Proceedings of 2013 International Symposium on Embedded Technology (ISET 2013)
Daegu Metropolitan City, 23-24 May, 2013.

How to consolidate legacy system codes using ARM TrustZone on multi-core systems

Sung Bae Yoo[*] Se-Won Kim[*] Hyun Woo Lee[†] Suk Young Rho[†] Chuck Yoo[*]

[*]College of Information and Communication
Korea University, Seoul, Republic of Korea
E-mail: {sbyoo, swkim, chuckyoo}@os.korea.ac.kr

[†]Research & Development Division
Hyundai Motor Company, Seoul, Republic of Korea
E-mail: {stoneno, lhwpro}@hyundai.com

Abstract

Whenever embedded system is developed newly, a lot of legacy codes are created. It is difficult to consolidate legacy system codes in newly developed system. We argue what the problem is and propose how to consolidate. Proposed method uses ARM TrustZone on multi-core processor. We show that it is possible to consolidate easily by just a few code changes.

Keywords: ARM, TrustZone, multi-core, legacy system

1. Introduction

To meet the increasing requirements of users, embedded systems have had more complex software and hardware architecture. Embedded system designers have appended new software functions and consolidated them and legacy software into same hardware. Traditional approach is that new software and legacy software run on same operating system. However such consolidation method has three problems. First, if new software has a bug, whole system may be unavailable even though bug-free legacy software that proven by long-term field test will be in unstable state. Second, in case of real time legacy software, newly appended software can interfere real time scheduling because recent embedded systems are increasingly taking on characteristics of general-purpose systems [1]. Lastly, even though new software qualified system reliability by single execution test, co-execution with legacy software can cause undiscovered problems. Therefore it is difficult to consolidate legacy and new software into one system. In this paper, we present how to consolidate legacy system codes to newly developed systems using ARM TrustZone on multi-core systems. Our method has a small code change, compare to other studies [2].

2. Background : ARM TrustZone

ARM TrustZone divides a physical processor into two worlds: secure world and normal world. Each world has states of CPU and an address space (page table registers). One OS is isolated in one world. OS which needs security runs in secure world and GPOS (General Purpose OS) runs in normal world generally. After processor initialization, processor's first world is secure world. So, if normal world is needed, processor should switch its world to normal world. For this, monitor mode is added in CPU modes newly (ARM CPU modes: FIQ, IRQ, Abort, Kernel, User, and Undefined.). Monitor mode is a mode to switch to another world. Monitor mode is a part of secure world. Therefore, instruction which

can be executed in secure world only can be also executed in monitor mode. To enter into monitor mode, SMC (Secure Monitor Call) instruction should be executed or IRQ/FIQ should be occurred [3].

3. Proposed Method

ARM TrustZone is used for isolation between legacy system codes and newly developed codes. In multi-core systems, each core should be initialized. For example, in Linux, function "secondary_startup" initializes the cores. This function executed first when core power is up. In proposed method, new function is added in front of this function. The new function checks if now core is allocated in legacy codes. If it is not allocated, new function returns to original function. Otherwise, it changes CPU Mode to monitor mode and sets MVBAR (Monitor Vector Base Address Register) and interrupt controller. And it copies legacy codes into memory and jumps to entry of legacy codes. Proposed method runs legacy system codes (commonly, RTOS) in normal world and runs newly developed codes (commonly, GPOS) in secure world unlike section 2, as in shown Figure 1. It has following advantages. First, it is difficult to porting newly developed codes on normal world because of its drivers that uses registers only accessible in secure world. Second, it needs only a few modifications adding a function call in front of existing function. Third, legacy codes scheduling isn't interfered through dedicating core to normal world.

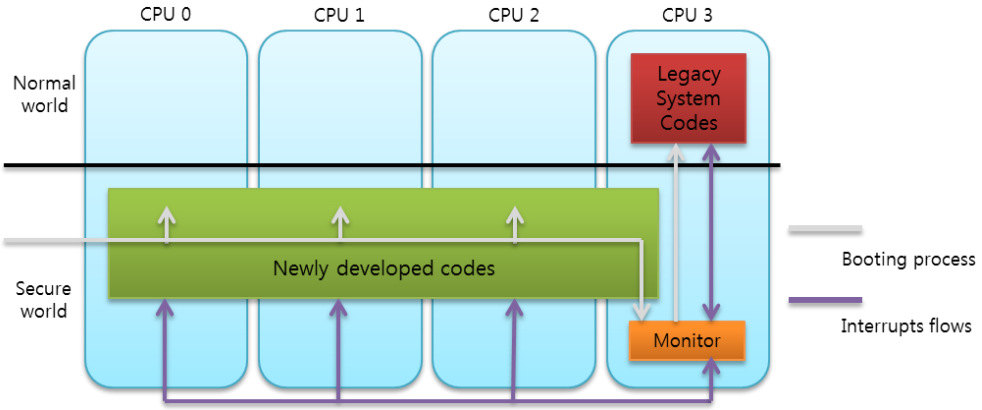

Figure 1. The architecture of proposed method

4. Conclusions

We consolidate uC/OS-II as legacy system codes on the real hardware device that uses Linux as newly developed codes. When we implement, it is possible to consolidate legacy system codes by just a few code changes. To insert our code, we modified only one file that includes secondary core initializing code.

Acknowledgement

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MEST) (No.2010-0029180) with KREONET.

References

[1] Gernot Heiser. 2008. The role of virtualization in embedded systems. In Proceedings of the 1st workshop on Isolation and integration in embedded systems (IIES '08), Michael Engel and Olaf Spinczyk (Eds.). ACM, New York, NY, USA, 11-16.

[2] Daniel Sangorrín, Shinya Honda, and Hiroaki Takada. Dual operating system architecture for real-time embedded systems. In Proceedings of OSPERT 2010, July 2010.

[3] ARM Limited. ARM security technology: Building a secure system using TrustZone technology. WhitePaper PRD29GENC-009429C, 2009