Safety-critical Control in Mixed Criticality Embedded Systems

Master Thesis
Royal Institute of Technology
Stockholm, Sweden

Emil Hjelm
emilhje@kth.se

February 20, 2017

Master Thesis MMK2017:Z MDAZZZ

Safety-critical Control in Mixed Criticality Embedded Systems

Emil Hjelm

Approved: (datum)	Examiner: Martin Törngren	Supervisor: Bengt Eriksson
	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 on the same system is crucial.

Examensarbete MMK2017:Z MDAZZZ

Säkerhetskritisk kontroll i blandkritiska inbyggda system

Emil Hjelm

Godkänt: (datum)	Examinator: Martin Törngren	Handledare: Bengt Eriksson
	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	Electronic Control Unit
MCS	Mixed Criticality System
EMC^2	Embedded Multi-Core systems for Mixed Criticality applications in dynamic and changeable real-time environments
RTOS	Real-Time Operating System
GPOS	General Purpose Operating System
FPGA	Field Programmable Gate Array
SIL	Safety Integrity Level
ASIL	Automotive Safety Integrity Level
DAL	Development Assurance Level
VMM	Virtual Machine Monitor
EMC^2DP	EMC^2 Development Platform
RM	Rate Monotonic
AUTOSAR	AUTomotive Open System ARchitecture

Chapter 1

Introduction

This chapter will introduce the subject of mixed criticality embedded systems and the EU project "EMC^2" to the reader.

1.1 Background

Today, modern automotive systems contain around 70-100 Electric Control Units (ECU)s [13]. Each ECU controls a subsystem of a specific criticality level such as safety-critical anti-lock brake system, or non-critical entertainment systems [17]. Having the ECUs isolated ensures that the numerous critical and non-critical applications do not interfere with each other, thus it is a simple task to certify an individual ECU. However, this approach leads to an inefficient use of system resources and expensive system implementation [5]. In order to lower the cost of the collective system and increase system efficiency (utilization), 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 [21]. 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 [21].

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 [5].

The EU project "Embedded Multi-Core systems for Mixed Criticality applications in dynamic and changeable real-time environments" (EMC^2) was founded in order to "find solutions for dynamic adaptability in open systems, provide handling of mixed criticality applications under real-time conditions, scalability and utmost flexibility, full scale deployment and management of integrated tool chains, through the entire lifecycle" [17].

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 [8] it is defined as "A system in which any failure or design error has the potential to lead to loss of life." Further, [6] defines safety-critical systems as "A computer, electronic or electromechanical system whose failure may cause injury or death to human beings." A Wikipedia article, [19], 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 Different levels of criticality

Different names of levels of criticality are typically Safety Integrity Level (SIL), Automotive Safety Integrity Level (ASIL) and Development Assurance Level (DAL). The IEC 61508 standard [10] defines four different levels and the ISO 26262 standard [11] and the DO-178C standard [7] define five different levels each. These levels range from low or no hazard up to life-threatening or fatal in the event of a malfunction requiring the highest level of assurance that the dependent safety goals are sufficient and have been achieved.

In this report the number of criticality levels will be restricted to two: "safety-critical" and "non-critical". This is due to the constraints presented in section 1.1.3 and 1.4.3.

1.1.3 EMC² development board

As a part of the EMC² project, Alten has developed a system for handling applications of mixed criticality on the same piece of hardware. The MCS developed at Alten is implemented on a Xilinx Zynq-7000 [20]. The development board is called EMC² Development Platform, or EMC²DP. It employs two operating systems to handle applications of different criticality. A General Purpose Operating System (GPOS) for non-critical applications and a Real-Time Operating System (RTOS) for safety-critical applications. A Virtual Machine Monitor (VMM) is used to alternate between the two.

Peripherals connected to the board are separated between safety-critical and non-critical via ARM TrustZone [1].

The board also has a Field Programmable Gate Array (FPGA).

For more detailed information, see the report by [21].

1.1.4 Platooning

"The platooning concept can be defined as a collection of vehicles that travel together, actively coordinated in formation. Some expected advantages of platooning include increased fuel and traffic efficiency, safety and driver comfort" [4].

1.2 Problem statement

An ideal MCS ensures partitioning between different criticality levels while still sharing resources efficiently.

