Leveraging Virtual Prototypes and Metamorphic Testing for Verification of Embedded Graphics Libraries [*]

Christoph Hazott, Florian Stögmüller, Daniel Große
Institute for Complex Systems, Johannes Kepler University Linz, Austria
christoph.hazott@jku.at, stoegmueller.f@gmail.com, daniel.grosse@jku.at

Abstract

In this extended abstract, we summarize our work from [6], where we proposed a novel approach focusing on integration testing of embedded graphics libraries. We leveraged *Virtual Prototypes* (VPs) and integrated them with *Metamorphic Testing* (MT). Additionally, we eliminated the need for physical hardware by virtualizing the displays within the virtual environment. On an extended RISC-V VP for the GD32V platform we found 15 distinct bugs for the widely used TFT_eSPI embedded graphics library.

1 Extended Abstract

In embedded systems, *Software* (SW) is closely tied to the *Hardware* (HW) it runs on. An important part of the embedded SW is the *Firmware* (FW) which provides low-level control for the device's specific HW. As many embedded systems include displays to visualize information as well as to enable easy interaction, FW libraries for these displays are very crucial components. Over the years, these FW libraries have become more and more powerful: One of the first implementations, where the term FW was already used, has been presented in [14]. This work interfaced different graphical terminals via the FW. Today, much more complex functionality is integrated in these *embedded graphics libraries*, extending the fundamental support of drawing simple geometric elements on different displays to advanced objects, fonts, and even features like sprites. Moreover, optimizations for different HW architectures are performed improving the rendering performance. Due to this increasing feature complexity, the importance of verification of embedded graphics libraries progressively amplifies.

To address the growing complexity, advancements in simulators and emulators are leveraged to enable the adaption of SW testing strategies for FW testing. The most fundamental strategy which is adopted is testing of individual components and functions in isolation, also referred to as unit testing [12]. While this approach is effective at uncovering numerous bugs, blind spots emerge because unit testing does not capture complex interactions among components, and overlooks integration challenges that can lead to functional and performance issues. *Integration testing* complements unit testing by focusing on these blind spots, capturing the interfaces and interactions among components [13, 11].

For successful integration testing on embedded devices, test inputs forcing the system into potential error cases have to be defined as well as comprehensive reference models are necessary to determine the test result. The latter may be very difficult to create as the effort to implement such models increases if more and more components (e.g. deeply layered functions) interact and if complex SW-to-HW stacks are involved. This challenge is well-known as *test oracle problem* [1] and in case of embedded graphics libraries it is even worse, as it typically means visual inspection of the results when executed on the HW. Altogether, this makes automating the verification very complicated.

In this extended abstract, we summarize our work from [6], where we present a novel approach focusing on integration testing of embedded graphics libraries. As first component, *Virtual Prototypes* (VPs) are leveraged targeting the need of HW for visual inspection. VPs are predominantly modeled in SystemC, a standardized C++ library [10]; for a broader overview on SystemC we refer the reader to [3, 8, 4]. VPs enable the development and execution of SW production code as if the physical HW were present on the table [2].

The second essential component of our approach to face the test oracle problem is *Metamorphic Testing* (MT) [16]. MT circumvents the need of a reference model and found an impressive amount of bugs in different domains [17]. MT is based on the core principle of using known relationships and properties among inputs and outputs to design effective test cases. These relationships are formalized within so-called *Metamorphic Relations* (MRs). An example of a MR for embedded graphics libraries is drawing a line: Assuming our line has the start position $A(x_1, y_1)$ and end position $B(x_2, y_2)$, if we switch A and B the output should be the same line. Based on this relation we are able to insert concrete values for x_1, y_1 and x_2, y_2 to generate *Metamorphic Test Cases* (MTCs). With this idea in mind we created 21 generic MRs for embedded graphics libraries and devised a framework to automatically generate MTCs based on the random testing strategy.

Fig. 1 provides an overview of our automated MT framework. In the upper left part (gray box), we consult the

[*]This work has partially been supported by the LIT Secure and Correct Systems Lab funded by the State of Upper Austria.

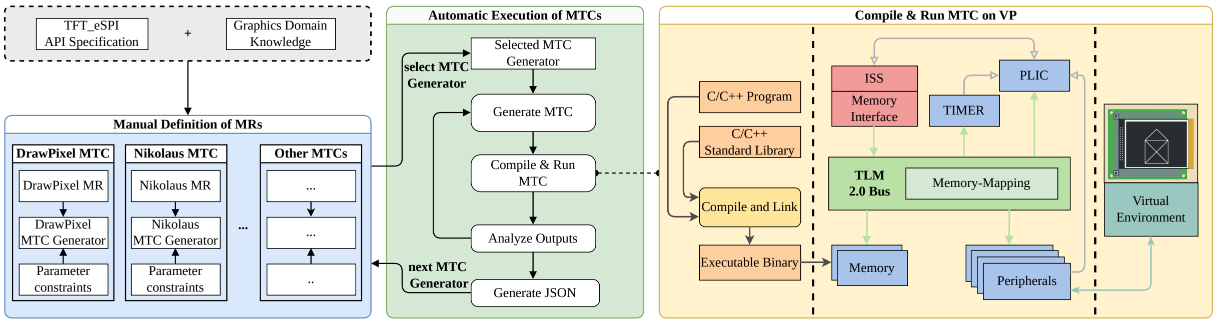
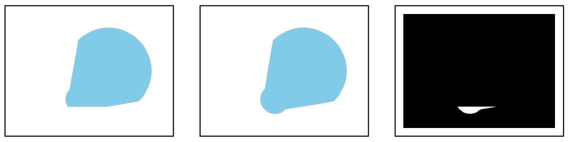


