engineering proceedings

MDPI

Proceeding Paper

Enhancing Safety-Critical Brake System Testing with Vector SIL over Complex Vector HIL †

János Májer [1,2,*], Dénes Fodor [3], Péter Panyi [1], Félix Tivadar Nagy [1] and Balázs István Németh [1]

[1] Active Safety and Controls (ASC) Group, Safety and Motion (SAM), Continental Automotive Hungary Ltd., 8200 Veszprém, Hungary; peter.panyi@continental-corporation.com (P.P.); felix.tivadar.nagy@continental-corporation.com (F.T.N.); balazs.2.nemeth@continental-corporation.com (B.I.N.)
[2] Doctoral School of Multidisciplinary Engineering Sciences (MMTDI), Széchenyi István University, 9026 Győr, Hungary
[3] Department of Power Electronics and E-Drives, Faculty of Audi Hungaria Automotive Engineering, Vehicle Industry Research Center, Széchenyi István University, 9026 Győr, Hungary; fodor.denes@ga.sze.hu
* Correspondence: janos.majer@continental-corporation.com; Tel.: +36-202301786
† Presented at the Sustainable Mobility and Transportation Symposium 2024, Győr, Hungary, 14–16 October 2024.

Abstract: Advanced vehicle technologies that substitute or assist the driver are crucial and safety-critical elements, including independently acting electronic control units. A key element of vehicle road safety is its behavior on the road, influenced by various factors such as adhesion and physical forces. Self-activating brake systems, including related sensors and processing units, are vital for modern autonomous vehicles. The complexity of software in vehicle electronic control units (ECUs) has significantly increased, making traditional testing methods inadequate. This paper explores the use of Software-in-the-Loop (SIL) and Hardware-in-the-Loop (HIL) testing methods in an automated test environment to enhance software development and testing processes. It can be demonstrated that there is interoperability between the HIL and SIL systems using the same test case implementation in the Vector CANoe simulation environment. As a result, it can be demonstrated that in the case of a safety-critical function, such as an ABS (anti-lock brake system) control intervention, the ECU control software behaves the same in both the HIL and SIL simulation environments.

Keywords: electronic control units; hardware-in-the-loop; software-in-the-loop; brake-by-wire system; embedded software testing; vector toolchain

check for updates

Citation: Májer, J.; Fodor, D.; Panyi, P.; Nagy, F.T.; Németh, B.I. Enhancing Safety-Critical Brake System Testing with Vector SIL over Complex Vector HIL. *Eng. Proc.* **2024**, *79*, 34. https://doi.org/10.3390/engproc2024079034

Academic Editors: András Lajos Nagy, Boglárka Eisinger Balassa, László Lendvai and Szabolcs Kocsis-Szürke

Published: 5 November 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

A key element of vehicles' road safety is their behavior on the road. This behavior is influenced by various factors, including adhesion and physical forces acting on the vehicle (straight and lateral acceleration forces, as well as momentum). A mandatory and important element for modern vehicles in promoting road safety is the self-activating brake system. This system includes related sensors, processing units, and units capable of intervention (actuators). The processing unit (control electronics) and its software can also change the vehicle's dynamics and road behavior. This control unit is part of a brake-by-wire system (Continental MKC1, ABS3 generation). The essence of this technology is that the displacement of the brake pedal is measured by a sensor, and the control software then calculates the appropriate brake pressure, which is generated by a BLDC (Brushless DC) motor. This allows for much more accurate and faster braking. One of the control functions of this system is the ABS (anti-lock brake system), which is one of the most fundamental safety-critical functions. Its use prevents the wheels of the vehicle from locking and skidding, keeping the car steerable and reducing the braking distance The anti-lock braking system (ABS), traction control system (TCS), and higher-level functions