The MCS developed at Alten (EMC²DP) 1.1.3 switches Operative System (OS) to enable partitioning between safety-critical and non-critical applications, which takes about 2 μs. This mode switch introduces additional deadlines which makes processor scheduling more difficult.

To evaluate the performance of the system, a distance keeping control algorithm for platooning will be implemented on it. A demonstrator will be constructed in the form of a RC car capable of following a vehicle in front of it at a specified distance. If the lead car exceeds a predefined maximum speed or deviates from the road, the following car should not exceed the maximum speed. The performance of the embedded controller and the control

algorithm will be measured during heavy non-critical computational load, and without any non-critical load altogether.

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. It should also be investigated at how high frequencies the control algorithm can operate while still maintaining functionality on the GPOS.

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?" [5]

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

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.

SafeCOP (Safe Cooperating Cyber-Physical Systems) is an European project that targets cyberphysical systems-of-systems whose safe cooperation relies on wireless communication [12]. SafeCOPs Use Case 3 (UC3) regarding "Vehicle control loss warning" together with the EMC2 goals tie well in with the problem statement and use case described in 1.2.

1.4 Goals

In this project there are both team goals and individual goals that do not always 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, lane detection 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, see 1.5. 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 Youssef Zaki [21].
The embedded computer is constrained to the Xilinx Zynq-7000 SoC [1].
The thesis is produced at Alten AB.
Scheduling on the processor will be restricted to Rate Monotonic (RM).

1.5 Research design

The plan is to make an confirmatory investigation using quantitative data/operations with a deductive approach. This is a quantitative research method where data is gathered during experiments and from simulations of the environment [9].

The position of the demonstrator will be read by a separate sensor of the same type as the one on the demonstrator. The performance of the control system and the embedded controller will be measured and compared with the same system without any non-critical computational load. This will also be done for a simulation of the system. The measures regarding control system performance will consist of

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

- Response time
- Overshoot
- Settling time

Data points for the performance of the embedded controller will be extracted from the RTOS, and the measures will consist of

- Missed deadlines
- CPU utilization

1.6 Ethical considerations

When designing a MCS it is crucial to ensure that errors made by a lower criticality application cannot propagate to higher criticality applications. This could have catastrophic consequences. Because of this the requirement of partitioning must have higher priority than the need of sharing.

Chapter 2

State of the art

This chapter will go through relevant articles and already known knowledge on the subject of mixed criticality systems. It will also explain the EMC2DP.

2.1 Mixed criticality systems

A MCS is achieved by letting applications of different criticality share resources. These resources could be computational power in the CPU, memory, peripherals, input/output ports. As of July 2016, the most explored area is sharing the CPU between multiple criticality levels [5].

2.1.1 Economical benefits of MCS

Potential benefits with pursuing MCS as opposed to distributed systems are reduced physical space required, reduced weight, reduced heat generation, reduced power consumption and reduced production costs [5]. This would all ultimately lead to economical benefits.

Potential downsides are increased complexity which could lead to higher system design costs. Building applications on the same platform to share resources could require engineering teams to work more closely together, potentially leading to administrative difficulties and costs. This needs to be investigated and could vary from industry to industry.
To combat the potential downsides, the EMC^2 project aims at creating platforms for easier development of MCS.
The EMC^2 project lists several goals [18]:

- Reduce the cost of the system design by 15%

- Reduce the effort and time required for re-validation and re-certification of systems after making changes by 15%
- Manage a complexity increase of 25% with 10% effort reduction
- Achieve cross-sectorial reusability of Embedded Systems devices and architecture platforms that will be developed using the ARTEMIS JU results.

2.1.2 Sharing processor

The area of MCS was first explored by Steve Vestal [16]. His paper showed that neither Rate Monotonic (RM) nor Deadline Monotonic (DM) priority assignment was optimal for MCS; however Audsley's optimal priority assignment algorithm [2] was found to be applicable.

This paper was followed by two publications in 2008 by Baruah and Vestal [59], and Huber et al. [206]. The first of these papers generalises Vestal's model by using a sporadic task model and by assessing fixed job-priority scheduling and dynamic priority scheduling. It contains the important result that EDF (Earliest Deadline First) does not dominate FP when criticality levels are introduced, and that there are feasible systems that cannot be scheduled by EDF. The latter paper addresses multi-processor issues and virtualisation (though it did not use that term). It focused on AUTOSAR and resource management (encapsulation and monitoring) with time-triggered applications and a trusted network layer."

