HLS-based Accelerated Simulation of Large Scale Cyber-Physical Systems on FPGAs

Maelic Louart, Jean-Christophe Le Lann, Frédéric Le Roy, Abdel Boudraa,
Jean-Jacques Szkolnik

HLS-based Accelerated Simulation of Large Scale Cyber-Physical Systems on FPGAs

Maelic Louart*†, Jean-Christophe Le Lann†, Frédéric Le Roy†, Abdel Boudraa*, Jean-Jacques Szkolnik*

†ENSTA Bretagne, Lab-STICC, Brest, France.
*Ecole Navale, IRENav, BCRM Brest CC 600, 29240 BREST Cedex 9, France.

Abstract—The design of cyber-physical systems remains challenging because of their highly heterogeneous nature that makes modeling, design and analysis hard. Despite extensive work in model-based approaches, few unified simulation tools are available today for such systems. This paper proposes a simulation strategy that benefits from the characteristics of recent FPGA platforms and advances in the high-level synthesis tools. Our proposal consists in using these tools to build a cyber-physical system simulator running at high-speed on a FPGA; in this view, high-level synthesis is used not only in the traditional prototyping phase of the embedded systems, but also to synthesize its physical environment, which is jointly simulated on the FPGA. Our paper proposes a case study illustrating this approach: the simulation of the automatic identification system required in maritime communications. The simulation executed on the latest FPGA generation is accelerated by a factor x654 compared to software alternatives demonstrating that FPGAs exhibit appealing characteristics for such simulations.

Index Terms—HLS, FPGA, CPS simulation

I. INTRODUCTION

As defined in [1], a cyber-physical system (CPS) is a tight integration of computation with physical processes. This definition applies at different scales: either at small scale, where such a CPS can be seen as a single embedded system with a physical environment, or at large scale where several embedded systems come into play in a distributed manner. In either cases, simulating such CPSs allows to design more robust embedded systems, by getting a realistic insight of their expected behaviors in a complex environment. However, CPS simulation remains difficult [2] because of their intrinsic heterogeneity [3]: the involved models of computation (MoCs) are varied in nature, mixing discrete and continuous phenomena and several temporal and synchronization characteristics. Moreover, such modeling activity requires multi-domain skills dealing with software and hardware components and multiple disciplines (mechanical, electrical and software engineering).

Ptolemy framework [4], Simulink , Simscape or Modelica are famous tools (and languages) developed to compose, in an hierarchical way, and simulate various models of computation. Pure cosimulation is another technique that allows linking standalone MoC simulators through functional mockup interface buses [5] [6] to simulate the behavior of the whole system. Unfortunately, these model-based design approaches make composition complex, model verification difficult [7] and to the best of our knowledge, do not provide immediate

solution to distribute simulation workloads efficiently and reduce simulation time. In addition, physical timing artifacts such as delays, skews, etc., as well as subtle logical causality errors, are produced because of the simulators connections and the execution models (interrupts, memory access and threads [8]), which deteriorates the overall simulation confidence [9].

We propose to explore an alternative CPS simulation approach. This approach relies on a single model expressed in C++ and resorts too high-level synthesis (HLS) to generate the full CPS simulator executable on a FPGA. Our experiment is inspired by the seminal work of Miller, Vahid, and Givargis on the use of FPGA for digital mockups [10]. They compiled various physiological models, based on partial differential equations, on an overlay-like structure on FPGA to simulate some human organs behavior (lungs, heart, etc). Such simulations allow to test the reaction of real devices (e.g ventilators, pacemakers) in various conditions. At that time (2011-2014), the capacities of FPGAs in terms of integration were much weaker than today's FPGAs; we demonstrate that we can now simulate both cyber and physical parts on the same FPGA. In this paper, we demonstrate that FPGAs offer a viable platform for full multi-scale CPS prototyping. This method can be considered "agile" because it bypasses a certain number of steps of a classic model-based "waterfall" approach, which is in line with [11]. A similar method is proposed in literature [12], but it does not rely on the combination of FPGA and HLS, and only simulates small scale CPSs.

In the next section, we present our CPS design method. Then we experiment our design method on a case study concerning a maritime communication protocol. We conclude by a discussion and delineate some future work.

