SE mode.

[3]Class W inputs were used for all the results.

[4]This is due to the deterministic routine of the codelet, as the region __cere__is_ranked_475 is verifying that a give array is well-sorted. Such tight results are harder to get on randomised or multithreaded code, see section 3.1.2.

[5]Taken from linux mint MATE 17.3 64 bits

than one x86 CPU in FS mode, that is why all the applications (including multicore's one) were run using a one-core configuration.

3.2 Simulation models

3.2.1 Hardware configuration

The chosen configurations are detailed in figure 3. All systems are set with 8 GB of 1600 MHz DDR3, the cacheline size is always kept at 64 B, and all CPUs are quad-core without hyper-threading.

3.2.2 Chosen codelets

We chose three codelets to efficiently reproduce different usages:

- NAS IS sequential: region __cere__is_ranked_475 which check whether the computed array is sorted or not.
- PARSEC[6] blackscholes: region __cere__blackscholes_m4__Z9bs_threadPv_first, an OpenMP region calculating the option value based on the Black & Scholes's equation.
- PARSEC freqmine: region __cere__tree8scan1_DBEP4Data_first which generates a hash from the tree dataset.
- PARSEC x264: region __cere__encoder_analyse_block_residual_write_cabac_745 used in the CABAC encoding of the video.

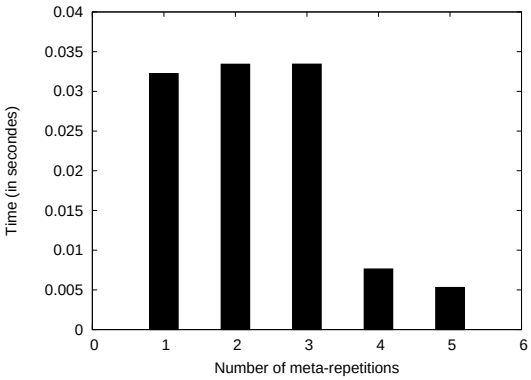

Figure 2: Variations of the codelet loop execution time on PARSEC Freqmine benchmark using the Cortex-A15 configuration with four AtomicSimpleCPU and simtest input.

Name	Frequency[7]	L1D and L1I associativity[8]	L2 Size	L2 Assoc.	L3
Cortex-A15	1 GHz	2	1 MB	16	No
i5-3550	3,3-3,7 GHz	8	4×256 kB	8	Yes[9]
i5-3337U	1,8-2,7[10] GHz	8[11]	4×256 kB	16	Yes[12]
Q9100	2,26 GHz	8	8 MB[13]	16	No

Figure 3: Parameters used for CPU simulations.

4 Results

The power consumption of each CPU is calculated roughly using gem5 output file by the MCPAT software: such value should not be taken absolutely but relatively to other CPU simulated. A power consumption of 10W is indeed too small for an one-core X86 desktop CPU running a benchmark.

One have to keep in mind that the CPUs use only using a generic x86 scheme, without hyper-threading or other technological improvements (pipeline aside), and own only one core. That is why only relative

[6] Version 3.0-beta-20150206

[7] Non-turbo - Turbo frequency when turbo techology is implemented

[8] The L1D and L1I size is always 32 kB.

[9] The size of the L3 cache is set to 8MB and its associativity to 16-way due to gem5 limitations, it should be 6MB and 12-way.

[10] Only the non-turbo frequency has been simulated

[11] It should be 6 MB.

[12] It should be 3 MB.

[13] It should be 6 MB and 12-way.

measured at fixed codelets are really meaningful.

4.1 Power/performance ratio and index

The performance of a CPU is measured by the execution time of the selected codelet. To keep an higher-is-better index, only $1/t_e$ values are used (where t_e is the execution time). The power-comsumption ratio is defined as

$$\frac{1}{P.t_e} \tag{1}$$

With P the power consumption of the CPU on the benchmark.

A higher ratio means either better performance or less comsumption, and so a better choice.

4.1.1 IS : A serial applications

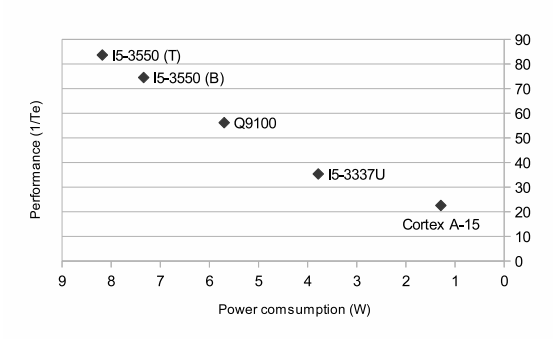

Figure 4: Power-Performance graph for the IS codelet.

