Failsafe ECU through dynamic partial-reconfiguration in FPGAs

Constantin Schieber, 01228774
Rupert Schorn, XXXXXXX
Andreas Hirtenlehner, XXXXXXXX
Peter Schober, XXXXXX

February 27, 2019

1 Introduction

In this lab, fail-safe mechanisms for Electronic Control Units (ECUs) are explored on the basis of Partial Reconfiguration (PR) in Field Programmable Gate Arrays (FPGAs). The introduced design contains typical characteristics of an automotive system. Communication of the different sub-systems is realized with a bus and connects the FPGA to the ECUs that handle operations during normal conditions. ECUs are monitored by a bus monitor in the FPGA for nominal operation characteristics and can be replaced by the instantiation of the very same ECU within the FPGA in case of failure.

Our ECUs are based on the Cortex-M1, an ARM-based microcontroller that became recently available as an intellectual property (IP)-package for Xilinx based FPGAs. By using a common hardware base we were able to streamline the development of the control-software which enables a faster development process and reduces the overhead of functional verification.

Brief overview, problem statement and final outcome.

1.1 Required Tools and packages

- Vivado 2018.3
- Vivado 2018.2
- uVision 5 (Windows only)
- ARM Keil (Windows only)
- ARM Cortex M1 IP for Vivado

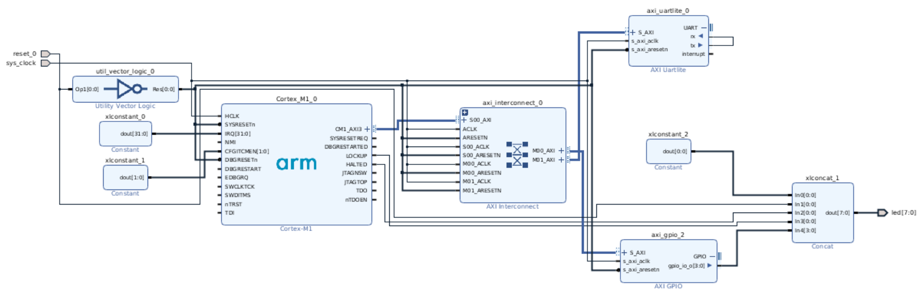

Figure 1: Block Diagram of the partial reconfiguration functionality

2 Cortex M1 / M3

Description of the setup of Cortex M1 and Cortex M3 Cores. [1] [2]

2.1 Usage of the Keil Toolchain

Link to tutorial enough? Maybe most basic steps (e.g. adaption of arty project for our needs). [3]

2.2 Usage of Cortex M1 in Vivado

Only global implementation / synthesis runs are permitted to obtain a working bitstream. If OOC (Out of Context) runs are used, everything except for the Cortex M1 will work fine. The Cortex M1 will enter a hard-fault state which does not allow recovery. This is indicated by a high bit on the *Lockup* port of the processor.

Also trivial, follow tutorial that is provided on ARM Website and adapt to own needs.

2.2.1 Code via Memory Initialization File

File is bound to synthesis process, how to change it...

2.3 How to mbed OS

How was the Cortex M1 Project adapted to support the Mbed OS? What is gained through the usage of mbed OS?

2.4 Implemented Functionality on the Cortex M1

How are the UART and IIC peripherals supported in the source code, which libraries are used, interrupt based or not, performance?

Show highlights of the code?

3 Bus and Peripherals

3.1 Used Peripherals

What Peripherals were used (Cortex M1/M3 Boards, which ones).

How to program / use them (only brief).

How are they connected to the Bus?

IIC and UART here or in next section.

3.2 Bus

How is it set up?

Made assumptions?

Document used / invented protocol.

4 Partial Reconfiguration

What is partially reconfigured - Cortex, uart, IIC.

Why not use AXI?

4.1 Integration Overview

[4], [5] Usage of ICAP.

How is the Zynq still used - Loading images and binary blobs from sd card into DDR.

4.2 Packaged IP

Why is the PR IP Packaged, how was it done? [**?**]

5 Organization

5.1 Assigned Sub-Tasks

5.1.1 Cortex M1

Evaluation of the provided Cortex-M1 from ARM and adaption of the provided toolchain to our needs.

Creation of a working stand-alone example, creation and integration of working IP-Packages for the Cortex-M1.

Provision of partial reconfiguration eligible IP-Packages.

5.1.2 Partial Reconfiguration

Evaluation of the partial reconfiguration flow in Vivado (project vs. script based).

Determination of restrictions that come with partial reconfiguration e.g. less flexible and limited use of clock spines within the pr region, no operations on clocks within the pr region, constant interface over all implementations, placement / alignment restrictions, naming conventions and conflicts for custom IP-Packages, binding usage of .vhd files instead of block diagrams with regards to pr modules.

Evaluation with Microblaze processors, as Cortex M1 is only available from the start of November 2018.

Generation of a Bootable Image that includes the partial reconfiguration bitstreams in a binary format.

Modification of the Zynq processing system on the Zedboard to allow for the usage of the ICAP instead of the PCAP, loading the partial bitstreams into DDR memory from the SD-Card.

Evaluation of the pr-controller and the available trigger sources / types.

5.1.3 Bus Design

Protocol, Standard, Implementation on FPGA / Cortex.

5.1.4 Hardware

Evaluation of hardware types (e.g. Nucleo Boards), possible connection to the Zedboard, CAN-Driver boards.

Task	Hirtenlehner	Schieber	Schober	Schorn
Cortex M1	X	X	X	
Partial Reconf.		X		
Bus Design				X
Hardware	X			X
Mbed OS	X			X

Table 1: Distribution of tasks within the group

5.2 Estimated Contribution

Contribution to the project was roughly the same for each group member. Table 1 shows the tasks and how they were assigned to the different team members.

References

[1] ARM, "Arm® Cortex®-M1 DesignStart™ FPGA-Xilinx edition User Guide," 2018. [Online]. Available: https://static.docs.arm.com/100211/0001/arm_cortex_m1_designstart_fpga_xilinx_edition_ug_100211_0001_00_en.pdf?_ga=2.121336076.523725493.1550477241-1711067005.1549732081

[2] ——, "Cortex-M1 Technical Reference Manual." [Online]. Available: https://static.docs.arm.com/ddi0413/d/DDI0413D_cortexm1_r1p0_trm.pdf

[3] "Getting Started." [Online]. Available: http://www2.keil.com/mdk5/install

[4] Xilinx, "Vivado Design Suite User Guide: Partial Reconfiguration (UG909)," 2018. [Online]. Available: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2018_3/ug909-vivado-partial-reconfiguration.pdf

[5] ——, "Vivado Design Suite Tutorial: Partial Reconfiguration (UG947)," 2018. [Online]. Available: https://www.xilinx.com/content/dam/xilinx/support/documentation/sw_manuals/xilinx2018_3/ug947-vivado-partial-reconfiguration-tutorial.pdf