Towards mixed time- and event-triggered execution for safety-critical embedded systems

Graham Hemingway
Vanderbilt University
2015 Terrace Place
Nashville, TN, 37203

Joseph Porter
Vanderbilt University
2015 Terrace Place
Nashville, TN, 37203

Nicholas Kottenstette
Vanderbilt University
2015 Terrace Place
Nashville, TN, 37203

Chis vanBuskirk
Vanderbilt University
2015 Terrace Place
Nashville, TN, 37203

Gabor Karsai
Vanderbilt University
2015 Terrace Place
Nashville, TN, 37203

Janos Sztipanovits
Vanderbilt University
2015 Terrace Place
Nashville, TN, 37203

ABSTRACT

Time-triggered execution provides a theoretically strong basis for the design of safety-critical embedded systems. Ideally, the design of such systems would not be limited to strictly time-triggered execution semantics. Indeed, in many situations event-triggered semantics provide a more natural and intuitive approach for designers. What is needed is a meaningful blending of time- and event-triggered execution semantics. Any approach should maintain the analytic behavior and robust fault tolerance of time-triggering while providing the flexibility of event-triggering. Obviously, any such solution will have trade-offs and compromises compared to purely time- or event-triggered execution.

In this paper we present an environment for modeling, analyzing, and executing systems with mixed time- and event -triggered semantics. First, we present the ESMoL modeling language which provides an intuitive means of describing mixed execution-type systems. Integrated into this environment is a tool for performing static schedulability analysis of time- and event-triggered systems. Once a system has been designed and analyzed, the ESMoL toolchain can synthesize a complete implementation that realizes the system and its behaviors.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

time-triggered, event-triggered

1. INTRODUCTION

Introduction here.

1.1 Example Model

Quadrotor example.

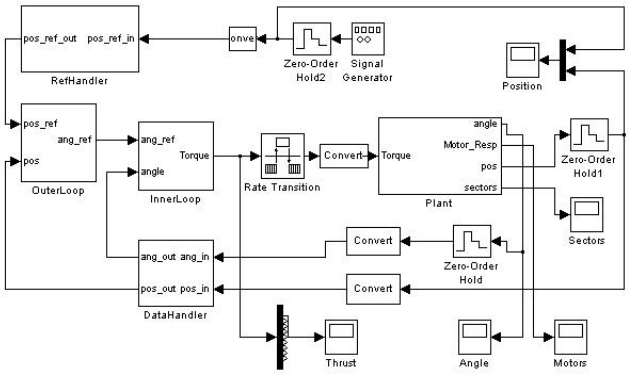

Figure 1: High-level quad-rotor controller model

2. BACKGROUND

Talk about the background here.

Talk about TTA maybe

2.1 ESMoL Toolchain

Talk about ESMoL

3. STATIC SCHEDULABILITY ANALYSIS

Talk about it here.

4. FRODO VIRTUAL MACHINE

The ESMoL modeling language and its attendant analysis tools capture the design intent of an embedded system and provide some reassurance as to its behavior. The effectuation of an ESMoL model into correct executable code is itself a significant challenge. The functional code necessary to implement the software components is synthesized, but the time-triggered behavior assumed by the time-triggered

architecture model of computation must also be realized. Time-triggered execution could be supported at the operating system level, but few such systems are available [2, 1]. Instead, an alternate approach is to implement a time-triggered execution layer on top of standard OS primitives [4, 3, **?**]. As long as all underlying TTA assumptions are maintained, this approach provides a portable and light-weight platform for robust execution. The FRODO v2 virtual machine (VM) provides time-triggered execution semantics for ESMoL models and has been integrated into the overall ESMoL toolchain.

5. RESULTS

There are three distinct results we present.

5.1 Schedulability

Here are some schedulability results.

5.2 FRODO Timing

FRODO is meant to be a highly portable platform. As such, we have built versions of FRODO capable of running on a wide variety of underlying operating systems (OS). For each OS we conduct a series of time-triggered assessments to determine the timing accuracy of the underlying OS and thus gain an understanding of expected performance should a model be deployed onto that OS. For each OS a test is run that attempts to execute >100,000 tasks and message transfers, all on a strictly time-triggered basis. During each execution the expected versus actual time is recorded. Post-test analysis gives each platform a minimum, maximum, and average variance between the expected and actual times. The results are shown in Table 1.

	Min(μs)	**Max(μs)**	**Avg(μs)**
OS X 10.6.4	<1	<1	<1
Windows XP SP 3	4747	7746	40
Linux 2.6.X	35	44	32
FreeRTOS 5.X	1	1	1

Table 1: FRODO Host Platform Timing Results

The maximum timing variance is the most useful metric from this test. The maximum variance provides a reasonable upper-bound for the amount of timing error we can expect for FRODO running on a given operating system.

5.3 Quadrotor Timing

Here are some example model results

6. CONCLUSIONS

This paragraph will end the body of this sample document. Remember that you might still have Acknowledgments or Appendices; brief samples of these follow. There is still the Bibliography to deal with; and we will make a disclaimer about that here: with the exception of the reference to the LaTeX book, the citations in this paper are to articles which have nothing to do with the present subject and are used as examples only.

7. ACKNOWLEDGMENTS

This section is optional; it is a location for you to acknowledge grants, funding, editing assistance and what have you. In the present case, for example, the authors would like to thank Gerald Murray of ACM for his help in codifying this *Author's Guide* and the **.cls** and **.tex** files that it describes.

8. REFERENCES

[1] Osek/vdx time-triggered operating system specification 1.0, version 1.0 edition, july 2001.

[2] 2005.

[3] P. Caspi, A. Curic, A. Maignan, C. Sofronis, S. Tripakis, and P. Niebert. From simulink to scade/lustre to tta: a layered approach for distributed embedded applications. In *Proceedings of the 2003 ACM SIGPLAN conference on Language, compiler, and tool for embedded systems*, pages 153–162. ACM New York, NY, USA, 2003.

[4] T. Henzinger, B. Horowitz, and C. Kirsch. Giotto: A time-triggered language for embedded programming. *Lecture Notes in Computer Science*, 2211:166–184, 2001.