Build Automation Framework for Design Validation of Automotive Gateway Controllers

Angela Gonzalez Mariño
Huawei Technologies Duesseldorf Gmbh
Munich, Germany
angela.gonzalez.marino@huawei.com

Nikhil Naganath Halinge
Huawei Technologies Duesseldorf Gmbh
Munich, Germany
nikhil.naganath.halinge@huawei.com

Francesc Fons
Huawei Technologies Duesseldorf Gmbh
Munich, Germany
francesc.fons@huawei.com

Juan Manuel Moreno Arostegui
Technical University of Catalunya (UPC)
Barcelona, Spain
joan.manuel.moreno@upc.edu

***Abstract*— Complex systems require an appropriate development methodology through the complete design lifecycle, from concept definition to system validation. In the end, the value and performance of a design are limited by how well it can be tested and validated. In this work, authors present an automatic framework for design validation of automotive Gateway controllers. The framework covers the validation process, going from test-case specification to results delivery in a fully automatic way. The framework relies on the use of standard Packet-CAPture (PCAP) files with frame traces, as a mean of injecting stimuli in the system and of output recording. This allows to repeat and reproduce experiments, following ACM guidelines. The data processing required to analyze the test results is done taking these PCAP files as input, allowing to automate the data analysis in a standardized way. The test files are automatically generated in the framework according to the design files and the test configuration selected by the system architect, providing a fast, flexible, scalable and human-error-free validation methodology. The design files of the system are also automated via a build automation framework for gateway design, introduced by authors in previous work. The design automation framework covered the automatic Design & Development of GW devices, and now the design validation framework covers the automatic Verification & Validation steps. Therefore, with the new build automation framework for design validation, the full design lifecycle of automotive gateway controllers following the V-model is completed. Finally, the validation framework provides both simulation and in-target testing capabilities, relying on the standard PCAP files strategy, getting the best of both worlds – simulation and in-target testing – blending digital with physical, and completing the last step of the product design life cycle, in a fully automatic way.**

Keywords—design methodology, system validation, automatic testing, in-vehicle networks, HW-SW co-design

I. INTRODUCTION

Autonomous driving, electrification, shared mobility and connected mobility shape the future vision of the automotive industry. Today, the scientific community is actively researching on the different technologies that need to be integrated into vehicles, in order to turn this vision into a reality. From the integration of new sensors that enhance environmental awareness of the vehicle, to machine learning applications that make decisions based on the data gathered, there is an essential part of the vehicle that needs to be safe, secure, robust and reliable: the In-Vehicle Network (IVN). Therefore, the design of IVNs and their components becomes of utmost importance when trying to reach higher levels of autonomy. Moreover, it is not only the design of the network and its components that takes a starring role in the vehicle, but also the verification and validation of this design, i.e. in order to guarantee the correct performance, a thorough validation needs to be in place proving that the requirements are met. This corresponds with the well stablished V-Model that guides the design lifecycle from Design and Development (D&D) to Verification and Validation (V&V), as shown in Fig. 1.

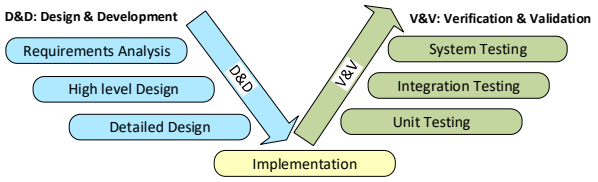

Fig. 1. V-Model design lifecycle

For the V&V, two complementary approaches can be followed. The first approach is to simulate the design of the network with its components, allowing to validate early concepts without the need for a real hardware (HW) deployment. Simulations provide full control of the network and devices parameters, allowing to inspect the behavior of the Device Under Test (DUT), especially in early stages. The second approach, is to run real test cases on the target HW, which provides the most accurate and realistic results. However, this is more costly, since it implies building a prototype. In both approaches, to maximize the benefits and insights on the design performance gained from the V&V stage, it is important to follow a methodology that allows for repeatable and reproducible experiments that lead to consistent conclusions and results.

