An Accurate Architectural Simulator for ARM1136

Hyo-Joong Suh[1] and Sung Woo Chung[2]

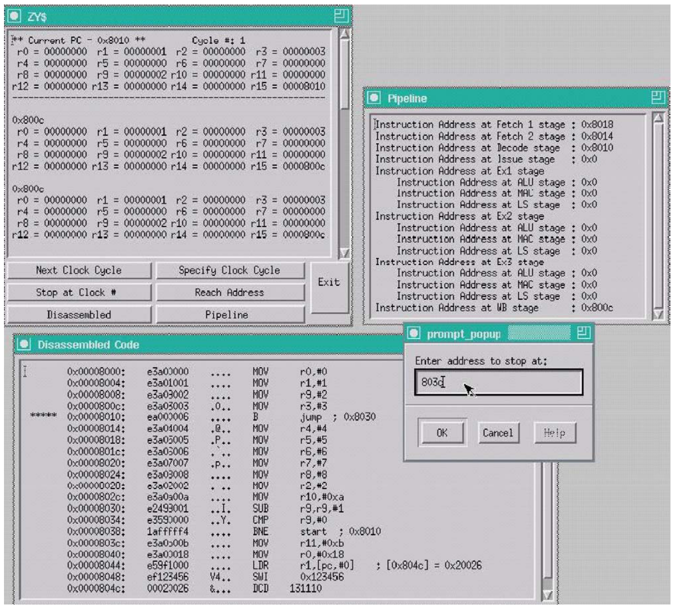
[1] School of Computer Science and Information Engineering,
The Catholic University of Korea
San 43-1 Yeoggog 2 Dong, Wonmi Gu, Bucheon, Gyeonggido, 420-743 Korea
hjsuh@catholic.ac.kr
http://esca.catholic.ac.kr
[2] Corresponding Author
Department of Computer Science
University of Virginia
Charlottesville VA 22901 USA
schung@cs.virginia.edu

Abstract. Cycle-accurate simulators are basic tools to evaluate performance improvements of computer architecture. Before confirming of the architecture improvements using cycle-accurate simulation, the simulator itself should be validated. However, off-the-shelf processors have been continuously improved, though the cycle-accurate simulators were not reflected the improved features. Simulation results show that the difference between the IPC (Instruction Per Cycle) of the modified model for ARM1136 (Sim-ARM1136) and the IPC of the original model for ARM7 (Sim-Outorder) is 19%, on average, which is large enough to mislead the impact of architecture improvements.

1 Introduction

Every architect try to improve the performance of microprocessors to achieve a high speed, low power consumption, and large bandwidth, by examining a real silicon takes a lot of resources, cost and time. Practically, however, we can not explore all architectural alternatives on the silicon, and most researchers, especially in academia, do not analyze the impact of their idea in the silicon level. Instead, they disbranch useless alternatives using architecture simulators [1][2][3][4][5]. According to detail of the simulation level, the architecture simulators can be classified into two classes : instruction-accurate and cycle-accurate. Instruction accurate simulators aim at instruction level performance evaluation, which does not includes the detailed cycle information, and the correctness is limited within the instruction level. Compared to the instruction accurate simulators, cycle accurate simulators are focused on the lower level that requires the performance evaluation as well as correctness check.