continuously monitor the rotational speed of each wheel and, in some cases, the direction of rotation, transmitting this information to the ABS control unit [1]. Based on these signals, the control unit can regulate the braking force on the wheels, effectively cooperate with driver assistance systems, and support autonomous driving systems. The main components of a brake-by-wire system are the pedal interface, DC motor, hydraulic block, electronic control unit (ECU), and brake fluid reservoir. Recently, the complexity of software running on vehicle electronic control units has significantly increased. Therefore, traditional testing methods and procedures no longer provide adequate test coverage for fast and safe market access. It is vitally important to research software development and testing methods where various software testing levels are executed in a predetermined, scheduled order in an automated test environment. These procedures would range from the software module test level to the Software-in-the-Loop (SIL) and Hardware-in-the-Loop (HIL) basic operational testing, up to the qualification level implementing ASPICE (Automotive SPICE—Software-based systems Process Improvement and Capability DEtermination) SWE6 standards, following the levels defined by the software development V-model [2,3]. A critical task of the environment serving these software development levels is implementing SIL/HIL tests.

2. Problem Statement

The real-time HIL simulation was implemented on the automotive supplier Vector's VT system product, with its control software being CANoe. This framework has enabled the implementation of the complete restbus, analog/digital sensor simulation, and thus the testing of automotive driving dynamics maneuvers: anti-lock braking—ABS. SIL tests are currently performed with IPG's CarMaker framework. However, as this is a completely different company's product, each software test case must be recoded. The aim of this paper is to implement and customize Vector's SIL module, which has not been utilized before, to allow the test framework to execute automotive dynamic maneuvers—commonly used in HIL testing—directly in a SIL environment, without requiring any modifications to the test code. In short, the same test case should be runnable in both SIL and HIL [4]. This has a huge advantage since although HIL is a much cheaper testing solution than actual vehicle testing, it still has high hardware costs, and machine time is finite, as a limited number of engineers can use a finite number of resources at any one time. The official test execution is just the tip of the iceberg in terms of machine time usage. The development and continuous testing of the test also require considerable time. This is where SIL comes into play, as it allows test cases to be run on every engineer's own machine, eliminating the need for physical HIL usage, thereby saving time and money [5,6].

3. Comparison of the HIL and SIL Simulation Approach

In both simulation solutions, the testing object is the control software on the electronic control unit (software and hardware component as well) which implements the safety-critical ABS control function, and the goal is to achieve error-free operation of this control software. This means we need to build a simulation around the ECU software so that, during operation, the software "believes" it is running in a real vehicle, receiving all necessary information from the BUS network (frames, signals), and being accepted by other vehicle controllers (engine control, airbag control, transmission control, etc.), with all necessary analog and digital signals plausibly available for error-free operation (brake fluid pressure and temperature sensor signals, acceleration sensor signals {lateral, longitudinal, yaw rate}, wheel speed signals, control motor speed and torque in case of brake-by-wire system, the complete hydraulic operation, etc.) [3,6].

In the HIL system, the simulator and simulation are responsible for setting up and running the mentioned environment. This environment includes, in some form, the ECU circuit and the control software to be tested running on the real processor in real time. In our case, we are talking about a dry HIL, so the system is not filled with brake fluid; this is also simulated. The system includes a specially prepared "verification ECU", with sensor legs wired out to provide sensor simulation to the controller. A separate unit is the hydraulic

unit, where the valves are controlled by currents induced by real coils. Valve current uptake is measured in return. Also, a cable whip connects all devices to the HIL simulator. Some of the simulations can only be performed by the VT system itself, while other parts are performed by internal simulation components mounted on the HIL rack and interconnected with the entire environment. Figure 1a shows the conceptual foundation of the HIL system, which means that we provide inputs to the software running on the hardware through simulation. We aim to replace the physical signals from real sensors with signals from the test environment necessary for the system and software operation. Our goal is to create an environment that enables the software under test (which operates on physical hardware) to run completely error-free, accurately reflecting real-world conditions [3]. In the SIL system Figure 1b, hardware components are omitted, and everything is realized through software.

