FPGAs as versatile configurable I/O devices in Hardware-in-the-Loop Simulation*)

Peter M. Visser, Marcel A. Groothuis and Jan F. Broenink
Twente Embedded Systems Initiative,
Cornelis J. Drebbel Institute for Mechatronics and Control Engineering,
Dept. of Electrical Engineering, University of Twente,
P.O.Box 217, NL-7500 AE Enschede, The Netherlands
Phone: +31 53 489 2788 Fax: +31 53 489 2223
E-mail: p.m.visser@utwente.nl

***Abstract* – This paper describes the use of FPGAs as versatile reconfigurable devices in Hardware-in-the-Loop Simulation. Advantages of software development (flexibility, short design cycles) now also apply to the I/O hardware development.**

The development method considers the implementation process as a stepwise refinement from physical system models and control laws to efficient control computer code, and that all phases are verified by simulation or HIL Simulation.

Experiments with a basic mechatronic set-up show that the HIL Simulated system behaves as the real system, and can be considered a useful addition in Systems Design. This can shorten the development time considerable, especially when the mechatronic product to be developed is complex, as is the case in our Boderc project.

***Keywords* – Embedded control systems, real-time, model-based approach, Hardware-in-the-Loop, Rapid Prototyping, FPGA, I/O, Simulation**

I. INTRODUCTION

Hardware-in-the-Loop Simulation is used as an aid in designing and testing complex, multidisciplinary engineering systems (in our case, mechatronic systems). Besides a more thorough means of testing, it can also sup*port concurrent engineering* in the design, allowing for an efficient use of human resources, and a shorter time to market.

Futhermore, it is posed that the flexibility of using FPGAs for the I/O allows for a *software* approach for developing I/O *hardware*. This means that advantages of software development (flexibility, short design / implementation cycles, and versatile functionality) can be applied to the development of the I/O hardware.

The work described here is carried out in the context of the Boderc project (Beyond the Ordinary, Design of Embedded Real-time Control). In this project, academia and industry work together to derive sophisticated methods for embedded controller design of complex mechatronic systems.

First, a brief introduction is given of Hardware-In-the-Loop Simulation (HIL Simulation or HILS), a kind of real-time simulation [1]. Then, HIL Simulation is applied in the context of a structured approach to embedded control systems software implementation as presented in [2]. A benefit in this case is that it allows for concurrent engineering in an early stage. Section four elaborates on the use of FPGAs as I/O devices.

A demonstration set-up shows a sample configuration and its results.

II. HARDWARE-IN-THE-LOOP SIMULATION

Testing and simulation of control algorithms is an important phase in the development of embedded control systems (ECSs). Different types of simulation are possible during the design process of a controller, ranging from simulation without time limitations, to partial real-time simulation in which only some parts of the complete control loop are simulated (see Figure 1). Real-time simulation means here that the simulation is performed such that the input and output signals show the same time-dependent values as the real component [1].

In this research, Hardware-In-the-Loop simulation (HILS) involves connecting the actual ECS to a computing unit with a real-time simulation model of the plant, the middle situation of Figure 1. The architecture of the actual experimental set-up is shown in Figure 2.

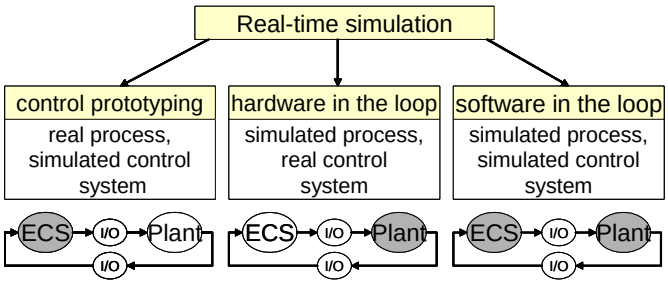

Figure 1: Real-time simulation methods (from [1]).

*) This work has been carried out as part of the Boderc project under the responsibility of the Embedded Systems Institute. This project is partially supported by the Netherlands Ministry of Economic Affairs under the Senter TS program.

