

Obtaining Diagnostic Coverage Metrics Using Rapid Prototyping of Multicore Systems

2011-01-1007

Published
04/12/2011Simon Brewerton
Infineon TechnologiesRolf Schneider
AUDI AGFabio Bruno
Infineon Technologies

Copyright © 2011 SAE International

doi:[10.4271/2011-01-1007](https://doi.org/10.4271/2011-01-1007)

ABSTRACT

With the introduction of the ISO26262 automotive safety standard there is a burden of proof to show that the processing elements in embedded microcontroller hardware are capable of supporting a certain diagnostic coverage level, depending on the required Automotive Safety Integrity Level (ASIL). The current mechanisms used to provide actual metrics of the Built-in Self Tests (BIST) and Lock Step comparators use Register Transfer Level (RTL) simulations of the internal processing elements which force faults into individual nodes of the design and collect diagnostic coverage results. Although this mechanism is robust, it can only be performed by semiconductor suppliers and is costly. This paper describes a new solution whereby the microcontroller is synthesized into a large Field Programmable Gate Array (FPGA) with a test controller on the outside. This platform can be used to inject faults into the internal state of the microcontroller while actual tests are running, such that real diagnostic capabilities of various mechanisms can be measured, and to check that reactions to these detected faults are handled as required.

INTRODUCTION

The cost and engineering effort required to design and develop a new microcontroller product is extremely significant, so much so that one error which is discovered after first silicon requiring a second tapeout can mean that the product will no longer be able to recover the total expenditure

invested in it over the lifetime and expected volume of the product. It has therefore long been the case that rigorous requirements gathering, architecture design, detailed design, implementation, verification, validation and integration testing have been common place, as semiconductor companies can literally not afford to get the devices wrong. Recently, the industry has also embraced the upcoming ISO26262 norm as a 'best practice' approach to follow when building Automotive grade devices which are to be used in safety relevant systems. The adoption of the process has only led to several minor changes in the development flow, but has added significantly to the degree of rigor in which the requirements and specification process are performed. The microcontroller is typically developed as a Safety Element Out Of Context (SEOOC), where by the semiconductor company makes many application based assumptions of the requirements and formally captures these in a requirements database. The process works very well until the FMEDA (Failure Mode Effect and Diagnostic Analysis) has to be performed, as the design is normally not known to a sufficient level of detail to know exactly the failure mode effects to any confidence. The process therefore allows the developers to make assumptions in the FMEDA as long as they also make some requirements that these assumptions are verified (by whatever means) later in the system verification.

The previous methods of verification at a system level were to ensure that the hardware was capable of detecting a possible safety violation (typically by design methods and simulation-based verification before the microcontroller

design is released for manufacture) and describing the mechanisms in a user manual so that software designers can implement a suitable driver that handles the potential violations when it is triggered. With previous 'state-of-the-art' microcontrollers this was not too much of an onerous task as there were not too many safety mechanisms performed by hardware, so software already had to run most of the detection mechanisms via special sequences (often as assembly code libraries) which looked for side effects of known failures, or triggered hardware support blocks, as previously described in several SAE papers [11][12]). The recent publication of the near-complete draft of the ISO26262 specification for Road Vehicle Functional Safety has clarified and strengthened the previously assumed requirements for silicon diagnostic metrics. That, together with the ever-present desire for more performance and 'system-on-a-chip' integration has made next generation microcontrollers simply too large and resource rich to be able to perform rigorous fault detection by software in a sensible amount of time (<10ms). Many new autonomous test mechanisms have now been implemented as hardware blocks, with their outputs triggering a central Safety Management Unit (peripheral) which has the potential to handle exceptions automatically and to notify the host software that a violation has been detected. This closer cooperation between hardware and software mechanisms has led to the need to create an environment to be able to develop these drivers. With this environment the correct cause and effect functionality can be verified. This is particularly relevant if the effect of the handler is not to just immediately turn off the system to establish a safe state, but to go through some error collection, arbitration and reporting scheme, and maybe provide an intermediate limited functionality (limp-home mode or temporary emergency mode) depending on the system mode in which the fault was detected and the actual fault detected. This environment can exist in C model environments for timing related investigations (such as Syntavision [19]), but the final verification that the real fault detection coverage metrics and reaction times are satisfied can require a significant portion of the external hardware outside the microcontroller to also be accurately modeled with suitable fidelity. The other alternative is to prototype the whole microcontroller as a digital electrical representation in a Field Programmable Gate Array (FPGA).