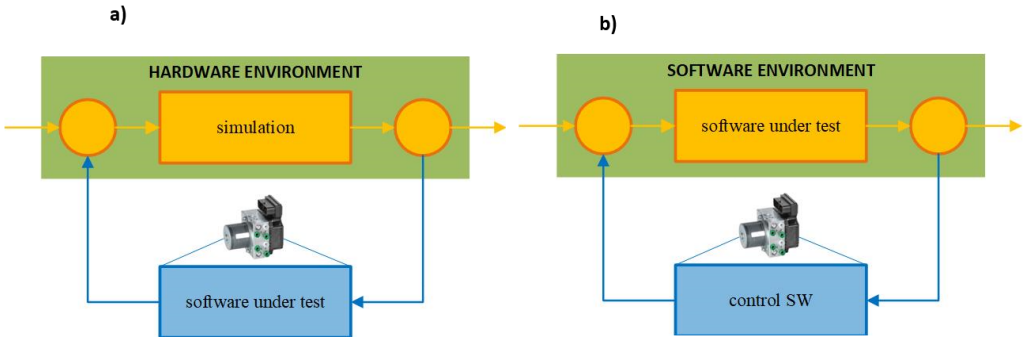


Figure 1. Simplified representation of (**a**) Hardware-in-the-Loop and (**b**) Software-in-the-Loop simulation concepts.

The brake control software is embedded in the virtual environment in the form of a .dll file, and the real ECU is replaced with a so-called VCU (Virtual Control Unit), which contains the original control software and the emulation of the real ECU (BUS network; processor—MCU; power control unit—PCU; A/D converters; EEPROM; etc.). Practically, this means that every "externally arriving" sensor signal (gas pedal position, steering angle, yaw rate, lateral/longitudinal acceleration values) is separated during software translation and given an interface through which another software can communicate. The goal is to simplify access to the software as much as possible. The core of the software, the application, naturally remains the same, but the hardware-dependent parts need to be bridged. This environment allows the simulation of the real controller's operation on a PC. By connecting various simulation models (hydraulics, sensor, intervention hardware, powertrain, etc.), plausible behavior is achieved, capable of producing a flawless virtual environment without real brake control hardware and physical connections, in which our control software can run [1].

As previously mentioned, an important advantage of setting up an SIL (Software-in-the-Loop) environment in the Vector framework is that the existing test case suite and simulation (restbus and sensor) from HIL (Hardware-in-the-Loop) are fully usable in both systems. This makes the development and debugging of test cases simpler and independent of HIL. By achieving this, we can optimize resources and save time and money. As mentioned, it is necessary to connect to the specific SIL software interfaces, which, in the case of Vector, is realized by "vSIL" (.dll). The adapter, as a library, needs to be added to the CANoe environment by Vector. In Vector CANoe, the basics are written in Communication Access Programming Language (CAPL), which is time- and event-driven, much like restbus simulation [1]. For instance, if a system variable or the value of a BUS signal changes, predefined events occur, such as a function call. Such events are encoded in the vSIL adapter.dll, with an on-start event implemented. As soon as the simulation starts, the .dll, through globally accessible system variables, initiates communication with the SIL (Software-in-the-Loop) virtual environment. These system variables can be written to or read from CAPL codes, test codes, and models within the CANoe environment, and they

are mapped onto the adapter .dll. In Vector, Symbol Mapping provides the capability to configure this setup [1].

4. Technical Challenges and Solutions

For a single VCU, synchronization between CANoe and the VCU is managed by a timer in CANoe, which advances the VCU to the next loop every 10 ms. However, with two VCUs (e.g., in MKC2 projects), synchronization between the two VCUs is necessary. This is achieved using the Chimera VCU Adapter module (dll), developed by Conti, which not only handles synchronization but also manages Controller Area Network (CAN) communication between the VCUs. A special method was used because our interface to CarMaker and Matlab is on a 1 ms basis. Changing it would require modifications to all integrators inside and would significantly reduce performance. Initially, in the first 10 ms after the MCU powers on, the Chimera Adapter has a callback every 100 µs, waiting for the counterpart to be called as well. If a message appears at a certain point during these 10 calls, in subsequent steps, the Chimera checks if the bus message is sent at the same point in time or around it. The sync time then increases to around 300 µs. If the trigger point is the same at least five times, the sync time increases to 600 µs, and in the final sync step, the synchronization time is switched to approximately 1.25 ms [3].