II. CPS SIMULATION APPROACH BASED ON FPGA PLUS HLS

Our method comes from three practical observations: the first is the quasi-systematic availability of highly domain-specific C/C++ codes able to simulate behavior parts of almost every CPS (physical, mechanical, chemical phenomena, etc.). These codes are rarely available in model-based specialized languages. The second observation is the growing robustness of HLS tools [13] able to synthesize hardware code from C++ code. The third is the interesting characteristics of reconfigurable architectures such as recent FPGAs [14]: they offer a high computational power, large amount of resources [15],
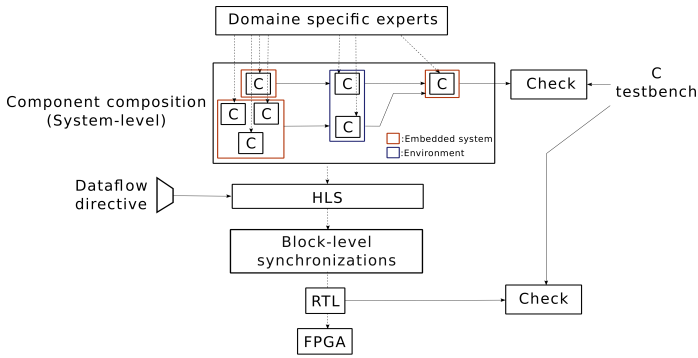
Fig. 1. Overview of the approach for CPS simulator design

Fig. 2. Our proposal: FPGAs as cyber-physical system Simulation platforms.

parallelism concepts easy to implement, and a native synchronization of computations. Considering these observations, we created a new CPS design method (displayed in Fig. 1) which, from a model expressed in C++, uses HLS to generate the full CPS simulator executable on FPGAs. Note that the functioning of the simulator can be tested thanks to the HLS tool, both at the C level and RTL level.

The method divides the CPS into components: one of them simulates the environment (physical part) and the others emulate the embedded systems (cyber part). The term "emulation" is used when the model under study reproduces a sequence of operations close or even identical to that of the system [16]. The description of the components behavior is done in C++ and the description of the interactions between components is done using HLS directives. Components are composed by applying the dataflow directive, which implements FIFO or Ping-Pong buffers between components. Computation can be expressed using float, fixed-point and integer with arbitrary precision data types. This method can be only applied to architecture limited to acyclic directed graph. At the macroscopic point of view (component point of view), the Kahn process networks (KPN) MoC is applied [1], while, at a microscopic point of view (FPGA logic point of view), the Synchronous Reactive MoC is applied.

Fig. 2 illustrates the generic architecture of the resulting synthesized simulators. Note the presence of embedded *Piloting*, allowing a classical interaction, from the host PC, with such a simulator: start, stop, step-by-step advancement commands. The embedded *Piloting* allows also to adjust the embedded systems and the simulated environment parameters. The user can, in addition, observe the dynamics of selected signals from the HLS, collected and read back to the host PC (*Monitoring*). These commands and observations come from interface registers described in a file (in s-expression format) and automatically generated with our Reggae tool [17]. The tool allows to generate a very simple combinatorial glue allowing to interface to the axi stream interface protocol used by the hls tool.
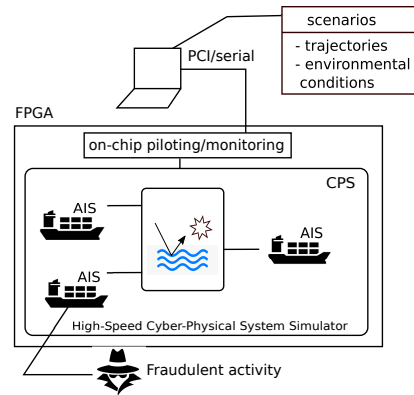
III. MOTIVATING CASE STUDY

