# Micro-architectural Simulation of In-order and Out-of-order ARM Microprocessors with gem5

Fernando A. Endo, Damien Couroussé and Henri-Pierre Charles
CEA, LIST, Département Architecture Conception et Logiciels Embarqués
F-38054 Grenoble, France
Email: first.lastname@cea.fr

***Abstract*—Heterogeneous multicore systems have gained momentum, specially for embedded applications, thanks to the performance and energy consumption trade-offs provided by in-order and out-of-order cores.**

**Micro-architectural simulation models the behavior of pipeline structures and caches with configurable parameters. This level of abstraction is well known for being flexible enough to quickly evaluate the performance of new hardware implementations, such as future heterogeneous multicore platforms.**

**However, currently, there is no open-source micro-architectural simulator supporting both in-order and out-of-order ARM cores. This article describes the implementation and accuracy evaluation of a micro-architectural simulator of Cortex-A cores, supporting in-order and out-of-order pipelines and based on the open-source gem5 simulator. We explain how to simulate Cortex-A8 and Cortex-A9 cores in gem5, and compare the execution time of ten benchmarks with real hardware. Both models, with average absolute errors of only 7 %, are more accurate than similar micro-architectural simulators, which show average absolute errors greater than 15 %.**

## I. Introduction

In embedded systems, reduced energy consumption is essential to provide more performance. One reason is that current energy storage technologies cannot scale with the increasing demands. Another reason comes from the fact that embedded systems usually have limited cooling capabilities, which limits the power dissipation and consequently the achievable performance.

A good way to reduce energy consumption is to use the most energy efficient hardware available for a given task. For example, in heterogeneous multicore systems, the energy consumption can be reduced by mapping applications to complex out-of-order cores, when performance is needed, or to simpler in-order cores, when less demanding tasks are executed [1].

Instruction set and RTL simulators are not adequate to estimate the performance and energy consumption of future heterogeneous multicore systems, because they are mostly calibrated to a specific hardware [2] and too complex to be modified, respectively. On the other hand, micro-architectural simulators offer a good trade-off between precision and simulation speed, and bring flexibility for architectural exploration.

The simulation of heterogeneous cores at the micro-architectural level is already a mature research topic. For example, Kumar et al. were the first to show the energy benefits of having different types of cores in a SoC [1]. Lukefahr et al. evaluated the possibility to push the heterogeneity of pipelines into the core [3]. In both work, the studies were performed in simulators of the Alpha ISA, currently not supported anymore. Nowadays, the ARM ISA is more relevant in embedded computing ecosystems. Then, the advantage of using ARM simulators is that researchers can evaluate their tools without porting or cross-compiling them to other platforms. For example, gem5 [4] is a popular simulation framework that permits to simulate unmodified ARM libraries and binaries, facilitating research studies of embedded systems. gem5 is famous for its precise micro-architectural models, simulating pipeline dependencies at each stage, before computing the result. This emphasizes the instruction timing and simulation accuracy (called *execute-in-execute* modeling) [4].

This paper is the first in a series of articles comparing energy and performance trade-offs of heterogeneous cores. We present a simulation platform based on gem5 [4], which simulates in-order and out-of-order pipelines of Cortex-A cores at the micro-architecture level. Our contribution is:

- We detail how we implemented an in-order pipeline in gem5, based on the out-of-order (O3) model. gem5 already provides the InOrder model for micro-architectural simulation, however it is not currently functional for ARM.
- We show how to simulate detailed Cortex-A8 (in-order) and Cortex-A9 (out-of-order) models in gem5.
- We validate the timing accuracy of these models by comparing the execution time of ten benchmarks with real hardware. To the best of our knowledge, this is the first work that directly evaluate the timing accuracy of the O3 model against real hardware.
- We show how to enhance the execution stage model in gem5 to better simulate FP data transfer penalties of the Cortex-A8.
- We compare the performance of a hypothetical dual Cortex-A8 vs a real dual Cortex-A9.

This paper is organized as follows. Section II describes the related work on micro-architectural simulation for ARM. Section III presents the simulation framework based on gem5. Section IV describes the reference and simulation models, and the benchmarks used to evaluate the accuracy of the Cortex-A models. The experiment results are presented in Section V. In

This work was supported in part by the European Project TOUCHMORE under Grant FP7-ICT-2011-7-288166.

Section VI, we show the architectural and micro-architectural exploration capability of gem5, by simulating hypothetical dual Cortex-A8 processors. Finally, we conclude in Section VII.

## II. Related work

SimpleScalar [5] was one of the first micro-architectural simulators, initially simulating a MIPS-like architecture and later also Alpha processors. A version supporting ARM was released, but only with the functional and timing accurate modes. After SimpleScalar, a huge number of x86 simulators appeared [6], [7], [8], [9]. A more recent simulation infrastructure, gem5 [4] is a cycle-accurate micro-architectural simulator, which supports the ARM ISA, including floating-point (VFP) and Advanced SIMD extensions (NEON). Butko et al. compared the execution time of gem5 modeling a Cortex-A9 with real hardware [10], but using the Timing Simple model that does not simulate at the micro-architectural level. A predecessor of gem5, the M5 simulator [11], whose out-of-order CPU model became the O3 model in gem5, was validated against two Compaq Alpha XP1000 systems embedding Alpha 21264 processors. However, this work focused on the network bandwidth evaluation. The maximum error obtained was no more than 11 %. To the best of our knowledge, there's no work that directly evaluates the timing accuracy of the O3 model in gem5. Our work differs from the previous ones by comparing in terms of execution time the micro-architectural model for ARM in gem5 against real hardware.

Not far from our approach, SimpleScalar also offered the possibility to simulate in-order issuing in their out-of-order model [5]. This technique has the advantage of virtually having two simulators, in-order and out-of-order, but only maintaining one implementation for both. In our work, we also based the in-order simulation on the out-of-order implementation, but we also disabled the effects of the register renaming, which may boost the performance of the in-order machine, but may not be cost-effective in real implementations.

