Performance-oriented PIC10F N-core simulator

Bogdan PETRIC[1)], Călin BÎRĂ[2)]

[1,2)] Departament of Devices, Circuits and Electronic Architectures (DCAE) , Faculty of Electronics, Telecommunications and Information Technology (ETTI), Politehnica University of Bucharest, Bucharest, Romania (bogdan.petric@stud.etti.upb.ro / calin.bira@upb.ro)

Keywords: performance-oriented simulator, PIC10F

Abstract: Testing of multiple programs running on the same core architecture is of use when a new microcontroller architecture is developed. However, this is performed sequentially even if the microcontroller to be developed is of very small compute power, making the testbench setup time dominate test runtime. This paper proposes a solution for testing multiple programs (hex-files) based on the PIC10F family[1] of MCUs from Microchip. A performance comparison will be made against the Microchip's MPLAB Simulator taking multiple precautions to enable a fair comparison.

1. Introduction and state of the art

Software-based simulators have always been used when no hardware was available (or was otherwise too expensive). They come in both online form, like CPUlator which simulates a NiosII, ARMv7 or MIPS cpu[2], HTML5 CPU Simulator[3] and offline form like CPU Sim3.1 [4], gem5[5], Sniper[6], MARSSx86[7], ZSim[8]. A performance comparison is made in paper [9] regarding an i7 CPU using gem5, Sniper, MARSSx86 and ZSim highlighting the problems encountered (cache misses, lack of uop cache, lack of fused uops). CPU-GPU system simulators like[10] and [11] have started to appear to fulfil the need of simulating heterogenous systems.

In [12], authors claim that 90% of the papers use the simulator as a performance-evaluation technique. A lot of simulators are simulating the x86 core as it is one of the most used architectures in the world, but very few simulators exist for microcontroller cores.

The objectives of this paper are to simulate 1-core of PIC10F microcontroller family, accelerate simulation of N-cores of PIC10F as a Multiple Instruction Multiple Data (MIMD) machine and benchmark the simulator against the Microchip's MPLAB v5.35 simulator (without using I/Os which one may argue that workstation's video subsystem might pollute the test)

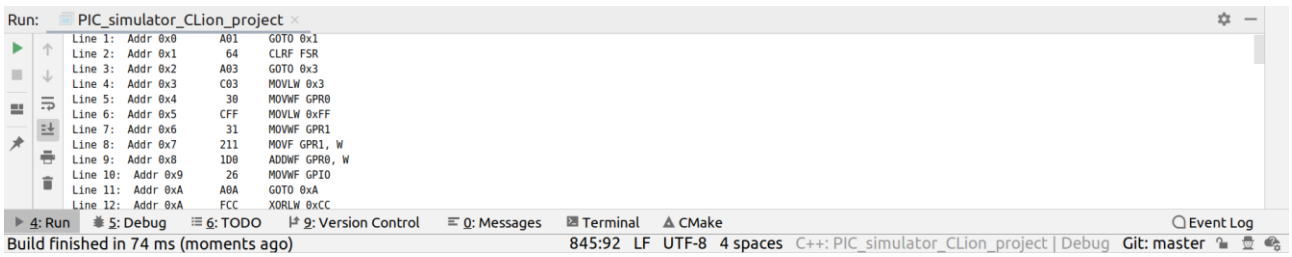

Figure 1. A hex-file is decoded and printed into the console for code inspection

2. Software simulator approach

Our simulator is made of multiple software components:

- the Intel HEX file format parser: its task is to read the HEX-file and assemble a vector of instructions
- the Instruction decoder takes each location of vector of instructions, assembles different instruction formats and presents them to the next layer, the instruction interpreter
- the Instruction interpreter executed the instruction based on the opcode and its arguments
- the 10F core model contains the register file, the flags and the program memory
- the N-core MIMD is obtained by running multi-threaded the 10F core model