A. AIS Maritime communication system

Our CPS case study consists in a set of vessels that move and exchange radio messages relying on a dedicated maritime protocol named automatic identifying system (AIS). AIS operates on two frequencies at 161.975 MHz and 162.025 MHz on the very high frequency (VHF) band. The sent messages comply with the national marine electronics association (NMEA) 0183 standard. The data rates are 9600 baud. Since 2002, the safety of life at sea agreement of the international maritime organization requires the installation of a class A AIS transponder for international ships over 300 gross tonnage and for passenger ships regardless of their size. This system improves the safety of maritime traffic by allowing the automatic exchange of navigational information (position, identity, port of departure, etc.) between ships. However, this system is vulnerable: the information sent by transponders can be easily falsified or spoofed (for piracy, illegal transport, etc.). This is why spoofing and falsification detection algorithms must be developed and integrated into new generation AIS transponders. The algorithms will be applied to both AIS messages and AIS signals, to detect, for instance, inconsistency in data (AIS messages) or to characterize the transponders radiometric signature (AIS signals).

B. Verification challenges

To develop a reliable system, simulations are required to test a wide range of scenarios reproducing use conditions of the product. However, getting comprehensive test streams from real field measures remains tedious. In our work, to solve this issue, a simulation framework was created to generate a wide variety of *synthetic* scenarios containing AIS messages and signals. The generated messages can be falsified and spoofed. The simulation framework emulates a variable number of AIS transponders, which communicate with each other in an simulated maritime environment. The simulation runs entirely on hardware, and is synthesized using HLS. An example of the simulation architecture is presented in Fig. 2, where three transponders are communicating.
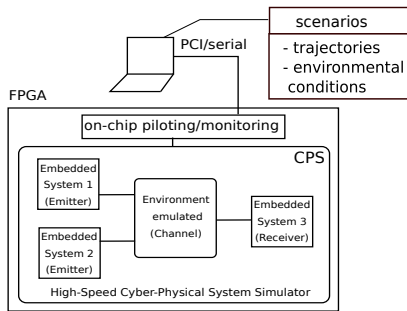
Fig. 3. Model of the CPS simulated on FPGA.

C. Model of the CPS simulated

Fig. 3 illustrates the general architecture of our CPS simulator in the same configuration as in Fig. 2. Three types of component are displayed and were modeled. An *Emitter* that acts as an AIS transponder which, from an NMEA frame (AIS message), sends a modulated AIS signal. A *Receiver* that acts as a AIS transponder, which only receives AIS signals and demodulates them to extract NMEA frame. Finally, a *Channel* model simulates the environment to apply some physical laws to the AIS signals. All components are discussed later. The *Emitter* and the *Receiver* were the CPS' embedded system and *Channel* was the CPS' environment. We also represent in Fig. 3 a specific block of *piloting* and *monitoring* of the simulation explained above in part II. Every component is modeled by a chain of blocks executed sequentially to the received signal. Note that the architecture is modeled by an directed acyclic graph: the *Emitters* send AIS signals to the *Channel*, and the *Channel* sends the signals to the *Receiver*.

The simulated signals were not modulated at the carriers frequencies (161.975MHz and 162.025MHz) because the frequencies are to high: FPGA should have huge memory and computation capacities. Thus, we simulated the signal only on baseband at a sample rate equal to 192kHz.

The *Emitter* component contains eight blocks. First block reads the NMEA frame. The following four blocks transform this frame according to the high-level data link control (HDLC) protocol. HDLC is a level 2 protocol (link layer) of the Open Systems Interconnection model. Each byte is inverted, a checksum is calculated and added at the end of the frame, then a stuffing of bits at 0 is applied to avoid the presence of 6 bits at '1'. Finally a start and end flag is inserted at the beginning and end of the frame. After, a conditioning sequence is added to the frame. Finally, a Non Return to Zero Inverted (NRZI) Coding and Gaussian minimum-shift keying (GMSK) modulation are applied. The signal is not modulated at the carriers frequencies because we only consider the baseband signal.

The *Channel* component contains four blocks and allows to apply varied environmental effects on the signals sent between the *Emitter* and *Receiver*. A first block adds a carrier frequency offset to the signal to characterize *Emitter* carrier frequency error caused by hardware imperfections. Another

block reproduces path loss attenuating the energy of the signal considering the Friis model and the distance between the *Emitter* and the *Receiver*. A third block adds a white Gaussian noise to the signal to characterize the propagation noise. The amplitude of the noise is fixed according to the signal-to-noise ratio (SNR) that we want to impose. Finally, a last block sums the signals from every *Emitter*.

