Krzysztof SIECZKOWSKI, **Tadeusz SONDEJ**
ELECTRONIC DEPARTMENT, MILITARY UNIVERSITY OF TECHNOLOGY THEM. JAROSLAW DABROWSKI
2 Gen. Sylvester Kaliski St., 00-908 Warsaw 49, Poland

A method for testing the performance of a dual-core microcontroller with integrated program and data memory

Abstract

The paper presents a set of test procedures for performing an extended test of dual-core microcontroller performance. The procedures described enable the performance of the system to be determined with regard to memory organization, mutual influence of cores and ability to verify the impact of code optimization level. The experimental trials were performed using an LPC4357 microcontroller with onboard FLASH and SRAM memory. The results obtained showed a slight performance drop for the various microcontrollers operating simultaneously in typical conditions. However, with frequent references to a shared memory, the performance drop may reach as much as 80%.

Keywords: microcontroller, multicore, performance, benchmark.

1. Introduction

Modern microcontrollers (MCU) or microprocessors (MPU) have increasingly advanced CPUs (Central Processing Units) as well as an integrated data memory (usually SRAM), program memory (usually FLASH) and a variety of peripherals. The performance of an MPU or MCU may be increased by multiplying CPU units for parallel processing [1]. That is why multi-core processors began to be introduced many years ago. These solutions were originally dedicated for PCs only. As the complexity and popularity of FPGA systems and mobile devices rose, multi-core systems began to be applied for a wider group of devices, including software processors [2]. For several years now, manufacturers have been offering multi-core microcontrollers, currently still rare. Estimating the performance of MPUs or MCUs is quite a complex but still relevant process. It is particularly difficult in multi-core systems [3-6]. It is not enough to merely perform a theoretical analysis of the core operating frequency and an analysis of the available memory resources and peripherals. Additional performance verification is required in the hardware resources. There are many algorithms for testing microprocessor systems [7], however they may not always be used on multi-core systems. Furthermore, in MCUs, the construction of internal buses and memory blocks is different that for MPUs. In most cases, MCUs have no cache memory and there must be competition for data memory, program or internal bus access. It is particularly important in multi-core microcontrollers. This paper proposes tests for determining the performance of multi-core microcontrollers using the example of a type LPC4357 dual-core NXP MCU microcontroller [8]. A detailed analysis of the system configuration options was performed and it was revealed what tests should be carried out in order to examine the multi-core microcontroller unit.

2. Elements of the dual-core microcontroller unit performance test

The execution of a performance test usually requires the measurement of a specified algorithm's runtime. The set of test procedures should be complete and should enable determining the mutual influence of used data and program memories for different configurations. In addition, there should be a test to determine the effect of one MCU core on the other core's performance and to use well-known test functions, such as Dhrystone, Whetstone [7] or CoreMark [9], the memory access time should be tested. For this purpose, assembler instructions can be used, which allow the number of writes/reads to be set manually. The rest of this paper describes the tests included in the proposed comprehensive testing suite for multi-core microcontrollers.

2.1. Test of impact of memory banks used on system performance efficiency

This test measures the test function runtime on all cores simultaneously. The test is divided into stages, featuring different configurations of core connection to the microcontroller's existing data and program memory.

2.2. Test of mutual influence of cores

In this test, the MCU cores perform a standard testing function. The standard function is defined as an approximation of a typical algorithm executed by the system. The trial involves performing tests on only one core active, and then more than one core is started. This test aims to identify the impact of one core's performance on another core's performance in typical operating conditions of the microcontroller.

2.3. Test of interaction between cores for frequent references to a shared data memory

This test involves verifying the effects of one core's performance on another core's performance in specialized applications. In this test, one core executes standard procedures, while others perform specialized calculations. The cores share a memory.

2.4. Test of compiler optimization on system operation speed

This trial involves performing a test of the speed of the cores for program compilations with various levels of optimization. In each test, for a selected optimization level, the cores perform the same algorithm simultaneously.