About a year ago, we had been trying to deploy an existing processor simulator for architecture explorations. However, insufficient degree of details made



```
** Current PC - 0x8010 **         Cycle #: 1
 r0 = 00000000  r1 = 00000001  r2 = 00000000  r3 = 00000003
 r4 = 00000000  r5 = 00000000  r6 = 00000000  r7 = 00000000
 r8 = 00000000  r9 = 00000002 r10 = 00000000 r11 = 00000000
r12 = 00000000 r13 = 00000000 r14 = 00000000 r15 = 00008010
-------------------------------------------------------------

0x800c
 r0 = 00000000  r1 = 00000001  r2 = 00000000  r3 = 00000003
 r4 = 00000000  r5 = 00000000  r6 = 00000000  r7 = 00000000
 r8 = 00000000  r9 = 00000002 r10 = 00000000 r11 = 00000000
r12 = 00000000 r13 = 00000000 r14 = 00000000 r15 = 0000800c

0x800c
 r0 = 00000000  r1 = 00000001  r2 = 00000000  r3 = 00000003
 r4 = 00000000  r5 = 00000000  r6 = 00000000  r7 = 00000000
 r8 = 00000000  r9 = 00000002 r10 = 00000000 r11 = 00000000
r12 = 00000000 r13 = 00000000 r14 = 00000000 r15 = 0000800c
```

```
Instruction Address at Fetch 1 stage : 0x8018
Instruction Address at Fetch 2 stage : 0x8014
Instruction Address at Decode stage  : 0x8010
Instruction Address at Issue stage   : 0x0
Instruction Address at Ex1 stage
    Instruction Address at ALU stage : 0x0
    Instruction Address at MAC stage : 0x0
    Instruction Address at LS stage  : 0x0
Instruction Address at Ex2 stage
    Instruction Address at ALU stage : 0x0
    Instruction Address at MAC stage : 0x0
    Instruction Address at LS stage  : 0x0
Instruction Address at Ex3 stage
    Instruction Address at ALU stage : 0x0
    Instruction Address at MAC stage : 0x0
    Instruction Address at LS stage  : 0x0
Instruction Address at WB stage      : 0x800c
```

```
        0x00008000:   e3a00000   ....   MOV    r0,#0
        0x00008004:   e3a01001   ....   MOV    r1,#1
        0x00008008:   e3a09002   ....   MOV    r9,#2
        0x0000800c:   e3a03003   .0..   MOV    r3,#3
*****   0x00008010:   ea000006   ....   B      jump  ; 0x8030
        0x00008014:   e3a04004   .@..   MOV    r4,#4
        0x00008018:   e3a05005   .P..   MOV    r5,#5
        0x0000801c:   e3a06006   .`..   MOV    r6,#6
        0x00008020:   e3a07007   .p..   MOV    r7,#7
        0x00008024:   e3a08008   ....   MOV    r8,#8
        0x00008028:   e3a02002   . ..   MOV    r2,#2
        0x0000802c:   e3a0a00a   ....   MOV    r10,#0xa
        0x00008030:   e2499001   ..I.   SUB    r9,r9,#1
        0x00008034:   e3590000   ..Y.   CMP    r9,#0
        0x00008038:   1afffff4   ....   BNE    start  ; 0x8010
        0x0000803c:   e3a0b00b   ....   MOV    r11,#0xb
        0x00008040:   e3a00018   ....   MOV    r0,#0x18
        0x00008044:   e59f1000   ....   LDR    r1,[pc,#0]    ; [0x804c] = 0x20026
        0x00008048:   ef123456   V4..   SWI    0x123456
        0x0000804c:   00020026   &...   DCD   131110
```

```
Enter address to stop at:
803c
```

Fig. 1. Sim-ARM1136 GUI

us hesitate to utilize the existing simulators as they are. We worried about the abstraction errors that may draw wrong conclusions when evaluating new techniques [6].

Cycle-accurate simulators are powerful tools to evaluate a performance sensitive architecture alteration with time and cost efficiency, while these tools must be validated against the processor architecture. Usually, cycle-accurate simulators tend to trade-off between exactness and simulation speed, and moreover, many improved features are added on off-the-shelf microprocessors, which weaken a rationale of outdated cycle-accurate simulator as an architecture evaluation tool. For example, the cycle-accurate simulator (called *Sim-Outorder*) from Simplescalar-ARM [5] was shown to be cycle-accurate by comparing the ARM 7 [7] silicon [8]. However, ARM 1136 [9] has a lot of different features from ARM 7 : number of pipeline stages, organization of execution units, write buffers, and so on. Nevertheless, many researchers use existing simulators not even modifying pipeline structure and functions in each pipeline, when they evaluate their embedded processor designs. In this paper, we will discuss the inaccuracy that arises from the use of an existing cycle-accurate simulator without modifications, and compare with modified simulator that reflects many architecture updates.

To discuss this issue, we modified Sim-Outorder from Simplescalar[10] to reflect the off-the-shelf processor (ARM1136) [9] features. The modified model is called *Sim-ARM1136* in this paper. We selected Sim-Outorder from Simplescalar to compare with, because it is known as one of the most popular cycle-accurate simulators for the computer architecture research. To validate Sim-ARM1136,