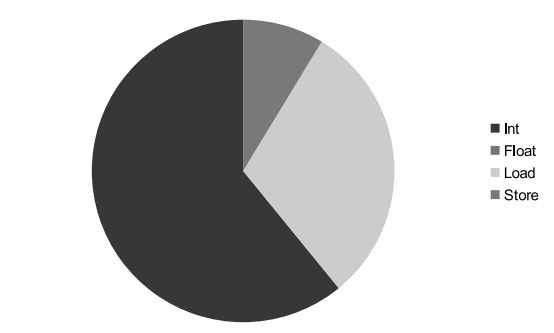

Figure 5: Repartition of the instructions during the execution of the IS codelet.

x264

?

4.1.2 Parallel applications

This parallel applications are run on a one-core configuration: these results are only bound to show the single-core performance-consumption differences, and not the manycore scaling. Such studys could easily be measured on ARM systems (see 6.1).

Freqmine

Freqmine was tested with a small data input. Therefore, all data can fit in cache. The Q9100 L2 cache is faster than the i5-3350 L3 cache, so the former is slightly faster than the latter at stock frequency on this benchmark - but consumes much regards to the improvement (figure 6). With the turbo enabled, the i5-3550 stays first.

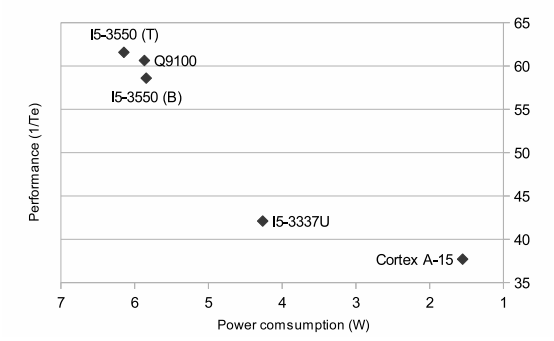

Figure 6: Power-Performance graph for the Freqmine codelet.

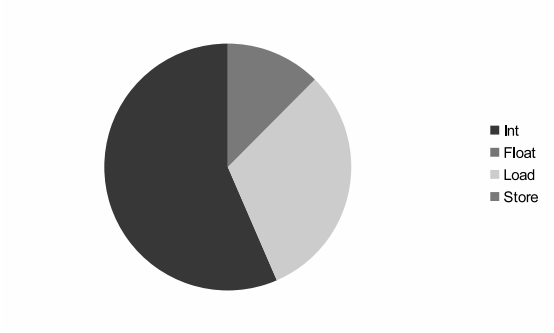

Figure 7: Distribution of the instructions during the execution of the Freqmine codelet.

Blackscholes

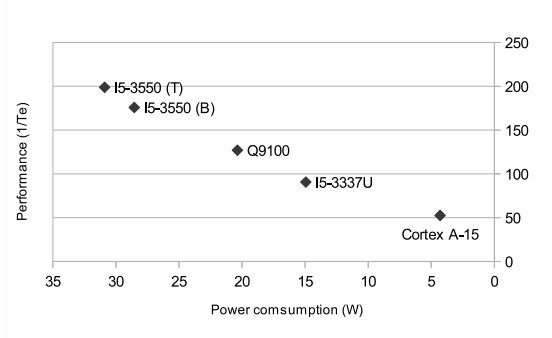

Figure 8: Power-Performance graph for the Blackscholes codelet.

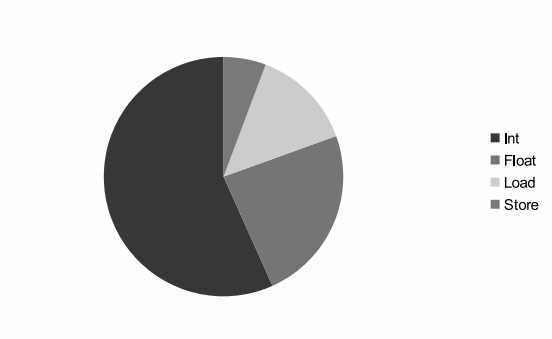

Figure 9: Distribution of the instructions during the execution of the Blackscholes codelet.