## III. Simulation framework

This section presents the simulation environment. First, in section III-A, we introduce the gem5 `arm_detailed` model, which configures the out-of-order model for ARM. Then, in section III-B, we explain how the out-of-order model was modified to simulate an in-order pipeline. Finally, in section III-C, we show how to configure the `arm_detailed` model to simulate in-order (Cortex-A8) and out-of-order (A9) cores. The following presentation is based on the gem5 stable version of June 2012 [12].

### A. Overview of gem5

gem5 is a performance simulator which provides four CPU models: Atomic Simple, Timing Simple, InOrder and O3. For micro-architectural simulation, only the InOrder and O3 models produce activity statistics of in-order and out-of-order pipelines, respectively. Moreover, two types of simulation environment can be used: System-call Emulation (SE), and Full-System (FS). In the SE mode, most of the system calls are emulated by passing them to the host operating system. On the other hand, the FS mode simulates a bare-metal machine, then an operating system must run to boot the machine and to support the running applications.

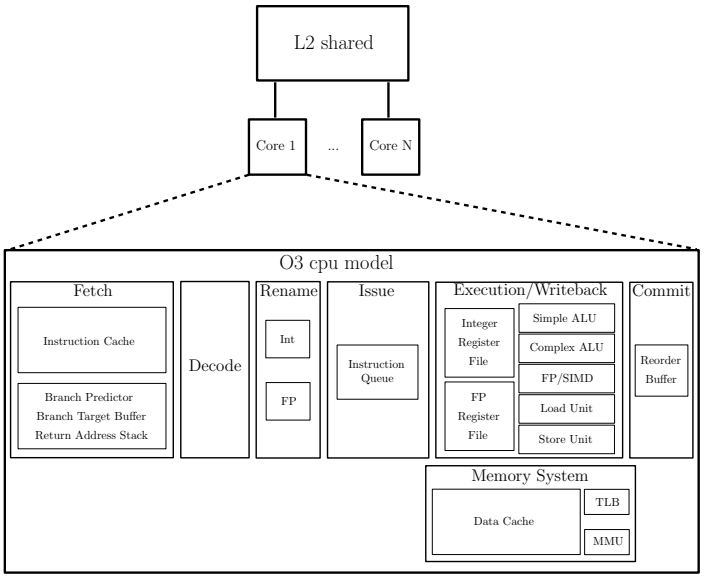


Fig. 1. The `arm_detailed` configuration of gem5.

For micro-architectural simulation with the ARM ISA, only the O3 model is currently functional. The released `arm_detailed` configuration simulates a multicore system composed of out-of-order cores with a simple memory hierarchy. Figure 1 presents the main components of this system.

*a) O3 CPU model:* The O3 CPU model simulates a generic out-of-order pipeline based on physical-register-file architectures, like the DEC Alpha design [13]. Seven pipeline stages are modeled: fetch, decode, rename, issue, execute, writeback and commit, but a variable number of stages can be effectively simulated by changing the signal delays between stages. The O3 model is highly configurable: for example, each parameter of the branch predictor (*tournament*, based on Alpha implementations [13]) can be configured. Other parameters include pipeline stage widths, and the size of buffers such as the instruction queue (IQ), reorder buffer (ROB), load/store queues (LSQ) and translation lookaside buffers (TLB). The execution stage is also highly flexible with variable number of functional units. For example: Simple ALU, Complex ALU (Mul/Div), FP/SIMD, Load and Store Units. Each unit accepts one or more operation classes with configurable latency and a pipelined flag[1]. Each operation class regroups one or more instructions.

*b) Memory hierarchy:* The `arm_detailed` model uses the Classic memory model. Two on-chip Data and Instruction caches form the first cache level. An off-chip L2 cache is the last level, connected to the SimpleMemory model. Stride prefetchers can be configured and added to any cache level. The Classic memory model puts the focus on the pipeline simulation, differently from the Ruby model that supports configurable cache coherence protocols, providing a detailed memory hierarchy simulation.

[1] In this version, the issue latency of operation classes is set to 1 to indicate that the functional unit is pipelined, or greater than 1, otherwise.

### B. Our approach for modeling an in-order pipeline with gem5

At the time of writing this paper, the InOrder CPU model was not functional for the ARM ISA yet. Instead of porting the current InOrder CPU model for ARM, we modified the O3 model to issue in program order and to disable the rename stage. First, we argue why this approach was chosen. Next, we explain how the O3 model was modified, then we present a qualitative explanation of the simulation precision of our approach.

*1) InOrder vs modified O3 model:* The InOrder model has a common simulation engine and architecture-dependent supporting code, which is partially ported to ARM. In our opinion, the InOrder model may not be adapted to simulate current high-end microprocessors, due to the lack of associative structures such as the IQ and LSQ, present in modern embedded microprocessors. In addition, FP and SIMD instructions are apparently not supported yet. On the other hand, the O3 model is based on high-performance machines. In our opinion, modifying the O3 model to simulate in-order cores of the Cortex-A series is faster and more stable to implement, compared to bring-up and enhance the InOrder model. However, the modifications to issue instructions in-order and to cancel the register renaming provide a cycle approximate in-order pipeline.

*2) Modification of the gem5 O3 model:* Ideally, to simulate an in-order pipeline using an out-of-order model, structures supporting out-of-order execution should be completely removed. However, instead of removing these structures, which can be complex, time-consuming and unstable, the original pipeline stages were kept intact. In summary, we bring two modifications over the original out-of-order pipeline:

- Issue in dispatch order;
- An extra register scoreboarding to disable register renaming effects.

The fetch, decode, execute and writeback stages are not modified. The commit will continue retiring instruction in program order.

*a) Issuing in program order:* We modified the instruction scheduler so that we only allow the oldest instruction in the instruction queue to issue if it's ready. The only exceptions to this rule are instructions that already requested a memory access and are re-issuing. In this case, we allow them to be issued again even if they are not the oldest instructions. This happens, for example, with load instructions of uncacheable data.