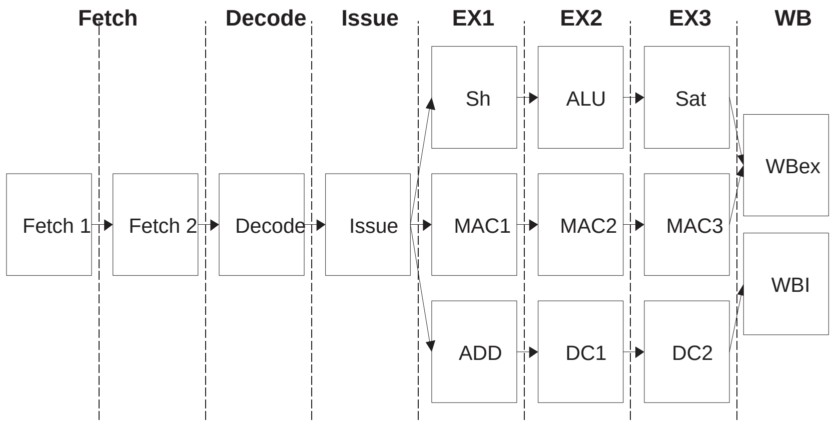


Fig. 2. Overall Pipeline Structure of Sim-ARM1136

we compare Sim-ARM1136 outputs with the RTL (register transfer level) simulation results of ARM1136. For convenience of operation, graphic interface was prepared on Sim-ARM1136, shown in Figure 1, and the user interface provided cycle-by-cycle validation against ARM1136 RTL by running some test programs.

In the next section, we describe Sim-ARM1136, focusing on the difference from Sim-Outorder. Section 3 describes the methodology that we used in this study. Section 4 shows the simulation results, and Section 5 concludes this paper.

2 Sim-ARM1136 Enhancements

In this section, we briefly describe Sim-ARM1136 enhancements compare with Sim-Outorder based on Simplescalar. Following features are the enhancement of Sim-ARM1136, and ARM1136 naturally.

– Pipeline stage modification

Sim-Outorder has 6-stage pipeline : fetch, decode, issue, execution, writeback, and commit. While Sim-ARM1136 is composed of 8-stage pipeline : fetch1, fetch2, decode, issue, execution1, execution2, execution3, and writeback as shown in Figure 2. Fetch and branch prediction consumes two cycles, resulting in two fetch stages, and execution is divided into three cycles.

In Sim-Outorder, one execution stage plays a role of multiple execution stages by setting latency and allowing multiple instructions in an execution unit. However, this structure can not support data forwarding in the execution stage. In Sim-ARM1136, we explicitly divide one execution stage into three execution stages for three kinds of execution units : Integer pipeline, MAC (Multiply and Accumulate) pipeline, and LS (Load/Store) pipeline.

Integer pipeline consists of Shift, ALU, and Saturation. If an instruction includes an operand to be shifted, the operand is shifted in the Shift stage. The

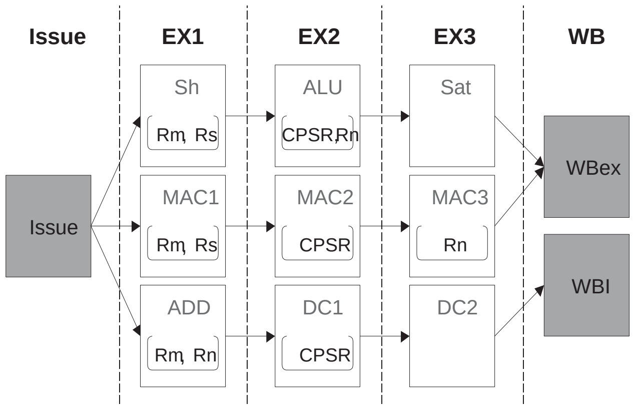


Fig. 3. Necessary Resolution of Data Dependency

ALU stage calculates the operands and the Saturation stage saturates the output from the ALU stage, if necessary. MAC pipeline consists of MAC1, MAC2, and MAC3. In the MAC1 stage and the MAC2 stage, operands are multiplied. In the MAC3 stage, the output from the previous stage is accumulated. The LS pipeline has ADD, DC1, and DC2. In the ADD stage, load/store address is calculated. If a shift operation is required to calculate the address in a load/store instruction, the shift operation in the Shift stage of the Integer pipeline should precede load/store instruction. In other words, if a shift operation is necessary for a load/store, the load/store instruction is implicitly split into two instructions. One goes through the Integer pipeline and the other goes through the LS pipeline. The data cache is accessed in the DC1 and DC2 stages.

The commit stage is merged into the writeback stage. We also modify the branch misprediction resolution mechanism to be appropriate for the modified pipeline structure.

– Interlock and data forwarding

In Sim-Outorder, instructions are issued after all data dependencies are resolved, while in Sim-ARM1136, instructions are issued regardless of data dependencies. In Sim-ARM1136, data dependencies are checked in every execution stage. If the required data dependency is not resolved, the instruction can not proceed until complete the resolution of the data dependency. Figure 3 shows the resolution of dependencies to proceed to the next stage. The dependency by the CPSR (Current Program Status Register) is also checked to match the condition code flags.

