Safety-critical control in mixed criticality embedded systems

Master Thesis
Royal Institute of Technology
Stockholm, Sweden

Emil Hjelm
emilhje@kth.se

January 24, 2017

Examensarbete MMK2017:Z MDAZZZ

Safety-critical control in mixed criticality embedded systems

Emil Hjelm

Approved: (datum)	Examiner: (examinator)	Supervisor: (handledare)
	Uppdragsgivare: Alten	Kontaktperson: Detlef Scholle

Abstract

Modern automotive systems contain a large number of Electronic Control Units, each controlling a specific system of a specific criticality level. To increase computational efficiency it is desired to combine multiple applications into fewer ECUs, this leads to mixed criticality embedded systems. The assurance of safety critical applications not being affected by non-critical applications is crucial.

Examensarbete MMK2017:Z MDAZZZ

Säkerhetskritisk kontroll i blandkritiska inbyggda system

Emil Hjelm

Godkänt: (datum)	Examinator: (examinator)	Handledare: (handledare)
	Uppdragsgivare: Alten	Kontaktperson: Detlef Scholle

Sammanfattning

Denna del kommer att inehålla en sammanfattning av arbetet på svenska.

Preface

Credit where credit is due.

Emil Hjelm
Stockholm

Contents

List of Figures

List of Tables

Abbreviations

Abbreviation	Description
ECU	Electric Control Unit
MCS	Mixed Criticality System
RTOS	Real-Time Operating System

Chapter 1

Introduction

This chapter will introduce the subject of mixed criticality embedded systems and the project to the reader.

1.1 Background

Today, modern automotive systems contain a large number of Electronic Control Units (ECU)s [], each controlling a specific system of a specific criticality level such as safety-critical distance keeping system (1.1.3) or non-critical entertainment systems []. This approach provides isolation for the numerous critical and non-critical applications in the collective system, and a simple mechanism to qualify an individual ECU. However, it yields an inefficient use of system resources [] and expensive system implementation []. In order to lower the cost of the system and increase system efficiency, applications of different criticality levels can be integrated into a single multicore platform, leading to a Mixed Criticality System (MCS). However, this approach increases system complexity, and hinders the certification of safety-critical systems []. In order to facilitate the design, test, and certification of such systems, spatial and temporal partitioning can be used in the architecture of the system as described by [5].

MCS

1.1.1 Definition of safety-critical systems

The term "Safety-critical system" has many definitions, most quite similar. Most definitions relate to systems with the potential to harm humans if the system malfunctions. According to [3] it is defined as "A system in which any

failure or design error has the potential to lead to loss of life." Further, [2] defines safety-critical systems as "A computer, electronic or electromechanical system whose failure may cause injury or death to human beings." A Wikipedia article, [4], defines a safety-critical system (or "life-critical system") as a system whose failure or malfunction may result in one (or more) of the following outcomes:

- death or serious injury to people
- loss or severe damage to equipment/property
- environmental harm

In this thesis, a safety-critical system will be defined as "a system whose failure may cause injury or death to human beings."

1.1.2 EMC2 Mixed criticality embedded system

The current MCS is implemented on a Xilinx Zynq-7000. It employs two operating systems to handle applications of different criticality. A Linux General Purpose Operating System (GPOS) for non-safety critical applications and a Real-Time Operating System for safety-critical applications. Hypervisor, FPGA, peripherals
An overview of the system can be seen in Figure 1.1.