*b) Disabling the rename stage:* The register renaming must be disabled, because it would incorrectly allow the concurrent execution of instructions that had register dependencies avoided by renaming their destination registers. Then, to cancel the effect of register renaming, an extra register scoreboarding was implemented over the original model. As a consequence, ready instructions are only issued if their source and destination registers are not being modified by any issued instruction under execution. Our register scoreboard only locks destination registers, because we consider that source registers are read during the first cycle of execution. Destination registers are then unlocked when the instruction writebacks.

*c) Out-of-order parameters:* The parameters that only exist in out-of-order pipelines should be configured properly to simulate an in-order machine. Table I lists and describes these parameters.

*3) Functionality:* Considering that we modified the instruction scheduler to only issue ready instructions respecting two extra restrictions (issue in-order and respect register dependencies), there is no reason that our approach would lead to wrong results. In other words, every ready instruction from the in-order pipeline point of view is always a ready instruction from the out-of-order point of view. We tested the correct execution of the model, by comparing the results produced by benchmarks. No errors were detected.

*4) Accuracy:* Our modification of the gem5 O3 model to simulate an in-order pipeline does not remove out-of-order structures. These remaining structures impact the accuracy of the simulator, albeit in a negligible way:

- We keep an "empty" rename stage. Nevertheless, we can consider the rename stage as an additional decode sub-stage.
- The commit stage continues managing the correct execution in the pipeline, even if the vast majority of the correct execution is insured by the issue in program-order modification.

In the Section V, we provide results and analysis that confirms the timing accuracy of the proposed approach.

### C. Configuring gem5 to simulate Cortex-A processors

This section presents the configuration parameters of the gem5 `arm_detailed` model to simulate Cortex-A8 and Cortex-A9 cores, with the VFP extension.

*1) System parameters:* The ARM Cortex-A9 is a dual-issue out-of-order pipeline with 8 to 11 stages [14], while the Cortex-A8 is a dual-issue in-order pipeline with 13 stages [15]. The main parameters of our models are summarized in Table II. The core clock and cache sizes were based on the development kits Snowball SKY-S9500-ULP-CXX series [16] and BeagleBoard-xM [17], respectively. The cache and memory latencies are typical values [18]. The cache line length is 32 Bytes in the Cortex-A9 [19], but this version of gem5 only accepts 64 as length. Most of the other parameters were taken from manuals and from the ARM website [20], [14], [19], [21], [15], [22]. Unfortunately, there's no official information about the following parameters (in parenthesis, our sources if any, otherwise we made an educated guess). For the Cortex-A9 model:

- L1 cache buffer sizes: miss status and handling register – MSHR – and write buffer (gem5 default values).
- Branch predictor – BP – and branch target buffer – BTB – parameters.
- ITLB and DTLB size.
- Number of IQ and ROB[2] entries (default values).

[2]Accordingly to Li et al. [23], the Cortex-A9 implements a out-of-order pipeline without a traditional ROB.

TABLE I. CONFIGURATION OF GEM5 O3 MODEL PARAMETERS[1] WHEN SIMULATING A DUAL-ISSUE IN-ORDER PIPELINE

| O3 model parameter | Ideal value | Chosen value | Description |
|---|---|---|---|
| *ToRenameDelay<br>renameTo*Delay<br>iewToCommitDelay<br>commitTo*Delay<br>renameToROBDelay | 0 | 1 | The rename and commit (which includes the ROB) stages do not exist in an in-order pipeline, then signal delays to and from them should be as small as possible. |
| numROBEntries<br>commitWidth<br>squashWidth | infinite | 512 | The ROB is an out-of-order structure that holds the information of instructions eventually executing out-of-order. An in-order pipeline does not need such buffer, because correct program execution must be insured before instructions writeback. To avoid the ROB stalling the pipeline, ideally there would be an infinite number of entries and an infinite commit and squash bandwidth. |
| renameWidth | infinite | numROBEntries | With an infinite bandwidth, the rename stage would not stall the pipeline due to instructions waiting to be renamed. In the O3 model, the renaming throughput is only limited by structures that receive the renamed instructions in the following stages, then we set its bandwidth as the number of entries in the largest structure. |
| dispatchWidth | infinite | numROBEntries | Same reasoning as renameWidth. The dispatch is between the rename stage and the following structures, then it has to be configured with the same bandwidth as the rename stage to not stall the pipeline. |
| numPhysIntRegs<br>numPhysFloatRegs | infinite | 44+numROBEntries<br>72+numROBEntries | In order to avoid stalls in the rename stage due to lack of physical registers, these parameters should be set as large as possible. For a finite ROB, the maximum number of physical registers used is equal the number of architectural registers plus the number of entries in the ROB. |

[1] To work properly, the out-of-order model needs two patches: issue in program order and disable register renaming.

- Number of LSQ entries.
- Pipeline stages depth (gem5 mailing-list[3]).

And for the Cortex-A8 model:

- L2 cache: MSHR and write buffer (default values).
- BP parameters.
- Pipeline stages depth.

*2) The execution stage:* The functional units were configured as shown in Table III, which only includes the VFP extension. The latency of instructions were obtained from ARM manuals [19], [24], [21]. While in the Cortex-A9, the VFP extension is pipelined for most instructions, the Cortex-A8 VFP extension in not pipelined. Note that the pipelined Advanced SIMD unit in the A8 can execute some single-precision instructions faster than the VFP, but we did not test this feature.

The execution stage of the in-order and out-of-order pipelines is modeled with two and three pipelines, respectively. Table IV shows the distribution of the functional units in their execution stages.

*3) Configuring the tournament branch predictor:* gem5 models the *tournament* branch predictor. A local and a global predictor are updated in parallel, and for each branch instruction, a third predictor (the chooser) decides which one will be used. Given that both Cortex-A9 and A8 have only the global history branch predictor [20], [21], the configuration of the *tournament* model was adapted. We set the local predictor with the smallest possible size, to reduce its influence, but the chooser was kept intact. The aim is that the local predictor, with a minute size, will loose every prediction to the global one.