– Hit under miss

When Sim-Outorder is set to run in in-order, multiple data cache misses can go out to fetch data from memory by absence of second level cache in ARM1136.

In case of a cache miss occurs, subsequent data cache access can fetch data only if there is a free entry in load store queue. In Sim-ARM1136, only one data cache miss can fetch data from memory. If there is more than one cache misses, pipeline stalled until there remains only one cache miss.

– Memory interface

In modern processors, there is a write buffer between a first level cache and memory, whose purpose is to decouple stores from the pipeline. In Sim-Outorder, the entry of the write buffer is not limited. To reflect ARM1136 features, Sim-ARM1136 has an 8-entry write buffer, which may affect performance much. CWF (Critical Word First) is also modeled in Sim-ARM1136.

We also implemented 64-bit wide internal bus between the pipeline and the first level cache, which support two-word transfer at a time in case of LDM (LoaD Multiple) or STM (STore Multiple) operations.

– Static branch prediction

In Sim-Outorder, there are only dynamic branch predictions, while in Sim-ARM1136, static branch prediction as well as dynamic branch prediction is modeled also. ARM1136 uses bimodal prediction for dynamic branch prediction, and forward taken/backward untaken for static branch prediction. In the first fetch stage, dynamic prediction is performed and BTAC (Branch Target Address Cache) is accessed that is similar to BTB (Branch Target Buffer). In the second fetch stage, static prediction is performed if a miss occurs in the BTAC.

– Other miscellaneous enhancements

In the cache model of Sim-Outorder, there is no option for turn off the write allocation policy, while in ARM1136, a cache line for a read operation should be allocated in the cache.

MAC pipeline can not process MAC instruction for every cycle. At least one cycle stall is required between adjacent MAC instructions, and if the former MAC instruction is 64-bit operation, two cycle stalls is required. Data dependency and forwarding of the MAC instructions were implemented also, but we think the model of MAC instructions need to be detailed in Sim-ARM1136.

3 Methodology

We built the many enhancement of ARM1136 in Sim-ARM1136 as described, we discuss a validation process of Sim-ARM1136 and compare the accuracy of Sim-ARM1136 to that of Sim-Outorder in this section.

1). Simple test programs are made to verify the latencies and enhanced features in ARM1136 TRM[9] with Sim-ARM1136. These program

Table 1. Processor Parameters

Parameter	Value
Issue Policy	In-order
Branch Predictor	Bimodal, 128-entry
BTB (Branch Target Buffer)	128-entry
Instruction Cache	16KB, 32B line size, 4-way set associative
Data Cache	16KB, 32B line size, 4-way set associative
Cache Latency	1 cycle
Memory Latency	8 (first chunk), 12 (second), 16 (third), 20 (fourth)

includes the validation of ALU execution, MAC execution, interlock forwarding, hit under miss, and branch prediction/recovery.

2). Execution of Dhrystone (1-iteration) using ARM1136 RTL, Sim-ARM1136 and Sim-Outorder.

3). We assume the enhanced features of simulator are evaluated and stabilized if it provides similar improvement/degradation across other validated environments. In order to validation the stability of the simulators, we repeat procedure 2) by turning off the branch predictor. Turning off the branch predictor in ARM1136 RTL simulation can be done, by changing a status register value.

4). Procedure 2) and 3) are repeatedly done on Dhrystone (100-iteration)

5). Six applications are selected from the EEMBC benchmark suite, and compare the applications on ARM1136 RTL and Sim-ARM1136 with 1-iteration.

6). We compare the IPC of Sim-ARM1136 to that of Sim-Outorder on all the applications with default number of iterations by running the EEMBC benchmark suite. We could not run the applications on ARM1136 RTL, due to the running default iterations in the RTL simulation requires excessively long time.

Other processor parameters are complying with the ARM1136 features that detailed in Table 1. Please note that Sim-Outorder only accepts ARM7 binaries. For an appropriate comparison, we use ARM7 binaries compiled by the RVCT (RealView Compilation Tools) [12] as an input for all three simulators (RTL simulation, Sim-Outorder, and Sim-ARM1136).

4 Simulation Results

4.1 IPC Comparison : ARM1136 vs. Sim-Outorder vs. Sim-ARM1136 (Dhrystone)

When running Dhrystone 1-iteration, the IPC (Instruction Per Cycle) from the ARM1136 RTL simulation is 0.19 as shown in Figure 4. In this case, the IPC from Sim-Outorder and that from Sim-ARM1136 is 0.22 and 0.20, respectively.