USE CASES OF A 'VIRTUAL MICROCONTROLLER'

The ISO26262 standard outlines the required design and development flow in terms of process, but also in terms of degree of rigor of performing the process for the required ASIL level of the particular functions and safety goals. The specific phases and work products that can benefit from this FPGA approach are as follows:

- 3-7 Hazard Analysis and Risk Assessment - Where the assumed FMEDA can be cross checked by error injection techniques, to ensure that the Effects of the Failure Modes are sufficiently accurate.
- 5-8 Hardware Architecture Metrics - Where the actual diagnostic coverage of the included safety metrics can be verified in an empirical way, to show that the given metrics are accurate.
- 5-9 Evaluation of Violation of the Safety Goal due to Random HW Failures - Where the actual diagnostic coverage metrics obtained in 5-8 are fed into a spreadsheet to build the Single Point Failure Metric and Multiple (Latent) Failure Metric.
- 5-10 Hardware Integration and Testing - Where the hardware blocks and interfaces (for instance the safety path disable hardware connections between the host microcontroller and external watchdog / safing engine) can be checked to ensure the safe states can be reached in the correct fault reaction time.
- 6-10 Software Integration and Testing - Where the Hardware / Software Interface Specification can be validated by the introduction of system level and microcontroller level faults, to show that the correct software paths are taken, and that there are no unhandled exceptions.
- 6-11 Verification of Software Safety Requirements - Where the specified Software Safety Requirements (generated from 6-6) can be validated by the simulation of safety goal violations.
- 4-9 Safety Validation - Where the complete and finished system can still offer error injection opportunities at a very low level, to ensure that all required error hooks are terminated, and that they have the required reaction time.