*4) Micro-architectural differences:* Compared to the O3 model, the two main differences of the Cortex-A9 are the absence of FP register renaming [15] and the out-of-order implementation without a traditional ROB [23]. Given that we have no detailed information about its pipeline implementation,

[3] A suggestion is setting equal number of sub-stages, except the rename stage, which is two times longer

TABLE II. PARAMETERS OF OUR GEM5 OUT-OF-ORDER AND IN-ORDER CORE MODELS

| Parameter | | | Out-of-order (Cortex-A9) | In-order (Cortex-A8) |
|---|---|---|---|---|
| Core clocks | | | 800 MHz | 800 MHz |
| DRAM | Size | | 256 MB | 256 MB |
| | Clock | | 400 MHz | 166 MHz |
| | Latency[1] | | 65 | 65 |
| L2 | Size | | 512 kB | 256 kB |
| | Associativity | | 8 | 8 |
| | Latency[1] | | 8 | 8 |
| | MSHRs | | 11 | 16 |
| | Write buffers | | 9 | 8 |
| L1-I | Size | | 32 kB | 32 kB |
| | Associativity | | 4 | 4 |
| | Latency[1] | | 1 | 1 |
| | MSHRs | | 2 | 1 |
| L1-D | Size | | 32 kB | 32 kB |
| | Associativity | | 4 | 4 |
| | Latency[1] | | 1 | 1 |
| | MSHRs | | 4 | 1 |
| | Write buffers | | 16 | 1 |
| | Stride prefetcher | Degree | 1 | N/A |
| | | Buffer size | 8 | N/A |
| Global BP | Entries | | 4096 | 512 |
| | Bits | | 2 | 2 |
| BTB entries | | | 4096 | 512 |
| Return address stack entries | | | 8 | 8 |
| ITLB/DTLB entries | | | 64 each | 32 each |
| Issue width | | | 2[2] | 2 |
| gem5 effective execution stage depth (`wbDepth`)[3] | | | 8 | 6 |
| Pipeline stages | | | 8 | 13 |
| Physical INT/FP registers | | | 62/256[4] | 556/584[5] |
| IQ entries | | | 32 | 16[6] |
| LSQ entries | | | 8 each[7] | 12 each[6] |
| ROB entries | | | 40 | 512[5] |

[1] Latencies in core clock cycles.
[2] We assume that the Cortex-A9 is dual-issue [14], although it may issue 4 instructions in some conditions [15].
[3] In gem5, `wbDepth` multiplied by the issue width represents the maximum allowed number of in-flight instructions in the execution stage.
[4] The Cortex-A9 does not rename FP registers [15]. The choice of the number of physical FP registers is explained in section III-C4.
[5] These structures do not exist in an in-order pipeline and the chosen values are explained in section III-B2c.
[6] We considered the corresponding structures in the NEON/VFP unit: one 16-entry instruction queue, one 12-entry load queue [21]. We assume that it has a store queue with 12 entries too.
[7] The Cortex-A9 has a store buffer with 4 slots and probably at least a 4-entry load queue (i.e., support for four data cache line fill requests) [20]. Without precise information, these parameters were hence tunned in the simulator.

TABLE III. CONFIGURATION OF THE FUNCTIONAL UNITS (FUS) FOR INTEGER AND VFP INSTRUCTIONS

| gem5 FU | gem5 opClass | Example of instructions | Out-of-order (Cortex-A9) | | In-order (Cortex-A8) | |
|---|---|---|---|---|---|---|
| | | | Latency | Pipelined | Latency | Pipelined |
| Simple ALU | IntAlu | MOV, ADD, SUB, AND, ORR | 1 | Yes | 1 | Yes |
| Complex ALU | IntMult | MUL, MLA | 4 | Yes | 5 | Yes |
| FP Unit[1] | SimdFloatAdd | VADD, VSUB | 4 | Yes | 10 | No |
| | SimdFloatCmp | VCEQ, VCGE, VCGT, VCLE, VCLT | 1 | Yes | 6 | No |
| | SimdFloatCvt | VCVT | 4 | Yes | 7 | No |
| | SimdFloatDiv | VDIV | 15 | No | 43 | No |
| | SimdFloatMisc | VMRS, VMSR, VMOV, VABS, VNEG | 1 | Yes | 4 | No |
| | SimdFloatMult | VMUL, VNMUL | 5 | Yes | 14 | No |
| | SimdFloatMultAcc | VMLA, VMLS, VNMLA, VNMLS | 8 | Yes | 22 | No |
| | SimdFloatSqrt | VSQRT | 17 | No | 40 | No |
| Load/Store Unit | MemRead | LDR, VLDR | 1 | Yes | 1 | Yes |
| | MemWrite | STR, VSTR | 1 | Yes | 1 | Yes |

[1] Although we simulate only the VFP extension, in gem5, both VFP and Advanced SIMD instructions are regrouped under the SimdFloat* operation classes.

TABLE IV. CONFIGURATION OF GEM5 EXECUTION STAGES[1]

| Pipeline | Functional unit(s) | |
|---|---|---|
| | Out-of-order (Cortex-A9) | In-order (Cortex-A8) |
| 1 | Simple ALU | Simple ALU, Complex ALU |
| 2 | Simple ALU, Complex ALU | Simple ALU, FP Unit, Load/Store Unit |
| 3 | FP Unit, Load/Store Unit | N/A |

[1] Table III describes the classes of instructions that each functional unit can execute.

we assume that these micro-architectural differences have a small impact in the execution time. We considered that the Cortex-A9 has a balanced pipeline, i.e. its structures are large enough to sustain the pipeline width in the absence of stalls. For example, to configure the FP renaming in gem5, we choose a sufficient large pool of physical FP registers (256). Compared to our in-order model, the Cortex-A8 has a considerable more complex micro-architecture. While gem5 only simulates one pipeline for all types of instruction (integer, FP and SIMD), the integer and SIMD/FP pipelines are completely separated in the Cortex-A8. SIMD and FP instructions have to go through the integer pipeline until completion, before being decoded and executed [21].

