Advanced Science and Technology Letters
Vol.75 (SIP 2014), pp.41-44
http://dx.doi.org/10.14257/astl.2014.75.11

Design of Dynamic Detector for Atomicity Races in ARINC-653 Applications

Eu-Teum Choi[1], Se-Won Park[2], Ok-Kyoon Ha[3], and Yong-Kee Jun[2]

[1] Aero Master Corporation, 345, Haeansaneop-ro, Sacheon, Republic of Korea
[2] Department of Informatics, Gyeongsang National University, 501, Jinjudea-ro, Jinju, Republic of Korea
[3] Engineering Research Institute, Gyeongsang National University, 501, Jinjdea-ro, Jinju, Republic of Korea
slateblue33@naver.com, swpark3179@nate.com, {jassmin, jun}@gnu.ac.kr

Abstract. This paper presents a dynamic detector, called Race-653 that locates atomicity races in ARINC-653 applications using an on-the-fly analysis technique. Race-653 consists of 653-Monitor and 653-Detector modules. The 653-Monitor collects monitored information during an execution of the applications, such as processes, semaphores, and accesses for each shared resource. The 653-Detector reports atomicity races by checking violations of a synchronization discipline based on semaphore. We implemented the detector as a PIN tool using PIN binary instrumentation framework and evaluated accuracy of the tool on a simulation system for integrated modular avionics.

Keywords: ARINC-653, IMA, reliability, atomicity races, dynamic detection.

1 Introduction

The ARINC-653 standard [1] for integrated modular avionics (IMA) [2] defines four intra-partition communication services: buffer, blackboard, semaphore, and event. The semaphore service allows atomic regions of program executions and synchronization objects to provide access to the shared partition resources for concurrent processes. Atomicity races [3] in ARINC-653 applications may occur when two processes which includes each atomicity region access to a shared resource without explicit synchronization, such as semaphore. It is important to detect atomicity races for guaranteeing safety critical executions of ARINC-653 applications, because they may lead to unpredictable behaviors or faults.

2 Background

ARINC-653 standard specifies two communication mechanisms for intra-partition communication [1]. The first one allows communication among processes on a partition via buffers and blackboards. The other is for process synchronization such as semaphores and events. The semaphore service for ARINC-653 applications is a

ISSN: 2287-1233 ASTL

synchronization object commonly used to create the atomic execution region and to control process accesses to the shared partition resources.

It is hard to locate atomicity races because there are many possible execution paths of the program and a lot of the defects are hard to reproduce. Hence, automated tools which employ sophisticated techniques for analyzing the program executions are used to locate atomicity races for debugging instead of identifying the bug manually. A previous tool, CodeSonar [4], which uses a static analysis technique identifies concurrency bugs, such as data races and deadlocks, in applications for avionics. The static analysis tool is sound, but imprecise due to their false alarms since it evaluates only source codes and impossible execution paths without any execution of the program. For the reliability of ARINC-653 applications, programmer must analyze the results reported by CodeSonar.

3 Dynamic Detector for Atomicity Races

To detect dynamically atomicity races in ARINC-653 applications, it must monitor the execution information of the application, such as processes, semaphores, and accesses for each shared resource, and analyze monitored information. In this paper, we present a dynamic detector, called Race-653 that locates atomicity races in ARINC-653 applications using an on-the-fly analysis technique.

Fig. 1 depicts the architecture of Race-653. Our dynamic detector consist of two modules: Monitor module which monitors these execution information during an execution of the application and Detector module which locates atomicity races by checking violations of a synchronization discipline based on semaphore.

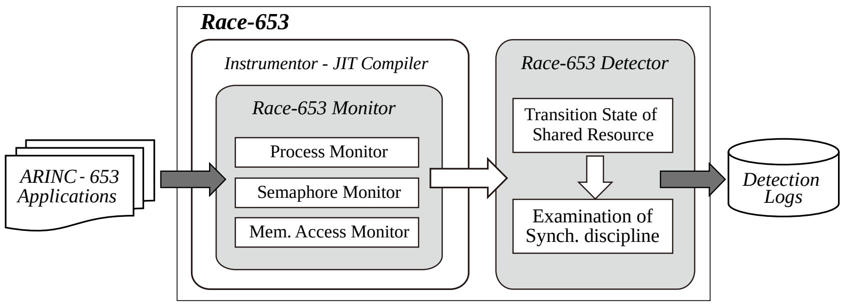

Fig. 1. The overall architecture of Race-653

Race-653 Monitor. Race-653 considers PIN software framework for dynamic binary instrumentation of ARINC-653 applications. The Monitor module consists of three functions, process monitor, semaphore monitor, and memory access monitor, which are used in an execution of the program by JIT compiler. Thus, this module monitors function calls related to process, such as creation, exit, and execution, and memory accesses to shared resources, such as read/write of shared variables and pointers. It also monitors semaphore services of ARINC-653 considering `CREATE_SEMAPHORE`, `WAIT_SEMAPHORE`, and `SIGNAL_SEMAPHORE`.