Following this philosophy, authors in [1] present a generic framework for IVN experiments following ACM repeatability and reproducibility guidelines [2], focusing on system level experiments with open source and commercial off-the-shelf solutions. In [3], authors introduce a build automation framework for architecture design of gateway (GW) controllers, which automates the D&D phases of GWs design. Now, in this work, we propose to take the same philosophy of generic, repeatable and reproducible experiments introduced in [1], to the GW device level. For this purpose, we present a build automation framework for design validation of GW controllers, covering the V&V stages of the design lifecycle. This, together with the work introduced in [3], provides a full automatic framework for the complete lifecycle of gateways. The main contributions (C) of this work are listed below.

- **C1. A full automatic validation methodology for GW devices** that blends the virtual and physical system providing the right tools to execute test cases in either or both of them.

- **C2. An automatic virtual system generation** based on a library of IPCores that enables fast simulation of the final system design accelerating the design lifecycle.

- **C3. A standard-based framework** that allows to generate and reproduce test cases both in virtual and physical system, to compare and merge the obtained results, and to automate the data post-processing in a standardized way.

Next, an analysis of the state of the art in several areas related to this work is presented. Later we expose the details of the presented framework and contributions. Finally we evaluate the proposal and discuss the insights gained during this work together with future research opportunities.

II. Related Art

In this section, an overview of the state of the art technologies and tools that are relevant for this work is given.

A. Automotive industry tools:

The automotive industry has traditionally developed Electronic Control Units (ECU) or computing platforms driven by software-based (SW) methodologies, following the Autosar SW stack [4]. For this purpose there are several professional tools that provide an integrated framework for the development of ECU software components, like Capital from Siemens or ASCET from ETAS. In the area of design verification and validation, the industry is moving towards automatic frameworks, however, as of today, there are still several challenges to overcome in this area, as pointed by authors in [5] and [6]. Regarding network protocols, until now the most widely used bus within IVNs was the Controller Area Network (CAN), for which several industrial validation tools exist, such as CANoe. Going in the direction of test automation, authors in [7] propose an automatic test framework for vehicle GWs based on CANoe. Nowadays, IVNs are shifting towards Ethernet based, therefore new frameworks for GW test automation are needed. The integration of Ethernet within IVNs imposes several challenges in functionality and design [8], but also brings a new world of possibilities both in design and validation space. Tools and standards for generic network debugging and validation can now be integrated into the automotive gateway design flow, allowing to leverage the knowledge developed in the wide Ethernet community, as explored in this work.

B. Time Sensitive Networking

Future IVNs have stringent requirements with regard to latency and Quality of Service. In order to meet these requirements over Ethernet, the IEEE is currently defining a set of standards that allow for providing the required determinism. These set of standards compose what is known as Time Sensitive Networking technologies (TSN) [9]. In the case of automotive, the P802.1DG [10] profile is currently under development. Based on these standards, there are some commercial offers to provide these features within Ethernet networks. TTTech Edge IP solution is targeting the specific area of TSN technologies integration within Industrial and IVNs. Their proposal is a library of IPCores designed to provide TSN capabilities. Some works in the literature also explore the configuration space of TSN technologies, proposing several approaches in order to automate the configuration [11]. The proposed protocol within TSN standards for the configuration of the TSN parameters is NETCONF, and the language used for data models definition is YANG. An overview of how this is integrated within TSN networks is described in [12]. As seen in the literature, the configuration space of TSN is very broad, which gives great opportunities for design exploration. However, it also makes the choice of the right tools, and the validation of these choices very complex. Hence the need for a framework that allows to automate network experiments supporting TSN capabilities.

C. HW-SW co-design

The design, prototyping and validation of novel HW computing architectures has been tightly coupled to the capabilities offered by SoC prototyping platforms. The need for fast development cycles and design validation capabilities in the industry generates a dependency with the offer from leading SoC providers. Furthermore, this dependency is not only device related, but also tool or framework related, i.e. in order to create a design to be prototyped in a specific SoC device, one needs to use the development framework of the SoC provider (e.g. Vivado/Vitis from Xilinx, Quartus from Intel). Trying to break this dependency where possible, there are some open source alternative HW-SW development tools which try to cover the full IC design, from high level system definition to ASIC deployment, e.g. Qflow [13] or Yosys [14]. There are also several professional tools that provide an integrated environment for the complete design lifecycle of SoC like Synopsys. Nevertheless, for prototyping purposes, the need to merge with vendor-specific tools is still there. With any of the aforementioned options, technical HW related expertise is needed in order to make resource-efficient and high-performance designs, for which a non-negligible learning curve is required.