Friis model stipulates that the ratio of received signal power and emitted signal power is given by:

$$\frac{P_r}{P_t} = G_r G_t \left(\frac{\lambda}{4\pi R} \right)^2 \qquad (1)$$

where P_r and P_t are respectively the received and transmitted powers. G_t and G_r are respectively the antenna gains in transmission and reception. λ the wavelength of the working frequency and R the distance separating the emitter from the receiver. We fix antennas gain to $G_t = G_r = 2dBi$, as suggested by [18], and the power of transmitter is $P_r = 12.5W$ as defined by AIS standard. Reference [18] shows that this model is reliable and reproduces the evolution of the energy of the signal up to thirty kilometers.

The *Receiver* component starts with the computation of the signal energy to detect the effective arrival of an AIS message characterized by a peak of energy. After, it demodulates the AIS signals to extract their NMEA frame. The demodulation is done symmetrically to *Emitter*. Thus, this component allows to extract AIS information from the signals to applied the algorithms. This receiver prototype is efficient because it extracts AIS message from true baseband signals recorded in the bay of Brest.

D. Hardware synthesis

We experimented our design method presented in Fig. 1. The whole CPS model represented about 5000 lines of C++ code using fixed point and integer with arbitrary precision. Note that some parts of the system are dedicated to pure simulation, while some elements (like our *Receiver*) can be seen as a viable prototype of a future embedded device. The results of the three components synthesis are presented in Table I. The synthesis target was one of the biggest FPGA proposed in the industry, the Xilinx Ultrascale+ HBM FPGA on a VCU128 board (about 10^6 System logic cells, 3K DSP slices, etc.). The HLS tool was Vitis HLS 2020.2. The occupancy percentage for a simulator with the three components is low (less than 7% for BRAM, DSP, FF, and LUT elements). However, the FPGA platform is not over-sized for such application. Indeed, the test-benches need to simulate synthetic environments with several *Emitters* moving physically and exchanging AIS messages at the same time. In addition, the simulated environment was too simple to replicate every spoofing and falsification scenario: special geographical maps (presence of islands, etc.) can be used by hackers to spoof AIS. These special geographical maps need high computation capabilities to be simulated. Note that the amounts of resources used by the components *Piloting* and *Monitoring* are not displayed in the table I because they are very small and can therefore be ignored.

	BRAM	DSP	FF	LUT
Simulator	293(7%)	66(0%)	30508(1%)	41665(3%)
Emitter	97	1	8718	16357
Channel	97	13	1478	4312
Receiver	99	52	19904	20754

Nb. Emitters	Soft.	Hard.	Gain	FPGA usage
1	576s	3.14s	183	7%
5	856s	3.14s	272	24%
10	1210s	3.14s	386	46%
22 (pred.)	2050s	3.14s	654	98%

IV. HIGH-LEVEL SYNTHETIC SCENARIOS

A. Objectives

To illustrate the ability of our simulator to generate ship trajectories that can contain falsified or spoofed messages (but also to prove the functioning of our channel model based on the Friis model), five scenarios were created and observed, and are displayed on Fig. 4.
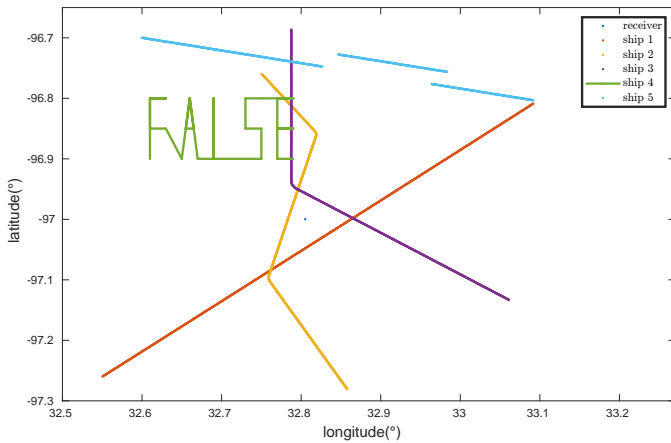

Fig. 4. Simulated trajectories on the FPGA.