4.2 Performance

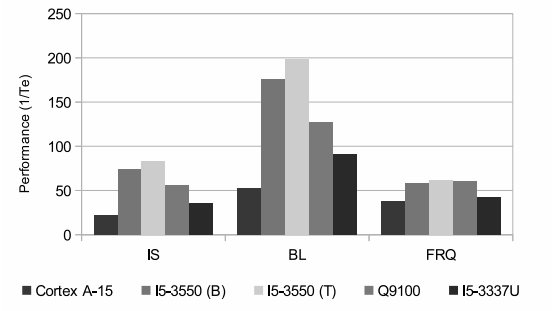

Figure 10: Performance of all tested CPUs on differents codelets.

X264 codelet is too fast too measure the execution time with gem5 (only 0,0000001 or 0,0000002 seconds are calculated), so this codelet will only be analysed for its power consumption.

IS and freqmine does execute on the same order of time (figure 10), even if freqmine codelet was extracted running the simtest input.

4.3 Power consumption

When the power consumption is on the x axis, the former is graduated backward to keep and higher-is-better index.

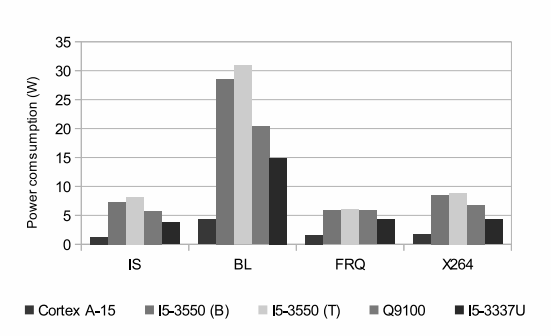

Figure 11: Power consumption for each CPU.

For example, the blackscholes codelets seems to consume more power than all the other codelets (figure 11). This is due to floating point operations, used significantly only in this codelet (see 4.1.2).

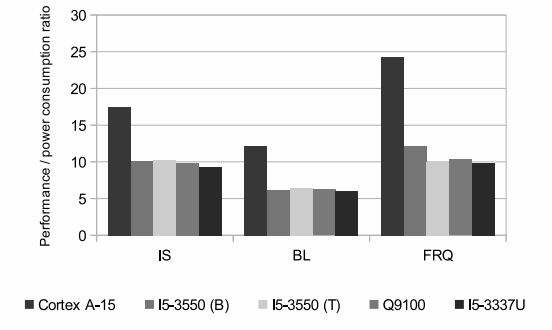

Figure 12: Performance-Power consumption ratio for each codelets.

4.4 Advantages

The use of codelets gives a powerful tool to fine-tune the architecture fitting to the codelets. Because of CERE limitations, only codelets replay *for* loops or *omp parallel for* loops are supported, but this is much lighter than running the full application. Moreover, codelets can improve individually each region of an application, which is extremely useful to choose on which CPU cluster run the application on a big.LITTLE system, or even in an heterogeneus compute server.

The gem5 simulator is a precise tool to reproduce the behaviour of a specific machine on a general computer. As the simulation is quite slow, the use of codelets could be really useful in term of time and resources in the conception of heterogeneous servers. As CERE operates as an IR-level, the operation does not need any additional work on the source code, and can be run in C, C++ or Fortran programs. Besides, the compilation of the codelets uses the power of the host machine and not the simulated one, which is really useful in term of compute time: the emulated system is used only when it is truly needed.

4.5 Inconvenient : Unknown precision

Gem5 is a simulator, and therefore cannot be entirely trusted. CERE too cannot exactly replay a whole execution of an application ; and the measures output by MCPAT are only an estimation of the effective power consumption of the CPUs: that's why the results could not fit exactly to the real-life experiments. Nevertheless, these simulations are bound to give an overview of the gains that could be archived by improving a specific part of a CPU, not to give absolute results.

The gem5 simulator has not been validated yet on x86 accuracy, and could be quite biased, as it emulates a generic x86 processor, whereas state of the art CPUs are even more complex.