D. Automatic testing:

In order to produce high quality designs and results, a good testing strategy is required. This test strategy quite often benefits from automation features mainly because within the validation process we often find repetitive tasks where no value is added after the first iteration, and also because automation skips an important source of mistakes: human errors. One of the most popular frameworks for test automation in the industry is LabView which offers a user friendly programmatic style that allows to implement complete test suites for many different applications. Going towards open source, Keysight Technologies and Nokia started the OpenTAP project providing a generic framework for development and execution of automated tests. For SoC and IC design validation, most efforts on automatic testing go in the direction of architectural error detection as seen in the literature [15], [16]. However, functional validation can't be automated in such a generic way, because of its intrinsic dependency with the system functionality. To overcome this, in network devices, the OSI layered approach can be followed, leaving the functionality to the application layer (payload of frames) and building generic test frameworks that are able to exchange frames with the DUT. For this purpose, PCAP files are the standard used to store test patterns that can be injected in the system. An overview of available frameworks for this purpose, and a proposal for FPGA test implementation is detailed by authors in [17]. In this work, we follow this functional approach based on standard PCAP files, extending the capabilities of prior works in terms of testing automation for networking devices.

E. Simulation frameworks:

An essential part of system design verification is the simulation step. All HW development tools (both proprietary and open source) offer some simulation capabilities that permit to validate the design from a functional perspective before the actual deployment in the HW platform. One of the most common simulators used within proprietary tools is QuestaSim. Within the open source landscape we find some simulation tools such as Verilator, which translates Verilog into C++, GHDL which directly translates VHDL into machine code resulting in fast simulations, or gem5 [18] which allows for simulating custom computing architectures. Authors in [19] give an overview of the available open source simulation alternatives and propose a framework to integrate RTL models in gem5.

Overall, we see that in the state of the art, there are several options targeting the development lifecycle of systems design, with some automation options for SW based products. However, none of the available solutions provides a complete and automatic framework to the design, development, verification and validation of HW based networking devices. In the work presented in this paper, we focus on the V&V part of the development lifecycle, providing a complete automation solution for GW devices.

III. GW Design Automation Framework

In this section, an overview of the design automation framework for GW devices described in [3] is given. For the design automation, the framework relies on a library of IPCores that provide the configuration capabilities required to accomplish a complete GW design. The system architecture proposed by authors for GW devices which covers all the functionalities required within IVNs is depicted in Fig. 2. Then, the framework used to configure this system architecture, and finally deploy different GW designs targeting from high-end to low-end vehicles is depicted in Fig. 3. Within a WebApp ① the GW designer can choose the design options which, after being processed, are stored into a JSON structure ②. These options affect both at the high level design of the gateway (number of ingress/egress ports) and at the low level design of the gateway, allowing to configure different parameters in the IPCores, e.g. network protocol per port, TSN standards in place, etc. Then the documentation for the design ③, the HDL code ④ and the system netlist ⑤ are automatically generated by the framework. Overall, this framework covers the D&D phases of the V-model (Fig. 1).

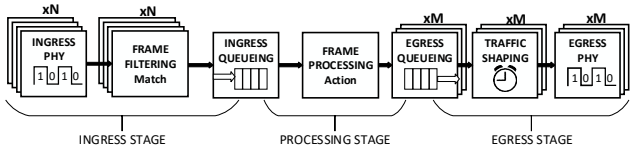

Fig. 2. High Level Design of Gateway Architecture

Some of the IPCores that compose the library with the configurable features per each IPCore are detailed in [20], [21], [22], [23], [24] and [25]. Basically, the design framework allows to automate the D&D of HW-based gateways by means of providing configuration options over a set of pre-defined IPCores. Then, this design is the one used within the validation framework as DUT, as seen in Fig. 4.