Among these scenarios, three reproduce a ship trajectory without falsification or spoofing of AIS messages, and two with falsification and spoofing. The ship 4 on Fig. 4 reproduces the type of spoofed reported in [19]. The ship 5 on Fig. 4 has a falsified trajectory reproducing the type of falsification reported in [20]. In this reference, a fishing vessel falsifies its position to fish in a restricted area. For all these trajectories, *Channel* added white Gaussian noise to the signal to reach a SNR equal to 10dB. To simulate these five scenarios at the same time, five *Emitter* components were implemented on a single FPGA, in addition to one *Channel* and one *Receiver* component.

B. Simulation results and analysis

We compare, in Table II, software's simulation time to hardware's simulation time for the five scenarios presented above. The software was a standard Intel Core I5 processor at 1.7GHz with 16Gb RAM, and the hardware, the FPGA presented above. During every simulation, 1000 messages are received by the *Receiver* successively from every *Emitter*. Other simulations, which simulated different trajectories, were done; one required only one *Emitter* and another ten. The

simulation times are displayed in Table II. To run these simulations, *Emitter* components were added or removed from the FPGA platform, this is why the FPGA usage rate is changing. Considering the results from these three simulations, we predict that the maximum number of *Emitters* that can be emulated at the same time in the FPGA is 22 because in this case the occupancy percentage of BRAM reaches 98%.

Performance gain is already x183 compared to software simulation for only one *Emitter*. The speed-up gain was obtained without applying optimization directives to the code of the blocks such as loop unrolling or pipeline, which still leaves a speed-up margin for the FPGA simulation. Only dataflow directive is applied to the code as mentioned in methodology part II. These directives pipeline the components execution. The FPGA acceleration reduces the simulation times from tens of minutes to only 3.14s. Moreover, although the number of components *Emitter* was increased, the hardware simulation time remained the same: the *Emitter* components were executed in parallel. This is why for 22 *Emitter* the performance gain is predicted to be 654. This fact shows the interest of the scalable aspect of FPGAs.

V. DISCUSSION

Our CPS design method is restricted to CPS whose architecture is an directed acyclic graph. Outside this type of architecture, Vitis HLS cannot synthesize systems. For instance, in our application case, the AIS transponders communicate with each other according to the TDMA (Time-division multiple access) method whose architecture is a complete undirected graph with as many nodes as boats. This TDMA method cannot be *directly* synthesized by Vitis HLS to be simulated on FPGA. Thus, other high-level synthesis tools, allowing actor-based modeling without limit in the complexity of the architecture, using, among others pure dataflow semantics [21], are to be experimented.

VI. CONCLUSION AND FUTURE WORK

In this paper, we proposed the use of FPGA as a simulation platform for complex cyber-physical systems. Our case study focuses on a set of AIS transponders operating in a maritime environment simulated. The progress of HLS tools and the increasing capabilities of FPGAs make such an approach possible. Compared to a pure software simulation, we have shown remarkable performance gains (up to a x654 acceleration factor). The user can interact with the simulator changing the environmental parameters, monitoring the simulation (start, stop, step-by-step) and observing the signals exchanged by the components. We point out some limitations of classical

HLS tools, that call for alternatives HLS approaches to be able to simulate a wide range of CPS architectures. Our paper demonstrates that FPGA can be seen as a platform of choice for the integration of multi-domain systems.

REFERENCES

[1] E. Lee and S. Seshia, *Introduction to embedded systems: A cyber-physical systems approach*. Mit Press, 2017.

[2] K.-D. Kim and P. R. Kumar, "Cyber–physical systems: A perspective at the centennial," *Proceedings of the IEEE*, vol. 100, no. Special Centennial Issue, pp. 1287–1308, 2012.

[3] H. S. Fadhlillah, K. Feichtinger, L. Sonnleithner, R. Rabiser, and A. Zoitl, "Towards heterogeneous multi-dimensional variability modeling in cyber-physical production systems," in *Proceedings of the 25th ACM International Systems and Software Product Line Conference-Volume B*, pp. 123–129, 2021.