## IV. EXPERIMENTAL SETUP

The main objective of our experiments is to evaluate the accuracy of our gem5 in-order and out-of-order models. To do so, we compared the execution time of benchmarks simulated by gem5 and measured in software development kits (SDKs). To insure a fair comparison, we ran the same binaries and dynamic libraries in all platforms, and core frequencies were fixed at 800 MHz.

### A. Reference models

The out-of-order reference model is the Snowball SDK, equipped with a dual Cortex-A9 processor [16]. The board runs the Linaro 11.11 distribution with a Linux 3.0.0 kernel.

For the in-order reference model, we chose the BeagleBoard-xM SDK, which has a processor with only one Cortex-A8 core [17]. The board runs a Linux 3.9.11 kernel with the Ubuntu 11.04 distribution released by the gem5 website. We insured that the NEON extension (which includes the VFP unit) accesses data in the L1 data cache (by default it accesses the L2).

### B. Simulation models

Section III-C details the simulation models. The in-order model is a Cortex-A8 core connected to a L2 cache, and its parameters were based on the BeagleBoard-xM SDK. The out-of-order model has a dual Cortex-A9 core connected to a shared L2 cache, representing the Snowball board. Only the VFP extension was tested.

### C. Benchmarks

To evaluate the timing accuracy of the models, we selected 10 of the 13 benchmarks of PARSEC 3.0 [25], a modern suite which covers several application domains. Three benchmarks were not tested, because:

- Canneal: does not support ARM yet;
- Facesim: the only released input set takes more than one minute to execute, which could take days to be simulated in gem5;
- Raytrace: depends on X11 development libraries, which were not available in the embedded environment;

To validate the gem5 Cortex-A9 model, we ran the benchmarks with one and two threads (except Vips with three threads) in order to evaluate the single and dual core behaviors. The A8 model was evaluated only with single-threaded benchmarks.

We used the Ubuntu 11.04 file system released by gem5 to natively compile the benchmarks in the Snowball with gcc 4.5.2. This file system together with the released Linux 2.6.38.8 kernel were used to boot the simulation models in the gem5 Full System mode.

The execution time was measured with the built-in `bash` command `time`, whose resolution is the millisecond, and we took the `real` measurements. In gem5, after booting, a script waits 10 s to calm down the initialization processes before running the benchmark under test. A similar procedure is carried in the reference boards. The PARSEC suite offers six input sets [25]. We chose the *simsmall*, which is adapted for micro-architectural simulations.

## V. RESULTS

This section first presents the accuracy evaluation of the Cortex-A models compared to real hardware. Then, we analyse

TABLE V. COMPARISON OF THE EXECUTION TIME (SECONDS) OF PARSEC BENCHMARKS EXECUTED IN THE GEM5 DUAL CORTEX-A9 MODEL AND THE SNOWBALL

| Benchmark | | Single thread | | | Two threads[1] | | |
|---|---|---|---|---|---|---|---|
| | | SDK | gem5 | Error (%) | SDK | gem5 | Error (%) |
| INT | Dedup | 2.65 | 2.58 | -2.57 | 2.02 | 1.85 | -8.36 |
| | Freqmine | 4.95 | 4.30 | -13.2 | 3.46 | 3.14 | -9.30 |
| | x264 | 6.45 | 7.23 | 12.1 | 3.96 | 4.45 | 12.4 |
| INT mean error | | | | -1.22 | | | -1.74 |
| INT mean absolute error | | | | 9.31 | | | 10.0 |
| FP | Blackscholes | 0.600 | 0.629 | 4.83 | 0.353 | 0.364 | 3.12 |
| | Bodytrack | 2.62 | 2.41 | -8.13 | 1.62 | 1.65 | 1.73 |
| | Ferret | 2.80 | 2.73 | -2.64 | 1.98 | 1.84 | -6.68 |
| | Fluidanimate | 3.10 | 2.98 | -3.97 | 1.94 | 1.90 | -2.07 |
| | Streamcluster | 3.48 | 3.53 | 1.44 | 1.76 | 1.84 | 4.42 |
| | Swaptions | 3.14 | 3.67 | 16.7 | 1.61 | 1.88 | 16.7 |
| | Vips | 6.73 | 6.15 | -8.66 | 3.76 | 3.44 | -8.49 |
| FP mean error | | | | -0.07 | | | 1.24 |
| FP mean absolute error | | | | 6.62 | | | 6.17 |
| Overall mean error | | | | -0.41 | | | 0.35 |
| Overall mean abs. error | | | | 7.43 | | | 7.33 |

[1] For Vips, we ran the benchmark with three threads in both platforms.

TABLE VI. COMPARISON OF THE EXECUTION TIME (SECONDS) OF PARSEC BENCHMARKS EXECUTED IN THE GEM5 CORTEX-A8 MODEL AND THE BEAGLEBOARD-XM

| Benchmark | | SDK | gem5 | Error (%) |
|---|---|---|---|---|
| Integer | Dedup | 3.40 | 3.75 | 10.5 |
| | Freqmine | 5.76 | 4.90 | -15.0 |
| | x264 | 6.50 | 6.71 | 3.18 |
| INT mean error | | | | -0.43 |
| INT mean absolute error | | | | 9.56 |
| Floating-point | Blackscholes | 1.86 | 1.72 | -7.54 |
| | Bodytrack | 7.78 | 7.40 | -4.97 |
| | Ferret | 7.60 | 6.64 | -12.7 |
| | Fluidanimate | 8.33 | 7.94 | -4.68 |
| | Streamcluster | 10.3 | 10.5 | 1.74 |
| | Swaptions | 10.1 | 8.51 | -15.8 |
| | Vips | 14.5 | 13.9 | -3.76 |
| FP mean error | | | | -6.81 |
| FP mean absolute error | | | | 7.30 |
| Overall mean error | | | | -4.89 |
| Overall mean abs. error | | | | 7.98 |

the behavior of the proposed in-order model. Finally, we propose a modification in the execution stage to improve even more the accuracy of our Cortex-A8 model.