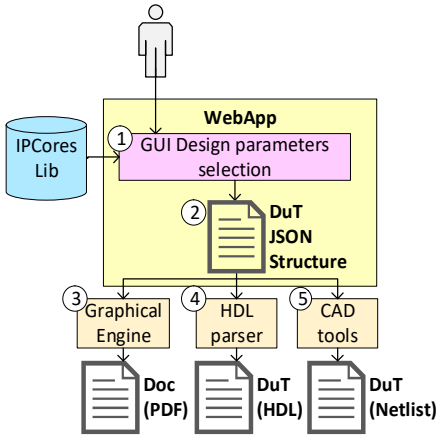

Fig. 3. Build Automation Framework design flow

IV. GW Validation Framework Proposal

In this section, a high level view of the proposed standard-based automatic test framework for gateway devices is described. This framework focuses on the V&V of the design generated with the previously described design framework, completing thus the product design lifecycle exposed in Fig. 1. The framework system architecture is depicted in Fig. 4, where it is shown that the input and output of the experiments are always standard PCAP files. This enables the scalability of the framework through easy standard-based extensions, and the reuse of data analysis tools. Within the system, there are two main blocks, corresponding to the two main options available within the framework. On the right side, there is the physical system, where real test cases can be deployed. On the left side, there is the virtual system, which allows for executing simulation based test cases. With both approaches, the framework covers from unit tests (IPCores validation) to integration tests (combination of some IPCores) and to system tests (full datapath with all required IPCores), providing a common, flexible and scalable framework for the whole validation phase of the GW design.

A. Physical System

The physical system is composed of two traffic management units (traffic generator and traffic logger) together with the HW prototype of the GW device in a SoC platform. For traffic generation and logging, several options can be used depending on the requirements of the test case. One option can be to use a regular PC with an open source traffic generator such as TRex or Ostinato. This provides an easy to use and low cost environment for network testing, with the limitations of a consumer PC. In order to overcome these limitations, a Network Interface Card can be used, providing a combination between SW and HW capabilities, as exposed by authors in MoonGen [26]. Another option when the test cases require higher data rates, real-time performance, or higher number of ports, is to use professional HW equipment such as Ixia or Spirent. The GW DUT is the design under test, which is automatically generated with the design automation framework previously described.

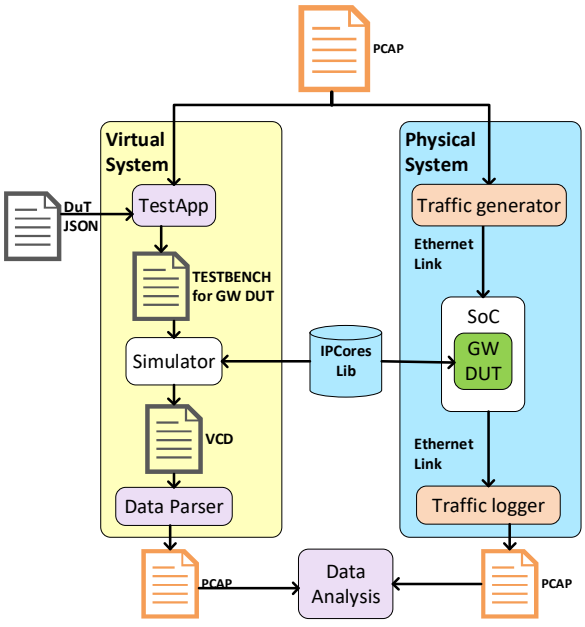

Fig. 4. Validation framework high level overview

B. *Virtual System*

The virtual system includes an automatically generated virtual version of the physical system. This is achieved by using the DUT generated within the design automation framework explained above, which is the same used within the physical system. For the validation, the system architect can select not only the test case through the PCAP file, but also other configuration parameters related to the simulation, e.g. clock frequency. The automatic flow of the virtual system is graphically exposed in Fig. 5. First, a TestApp serves as user interface where the test configuration options can be selected ①. Afterwards, an HDL testbench is automatically generated based on the selections and the design of the DUT ②. This testbench allows to read as many PCAP files as input ports and to inject the corresponding HDL stimuli to the DUT. For the simulation, the automatically generated testbench together with the IPCores in the library and the automatically generated GW DUT design are used. The simulation is executed within a simulation framework ③ and the result is recorded in VCD format (Value Change Dump) since most simulators provide this output. Finally the results are translated into a PCAP file ④ achieving thus the standard compliance.

Next, the main advantages of the design validation framework proposed in this work are summarized:

- Application Specific Automatic validation framework: The framework is highly optimized for network devices validation and provides a user friendly interface while accelerating system testing, from unit test to system test.

