

High-Integrity GPU Designs for Critical Real-Time Automotive Systems

Sergi Alcaide*[†], Leonidas Kosmidis*, Carles Hernandez*, Jaume Abella*

*Barcelona Supercomputing Center, Barcelona, Spain

[†]Universitat Politècnica de Catalunya, Barcelona, Spain

E-mail: {sergi.alcaide, leonidas.kosmidis, carles.hernandez, jaume.abella}@bsc.es

Keywords—GPU, reliability, functional safety, diverse redundancy

I. EXTENDED ABSTRACT

The advent of autonomous driving (AD) makes automotive industry embrace high-performance hardware such as accelerators to execute performance-hungry tasks (e.g. object recognition and tracking) timely. However, while those accelerators have been widely deployed in the consumer electronics market, where performance within given power and thermal envelopes is the main concern, critical real-time systems, such as those in AD, pose a set of different challenges related to functional safety. In particular, safety-related automotive systems (e.g. braking, steering, etc), which include most AD functionalities, need to meet specific requirements described in the ISO26262 functional safety standard [1] to be deployed in cars. Those requirements, which mostly relate to the ability of the system to detect faults and prevent hazardous situations, impose strict verification and validation (V&V) requirements on the system and its components thereof. Hence, high-performance accelerators deployed in cars for AD must adhere to those requirements, which needs to be conveniently proven.

In the context of ISO26262, functionalities are classified in different Automotive Safety Integrity Levels (ASIL) based on the type of hazard they can cause, their severity, their exposure and the controllability upon a failure. In particular, safety-related functionalities are ranked from ASIL-D (the highest integrity level) to ASIL-A (the lowest), being ASIL-D components involved in ASIL-D functionalities those subject to the strictest V&V processes.

GPUs are becoming the most popular accelerator for AD, and they are already included in specific AD commercial platforms, such as RENESAS R-Car H3 [2] and NVIDIA Xavier [3] platforms. Those platforms include general purpose microcontrollers (e.g. ARM or Infineon cores) proven ASIL-D capable, as well as high-performance accelerators whose adherence to ASIL-D must also be proven so that they can perform AD-related activities. In particular, as detailed in ISO26262, ASIL-D systems must not allow a single fault lead the system to a hazardous situation. Appropriate safety measures include some form of independent redundancy, thus ensuring that a single fault will not lead redundant elements to identical erroneous outputs. For instance, Error Detection and/or Correction Codes are often used for storage and communication interfaces, whereas Dual Core LockStep (DCLS) with some source of diversity (e.g. staggered execution) is used for computation elements so that a single fault affecting all redundant components (e.g. a voltage droop) does not

cause them to fail identically (see example on Figure 1). In the case of GPUs, they have already been proven ASIL-B compliant, but, to our knowledge, ASIL-D compliance has only been achieved by implementing expensive independent redundancy means. In particular, either full system replication or heterogeneous implementations are used. The former, for instance, performs object recognition based on cameras and LIDAR, with different software implementations and, potentially, different hardware support. The latter, for instance, performs object recognition based only on cameras, but software is implemented and deployed for different accelerators (e.g. a GPU and a Deep Neural Network – DNN – accelerator) [4]. In both cases, design and V&V costs are duplicated, which is against efficiency and costs. For instance, these approaches clash with that followed for ASIL-D microcontrollers, which build upon diverse DCLS. Hence, it is critically important enabling some form of diverse DCLS on GPUs so that ASIL-D compliance can be achieved without needing to design and certify multiple heterogeneous software and hardware components.

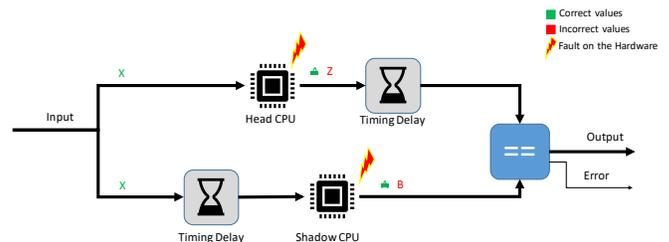


Fig. 1. Example of a Dual Core LockStep with Staggered Execution

This study tackles this challenge by identifying the main requirements to enable ASIL-D compliance for Commercial Off-The-Shelf (COTS) GPUs, assessing to what extent they have the potential to meet ASIL-D requirements, and providing a set of lowly-intrusive modifications that allow adhering to ASIL-D requirements without diminishing their performance for non-safety-related functionalities. Those modifications allow GPU vendors to reuse their designs avoiding a significant increase of their Non-Recurring Expenses (NRE). In particular, we perform our analysis on an NVIDIA COTS GPU, implement modifications on a GPU simulator where we can assess both performance and ASIL-D compliance, and evaluate what the performance impact would be on the COTS GPU.

REFERENCES

- [1] International Standards Organization, *ISO/DIS 26262. Road Vehicles – Functional Safety*, 2009.
- [2] “RENEAS R-Car H3,” <https://www.renesas.com/en-us/solutions/automotive/products/rcar-h3.html>.
- [3] D. Shapiro, “Introducing Xavier, the NVIDIA AI Supercomputer for the Future of Autonomous Transportation,” *NVIDIA blog*, 2016. [Online]. Available: <https://blogs.nvidia.com/blog/2016/09/28/xavier/>
- [4] NVIDIA, “NVIDIA Announces World’s First Functionally Safe AI Self-Driving Platform,” <https://nvidianews.nvidia.com/news/nvidia-announces-worlds-first-functionally-safe-ai-self-driving-platform>.

Sergi Alcaide received his BSc degree in Computer Engineering from Universitat Politècnica de Catalunya (UPC), Barcelona in 2016. Then he joined the Computer Architecture - Operative Systems (CAOS) group in Barcelona Supercomputing Center (BSC) while studying his master studies. In 2018, he completed his MSc degree, Master in Innovation and Research in Informatics (MIRI) from Universitat Politècnica de Catalunya (UPC). Now he is a PhD student from UPC in the Computer Architecture - Operative Systems (CAOS) group in BSC.