3. Measurement setup

The comprehensive test of a multi-core microcontroller was performed using an example LCP4357 system, placed on an MCB4300 evaluation board by ARM Keil. The LPC 4357 microcontroller structure includes two heterogeneous Cortex-M4 (CM4) and Cortex-M0 (CM0) cores which can operate at a maximum frequency of 204 MHz. The LPC4357 microcontroller has on-board SRAM and FLASH memory. The memory is divided into banks. None of the cores has its own cache memory. Access to any memory bank is realized by an AHB Multilayer Matrix. Fig. 1 shows a block diagram of the test system. According to the manufacturer, the CM4 core is the master core, activated on power-up. The core CM0 is the slave core, triggered from the CM4 core. The program code for CM4 is always placed in the FLASH memory. For the core CM0, it may be in the FLASH or the SRAM memory. For the core CM0 to the reference SRAM as a program memory, the program must be originally copied by the core CM4 from the FLASH memory to the SRAM memory. For time measurement, an internal 32-bit RIT (Repetitive Interrupt

Timer) counter was used, operating at the same clock frequency as the cores.

This enables a resolution of approx. 5 ns with a range of approx. 21 s. After the test, the results were sent via UART to a computer. It was also possible to set the control pins which connected to a logic analyzer. The LEDs signaled the test execution stage. Fig. 2 shows the measuring position block diagram.

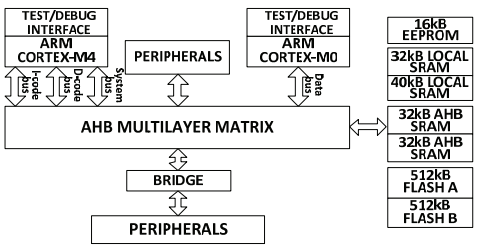


Fig. 1. Measurement setup - LPC4357 system block diagram

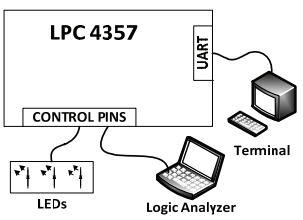


Fig. 2. Measurement setup - measuring position block diagram

Figs. 3 - 6 show the LPC4357 microcontroller configurations which allow the tests proposed earlier to be performed.

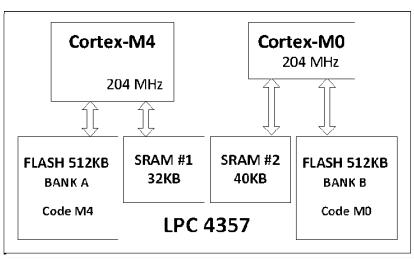


Fig. 3. Memory configuration in testing: - separate SRAM and FLASH

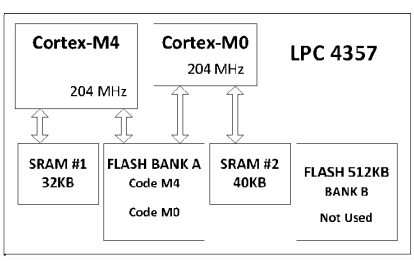


Fig. 4. Memory configuration in testing: - shared FLASH memory

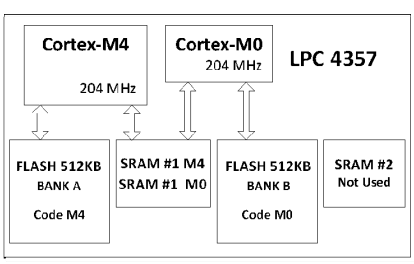


Fig. 5. Memory configuration in testing: - shared SRAM memory

The Whetstone algorithm and a specially prepared code in the assembler were used as the testing function. The Whetstone algorithm was chosen because many modern 32 - bit microcontrollers have FPU unit, particularly in the case of multi-core MCUs. Furthermore, the main goal of the tests was to examine the mutual influence of the two cores, while both MCU referred to the shared memory.