- Standard compliance: Through the use of PCAP files that permit to reproduce real scenarios in the test environment, allowing for repeatability and reproducibility of experiments.

- Blend of virtual and physical system: Capability of executing identical tests in simulation and real platform, providing insights gained from exploratory simulations as well as from real platform outcomes. This combination gives the greatest overview of system performance and allows for a better understanding of the system limitations.

- Automatic processing of test results: Getting standard PCAP files both from virtual and physical systems allows for simplifying results analysis since the same tools can be used in both cases. Depending on the designer needs, data analysis can range from visually inspecting the PCAP results in WireShark, to the development of custom scripts that extract the required metrics, e.g. observing input and output timestamps to determine the latency of frames.

- Flexibility and Scalability: The validation framework is completely flexible allowing to validate any design combination generated by the design automation framework, with any number of ingress/egress ports. The virtual system is also flexible allowing to use different versions of the IPCores library for exploratory research. Furthermore, the framework provides the possibility to easily extend analysis capabilities via python scripts that extract the information required from each experiment.

To the best of our knowledge, there is no automatic framework for GW devices validation in the state of the art, combining simulation and real platform testing, and providing the same level of automation, flexibility and scalability as our proposal, being this the main contribution of this work.

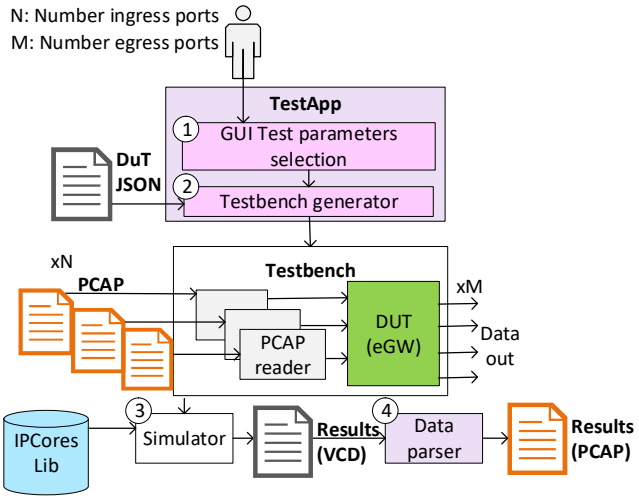

Fig. 5. Build Automation Framework validation flow

V. Analysis and Discussion

In this section we discuss the Proof of Concept implementation of the proposed automatic design validation framework. First we give an overview of the different elements used in order to compose the whole framework, combining third party tools with custom tools. For each of them we provide the reasoning behind this choice and possible alternatives for implementation. Then we show an example of design validation using the framework. Afterwards we discuss the challenges faced when defining the framework concept and how they are solved in our proposal. Then we also identify some opportunities that are open within the framework and finally we discuss possible future works.

A. *Proof of Concept implementation*

In order to prove the feasibility of the framework concept, a minimum viable product (MVP) has been implemented running the full flow through it. On the virtual system, a web application is developed for the TestApp component, joining this framework with the build automation design framework introduced in [3]. The TestApp offers the minimum set of configuration parameters required for the system, i.e. choice of PCAP files used as input, interframe gap for input frames and clock frequency for the execution. After the selection of configuration options, a python script automatically generates a HDL testbench, following the structure shown in Fig. 5. These two custom developments provide the required flexibility and scalability to automate the test generation and, to the best of our knowledge, are not possible with this level of integration in state of the art applications. Afterwards, the simulation can be run from the application with a button click. In this implementation, the simulator used is GHDL. This simulator is chosen for its simplicity of use, fast simulation results and open source. However any of the simulators discussed in Section II could be integrated. The application launches a python script that executes the required GHDL commands in order to obtain a VCD waveform file with the simulation results. Finally, this VCD file is translated to PCAP via a new python script, achieving thus the standard compatibility. With the integration of all these steps, an MVP of the virtual system and the automatic generation of a virtual system has been achieved. This MVP allows for proving that the full automation from user test definition to execution and results deployment is feasible within the proposed framework.