1. (audsley) 2. EDF-VDL
For a more complete review of work done on MCSs with a shared processor, see the paper by Alan Burns [5].

2.1.3 Different criticality on different processors

2.1.4 Sharing memory

2.2 Current system

The EMC2DP uses two operating systems, a RTOS for safety-critical applications and a GPOS for non-critical applications. A Virtual Machine Monitor (VMM) or "Hypervisor" is used to alternate between safety-critical RTOS and non-critical GPOS. The RTOS is TOPPERS FMP kernel [14], and the

GPOS is a custom modified Linux distribution. The VMM used is SafeG [15]. It switches processor state via a hardware switch. See figure 2.1.

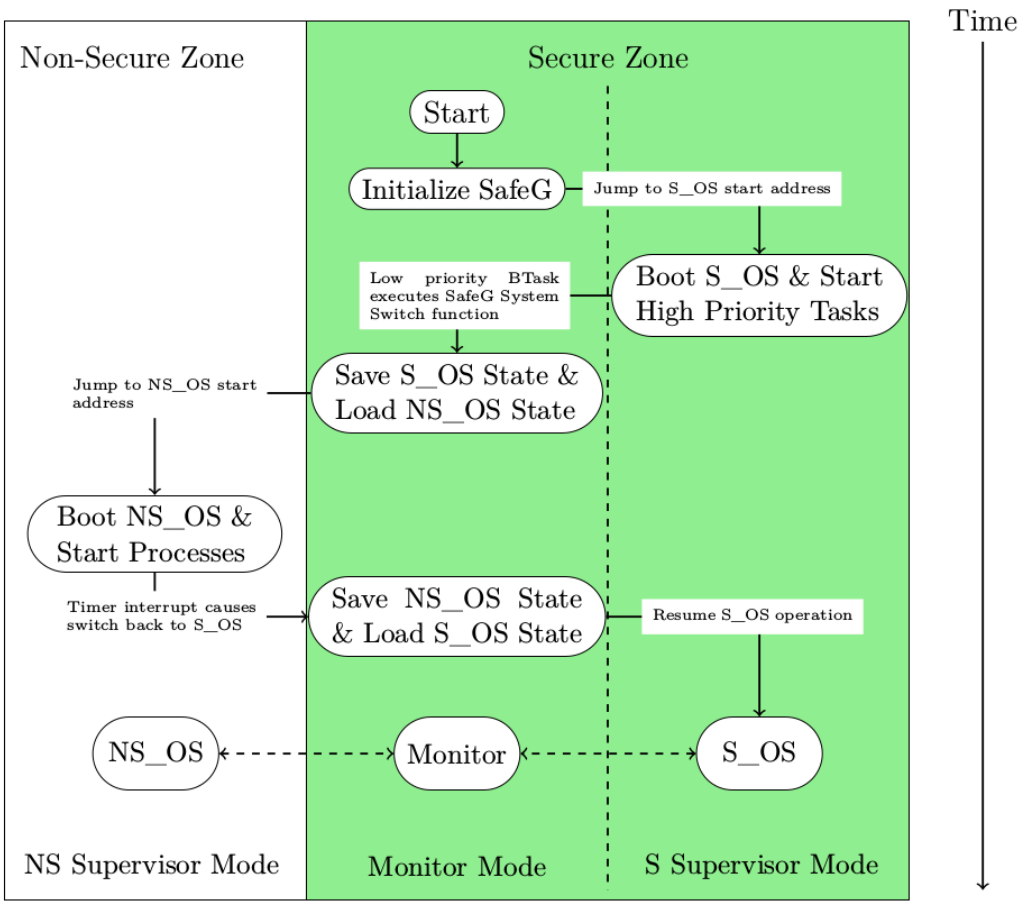

Figure 2.1: Flowchart of the boot sequence of the CPU. [21]

A Field Programmable Gate Array (FPGA) as interface between processor and board. Peripherals are separated as secure and non-secure using ARM TrustZone [1], see section 2.2.1.

An overview of the system can be seen in Figure 2.2.