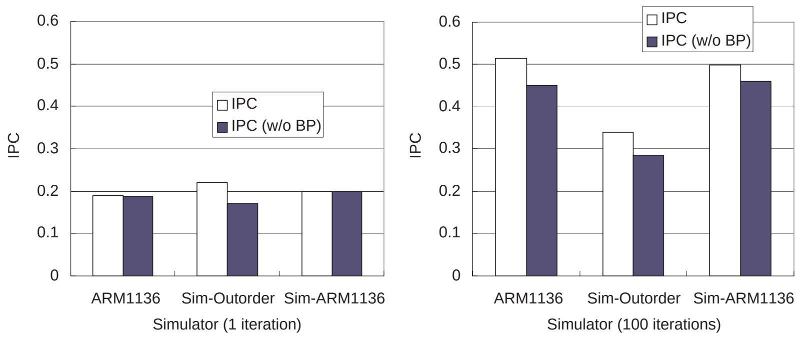


Fig. 4. IPC Comparison (Dhrystone)

Table 2. IPC Comparison between ARM1136 and Sim-ARM1136 (EEMBC, 1-iteration)

Applications	IPC (ARM1136)	IPC (Sim-Outorder)	Difference (%)	IPC (Sim-ARM1136)	Difference (%)
autcor00	0.57	0.73	-28.07	0.62	-8.77
bitmnp01	0.71	0.75	-5.63	0.71	0.86
bitmnp816	0.60	0.63	-5.00	0.55	8.33
memacc816	0.32	0.35	-9.38	0.35	-9.38
pntrch816	0.53	0.48	9.43	0.51	3.77
rspeed816	0.19	0.22	-15.79	0.25	-31.58
Mean	0.48	0.60	12.22	0.58	10.45

All the results are similar. When simulating without the branch predictor, the relative performance loss in Sim-ARM1136 is as close as that in ARM1136. However, the performance loss in Sim-Outorder is as much as 24%. Different from Sim-ARM1136, Sim-Outorder can issue multiple outstanding data cache miss requests to memory. However, without the branch predictor, the effect of multiple outstanding data miss requests decreases.

When running Dhrystone 100-iteration, the results from ARM1136 and Sim-ARM1136 shows less than 3% difference. Sim-Outorder has 36% lower IPC, compared to the ARM1136. There are two reasons for that. First, running more iterations reduce data cache miss rate, leading to decrease of the multiple data cache miss effect. Second, data dependency prevents instructions from being issued in Sim-Outorder, where only dependency-resolved instructions can be issued.

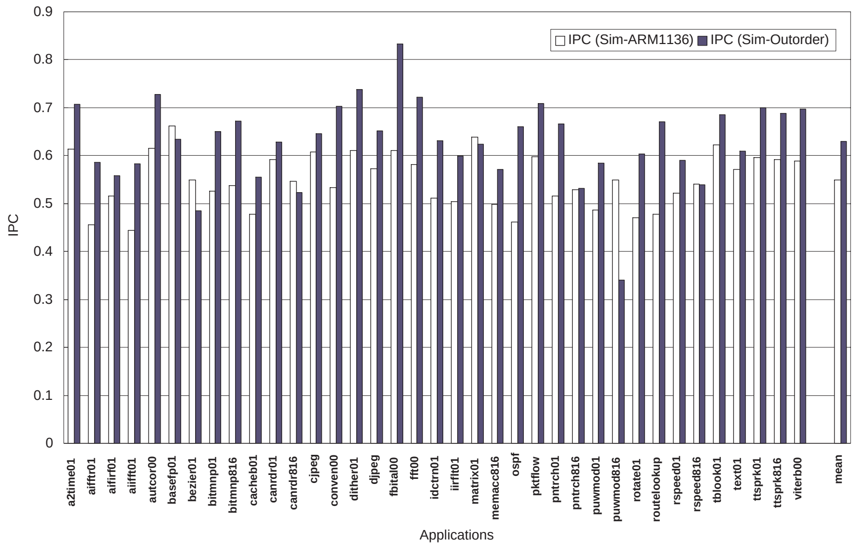


Fig. 5. IPC Comparison between Sim-ARM1136 and Sim-Outorder (EEMBC,default-iteration)

4.2 IPC Comparison : ARM1136 vs. Sim-Outorder vs. Sim-ARM1136 (EEMBC, 1-teration)

Table 2 shows the accuracy of Sim-Outorder and Sim-ARM1136 to that of Sim-Outorder by running EEMBC benchmark suite [11]. The average difference is 12.22% and 10.45% in Sim-Outorder and in Sim-ARM1136, respectively. Main source of differences may be caused by not modeled features (more detailed MAC pipeline, return address stack, branch folding and so on).