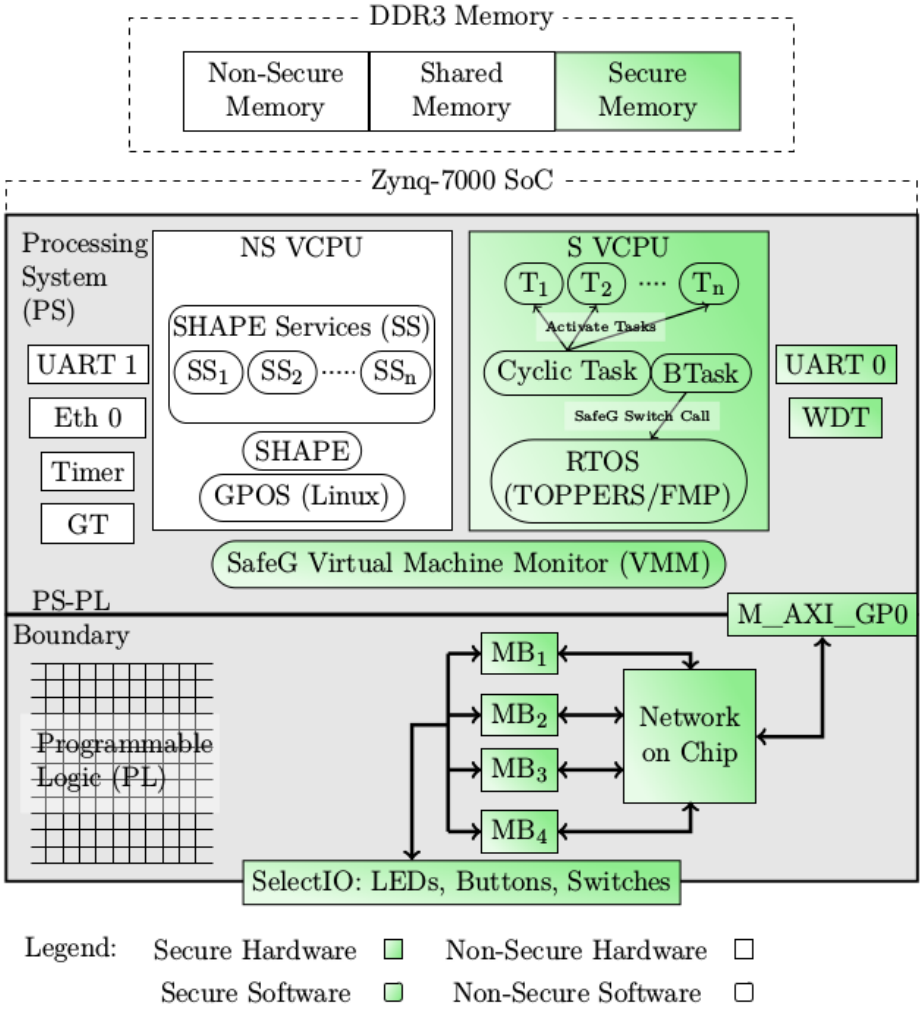

Figure 1.1: System overview of the MCS in place. [5]

For more detailed information, see the entire report by [5].

1.1.3 Platooning

Describe platooning.

1.2 Problem statement

A distance keeping control algorithm for platooning will be implemented on the embedded system described in 1.1.2. A demonstrator will be constructed

in the form of a RC car capable of following a vehicle in front of it at a certain distance. It should be verified that no matter the computational load and eventual crashes of the Linux based non-critical system, the distance keeping algorithm on the RTOS never crashes. The performance of the control system should be measured and compared with the same controller without any non-critical computational load. Missed deadlines? CPU usage? Evaluate different scheduling methods.

This problem leads to the research question:

- How well can a safety-critical control system perform when implemented on a mixed criticality system using virtualization?

alternatively:

- How, in a disciplined way, to reconcile the conflicting requirements of partitioning for safety assurance and sharing for efficient resource usage? [1]

alternatively:

- Is virtualization an efficient approach when trying to reconcile the conflicting requirements of partitioning for safety assurance and sharing for efficient resource usage when implementing a safety-critical control system?

1.3 Purpose

Protecting the integrity of a component from the faults of another is desired in all systems hosting multiple applications. However, it is of higher significance if the different applications have different criticality levels. Without such protection all components on the same system would need to be engineered to the standards of the highest criticality level, potentially massively increasing development costs [1].
Reducing the amount of computers in automotive systems would have many effects. Manufacturing costs would decrease and with fewer physical components maintenance costs would also decrease. However, the system complexity would increase and thereby increasing time and cost to design the system.

1.4 Goals

In this project there are both team goals and individual goals that do not necessarily align with each other.

1.4.1 Team goal

The team consists of five master thesis students. The students areas of work are: control theory and system modeling, data aggregation, safety-critical communication in MCS, line following and system testing and finally safety-critical control in MCS. Together the team will build a vehicle capable of following a vehicle ahead of it while keeping inside road markers.

1.4.2 Individual goal

Verify quantitatively the performance of safety-critical distance keeping controller. Evaluate different task scheduling methods in the specific system. Solve the problems described in 1.2.

1.4.3 Scope

The work of this thesis and the implementation on the demonstrator will build upon the work of [5].
The embedded computer is constrained to the Xilinx Zynq-7000 [1].
The thesis is produced at Alten AB.

1.5 Method

[1] `https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html`

Chapter 2

State of the art

This chapter will go through relevant articles and already known knowledge on the subject.

Chapter 3

System design

This chapter will derive the design of the controller to be implemented.

Chapter 4

Implementation

This chapter will describe the implementation of the control system in the demonstrator.

Chapter 5

Results

This chapter will present results from the demonstrator to the reader.

Chapter 6

Discussion

Discussion about the results produced by the thesis.

Chapter 7

Future work

This chapter will contain thoughts and ideas for future work building on this thesis.

Bibliography

[1] Alan Burns and Robert I. Davis. Mixed criticality systems - a review. Available online: https://www-users.cs.york.ac.uk/burns/review.pdf, July 2016.

[2] Dictionary.com. safety-critical system, January 2017. http://www.dictionary.com/browse/safety-critical-system.

[3] Encyclopedia.com. safety-critical system, January 2017. http://www.encyclopedia.com/computing/dictionaries-thesauruses-pictures-and-press-releases/safety-critical-system.

[4] Wikipedia.com. Life-critical system, January 2017. https://en.wikipedia.org/wiki/Life-critical_system.

[5] Youssef Zaki. An embedded multi-core platform for mixed-criticality systems: Study and analysis of virtualization techniques. Master's thesis, KTH, School of Information and Communication Technology (ICT), 2016.