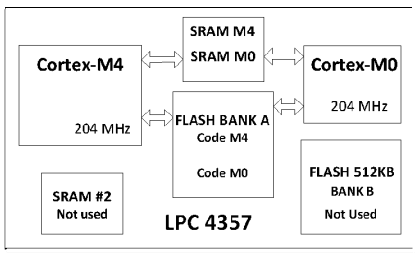


Fig. 6. Memory configuration in testing: - shared SRAM and FLASH memory

With respect to the test 2.1, four individual tests were carried out with different memory configurations. The configurations were as follows: the cores referred to individual (separate) SRAM and FLASH memories (Fig. 3), referred to a shared FLASH memory (Fig. 4), referred to a shared SRAM memory (Fig. 5), referred to a shared SRAM and FLASH memory (Fig. 6). The test 2.2 was first carried out for an active CM4 core (CM0 inactive), then for an active CM0 core (CM4 inactive), and the last part for the two cores CM0 and CM4 active. The test 2.3 was divided into three smaller tests. In the first test, the CM4 was executing the Whetstone algorithm, while at the same time the CM0 was performing assembly language instructions that referred to the memory once an interval. The testing procedure written in the assembler code is shown in Fig. 7.

```
void test_fun(void) {
   __asm {
      STR r0, r1
      STR r0, r1
      STR r0, r1
      LDR r0, r1
      STR r0, r1
      STR r0, r1
      STR r0, r1
      STR r0, r1
      LDR r0, r1
      STR r0, r1
   }
}
```

Fig. 7. Assembler code for the testing procedure

The assembler code consists of write instructions (STR) and reading instructions (LDR). The full cycle of testing instructions was performed 50 times. This allowed the number references core to be freely set to the program memory.

In the second test, while CM4 was performing the Whetstone test, CM0 was referring to the memory through assembly language instructions almost all the time. In the final test of this test, the whole time both cores referred to the memory very frequently. In each test included in this test, function (Whetstone or assembly language instruction set) execution by the core CM4 was measured. Three cases were examined: CM0 offline, CM0 referred to a different memory than CM4, CM0 referred to the same memory as CM4.

In the last test, the execution of the testing function by the cores CM4 and CM0 was measured for various compiler optimization levels. In addition, the cores are configured so that they can refer to the individual FLASH and SRAM banks. The default settings assume that each core has compilation optimization disabled and, in addition, the core CM4 has the FPU coprocessor disabled.

4. Results

All the tests used Keil's ARM MDK-Lite, version 4.73 development environment. Fig. 8a and 8b show the function execution time for the test presented in, respectively, 2.1 and 2.2 subsection.

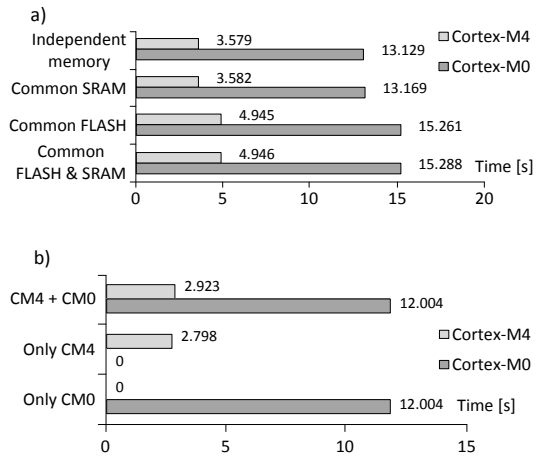


Fig. 8. Results for test presented in 2.1 (a) and 2.2 (b) subsection

The results show that the mutual impact of the cores for different memory configurations is not serious. Core performance degradation occurs when using a shared memory. Table 1a shows the obtained times of function execution by the core CM4 for three core configurations, for three instances of CM0 core operation. Table 1b shows the times of testing function execution by two cores working simultaneously for different optimization variants. The largest decrease in performance occurs with very frequent references to a shared data memory.