For the MVP validation, several GW designs with different features are generated with the design automation framework, and for each of them a testbench is generated with the validation framework. In order to showcase the capabilities of the validation framework, we show an example of evaluation of TSN technologies integration within the gateway. In particular we look at the impact of using Credit Based Shaper (CBS) algorithm [9] in video and audio streams. CBS algorithm allows to limit the amount of bandwidth (BW) used by a flow by giving it a specific credit. When flows run out of credit, they cannot transmit, allowing other flows to make use of the available BW. The internal details of the design evaluated here and a more complete set of experiments evaluating more TSN algorithms are available in [22].

According to the TestApp options, it is possible to select the input PCAP file for each port and run a simulation with it. In this case we follow the definition for IVN traffic presented in [22] and [27]. Finally, the result simulation is converted to PCAP, and this PCAP is processed with a Python script in order to evaluate metrics such as latency per frame or message drops. For the purpose of this MVP, the traffic is generated synthetically, since the focus is on the framework evaluation. In future work, we plan to extend these tests by using recorded traffic of real sensors and actuators present in the IVN in order to evaluate more realistic scenarios. Regarding data analysis, for the MVP the basic KPIs of network devices are measured: frame latency and drops. After the execution of the test either in virtual system or physical system, a report is generated including this information.

Table I shows the results obtained for two experiments. The first one runs over a gateway design that is not implementing CBS, and the second one over a gateway design that is using CBS to limit the BW consumption of video flows. For both cases, a testbench is automatically generated in the tool, and the same input traffic is used. As seen in the table, when CBS is not used, audio flows suffer a great delay because of high consumption by video flows. When CBS is used, video flows leave space for audio flows and the maximum and average latencies in the overall system improve. With this example we show how the validation framework allows to evaluate different gateway designs in a simple, standardized and flexible way. For more details on the evaluation of TSN algorithms in automotive GWs see [22].

TABLE I. DATA ANALYSIS REPORT

Experiment I: No CBS					
Flow	**Ingress**	**Drops**	**Latency**		
# Flow type	# frames in	# drops	Min.	Max.	Avg.
Video ADAS	167	0	3	2177	178
Video vision	501	0	3	3756	206
Audio	32	0	4234	9794	6723
Experiment II: CBS in video flows					
Flow	**Ingress**	**Drops**	**Latency**		
# Flow type	# frames in	# drops	Min.	Max.	Avg.
Video ADAS	167	0	4	3923	633
Video vision	501	0	4	4494	738
Audio	32	0	2718	5861	3831

On the physical system, the same GW designs previously simulated, can now be validated in the real HW. For traffic generation, several options are available as discussed in the previous section, depending on the HW availability and requirements of the system test. Since the traffic generation and logging is standardized by using PCAP files, the same synthetic/real traffic used in the virtual system can be also injected in the target system, validating the final design.

B. Challenges and Opportunities

When defining the concept and design of the validation framework, we faced several challenges (Ch.) that in one way or another, contributed to shape the final definition. The most relevant ones and how we overcame them are detailed below.

- *Ch. 1: How to manage the test automation for a complex system like a gateway?* Gateway devices are complex designs with many variables to be taken into account. Trying to handle everything in an automatic manner for validation imposes a great challenge. The approach followed in the proposed framework is to decouple application functionality from network functionality, i.e. to focus the validation on the networking behavior of the device, and keep the application level functionality validation separate. This is possible through the use of standard PCAP files, which allow for providing this level of abstraction at test automation level.

- *Ch. 2: Ok, I have the PCAP results for analysis, where do I start?* The PCAP format provides a standardized way of inspecting packets with great detail, but is also a very low level view of what is happening on the network. This is why we keep the data analysis step open and easily extendable via custom scripts. For initial analysis, the typical metrics to inspect are packet drops and latency. Both of them can be achieved by inspecting frame IDs and timestamps. For further analysis, other aspects can become relevant such as traffic filtering and policing rules.