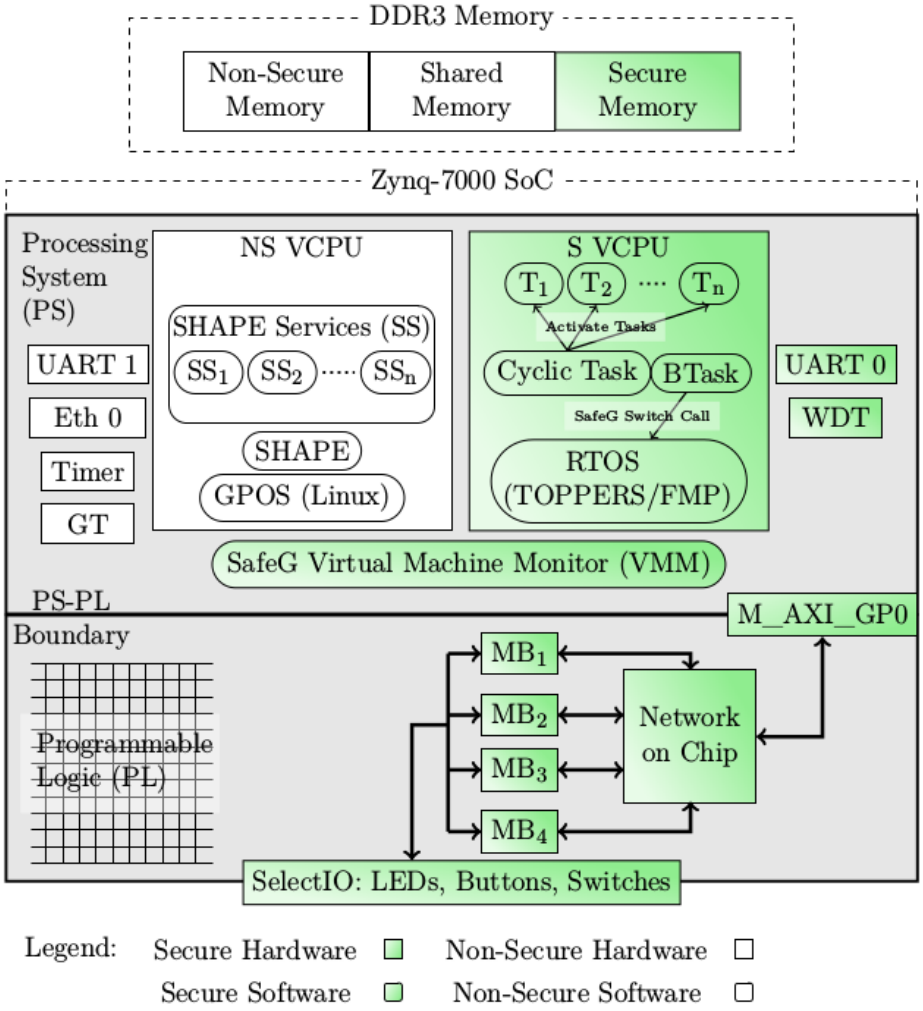

Figure 2.2: System overview of the MCS in place. [21]

2.2.1 TrustZone

"TrustZone is a security extension available in modern ARM processors that creates a security infrastructure designers can use to protect critical system assets. This infrastructure is achieved by enabling the partition of system components, both hardware and software, into either a Secure and a Normal world (or zone). Resources that are marked as normal are not permitted to access Secure zone components. This mechanism is enforced by the AMBA3 (Advanced Microcontroller Bus Architecture) AXI (Advanced eXtensible Interface) bus system. It contains an extra control signal for each of the read

and write channels (Non-secure or NS bits) that dictate the access rights Non-secure bus masters to the Secure slaves. Each processor with an enabled TrustZone security extension can be partitioned into a Normal and a Secure virtual CPU. The virtual processors execute in a time-multiplexed fashion, and use the "Monitor Mode" state to create an efficiently switching mechanism between Normal and Secure zones. The NS-bit, bit[0] of the Secure Configuration Register (SCR) in the System Control Coprocessor (CP15), controls the activation of secure state of the processor. Whenever the NS-bit set high, the processor state immediately switches to the Normal world. However, if the processor is in Monitor Mode, it remains in the Secure world regardless of the state of the SCR NS-bit. The processor state can nter monitor mode by either issuing a special instruction, SMC (Secure Monitor Call), in software or by hardware exception mechanisms such as IRQ, FIQ, external Data Abort, or external Prefetch Abort. In general, the software running in monitor mode serves the purpose of saving the state of the current world, and loading the state the other." [1]

2.3 Standards

Different standards to regulate and ensure safety.