### A. Accuracy evaluation of the Cortex-A models

Table V and VI show the reported execution time of each benchmark and the percentage errors of the Cortex-A9 and Cortex-A8 models, respectively. The simulations took between 1 and 8 hours. In average, the Cortex-A9 model estimates the execution time with an absolute error of only 7.4 %, ranging from 1 to 17 %. The Cortex-A8 model performs as well as the A9, in average, estimating the execution time with an absolute error of 8.0 %, ranging from 2 to 16 %.

Even considering the generic modeling of gem5, these magnitudes of error can be considered as good results for a micro-architectural simulator. For example, two models of the SimpleScalar simulator were compared to a real Alpha 21264 processor [26]. The `sim-outorder`, which models a generic out-of-order pipeline, showed an average absolute error of 37 %, going up to 77 %. The `sim-alpha`, which models the actual chip, performed better with an average absolute error of 18 %, going up to 43 %. For x86 platforms, PTLSim showed only 5 % of timing error compared to the real AMD's K8 architecture, but the only benchmark considered was the `rsync` command found in Linux systems [6]. Zesto was preliminarily validated against Intel's Merom micro-architecture, showing average absolute errors between 5 and 6 %, although, as the authors noted, the selected micro-benchmarks are too simple to fully evaluate the timing accuracy of their simulator [9]. A more recent x86 simulator, McSimA+, was validate against an Intel Xeon E5540. By comparing the result of a large number of Splash-2 and SPECCPU2006 benchmarks, McSimA+ achieved an average IPC absolute error of 15.1 %, going up to almost 40 % [27].

In our opinion, most of the timing errors comes from the generic purpose of gem5 and the parameters incertitude. We purposely chose the Cortex-A8 and A9 because they are the only two microprocessors of the Cortex-A series whose execution stage parameters are publicly available. Such official information is absolutely useful to reduce the parameters incertitude, but as explained in Section III-C4, these modern processors have complex micro-architectures, and it's not easy to configure a generic model to simulate them.

### B. In-order model behavior

*1) O3 model parameters for in-order simulation:* To insure that the modified O3 model behaves as an in-order pipeline, we verified that the values chosen in Table I, except the delays, do not produce unwanted stalls in the pipeline. For example, by looking at the gem5 statistics, we confirmed that the rename stage does not lacks physical registers, because the counter of physical registers limit reached (`FullRegisterEvents`) was virtually zero[4] in all benchmarks. In addition, we also inserted warning messages in the code, for example, to tell if instructions are issued out-of-order. In more detail, we compare the fetch sequence number of the issuing instructions to check that this number never decreases (except instructions re-issuing, which are verified separately). All benchmarks issued all instructions in-order.

*2) Further improvements:* We observed that memory ordering violations still happen, although, in the worst case, Ferret had only 884 ordering violation among 700 million memory references. After investigation, we found that even if all instructions are issued and re-issued in-order, some loads can receive a value before a preceding instruction stores data to the same address. The out-of-order mechanism simply squashes the violator (load) and the newer instructions, which is not realistic for an in-order pipeline, because such events should be detected and avoided before instructions writeback. Finally, to better identify unwanted behaviors like this, we propose the following verification. To simulate an in-order over an out-of-order pipeline, ideally, after an instruction effectively writebacks [5] it must not be squashed from the ROB (if so, the

[4] Only in the Freqmine benchmark, the `FullRegisterEvents` counter was not zero. However, only 22 events out of 4.2 billion renamed operands is negligible. This behavior indicates that we underestimated the maximum number of physical registers used (Table I). Indeed, in the worst case, each instruction in the ROB could have two destination registers.

[5] gem5 does not directly simulate data bypass between functional units. The proposed solution is that instructions writeback just after computing the results, respecting the result latency. By doing so, the depending instructions can issue and read their inputs in the register file in the next cycle, just as if the result was bypassed. For example, an ALU instruction has a result latency of one cycle, after which the result is writtenback. But the integer pipeline in the execution stage has more than one sub-stage, e.g. five in the Cortex-A8 [21], then the effective writeback may occur five cycles after issuing.

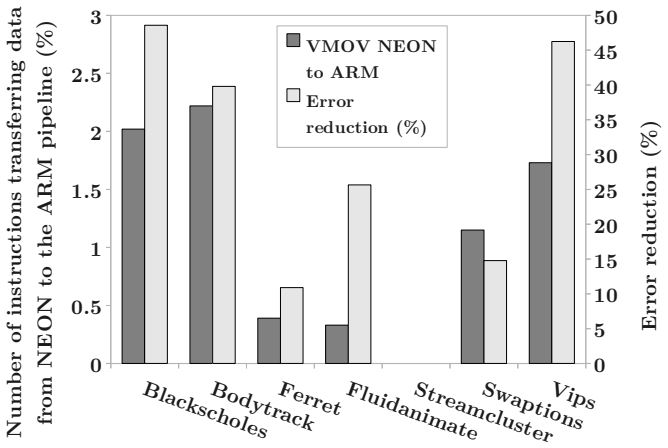


Fig. 2. Correlation between the number of VMOV instructions transferring data from NEON to the ARM pipeline and the error reduction obtained by the introduction of delay penalties is such instructions (Cortex-A8 model).

writtenback values are lost, because the rename map will be set to a previous state). Our proposal is to warn when such events happen.

### C. Modeling FP data transfer penalties of the Cortex-A8

The validation of our Cortex-A8 model shows that an out-of-order model can be adapted to approximately simulate an in-order pipeline with good precision. However, as Table VI shows, in average, the gem5 Cortex-A8 model is faster than the BeagleBoard-xM. Among FP benchmarks, 6 of 7 are estimated with a faster execution time than the real board. In order to explain this behavior, we therefore focused our analysis on the FP benchmarks. We discovered that VMOV instructions have latency penalties when data is transferred from NEON/VFP to the ARM pipeline [21]. This means that the original execution stage model in gem5 tends to be faster, because such penalties are not modeled.