- *Ch. 3: What is the best option for traffic generation / logging?* As introduced before, traffic management is an important part of the real system deployment. Here there are several aspects to consider when selecting the best traffic generation / logging option. On one side, there are several SW based applications for traffic management, like TRex or Ostinato. These are good options for fast prototyping, with low HW cost and high availability, since they run in a regular PC. However, there are several limitations in terms of performance that are related to the PC used, e.g. the number of ports available, or the maximum frequency at which the ports can operate. These limitations can be improved to some extent with the integration of NICs, as in [23]. On the other side, there are professional solutions such as Ixia or Spirent. These overcome the performance issues mentioned before, but come at a non-negligible cost. Additionally, due to the cost factor, this kind of systems are usually shared by teams within research labs and therefore availability is not as convenient as in SW solutions. Considering all these items, there is a tradeoff between the availability and cost of the traffic management and the performance of the test cases that can be deployed. From authors' perspective, the ideal solution is a combination of both: using the SW options for initial debugging of the system (when usually test requirements are not high, but the system availability is key to ensure a fast start up), and changing to the professional solutions for more in depth and mature evaluations (which can be planned according to environmental constrains). The framework proposed in this work allows for this alternation, since the traffic management is separate from the framework.

During the concept definition and design of the framework we also identify several opportunities (Opp.) for extension.

- *Opp. 1: Open data analysis.* As mentioned before, the analysis of data is kept open in the framework due to the wide variety of alternatives that can fit within this feature. The development of advanced data analysis applications is one of the future research lines considered by authors.

- *Opp. 2 → Alternative tools integration.* In our proposal, the different steps of the framework are self-contained and interconnected via standard interfaces. This provides a modular approach where the different pieces can be replaced by new ones with further functionalities and highlights the flexibility and scalability of the framework.

C. Future work

With the current implementation of the design validation framework as a baseline, there are two main lines of future research. On one side, it is interesting to perform the evaluation of more complex GW designs, both within the virtual and physical system. It is expected that this evaluation will provide new insights into the architecture design and allow to iterate the design of complex GW systems. It is of special interest to evaluate the integration of TSN technologies within the GW architecture, since this is a key technology to be integrated within IVNs, as seen in Section II. Actually, TSN is a good fit for this framework, since the combination and orchestration of TSN standards within IVNs is an active area of research where design and test automation can play an important role to accelerate results. On the other side, the design and development of further data analysis applications is another interesting research line. To define the right metrics to analyze within the GW, and to introduce the evaluation of TSN technologies in the results analysis will be essential to achieve the full automatic validation. The results of these future research lines are expected to derive in one or several follow-up publications.

VI. Conclusions

In this work, authors present a standard-based design validation framework for GW devices. The framework is oriented to automate the validation phase of GWs, eliminating human errors, specifically for IVNs. The framework focuses on generating generic, repeatable and reproducible experiments for network devices. It follows a standard approach, using PCAP files as input and output to the framework, being able to leverage all the existing artifacts and tools related to PCAP files processing in the state of the art. Furthermore, this approach allows for repeatability of experiments and reproducibility of real world scenarios within the testbed. The framework provides both a virtual system to perform simulation based tests, and the capability to execute tests in the target HW platform. This allows for comparing simulation and real world results. With this, the system and the test itself can be rapidly evolved, accelerating the research in this area. Both the virtual and physical system generate a PCAP file with the output of the test, allowing for a common evaluation of the results via specific data analysis frameworks. To the extent of its possibilities, the proposed framework relies on open source solutions allowing also a broader community to benefit from this contribution. For instance, the simulation and traffic generation steps are based on open source available options. However, the real HW platform prototyping is tightly linked to vendor specific CAD tools.

An MVP of the framework is discussed and the main challenges and opportunities identified during the definition and development of the framework are presented. Future lines of research derived from this work are introduced as well. To conclude, the work presented in this paper represents a step forward in the state of the art of GW devices validation, providing a complete automatic framework and method to evaluate the performance of the key component of the In-Vehicle Networks of the future.

References

[1] F. Rezabek et al. “EnGINE: Developing a Flexible Research Infrastructure for Reliable and Scalable Intra-Vehicular TSN Networks”. In: HiPNet. 2021.

[2] Artifact Review and Badging - Current. https://www.acm.org/publications/policies/artifact-review-and-badging-current.

[3] A. Gonzalez, N. Halingue, F. Fons and J.M. Moreno, “Build Automation Framework for Architecture Design of Automotive Elastic Gateway Controllers”, Embedded World Conference 2022

[4] “AUTOSAR”—Automotive Open System Architecture Rel. 4.1. 2018.

[5] Katharina Juhnke, Matthias Tichy, and Frank Houdek. 2018. Challenges with automotive test case specifications. (ICSE '18). ACM

[6] Harald Altinger, Franz Wotawa, and Markus Schurius. 2014. Testing methods used in the automotive industry: results from a survey.