5 Conclusion

To our knowledge, no other work has been produced before using codelets to do architectural-tuning inside a simulator. TODO

References

[1] Chadi Akel, Yuriy Kashnikov, Pablo de Oliveira Castro, and William Jalby. Is source-code isolation viable for performance characterization? In *42nd International Conference on Parallel Processing, ICPP 2013, Lyon, France, October 1-4, 2013*, pages 977–984. IEEE Computer Society, 2013.

[2] David H. Bailey, Eric Barszcz, John T. Barton, D. S. Browning, Robert L. Carter, Leonardo Dagum, Rod A. Fatoohi, Paul O. Frederickson, T. A. Lasinski, Robert Schreiber, Horst D. Simon, V. Venkatakrishnan, and Sisira Weeratunga. The nas parallel benchmarks. *IJHPCA*, 5(3):63–73, 1991.

[3] Christian Bienia. *Benchmarking Modern Multiprocessors*. PhD thesis, Princeton University, January 2011.

[4] Nathan L. Binkert, Bradford M. Beckmann, Gabriel Black, Steven K. Reinhardt, Ali G. Saidi, Arkaprava Basu, Joel Hestness, Derek Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish, Mark D. Hill, and David A. Wood. The gem5 simulator. *SIGARCH Computer Architecture News*, 39(2):1–7, 2011.

[5] Maximilien Breughe and Lieven Eeckhout. Selecting representative benchmark inputs for exploring microprocessor design spaces. *TACO*, 10(4):37, 2013.

[6] Anastasiia Butko, Rafael Garibotti, Luciano Ost, and Gilles Sassatelli. Accuracy evaluation of GEM5 simulator system. In Leandro Soares Indrusiak, Guy Gogniat, and Nikolaos S. Voros, editors, *7th International Workshop on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC), York, United Kingdom, July 9-11, 2012*, pages 1–7. IEEE, 2012.

[7] Pablo de Oliveira Castro, Chadi Akel, Eric Petit, Mihail Popov, and William Jalby. CERE: llvm-based codelet extractor and replayer for piecewise benchmarking and optimization. *TACO*, 12(1):6, 2015.

[8] Fernando A. Endo, Damien Couroussé, and Henri-Pierre Charles. Micro-architectural simulation of in-order and out-of-order ARM microprocessors with gem5. In *XIVth International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation, SAMOS 2014, Agios Konstantinos, Samos, Greece, July 14-17, 2014*, pages 266–273. IEEE, 2014.

[9] Sheng Li, Jung Ho Ahn, Richard D. Strong, Jay B. Brockman, Dean M. Tullsen, and Norman P. Jouppi. Mcpat: an integrated power, area, and timing modeling framework for multicore and manycore architectures. In David H. Albonesi, Margaret Martonosi, David I. August, and José F. Martínez, editors, *42st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-42 2009), December 12-16, 2009, New York, New York, USA*, pages 469–480. ACM, 2009.

[10] ARM ltd. Arm information center, 2015.

6 Annexe

6.1 ARM Simulation

As multicore benchmarks seem to hang on X86 simulation, some work has been done to adapt the codelet on ARM simulation. The multicore simulation works with linaro-minimal

As capturing on ARM required an ARM machine, a few cross-compile instructions has been added in CERE. Capturing en gem5 may indeed take several days, and require moreover an ARM version of CERE pre-installed on the virtual machine, which is too heavy to be implemented yet. The goal is then to use an x86 memory dump and replay it on ARM systems. The following changes have been made in CERE:

- Changed objdump to aarch64-linux-gnueabi-objdump.
- Added clang cross-compile option.

After thoses changed, the compilation ran successfully, but the execution outputs *"Killed"*, even before the main() starts.

We found that the aarch64 architecture limits the application space to the address 0x00000000_00000000 to 0x0000fff_fffffff[10]. But the stack is placed on x86 linux at address 0x07fda3c7_b0000000 so the kernel kills the application as soon as the X86 stack region is reserved. To avoid this issue, the stack has manually been moved to address 0x000003c7_b0000000.

NAS IS (codelet __cere__is_ranked_475) has been successfully replayed on a juno board (linaro-image-minimal-genericarmv8 system) using this trick. Nevertheless, further adaptations have to be done in order to safely convert x86 dumps to ARM dumps[14].

6.2 About the UVSQ laboratory

6.3 Gem5 patches

[14] Especially on pointers of stack address which need to be updated to the new stack position at the codelet compilation.