[4] J. Buck, S. Ha, E. Lee, and D. Messerschmitt, "Ptolemy: A framework for simulating and prototyping heterogeneous systems," 1994.

[5] C. Gomes, T. Blochwitz, C. Bertsch, K. Wernersson, K. Schuch, R. Pierre, O. Kotte, I. Zacharias, M. Blesken, T. Sommer, *et al.*, "The fmi 3.0 standard interface for clocked and scheduled simulations," in *Modelica Conferences*, pp. 27–36, 2021.

[6] S. E. Saidi, A. Charif, T. Sassolas, P.-G. Le Guay, H. V. Souza, and N. Ventroux, "Fast virtual prototyping of cyber-physical systems using systemc and fmi: Adas use case," in *Proceedings of the 30th International Workshop on Rapid System Prototyping (RSP'19)*, pp. 43–49, 2019.

[7] J. Porter, P. Volgyesi, N. Kottenstette, H. Nine, G. Karsai, and J. Sztipanovits, "An experimental model-based rapid prototyping environment for high-confidence embedded software," in *2009 IEEE/IFIP Int. Symp. on Rapid System Prototyping*, pp. 3–10, IEEE.

[8] E. A. Lee, "The problem with threads," *Computer*, vol. 39, no. 5, pp. 33–42, 2006.

[9] H. Carlsson, B. Svensson, F. Danielsson, and B. Lennartson, "Methods for reliable simulation-based plc code verification," *IEEE Transactions on Industrial Informatics*, vol. 8, no. 2, pp. 267–278, 2012.

[10] B. Miller, F. Vahid, and T. Givargis, "Application-specific codesign platform generation for digital mockups in cyber-physical systems," in *Electronic System Level Synthesis Conf. (ESLsyn)*, pp. 1–6, IEEE, 2011.

[11] D. Roy, M. Balszun, T. Heurung, S. Chakraborty, and A. Naik, "Waterfall is too slow, let's go agile: Multi-domain coupling for synthesizing automotive cyber-physical systems," in *Proc. of the Int. Conf. on Computer-Aided Design*, ICCAD '18, (New York, USA).

[12] S. Sood, A. Malik, and P. Roop, "Robust design and validation of cyber-physical systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 18, no. 6, pp. 1–21, 2019.

[13] S. Lahti, P. Sjövall, J. Vanne, and T. D. Hämäläinen, "Are we there yet? a study on the state of high-level synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 5, pp. 898–911, 2018.

[14] T. Grimm, B. Janßen, O. Navarro, and M. Hübner, "The value of fpgas as reconfigurable hardware enabling cyber-physical systems," in *2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*, pp. 1–8, IEEE, 2015.

[15] "Xilinx Announces General Availability of VU19P, World's Largest FPGA."

[16] I. McGregor, "The relationship between simulation and emulation," in *Proceedings of the Winter Simulation Conference*, vol. 2, pp. 1683–1688, IEEE, 2002.

[17] J.-C. L. Lann, "Reggae : FPGA register map VHDL generation," Jan. 2022. original-date: 2022-01-27T14:54:25Z.

[18] F. Mazzarella, M. Vespe, A. Alessandrini, D. Tarchi, G. Aulicino, and A. Vollero, "A novel anomaly detection approach to identify intentional ais on-off switching," *Expert Systems with Applications*, vol. 78, pp. 110–123, 2017.

[19] M. Balduzzi, A. Pasta, and K. Wilhoit, "A security evaluation of ais automated identification system," in *Proc. of the 30th annual computer security applications Conf.*, pp. 436–445, 2014.

[20] F. Katsilieris, P. Braca, and S. Coraluppi, "Detection of malicious ais position spoofing by exploiting radar information," in *Proc. of the 16th Int. Conf. on information fusion*, pp. 1196–1203, IEEE, 2013.

[21] J. Janneck, I. Miller, D. Parlour, G. Roquier, M. Wipliez, and M. Raulet, "Synthesizing hardware from dataflow programs," *Journal of Signal Processing Systems*, vol. 63, no. 2, pp. 241–249, 2011.