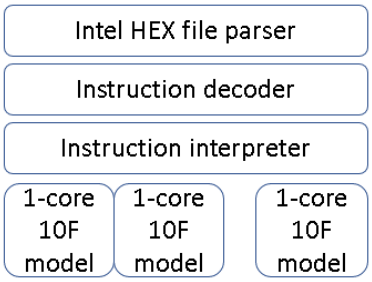

Figure 2. Software stack of the proposed simulator

3. Testing procedure

We have used the same i7-8550U machine @ 1.8GHz with 16 GB of RAM as measurement setup. To be able to measure as precisely as possible and lessen the impact of MPLAB's simulator start-up, we took the following precautions:

- we have created 4 long-running C programs that stress the ALU unit of the PIC10F. The programs were compiled with latest available compiler (XC8 2.20), but the resulting hex-file was simulated
- all peripherals were disabled, including the WDT
- we used the mdb (command-line version of MPLAB) to cancel the GUI impact
- measurement was performed 10 times for each case: timer started when the mdb reported "Running" state and was stopped when the mdb hit the first time, the breakpoint at the infinite loop
- the same hex-file generated by the compiler toolchain, was used for both mdb and our simulator (to ensure exactly same instructions are ran, in the same order)
- same machine (i7-8550U / 16GB RAM, Ubuntu LTS 18.04) was used when running all performance-oriented programs

All performance-oriented programs were following the recipe:

```
int main() {
    for (short int ii = 0; ii < 10000; ii++)
        some_alu_computation;
    while(1); // a breakpoint set here
}
```

A Java-based test bench program was developed to start a new thread with the mdb running a command script where a breakpoint was inserted in the when the infinite loop has been hit; the program waited for the mdb to report "Running" state and the moment the breakpoint was hit. The measurements were performed 10 times, taking the best, the worst and average time. The same setup was used for timing our simulator.

Four applications were developed and benchmarked, that were specifically designed to stress the ALU, and to be long enough to suppress any simulator launch latency or measurements resolution. The fifth was a very short app, used to measure the measurement precision of both simulators.

Application	Number of instructions ran until end
SPEED1.hex	20055018
SPEED2.hex	17314779
SPEED3.hex	4759216
SPEED4.hex	14895025
SPEED5.hex	6

Table 1. Number of simulated instructions, for each application

4. Results

The code we used is located here: https://gitlab.dcae.pub.ro/research/simd_pic_accelerator/-/tree/ximd.
We ran each measurement 10 times, keeping min / max and then computing the avg values (in milliseconds).
MPLAB simulator was used via mdb, its command-line tool, to avoid the GUI overhead.

	MPLABv5.35 simulator			Our implementation of simulator		
Application / Time	Min (ms)	Avg (ms)	Max (ms)	Min (ms)	Avg (ms)	Max (ms)
SPEED1.hex	5646	5882	6271	195	236	368
SPEED2.hex	6904	7162	7604	217	269	408
SPEED3.hex	6124	6583	7319	208	259	407
SPEED4.hex	1904	2098	2405	54	64	77

Table 2. Speed comparison between MPLAB's simulator and ours

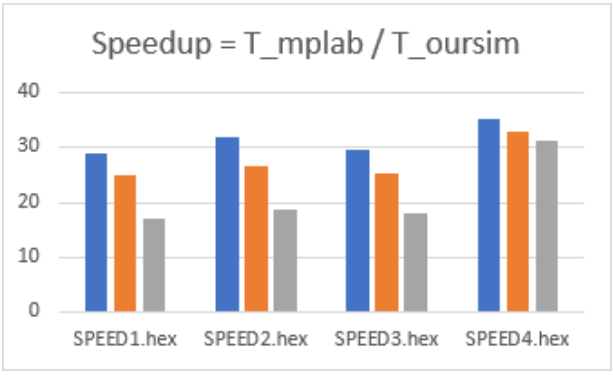

Figure 3. Speedup of our implementation against MPLAB's simulator

In figure 2, we checked whether running MPLAB with multiple instances, can improve performance: it can but not 4x, only 2x, no matter how many instances we would create and run in parallel on our 4-core machine. The conclusion is that MPLAB is using always only two cores.