To confirm our hypothesis, gem5 was modified with the VMOV instructions that move data from NEON to ARM regrouped in a separate operation class. This new class has a latency of 20 cycles and instructions can be issued back-to-back (i.e., the functional unit is pipelined to those operations) [21]. Table VII compares the simulation error of the original gem5 model and our improved model, executing FP benchmarks. In average, this modification reduced the simulation error of FP benchmarks by 27 %. The graph is Figure 2 shows the absolute percentage error reduction of our modification correlated with the number of such VMOV instructions. The correlation is not perfect, because the slowdown and in consequence the error reduction depends on the relative place of such VMOV instructions and the critical paths, for example. This improved Cortex-A8 model, in average, estimates the execution time of the ten benchmarks with an absolute error of 6.8 %, ranging from 2 to 15 %.

## VI. Architectural and micro-architectural exploration

To illustrate the design space exploration capabilities of gem5, we simulated the performance of two hypothetical dual

TABLE VII. FP benchmark results of the improved Cortex-A8 model with the execution stage modeling data transfer penalties from NEON to the ARM pipeline.

| Benchmark | Error (%) | | Abs. error reduction (%) |
|---|---|---|---|
| | Original model | Improved model | |
| Blackscholes | -7.54 | -3.88 | 48.6 |
| Bodytrack | -4.97 | 2.99 | 39.8 |
| Ferret | -12.7 | -11.3 | 10.9 |
| Fluidanimate | -4.68 | -3.48 | 25.6 |
| Streamcluster | 1.74 | 1.74 | 0[1] |
| Swaptions | -15.8 | -13.4 | 14.8 |
| Vips | -3.76 | 2.02 | 46.2 |
| Mean | -6.81 | -3.62 | 26.6 |
| Absolute Mean | 7.30 | 5.55 | |

[1] Streamcluster had no improvement because only a negligible number of VMOV instructions transferring data from NEON to the ARM pipeline are executed.

Cortex-A8 processors (the Cortex-A8 does not support multi-core configurations [15]). In both processors, we configured two Cortex-A8 cores using our improved model of Section V-C connected to a shared L2 cache. The difference resides in the FP pipeline: one model has the original non-pipelined VFP unit, while the other has a pipelined version with the same parameters as the Cortex-A9 VFP. This modified model offers a more fair comparison between almost equivalent in-order and out-of-order pipelines.

Table VIII shows the speedup of the dual over the mono A8 and the speedup of the dual A9 over the dual A8. In average, the dual Cortex-A8 allows a speedup of 1.77 over its mono-core version, achieving an almost perfect speedup in the Swaption and Streamcluster benchmarks. The comparison of the out-of-order dual A9 and the in-order dual A8 is more interesting: For integer benchmarks, there is no significant speedup (between 4 and 6 percent is beyond the precision of our simulator), which may be explained by the fact that small-latency integer operations benefit less from out-of-order pipelines than long-latency floating-point ones. For FP benchmarks, as expected, the original non-pipelined VFP in the A8 leads to poor results compared to the A9 equipped with a pipelined VFP, in average executing FP benchmarks two and a half times slower. On the other hand, an A8 equipped with the same VPF as in the A9 shows relatively better performance with an average slowdown factor of 1.37, for FP benchmarks. This performance difference comes not only from the dynamic scheduling capacity of the Cortex-A9, but may also come from differences in their memory systems: for example, slower memory frequency, smaller L2 cache and the blocking cache behavior in the Cortex-A8.

This experiment show how useful a micro-architectural simulator can be. They allow researchers to explore the architectural and micro-architectural design space, to test new ideas and to simulate hypothetical or emerging hardware implementations.

## VII. Conclusion

In this paper, we detailed the simulation of in-order and out-of-order pipelines with gem5. We presented a fast and stable modification to simulate an in-order pipeline modifying the out-of-order model. We showed how to configure Cortex-A9 and Cortex-A8 cores and compared the execution time of ten benchmarks with real hardware. In average, these models estimate the execution time with absolute errors around 7 %

TABLE VIII. PERFORMANCE OF PARSEC BENCHMARKS EXECUTED IN HYPOTHETICAL DUAL CORTEX-A8[1] PROCESSORS, COMPARED TO THE SINGLE CORTEX-A8 MODEL[1] AND THE DUAL CORTEX-A9 (SNOWBALL)

| Benchmark[2] | | Speedup of dual over mono A8 (Original VFP[3]) | Speedup of dual A9 over dual A8 | |
|---|---|---|---|---|
| | | | Original VFP[3] | Same VFP as A9[3] |
| INT | Dedup | 1.76 | 1.06 | 1.11 |
| | Freqmine | 1.40 | 1.01 | 1.01 |
| | x264 | 1.60 | 1.06 | 1.06 |
| INT mean | | 1.59 | 1.04 | 1.06 |
| FP | Blackscholes | 1.87 | 2.70 | 1.37 |
| | Bodytrack | 1.69 | 2.93 | 1.42 |
| | Ferret | 1.83 | 1.86 | 1.19 |
| | Fluidanimate | 1.79 | 2.32 | 1.30 |
| | Streamcluster | 1.96 | 3.04 | 1.64 |
| | Swaptions | 1.97 | 2.75 | 1.39 |
| | Vips | 1.83 | 2.15 | 1.31 |
| FP mean | | 1.85 | 2.53 | 1.37 |
| Overall Mean | | 1.77 | 2.09 | 1.28 |

[1] Based on the improved Cortex-A8 model of Section V-C
[2] The mono-core A8 ran the benchmarks with one thread, while the dual-core processors ran with two threads, except Vips with three.
[3] The original VFP extension in the A8 is not pipelined, while in the A9 the VFP is pipelined for most instructions.

and within 17 %. Similar simulators also extensively validated show average absolute errors between 15 and 37 %. Our results confirm the timing accuracy of the O3 model in gem5 and of the proposed in-order simulator. We plan to send a patch of the Cortex-A9 configuration to the gem5 community.

## ACKNOWLEDGMENT

The authors would like to thank Alexandre Aminot, Victor Lomüller and anonymous readers for their reviews and suggestions.