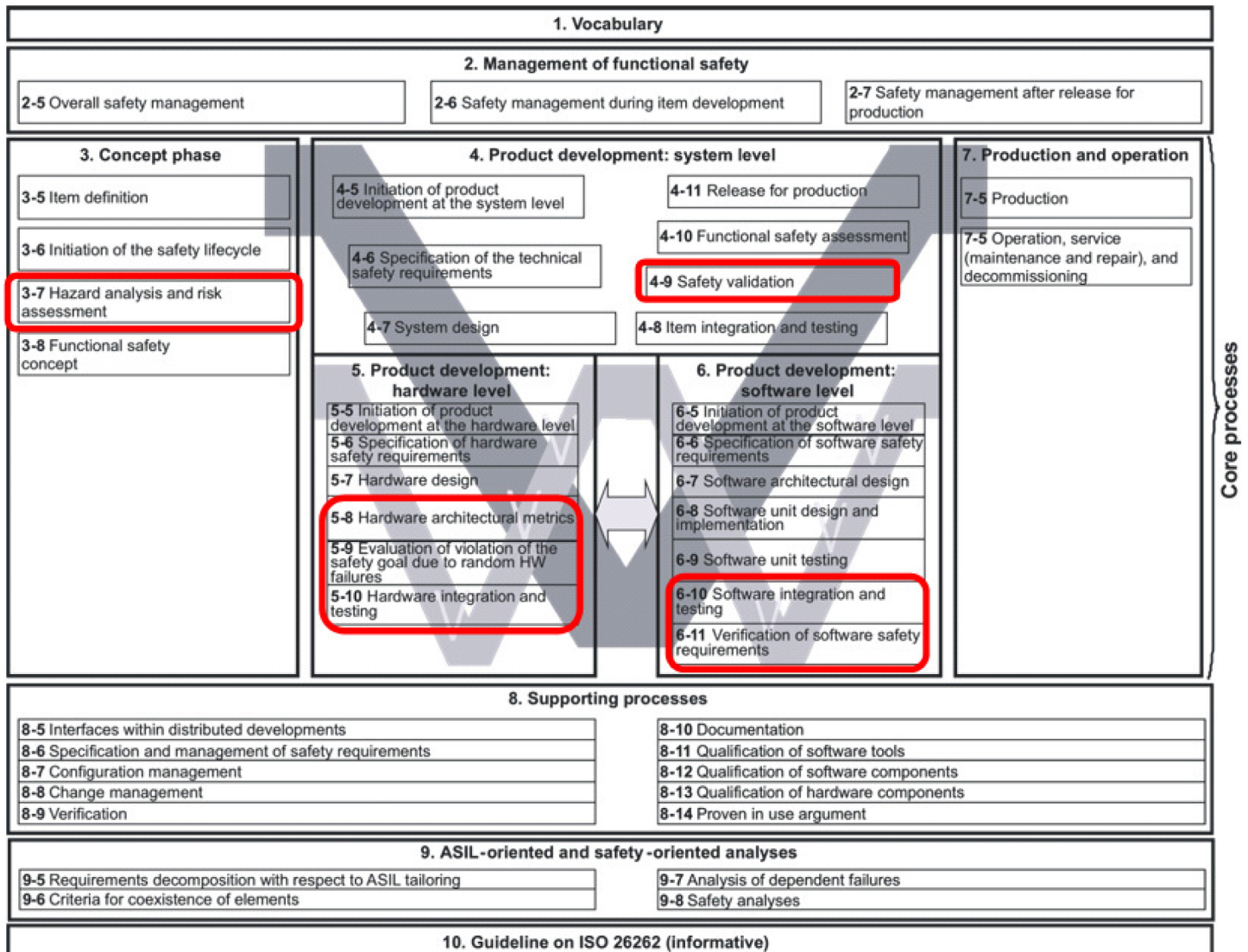


Fig 1. Overview of ISO26262 Methodology, highlighting tasks where FPGA rapid prototyping is beneficial.

FPGA TECHNOLOGY BENEFITS AND DRAWBACKS

The FPGA is effectively an array of various digital cells, memories, clocks trees, and 10 ports that can be configured to represent the netlist of the digital components of the microcontroller (or indeed any other digital logic). The FPGA chosen for this system is a Xilinx Virtex 6 [20], which is supported by a suite of synthesis tools and programming tools to allow digital RTL and Verilog designs to be targeted onto the gate array. This flexible FPGA has a certain gate count, quantity of SRAM, Ethernet MACs, special purpose cells, as well as generic cells. The microcontroller design can be compiled and synthesized to the available resources of the FPGA. However, due to the flexible nature of the FPGA, compared to the customized resources of the final product - the mapped design does not achieve an optimized clock speed for the FPGA implementation, so although the FPGA is rated for 800MHz, the actual simulated microcontroller design may

be limited to around 100MHz at room temperature. FPGA technology is also primarily a digital prototyping environment, but automotive microcontrollers typically have several tens of onboard 5V, 10 bit analog to digital converter channels. This must be addressed by companion devices that can support analog functions in a way that can be easily interfaced to the FPGA digital environment. The other (significant) drawback is that the final production silicon of the microcontroller in volume production may be targeting less than €10, whereas the generic FPGA in single pieces costs upwards of €5000 per piece. When integrated into the Trusted Compute Platform (TCP), this comes to a whole board cost of approximately €10000. It should be noted however, that this TCP is not just a single point solution, but can in reality be reused for several years, and across different application domains (body systems, electric motor drives, safety system, and vision systems), so the up front cost can be justified easily by its extended application.