Race-653 Detector. Race-653 reports atomicity races using an on-the-fly analysis technique [5-6], called lock-set based analysis [5]. The Detector module checks violations of a synchronization discipline which stipulates that any two different processes access a shared resource with a common semaphore. For detecting atomicity races, the Race-653 Detector maintains a candidate set of semaphores *Cx* that is held by all processes during a program execution for a shared resource *x*. Therefore, it locates an atomicity race whenever any two accesses on different processes access a shared resource with at least one write, and the two accesses are not protected by a common semaphore. Given two processes *Pi* and *Pj* that access to a shared resource *x* is occur an atomicity race if they satisfy following condition: $(Pi = Write \lor Pj = Write) \land (Pi \neq Pj \land Cx = \emptyset)$, where *Cx* maintains a set of locks by intersecting itself with the set of semaphores held by the current process.

4 Experimentation

We implemented Race-653 as a PIN tool on top of PIN binary instrumentation framework which uses a just-in-time (JIT) compiler to recompile target binaries for dynamic instrumentation. Our implementation and experimentation were carried on a system with Intel Xeon Dual-core 2 CPUs and 8GB main memory under a simulation system for integrated modular avionics (SIMA).

To evaluate the accuracy of Race-653, we developed synthetic programs for ARINC-653. These synthesis consist of two criteria, such as using shared variables and using pointers of shared resources in atomicity regions. Table 1 shows design of these synthesis. For the experiments, these synthesis was recorded for ten executions, and each synthetic ran with two partitions on the standalone mode of the SIMA.

Table 1. Design of synthetic programs for ARINC-653 applications

Types	Races	Number	Name	Types	Races	Number	Name
	○	01	SV-R01		○	01	SP-R01
Shared	○	02	SV-R02	Shared	○	02	SP-R02
Variable	×	01	SV-N01	Pointer	×	01	SP-N01
	○	02	SV-N02		○	02	SP-N02

We acquired the results of reported atomicity races to evaluate the precision of our dynamic detector. The results appear in Table 2. From the table, Race-653 precisely detect atomicity races in SV-R01, SV-R02, SP-R01, and SP-R02 which are incurring an atomicity race in each program execution. Naturally, it also precisely detect no atomicity races in SV-N02 and SP-N02 which use a common semaphore for an explicit synchronization. However, Race-653 reports false positives in case of SV-N02 and SP-N02 because these programs use implicit synchronization by two different semaphores. For optimized executions, these programs should be modified to use only one of two semaphores due to the fact that it is safe to use one semaphore for a shared resource and may incur more serious faults, such as dead locks.

Table 2. The results of atomicity race detection under Race-653

Types	Races	Name	Results	Types	Races	Name	Results
	○	SV-R01	○		○	SP-R01	○
Shared	○	SV-R02	○	Shared	○	SP-R02	○
Variable	×	SV-N01	×	Pointer	×	SP-N01	×
	×	SV-N02	○		×	SP-N02	○

5 Conclusion

Atomicity races must be detected for the reliability of safety critical system, such as avionics. To detect precisely atomicity races in ARINC-653 applications, it must monitor execution information of the application, such as processes, semaphores, and accesses for each shared resource, and analyze monitored information. We presented Race-653 which is a dynamic detector for atomicity races in ARINC-653 applications using an on-the-fly analysis technique and implemented the detector as a PIN tool using PIN binary instrumentation framework on top of SIMA. The evaluated results show that Race-653 is useful whether an ARINC-653 application runs into atomicity races due to the fact that the detector is sound and provides reduced false positives.

Acknowledgments. This work was supported by the BK21 Plus Program (Research Team for Software Platform on Unmanned Aerial Vehicle, 21A20131600012) funded by the Ministry of Education (MOE, Korea) and National Research Foundation of Korea (NRF) and was also supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2013R1A1A2011389).

References

1. Airlines Electronic Engineering Committee (AEEC): Avionics Application Software Standard Interface - ARINC Specification 653 - Part 1 (Supplement 2 - Required Services), ARINC (2006)
2. Ha, O.-K., Tchamgoue, G.M., Suh, J.-B., Jun, Y.-K.: On-the-fly healing of race conditions in ARINC-653 flight software. In: Proceedings of the 29th Digital Avionics Conference. pp. 5.A.6-1 - 5.A.6-11. IEEE, Salt Lake City (2010)
3. Netzer, R.H.B., Miller, B.P.: What are race conditions?: Some issues and formalizations. ACM Lett. Program. Lang. Syst. 1, 74-88 (1992)
4. GrammaTech, CodeSonar, http://www.grammatech.com/codesonar
5. Savage, S., Burrows, M., Nelson, G., Sobalvarro, P., Anderson, T.: Eraser: A dynamic data race detector for multithreaded programs. ACM Trans. on Comput. Syst. 15, 391-411 (1997)
6. Ha, O.-K., Kuh, I.-B., Tchamgoue, G. M., Jun, Y.-K.: On-the-fly detection of data races in OpenMP programs. In: Proceedings of the 10th Workshop on Parallel and Distributed Systems: Testing, Analysis, and Debugging. pp. 1-10. ACM, Minneapolis, MN. (2012)