Figure 1 Automated MT Framework



(a) Source FW **(b)** Follow-up FW **(c)** Difference

Figure 2 `drawWedgeLine` Bug

API specification of the TFT_eSPI, and combine this information with general knowledge from the graphics domain. The result serves as basis to define the MRs (cf. blue box). In addition, we need to implement a generator for each MR within our MT framework. According to a given MR the generator will produce concrete values for two FW, i.e. a source FW and a follow-up FW, constituting a MTC. Since the generation is automated, multiple MTCs per MR can be generated.

The main part of our MT framework (green box in the center), selects and starts the different MTC Generators. The generated source FW and follow-up FW of an MTC are then compiled and executed on the VP. In Fig. 1, the VP is depicted within the yellow box on the right. To compile and run our MTCs we enhanced the RISC-V VP from [7, 9] to support the GD32V and added a virtual display. The enhancements are available in the open-source *RISC-V VP++* on GitHub[1] together with general improvements [15]. The developed virtual display can also capture images. This is used to store the outputs produced by the source FW and the follow-up FW, which are compared at the end on an MTC Execution.

In the evaluation, we were able to find 15 distinct bugs for the widely used TFT_eSPI embedded graphics library, confirming the strength of our approach. One of those bugs is shown in Fig. 2. This is also the most severe one found by our approach. When calling the `drawWedgeLine` method of TFT_eSPI with a specific combination of valid arguments, the resulting wedge line is incomplete. We found this issue to be infrequent, occurring only twice out of 817 MTCs, which suggests that the bug is only triggered by very specific arguments.

In future work, we plan to integrate a detailed test analysis based on dynamic instrumentation techniques as recently presented in [5].

[1]`https://github.com/ics-jku/riscv-vp-plusplus`

2 Literature

[1] E. T. Barr, M. Harman, P. McMinn, M. Shahbaz, and S. Yoo. The oracle problem in software testing: A survey. *IEEE Transactions on Software Engineering*, 41(5):507–525, 2015.

[2] T. De Schutter. *Better Software. Faster!: Best Practices in Virtual Prototyping*. Synopsys Press, March 2014.

[3] D. Große and R. Drechsler. *Quality-Driven SystemC Design*. Springer, 2010.

[4] M. Hassan, D. Große, and R. Drechsler. *Enhanced Virtual Prototyping for Heterogeneous Systems*. Springer, 2022.

[5] C. Hazott and D. Große. DSA monitoring framework for HW/SW partitioning of application kernels leveraging VPs. In *Design and Verification Conference and Exhibition Europe*, 2023.

[6] C. Hazott, F. Stögmüller, and D. Große. Verifying embedded graphics libraries leveraging virtual prototypes and metamorphic testing. In *ASP Design Automation Conf.*, pages 275–281, 2024.

[7] V. Herdt, D. Große, H. M. Le, and R. Drechsler. Extensible and configurable RISC-V based virtual prototype. In *Forum on Specification and Design Languages*, pages 5–16, 2018.

[8] V. Herdt, D. Große, and R. Drechsler. *Enhanced Virtual Prototyping: Featuring RISC-V Case Studies*. Springer, 2020.

[9] V. Herdt, D. Große, P. Pieper, and R. Drechsler. RISC-V based virtual prototype: An extensible and configurable platform for the system-level. *Journal of Systems Architecture - Embedded Software Design*, 109:101756, 2020.

[10] *IEEE Standard SystemC Language Reference Manual*. IEEE Std. 1666, 2011.

[11] C. Kaner and R. L. Fiedler. *Foundations of Software Testing*. Cengage Learning, 2013.

[12] P. Liggesmeyer and M. Trapp. Trends in embedded software engineering. *IEEE Software*, 26(3):19–25, 2009.

[13] A. Marrero Perez and S. Kaiser. Integrating test levels for embedded systems. In *TAICPART*, pages 184–193, 2009.

[14] G. A. Rose. "intergraphic," a microprogrammed graphical-interface computer. *IEEE Trans. on Electronic Comp.*, EC-16 (6):773–784, 1967.

[15] M. Schlägl and D. Große. GUI-VP Kit: A RISC-V VP meets Linux graphics - enabling interactive graphical application development. In *ACM Great Lakes Symposium on VLSI*, pages 599–605, 2023.

[16] S. Segura and Z. Q. Zhou. Metamorphic testing 20 years later: A hands-on introduction. In *International Conference on Software Engineering*, pages 538–539, 2018.

[17] S. Segura, G. Fraser, A. B. Sanchez, and A. Ruiz-Cortés. A survey on metamorphic testing. *IEEE Transactions on Software Engineering*, 42(9):805–824, 2016.