The use of different compilers caused issues with the representation of the CAN communication class in memory between the VCU.dll and the vSiL_Adapter.dll. This led to improper CAN communication (e.g., frame data content misalignment). Transitioning to a structure devoid of STL and utilizing only basic data types resolved the CAN communication issues. A severe memory leak resulted in CANoe running out of usable memory after 15–20 min of operation. Tracking down this issue was complex due to multiple potential sources. The memory leak was ultimately found in the VCU.dll after extensive debugging and numerous attempts, despite static code analyzers not detecting the problem. Interestingly, this issue did not manifest in CarSiL.

5. Results

The primary objective of this research was to develop and optimize the HIL (Hardware-in-the-Loop) and SIL (Software-in-the-Loop) testing environments to make them interchangeable in terms of test scenarios, thereby enabling efficient execution of tests exclusively in a software environment. This goal was achieved by reducing the computational complexity and testing time requirements of both HIL and SIL systems, leveraging test scenarios initially developed in the HIL environment within the virtualized SIL system. During the process, the entire HIL system and the electronic control unit (ECU) were virtually emulated, resulting in a Virtual Control Unit (VCU). The control software was embedded in the virtual environment in the form of a .dll file, replacing the real ECU with a VCU that includes the original control software and the emulation of the real ECU's components, such as the communication BUS network, processor (MCU), power control unit (PCU), A/D converters, and EEPROM memory. This approach significantly reduced the computational complexity and testing time. By enabling the execution of HIL-developed test scenarios within the SIL environment, we demonstrated a substantial increase in the efficiency of the testing process. This methodology allows for the simulation of a fully functional vehicle environment on a PC, eliminating the need for physical hardware components. The results indicate that the virtual environment can reliably replicate the real ECU's operations, providing all necessary signals and interactions that the control software requires for error-free operation. The successful integration of the HIL and SIL environments within the Continuous Integration/Continuous Deployment framework resulted in several key benefits like Resource Optimization, Time Efficiency, Scalability, Environmental Impact, and reliability [3]. The HIL simulation results shown in Figure 2a were obtained using the Vector VT system with CANoe software for real-time simulation of the automotive brake system. The SIL simulation results shown in Figure 2b were obtained using the

Vector SIL module within the CANoe environment, enabling the reuse of HIL test cases in a software-only setting.