Tab. 1. Results for the test presented in 2.3 (a) and 2.4 (b) subsection

a)

	CM0 inactive	CM0 another SRAM	CM0 the same SRAM
CM4-Whetstone CM0-rare references	13.3%	13.3%	13.3%
CM4-Whetstone CM0-frequent references	13.3%	13.8%	13.7%
CM4, CM0-frequent references	52.7%	52.9%	100%

b)

	Cortex-M4	Cortex-M0
No optimization	15.7%	54.8%
Cross-Module Optimization (CMO)	15.6%	54.5%
CMO, Optimization: Level 3	14.9%	54.5%
CMO, Optimization: Level 3, Optimize for Time	14.8%	54.2%
CMO, Optimization: Level 3, Optimize for Time, FPU (CM4)	13.2%	54.2%
CMO, Optimization: Level 3, Opt. for Time, FPU (CM4), MicroLIB	28.3%	100%

5. Conclusions

The test microcontroller had an onboard FLASH memory and SRAM memory divided into banks. When the cores referred to a shared FLASH memory, the performance for the core CM4 and CM0 dropped by over 36% and 16%, respectively. In typical operation, the cores interacted with each other slightly. A slight performance drop (approx. 4%) for the core CM4 was observed when the core CM0 was active or remained offline. During execution of specialized tasks, when the cores often referred to a shared RAM data memory, a decrease in the efficiency of up to 80% was observed. The measurement setup was a laboratory setup for experimental purposes, but it showed the effect of cores competing for shared hardware resources. The proposed suite of test trials enables the actual performance of a multi-core microcontroller unit to be evaluated and the impact of memory organization on performance to be determined.

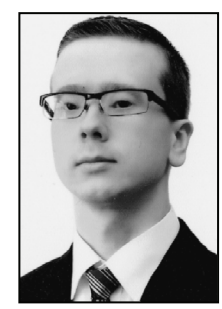
This work has been supported by the Military University of Technology, Warsaw, Poland, as a part of the project PBS918.

6. References

[1] Gebali F.: Algorithms and Parallel Computing. John Wiley & Sons, Hoboken, 2011.
[2] Sondej T., Zagoździński L., Pełka R.: Porównanie wydajności sprzętowego i programowego procesora w układzie FPGA Xilinx Virtex-4. Pomiary Automatyka Kontrola, nr 7bis, 2006, pp. 20-22.
[3] Gal-On S., Levy M.: Measuring Multicore Performance. Computer, Vol. 41, Issue: 11, 2008, pp. 99-102.
[4] Meihua Xu, Huacheng Yao, XuanHuan: Performance test of dual-core processor system based on NIOS II. 2012 IEEE Symposium on Electrical & Electronics Engineering (EEESYM), 2012, pp. 82-85.
[5] Dukyoung Yun, Sungchan Kim, Soonhoi Ha: A Parallel Simulation Technique for Multicore Embedded Systems and Its Performance Analysis. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 31, Issue: 1, 2012, pp. 121-131.
[6] Pengcheng Nie, Zhenhua Duan: Efficient and scalable scheduling for performance heterogeneous multicore systems. J. Parallel Distrib. Comput., Vol. 72, Issue: 3, 2012, pp. 353-361.
[7] Weicker R.P.: An overview of common benchmarks. Computer, Vol. 23, Issue: 12, 1990, pp. 65-75.
[8] NXP Semiconductors, LPC43xx ARM Cortex-M4/M0 multi-core microcontroller. User manual, Rev. 1.9, 2015.
[9] Gal-On S., Levy M.: Creating Portable, Repeatable, Realistic Benchmarks for Embedded Systems and the Challenges Thereof. ACM SIGPLAN NOTICES, Vol. 47, Issue: 5, 2012, pp. 149-152.

Received: 07.04.2015 Paper reviewed Accepted: 02.06.2015

Krzysztof SIECZKOWSKI, MSc



Born 20th July 1990 in Łowicz. Received his BSc and MSc degrees in electronics and telecommunications from Military University of Technology, Poland, in 2014 and 2015, respectively. His main scientific interests are measurement and analysis of biometric signals and energy-efficient microprocessor systems.

e-mail: ksieczkowski9007@gmail.com

Tadeusz SONDEJ, PhD

He received the M.Sc. degree in electronic engineering and the PhD degree in applied sciences from the Military University of Technology (MUT), Warsaw, Poland, in 1997 and 2003 respectively. Since 1998, he has been with the MUT, where he has been working on design and programming of embedded systems. His current interests are in the field of design, optimization and programming of System-on-Chip based digital systems, especially for biomedical applications.

e-mail: tsondej@wat.edu.pl