Poster: ARM Errata and their Software Workarounds

Nisal Menuka and Lin Zhong
Rice University, Houston, TX
{nisal.menuka, lzhong}@rice.edu

Like software, hardware also has bugs. These bugs, or *errata*, can cause unexpected behaviors, reducing performance and causing malfunctions in the entire system. As there is no way to fix an erratum after hardware is deployed, its harmful effects are often mitigated by software workarounds, usually in low-level software such as operating system. The goal of our work is to ensure system correctness and security against the harmful effects of errata and their workarounds. The first step is to systematically understand them in the wild.

In this poster, we present our early results from analyzing errata in the ARM Cortex-A family of microarchitecture, introduced by ARM itself, and their workarounds. We studied publicly available ARM errata documentations and examined source code of low-level software, namely Linux kernel, U-Boot and ARM Trusted firmware. ARM classifies errata in to three categories: A - critical error with no acceptable workaround available, B - significant error with acceptable workaround available and C - minor error. Figure 1 (Left) shows distribution of the errata we analyzed according to these categories.

Unfixed Errata and Spurious Workarounds

We found most errata do not have workarounds in the low-level software we examined. We found workarounds for only about 50 errata, less than 20% of those documented by ARM. Figure 1 (Right) shows the workarounds we identified according to applicable ARM cores. Out of the 115 Category B errata, less than half have workarounds. Recall these errata can cause significant errors and their workarounds are available according to ARM.

On the other hand, we found that workarounds intended for one architecture are sometimes enabled for another. For example, the official kernel repository for Cortex-A8 based BeagleBone Black [1] had three errata workarounds intended for Cortex-A9 in its default configuration. The same problem also was found in [2] and the maintainer disabled four spurious errata workarounds after we reported them.

How and Where Errata were Fixed

We found workarounds fix errata in two different ways. The more common way is to configure the processor during initialization so that the erratum would never occur. Such workarounds usually

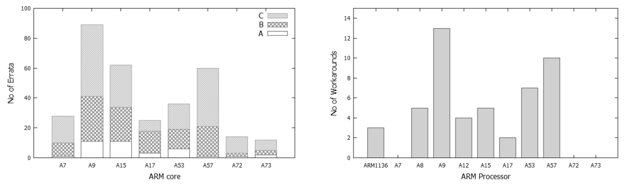

Figure 1: (Left) Errata distribution according to categories; (Right) Workarounds found in Linux kernel, U-Boot and ARM Trusted Firmware categorized according to applicable ARM core.

modify processor control registers. 22 out of 32 workarounds we identified in the Linux kernel falls in to this category and require the kernel to run in the secure state. For example, cause of ARM errata 742230 which affects Cortex-A9 is faulty execution of the Data Memory Barrier (`DMB`) instruction. The workaround for this erratum is to configure a secure state register to decode `DMB` as the Data Synchronization Barrier (`DSB`) instruction. `DSB` functionally subsumes `DMB`. This workaround can reduce system performance slightly.

The rest of the 32 workarounds modify necessary code, e.g., by adding new instructions and replacing instructions. For example, the workaround for ARM errata 754322 for Cortex-A9 adds a `DSB` instruction in the code for context switching, which increases the latter's overhead. Note that this workaround was enabled spuriously for Beaglebone Black in [2], where we measured the overhead to be 1.3% on average for each context switch.

Errata workarounds that modify code must be applied in the kernel, e.g., that for ARM errata 753422, for obvious reasons. However, errata workarounds that configure the secure state registers appear to be better applied before the Linux kernel boots, in the bootloader. This is because the bootloader itself may trigger an erratum and must work around it. Additionally, if the kernel boots in a non-secure state, it cannot write into secure state registers. The bootloader is more likely to run in the secure state. However, with the ARM TrustZone technology, a normal world bootloader does not run in the secure state. As a result, workarounds that require access to secure state registers must be applied even early, e.g., in the firmware.

We observe that some workarounds for the same errata exist in both the Linux Kernel and the bootloader. The Linux kernel community has already reached the consensus not to accept any more errata workarounds if they can be applied in the bootloaders.

References

[1] The official beagleboard and beaglebone kernel repository. https://github.com/beagleboard/linux. Accessed 03/01/2017.

[2] Robertcnelson/bb-kernel. https://github.com/RobertCNelson/bb-kernel. Accessed 03/01/2017.

MobiSys'17 June 19-23, 2017, Niagara Falls, NY, USA
© 2017 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-4928-4/17/06.
DOI: http://dx.doi.org/10.1145/3081333.3089323