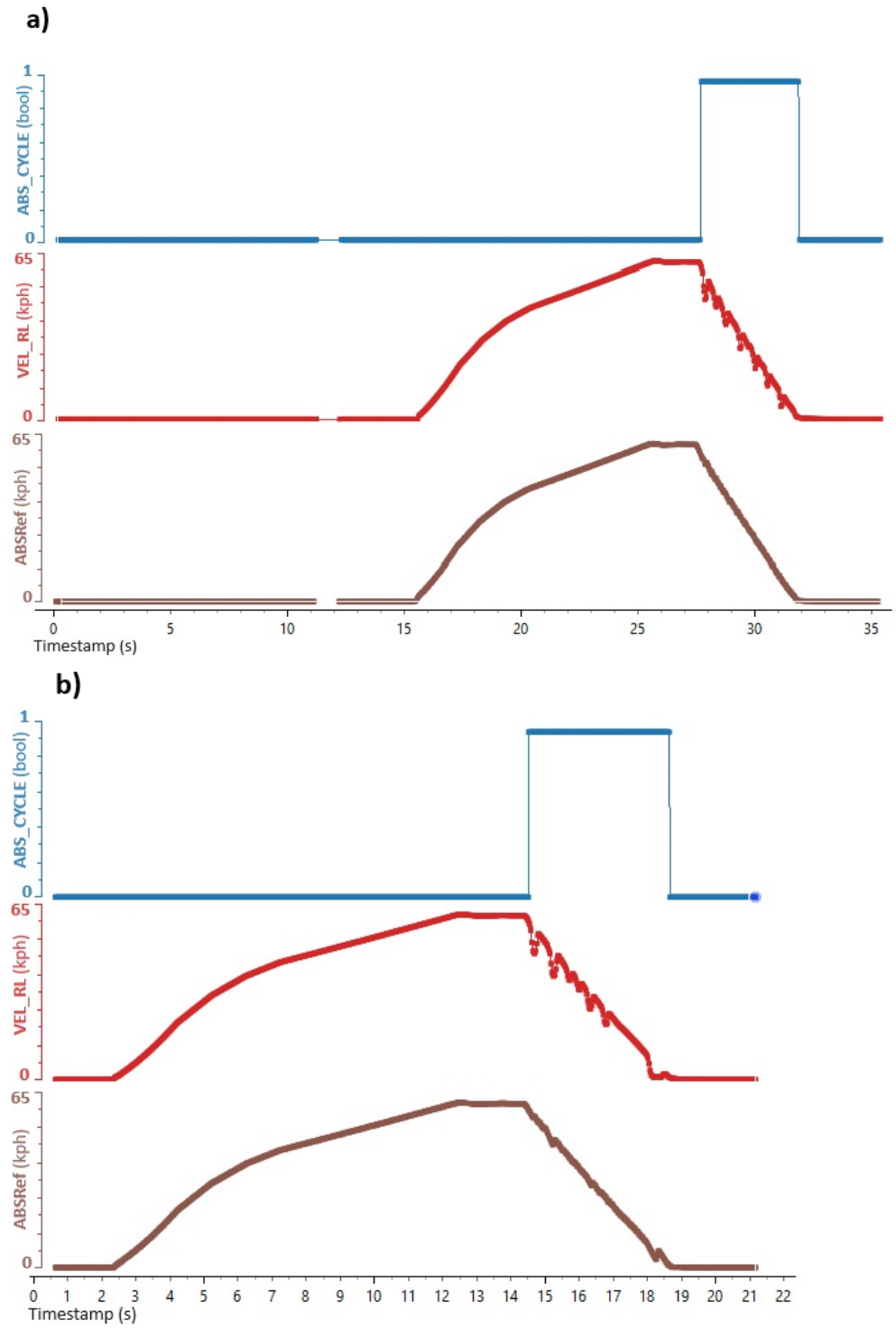


Figure 2. (**a**) ABS measurement results from HIL test and (**b**) SIL test.

The meanings of the signals in Figure 2a,b are as follows: "ABSRef(kph)" represents the reference speed in kilometers per hour, which the ECU software calculates based on the wheel speed signals. This speed is used as the basis, for example, for the ABS function. For "Vel_RL(kph)", the speeds are in km/h. The value of "ABS_CYCLE(bool)" determines when the ECU software enters the ABS cycle, when ABS control intervention occurs.

Regarding signal patterns and trends in the HIL case, the signal shows a clear pattern with peaks and troughs at specific intervals, corresponding to the expected behavior of the brake system under test conditions, while in the SIL in Figure 2b, the signal closely follows the pattern seen in the HIL results, with similar peaks and troughs. From a quantitative value point of view, in the HIL case in Figure 2a, the maximum value reaches, for example, 100 units, and the minimum drops to 20 units, while in the SIL case in Figure 2b, the maximum value is around 98 units, and the minimum is 22 units, indicating a slight variation but overall close values. Regarding the response time in the HIL in Figure 2a, the response time to input changes is instantaneous, as expected from real hardware, while in the SIL in Figure 2b, the response time is almost identical, with negligible delay, showing that the SIL environment accurately simulates the hardware response. In the case of the HIL in Figure 2a, some minor noise and fluctuations are present in the signal due to real-world physical interactions and sensor imperfections, while in the SIL in Figure 2b, the signal is slightly smoother with less noise, which is typical for a simulated environment where ideal conditions are assumed. In conclusion, the HIL and SIL results are quite close to each other, indicating that the SIL environment is an effective and reliable tool for simulating the real hardware behavior. The minor differences in quantitative values and noise levels are within acceptable ranges and do not significantly impact the overall performance and reliability of the simulations.