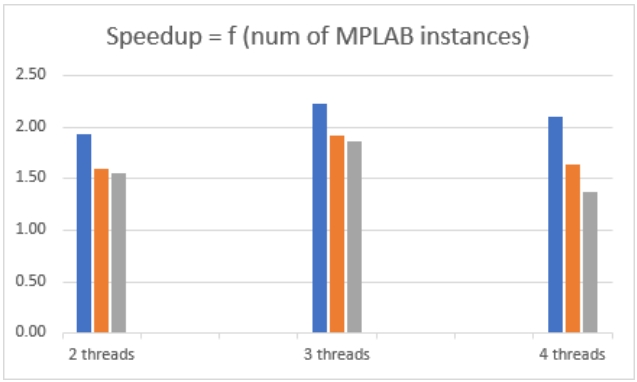

Figure 4. Speedup obtained while running multiple MPLAB instances

In addition, we want to see how much faster are the simulators against the real hardware. Our simulator proved to be at least 18 times faster, depending on the application, but MPLAB's simulator somewhat struggled to beat the hardware (PIC10F206) with 0.7 – 7.1x, even if it was running on a 4-core 1.8 GHz x86 machine (which is at least 1000 times more powerful).

Application	Speedup (Our Sim vs HW)	Speedup (MPLAB vs hw)
SPEED1.hex	85.0	3.4
SPEED2.hex	64.4	2.4
SPEED3.hex	18.4	0.7
SPEED4.hex	232.7	7.1

Table 3. Speed comparison of simulators against the 4 MHz hardware MCU

Lastly, we have built an empty program (main + infinite loop) to measure the possible latency introduced in our measurement, by not being able to time specifically the moment the simulator starts simulating and the simulator's stop moment: only 2 microseconds in our simulator, and around 15 milliseconds for the MPLAB, which was to be expected (the synchronization between the measurement setup and MPLAB is via Linux I/O system, as they are different processes)

5. Conclusions

We have built a fast, instruction-accurate PIC10F core simulator with its components (Intel HEX-file parser, instruction decoder and interpreter, N-core MIMD data representation (program memory, registers) running in multithreaded environment), a testbench of hex-files produced either by the assembler or the compiler to ensure correct functional behavior (tested when running in both MPLAB or our simulator) and found that our simulator is at least 26 times faster that the MPLAB's simulator (running in the same conditions: single-instance, same machine).

References

[1] PIC10F datasheet: http://ww1.microchip.com/downloads/en/DeviceDoc/40001239F.pdf

[2] https://cpulator.01xz.net/

[3] https://github.com/pddring/cpu-simulator

[4] D. Skrien, "CPU Sim 3.1: A tool for simulating computer architectures for computer organization classes" DOI 10.1145/514144.514731, Journal on Education Resources in Computing, dec. 2001

[5] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness,D. R. Hower, T. Krishna, S. Sardashti,et al., "The gem5 Simulator,"SIGARCH Comp.Arch. News, vol. 39, pp. 1–7, May 2011

[6] W. Heirman, T.E.Carlson, Sinper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation DOI: 10.1145/2063384.2063454

[7] K. Vollmar, P. Sanderson "MARS: an education-oriented MIPS assemble language simulator", ACM SIGCSE Bulletin 38(1):238-243, March 2006

[8] D.Sanchez, C.Kozyrakis "ZSim: fast and accurate microarchitectural simulation of thousand-core systems", ACM SIGARCH Computer Architecture News 41(3): 475, July 2013

[9] A. Akram, L.Sawalha Ä Comparison of x86 Computer Architecture Simulator", IEEE 34th International Conference on Computer Design (ICCD), Nov 2016

[10] J.Ma, L.Yu, J.M.Ye, T.Chen "MCMG simulator: A unified simulation framework for CPU and graphic GPU", Journal of Computer and System Sciences, vol 81, issue 1, Feb 2015, pp 57-71

[11] J.Power, J.Hestness, M.S.Orr, M.D.Hill, D.A.Wood "gem5-gpu: A Heterogenous CPU-GPU Simulator", IEEE Computer Architecture Lettters, Jan-June 2015, pp.34-36, vol 14

[12] K. Skadron, M. Martonosi, D. I. August, M. D. Hill, D. J. Lilja, and V. S. Pai, "Challenges in Computer Architecture Evaluation,"Computer, vol. 36, pp. 30–36, August 2003