2.3.1 AUTOSAR

"AUTOSAR (AUTomotive Open System ARchitecture) is a worldwide development partnership of automotive interested parties founded in 2003. It pursues the objective of creating and establishing an open and standardized software architecture for automotive electronic control units (ECUs) excluding infotainment. Goals include the scalability to different vehicle and platform variants, transferability of software, the consideration of availability and safety requirements, a collaboration between various partners, sustainable utilization of natural resources, maintainability throughout the whole "Product Life Cycle"." [3]

2.3.2 ISO26262

New automotive standard.

Freedom from interference

ASILs

Different levels relating to hazard and risk.

Severity class	Probability	Controllability class		
		C1	C2	C3
S1	E1	QM	QM	QM
	E2	QM	QM	QM
	E3	QM	QM	A
	E4	QM	A	B
S2	E1	QM	QM	QM
	E2	QM	QM	A
	E3	QM	A	B
	E4	A	B	C
S3	E1	QM	QM	A
	E2	QM	A	B
	E3	A	B	C
	E4	B	C	D

Table 2.1: ASIL as a function of severity, probability and controllability.

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 or in the area of MCS in general.

7.1 Using virtualization

Facilitate for more than two different criticality levels.
Examine different scheduling methods.

7.2 Using other means of partitioning

Examine limitations for other configurations of MCS, for example different CPUs for different criticality levels.

7.3 Economical benefits for pursuing MCS

It is not clear how much the potential economical benefit would be from pursuing MCS. The economical impacts of MCS might be different in different industries. It must be calculated more exactly how large the potential benefits would be to gauge the need for pursuing MCS.

Bibliography

[1] ARM Ltd. ARM TrustZone, February 2017. https://www.arm.com/products/security-on-arm/trustzone.

[2] N C Audsley. On priority assignment in fixed priority scheduling. *Information Processing Letters*, 79(1):39–44, 5 2001.

[3] AUTOSAR. AUTOSAR Homepage, February 2017. http://www.autosar.org/.

[4] Carl Bergenhem, Henrik Pettersson, Erik Coelingh, Cristofer Englund, Steven Shladover, and Sadayuki Tsugawa. Overview of platooning systems. In *Proceedings of the 19th ITS World Congress*, Vienna, Austria, October 2012.

[5] 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.

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

[7] Software considerations in airborne systems and equipment certification. Standard, Radio Technical Commission for Aeronautics, Washington, DC, USA, December 2011.

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

[9] Anne Håkansson. Portal of research methods and methodologies for research projects and degree projects, July 2013.

[10] Functional safety of electrical/electronic/programmable electronic safety-related systems – parts 1 to 7. Standard, International Electrotechnical Commission, Geneva, CH, April 2010.

[11] Road vehicles – functional safety – part 9: Automotive safety integrity level (asil)-oriented and safety-oriented analyses. Standard, International Organization for Standardization, Geneva, CH, November 2011.

[12] SafeCOP. SafeCOP - part B, October 2016.

[13] H. Thompson. Mixed criticality systems, February 2012.

[14] TOPPERS Project, Inc. Introduction to the TOPPERS/FMP kernel, February 2017. `https://www.toppers.jp/en/fmp-kernel.html`.

[15] TOPPERS Project, Inc. TOPPERS SafeG, February 2017. `https://www.toppers.jp/en/safeg.html`.

[16] Burns Vestal. Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance, 2007.

[17] W. Weber, A. Hoess, F. Oppenheimer, B. Koppenhöefer, B. Vissers, and B. Nordmoen. EMC2 a Platform Project on Embedded Microcontrollers in Applications of Mobility, Industry and the Internet of Things. In *2015 Euromicro Conference on Digital System Design*, pages 125–130, August 2015.

[18] Werner Weber. A Platform Project on Embedded Microcontrollers in Applications of Mobility, Industry and the Internet of Things, Mars 2015. `https://artemis-ia.eu/publication/download/1131.pdf`.

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

[20] Xilinx Inc. Zynq-7000 All Programmable SoC, February 2017. https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html.

[21] 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.