## REFERENCES

[1] R. Kumar, K. I. Farkas, N. P. Jouppi, P. Ranganathan, and D. M. Tullsen, "Single-ISA heterogeneous multi-core architectures: The potential for processor power reduction," in *Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO 36. Washington, DC, USA: IEEE Computer Society, 2003, pp. 81–. [Online]. Available: http://dl.acm.org/citation.cfm?id=956417.956569

[2] N. Fournel, "Estimation et optimisation de performances temporelles et énergétiques pour la conception de logiciels embarqués," Ph.D. dissertation, École Normale Supérieure de Lyon, 2007.

[3] A. Lukefahr, S. Padmanabha, R. Das, F. M. Sleiman, R. Dreslinski, T. F. Wenisch, and S. Mahlke, "Composite cores: Pushing heterogeneity into a core," in *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 317–328. [Online]. Available: http://dx.doi.org/10.1109/MICRO.2012.37

[4] 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

[5] D. Burger and T. M. Austin, "The SimpleScalar tool set, version 2.0," *SIGARCH Comput. Archit. News*, vol. 25, no. 3, pp. 13–25, Jun. 1997. [Online]. Available: http://doi.acm.org/10.1145/268806.268810

[6] M. T. Yourst, "PTLsim: A cycle accurate full system x86-64 microarchitectural simulator," in *ISPASS 2007: IEEE International Symposium on Performance Analysis of Systems and Software*, IEEE Comp Soc. IEEE COMPUTER SOC, 2007, Proceedings Paper, pp. 23–34.

[7] H. Zeng, M. Yourst, K. Ghose, and D. Ponomarev, "MPTLsim: a simulator for x86 multicore processors," in *Proceedings of the 46th Annual Design Automation Conference*, ser. DAC '09. New York, NY, USA: ACM, 2009, pp. 226–231. [Online]. Available: http://doi.acm.org/10.1145/1629911.1629974

[8] A. Patel, F. Afram, S. Chen, and K. Ghose, "MARSS: a full system simulator for multicore x86 CPUs," in *Proceedings of the 48th Design Automation Conference*, ser. DAC '11. New York, NY, USA: ACM, 2011, pp. 1050–1055. [Online]. Available: http://doi.acm.org/10.1145/2024724.2024954

[9] G. H. Loh, S. Subramaniam, and Y. Xie, "Zesto: A Cycle-Level Simulator for Highly Detailed Microarchitecture Exploration," in *ISPASS 2009: IEEE International Symposium on Performance Analysis of Systems and Software*, IEEE; IEEE Comp Soc. IEEE COMPUTER SOC, 2009, Proceedings Paper, pp. 53–64.

[10] A. Butko, R. Garibotti, L. Ost, and G. Sassatelli, "Accuracy Evaluation of GEM5 Simulator System," in *2012 7th International Workshop on Reconfigurable and Communication-centric Systems-on-Chip (ReCoSoC)*, Indrusiak, LS and Gogniat, G and Voros, N, Ed. IEEE, 2012, Proceedings Paper.

[11] N. L. Binkert, R. G. Dreslinski, L. R. Hsu, K. T. Lim, A. G. Saidi, and S. K. Reinhardt, "The M5 simulator: Modeling networked systems," *IEEE Micro*, vol. 26, no. 4, pp. 52–60, Jul. 2006. [Online]. Available: http://dx.doi.org/10.1109/MM.2006.82

[12] gem5.org. (2012, June) gem5 stable version of june 2012. Tag: stable_2012_06_28. [Online]. Available: http://repo.gem5.org/gem5-stable

[13] D. Leibholz and R. Razdan, "The Alpha 21264: A 500 MHz Out-of-Order Execution Microprocessor," in *Proceedings of the 42nd IEEE International Computer Conference*, ser. COMPCON '97. Washington, DC, USA: IEEE Computer Society, 1997, pp. 28–. [Online]. Available: http://dl.acm.org/citation.cfm?id=792770.793692

[14] ARM website. Cortex-A9 Processor. [Online]. Available: http://www.arm.com/products/processors/cortex-a/cortex-a9.php

[15] ARM, *Cortex-A Series Programmer's Guide*, June 2012, version: 3.0.

[16] Calao Systems, *SKY-S9500-ULP-CXX (aka Snowball PDK-SDK) Hardware Reference Manual*, July 2011, revision 1.0.

[17] BeagleBoard.org, *BeagleBoard-xM Rev C System Reference Manual*, April 2010, revision 1.0.

[18] ARM, *CoreLink Level 2 Cache Controller L2C-310 Revision: r3p3 Technical Reference Manual*.

[19] ——, *Cortex-A9 Technical Reference Manual*, June 2012, revision: r4p1.

[20] ——, "The ARM Cortex-A9 Processors," *ARM White paper*, 2009, document Revision 2.0 Sept 2009.

[21] ——, *Cortex-A8 Technical Reference Manual*, May 2010, revision: r3p2.

[22] Texas Instruments, *AM/DM37x Multimedia Device Silicon Revision 1.x Technical Reference Manual*, May 2010, version R.

[23] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "The McPAT framework for multicore and manycore architectures: Simultaneously modeling power, area, and timing," *ACM Trans. Archit. Code Optim.*, vol. 10, no. 1, pp. 5:1–5:29, Apr. 2013. [Online]. Available: http://doi.acm.org/10.1145/2445572.2445577

[24] ARM, *Cortex-A9 Floating-Point Unit Technical Reference Manual*, June 2012, revision: r4p1.

[25] C. Bienia, "Benchmarking modern multiprocessors," Ph.D. dissertation, Princeton University, January 2011.

[26] R. Desikan, D. Burger, and S. W. Keckler, "Measuring experimental error in microprocessor simulation," in *Proceedings of the 28th Annual International Symposium on Computer Architecture*, ser. ISCA '01. New York, NY, USA: ACM, 2001, pp. 266–277. [Online]. Available: http://doi.acm.org/10.1145/379240.565338

[27] J. H. Ahn, S. Li, O. Seongil, and N. P. Jouppi, "McSimA+: A manycore simulator with application-level+ simulation and detailed microarchitecture modeling," in *Performance Analysis of Systems and Software (ISPASS), 2013 IEEE International Symposium on*. IEEE, 2013, pp. 74–85.