[7] Y. Xu, P. Ou and Y. Li, "Design of Vehicle Gateway Automatic Test System Based on CANoe," (CAC), 2019

[8] Lucia Lo Bello. 2011. The case for ethernet in automotive communications. SIGBED Rev. 8, 4 (December 2011)

[9] “IEEE Standard for Local and Metropolitan Area Network– Bridges and Bridged Networks”. In: IEEE Std 802.1Q-2018, pp. 1–1993.

[10] 2020. IEEE P802.1DG/D1.4 Draft Standard for Local and metropolitan area networks — Time-Sensitive Networking Profile for Automotive In-Vehicle Ethernet Communications. (2021)

[11] Yan, Jinli & Quan, Wei & Yang, Xiangrui & Fu, Wenwen & Jiang, Yue & Yang, Hui & Sun, Zhigang. (2020). TSN-Builder: Enabling Rapid Customization of Resource-Efficient Switches for Time-Sensitive Networking.

[12] M. Gutiérrez, A. Ademaj, W. Steiner, R. Dobrin and S. Punnekkat, "Self-configuration of IEEE 802.1 TSN networks," (ETFA), 2017

[13] T. Frangieh and P. Athanas, "A design assembly framework for FPGA back-end acceleration," ReConFig, 2012

[14] D. Shah, E. Hung, C. Wolf, S. Bazanski, D. Gisselquist and M. Milanovic, "Yosys+nextpnr: An Open Source Framework from Verilog to Bitstream for Commercial FPGAs," (FCCM), 2019

[15] Vanitha, K. & Moorthy, C.A.. (2013). Implementation of an integrated FPGA based automatic test equipment and test generation for digital circuits. ICICES 2013

[16] Hulle, R., Fiser, P., Schmidt, J., & Borecký, J. (2016). SAT-ATPG for application-oriented FPGA testing. 2016 (BEC)

[17] A. Oeldemann, T. Wild and A. Herkersdorf, "FlueNT10G: A Programmable FPGA-based Network Tester for Multi-10-Gigabit Ethernet," (FPL), 2018

[18] Jason Lowe-Power, Abdul Mutaal Ahmad, Ayaz Akram, Mohammad Alian, Rico Amslinger, et al. 2020. The gem5 Simulator: Version 20.0+. CoRR abs/2007.03152 (2020).

[19] Guillem López-Paradís, Adrià Armejach, and Miquel Moretó. 2021. Gem5 + rtl: A Framework to Enable RTL Models Inside a Full-System Simulator. (ICPP 2021).

[20] A. G. Mariño, F. Fons, A. Gharba, L. Ming and J. M. Moreno Arostegui, "Elastic Queueing Engine for Time Sensitive Networking," (VTC2021-Spring)

[21] A. Gonzalez, F. Fons, L. Ming, and J.M. Moreno, “PDU Normalizer Engine for Heterogeneous In-Vehicle Networks in Automotive Gateways”. In: (ARC 2021)

[22] A. Gonzalez, F. Fons, L. Ming, and J.M. Moreno. Traffic Shaping Engine for Time Sensitive Networking Integration within In-Vehicle Networks, In: IEEE Vehicular Network Conference. (2021)

[23] A. Gonzalez, F. Fons, H. Zhang, and J.M. Moreno. 2021. Loopback Strategy for TSN-compliant Traffic Queueing and Shaping in Automotive Gateways, In: IEEE NFV-SDN. (2021)

[24] A. Gonzalez, F. Fons, H. Zhang, and J.M. Moreno. 2021. Loopback Strategy for In-Vehicle Network Processing in Automotive Gateway Network on Chip , (NoCArc’21)

[25] A. Gonzalez, A. Kane, F. Fons and J.M. Moreno. 2021. Enhancements for Hardware-based IEEE802.1CB embedded in Automotive Gateway System-on-Chip, In: (ANCS’21)

[26] Paul Emmerich, Sebastian Gallenmüller, Daniel Raumer, Florian Wohlfart, and Georg Carle. 2015. MoonGen: A Scriptable High-Speed Packet Generator (IMC '15).

[27] “J. Migge et al. “Insights on the Performance and Configuration of AVB and TSN in Automotive Ethernet Networks” (2018)