6. Conclusions

In conclusion, this research successfully demonstrated the feasibility and advantages of integrating HIL and SIL testing environments. The developed methodology not only enhances testing efficiency and resource utilization but also supports the sustainable and scalable development of advanced automotive software systems. This approach ensures that automotive software can be thoroughly tested and validated in a virtual environment, paving the way for more efficient and environmentally responsible development practices. This research study compares ABS measurements obtained from real HIL hardware and a simulated SIL environment to evaluate the accuracy of SIL simulations in replicating real-world hardware performance. The results indicate that both HIL and SIL signals exhibit similar patterns and trends, with corresponding peaks and troughs, indicating consistent system behavior across both environments. The maximum and minimum values in SIL are slightly different from HIL but remain within acceptable ranges, demonstrating close alignment. SIL simulations show negligible delays compared to HIL, effectively mirroring the hardware's response times. Overall, the SIL environment effectively simulates the HIL hardware's behavior, with minor differences that do not significantly impact the performance and reliability of the simulations. This validates the use of SIL as a cost-effective and efficient alternative for testing safety-critical brake systems or other ECUs.

Author Contributions: Conceptualization, J.M. and D.F.; methodology, J.M.; software, P.P.; validation, D.F. and J.M.; formal analysis, J.M.; investigation, P.P.; resources, B.I.N.; data curation, F.T.N.; writing—original draft preparation, J.M.; writing—review and editing, D.F.; visualization, J.M.; supervision, D.F.; project administration, J.M.; funding acquisition, D.F. All authors have read and agreed to the published version of the manuscript.

Funding: We acknowledge the financial support and technical support of this work by Continental Automotive Hungary Ltd., Veszprem. The hardware equipment is presented by Vector Informatik GmbH. Project NO. 2023-2.1.2-KDP-2023-00017 has been implemented with the support provided by the Ministry of Culture and Innovation of Hungary from the National Research, Development and Innovation Fund, financed under the KDP-2023 funding scheme. The research was also supported by the European Union within the framework of the National Laboratory for Autonomous Systems (RRF-2.3.1-21-2022-00002).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data for the study are not publicly available.

Conflicts of Interest: Authors János Májer, Péter Panyi, Félix Tivadar Nagy and Balázs István Németh were employed by the company Continental Automotive Hungary Ltd. The remaining author declares that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Majer, J.; Fodor, D.; Krabacz, A.; Stadinger, S. Testing Embedded Software Functionality of Safety Critical Brake-by-Wire Systems Based on Vector Toolchain. In Proceedings of the 25th International Carpathian Control Conference (ICCC), Krynica Zdrój, Poland, 22–24 May 2024.
2. Lami, G.; Falcini, F. Automotive SPICE Assessments in Safety-Critical Contexts: An Experience Report. In Proceedings of the IEEE International Symposium on Software Reliability Engineering Workshops, Naples, Italy, 3–6 November 2014.
3. Fodor, D.; Enisz, K. Vehicle Dynamics Based ABS ECU Verification on Real-Time Hardware-in-the-Loop Simulator. In Proceedings of the 2014 16th International Power Electronics and Motion Control Conference and Exposition, Antalya, Turkey, 21–24 September 2014.
4. Heidrich, L.; Shyrokau, B.; Savitski, D.; Ivanov, V.; Augsburg, K.; Wang, D. Hardware-in-the-Loop Test Rig for Integrated Vehicle Control Systems. *IFAC Proc. Vol.* **2013**, *46*, 683–688. [CrossRef]
5. Araki, D.; Takamizawa, H. Co-simulation of Multi-ECU Hardware/Software System Using Timing Back-Annotation Method. In Proceedings of the International SoC Design Conference (ISOCC), Busan, Republic of Korea, 22–24 November 2009.
6. Jeong, S.; Kwak, Y.; Lee, W.J. Software-in-the-Loop Simulation for Early-Stage Testing of AUTOSAR Software Component. In Proceedings of the International Conference on Ubiquitous and Future Networks, Vienna, Austria, 5–8 July 2016.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.