Since the first iteration incurs much more data cache misses that account for large portion of execution time, the IPC of Sim-Outorder is very close to that of Sim-ARM1136. In the next Subsection, we show the difference between Sim-ARM1136 and Sim-Outorder by running the default number of iterations of all applications in the EEMBC benchmark suite.

4.3 IPC Comparison : Sim-ARM1136 vs. Sim-Outorder (EEMBC, default-iteration)

As shown in Figure 5, the simulation results show that Sim-Outorder has higher IPC than Sim-ARM1136 in most applications. These results are reverse, compared to the results in Section 4.1. The reason is that EEMBC benchmark suite incurs much more data cache misses than Dhrystone, which results in performance gain from multiple outstanding cache misses in Sim-Outorder. Though Sim-ARM1136 allows the instructions whose data dependencies are not resolved, this effect is compensated by hit under miss feature (explained in Section 2) that allows only one outstanding data cache miss. The average IPCs for Sim-ARM1136 and for Sim-Outorder are 0.55 and 0.63, respectively. The average difference is as much as 19.0%. Moreover, the increased IPC in Sim-Outorder is not proportional among the applications. Therefore, the amount of performance improvement or degradation by researcher's new idea would be different between in Sim-ARM1136 and in Sim-Outorder.

Though Sim-ARM1136 shows lower IPC in most applications, Sim-ARM1136 shows higher IPC in some applications that are expected to have a lot of data forwarding and utilize the 64-bit wide internal bus. Especially, in puwmod816 that has much more LDM/STM, the IPC difference is as much as 38%. Since LD/ST generates two operations (shift operation for address calculation and practical LD/ST), it seriously increases data dependencies. In this case, data dependency can be resolved by data forwarding, resulting in relative performance gain in Sim-ARM1136.

5 Conclusions

Simulators are essential tools for most researchers to analyze their valuable idea without costly silicon implementation. However, state-of-the art embedded processors have many architectural improvements than the simple simulators. By the absence of improved features of embedded processors on simulators, if researchers put ideas in the simulators without reflecting of the improvements, simulation results can be distorted by impacts of added features. Thus, this might unintentionally conclude valuable idea as useless.

In this paper, we validated Sim-ARM1136, which simulates the processor parameters of ARM1136 as well as the extended features. We also presented the difference between Sim-ARM1136 and Sim-Outorder where only processor parameters are changed without modifying pipeline features. The simulation results show the difference that might mislead to incorrect performance evaluation. Now, Sim-ARM1136 is adopted by Samsung electronics to evaluate of high-performance embedded processors.

References

1. V. Pai, P. Ranganathan, and S. Adve: RSim: A Simulator for Shared Memory Multiprocessor and Uniprocessor Systems that Exploit ILP In Proceedings of the 3rd Workshop on Computer Architecture Education (1997)
2. M. Rosenblum, S. Herrod, E. Witchel, and A. Gupta: Complete Computer Simulation : The SimOS Approach In IEEE Parallel and Distributed Technology (1995)
3. D.M. Tullsen, S.J. Eggers, and H.M. Levy: Simultaneous Multithreading: Maximizing On-Chip Parallelism In 22nd Annual International Symposium on Computer Architecture, June (1995)
4. Virtutech: Simics Simulator. available at http://www.virtutech.com.
5. Simplescalar LLC: Simplescalar 3.0 available at http://www.simplescalar.com.

6. Bryan Black and Jhon Paul Shen: Calibration of Microprocessor Performance Models Computer, 31 (5):41-49, May (1998)
7. ARM Corp.: ARM 7 Technical Reference Manual available at http://www.arm.com.
8. Simplescalar LLC: Simplescalar Tutorial available at http://www.simplescalar.com.
9. ARM Corp.: ARM 1136 Technical Reference Manual available at http://www.arm.com
10. Simplescalar LLC: Pre-release of Simplescalar 4.0, (2004)
11. The Embedded Microprocessor Benchmark Consortium: EEMBC Benchmark Suite available at http://www.eembc.org.
12. ARM Corp.: RealView Compilation Tools available at http://www.arm.com.