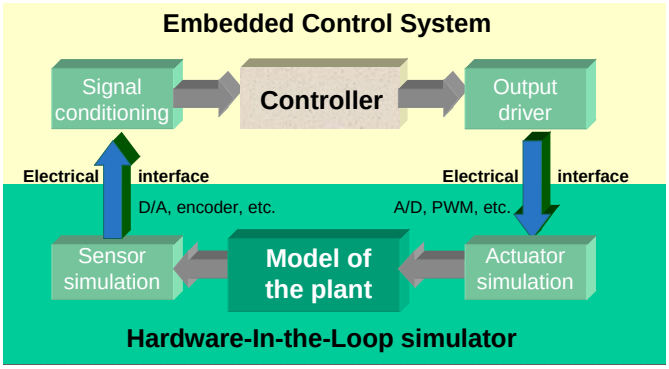

Figure 2: Used HILS set-up.

Compared to 'ordinary' simulation, extra computer hardware is needed: the simulation has to run in real time, the I/O interfaces have to be available. Furthermore, conversion from the I/O-signals back to computer numbers must be constructed (the blocks "Sensor simulation" and "Actuator simulation" in Figure 2). These interfaces are not commonly available, and thus might be costly to produce.

The main advantages of HIL Simulation are:

- Plant models used during off-line design and simulation for the controller development can now be used for the ECS testing. This implies that, at software testing, the stubs representing the plant now can be proper models instead of simple signal generators. The system to be tested is now closed loop, which better resembles the final situation. This results in better quality of the ECS tests, allowing for a less complicated integration phase.
- Software design and testing can be moved to an earlier design phase, i.e. before a first physical/mechanical prototype is available, allowing concurrent engineering between the different design disciplines. This results in a shorter time to market.
- The ECS software updates can easily be checked for consistency with the design. Test benches written in the control design stage can be reused easily.

An additional benefit is that plant models used for off-line design and simulation during the control development can be reused for the when testing the ECS in the final stage.

III. Design Trajectory

The design trajectory proposed in [2] is used, see Figure 3.

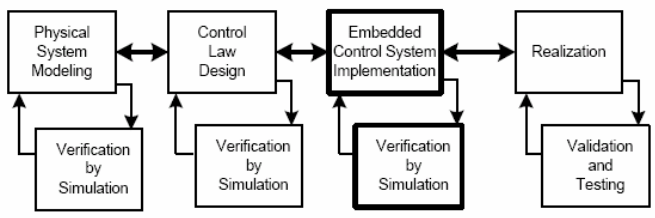

Figure 3: Design trajectory for embedded control systems

The trajectory is summarized into the following steps:

- Model the plant and controller; verify them by simulation.
- ECS implementation; verify by simulation.
- Realization: real prototype or plant with real ECS: validate, measure and test.

In combination with hardware-in-the-loop simulation this trajectory is *not* a waterfall approach (followed from the left to the right) but iterated in a micro-cycle fashion [3]. A simple physical model is made, which is based on basic physical principles. A corresponding simple control law is designed. A simple ECS implementation is made. This will be run on the hardware-in-the-loop simulation and tested. Next the physical model is extended, the control law is extended and the ECS implementation is extended and so on. This allows concurrent engineering in an early stage and tries to avoid the pitfall of system integration problems.

By using FPGAs as I/O devices, there are two distinct modes for the HIL-Simulation to run: a direct I/O link mode and a hardware compatible I/O mode. In the former case there is no real I/O, the controller and plant are connected through signals in the exact same manner as in the model. In the latter case effects due to non-idealness of computer hardware [2] are taken into account by fully implementing the I/O as depicted in Figure 2. Also the HILS and the plant can be exchanged without making changes to the ECS.

So starting with a simple design the direct I/O link mode is used and iterating to a more detailed, and final design the hardware compatible I/O mode is used.

IV. Use of FPGA's as I/O Devices

Figure 2 shows four blocks that represent the I/O devices between the controller and the model of the plant. (signal conditioning, output driver, sensor simulation and actuator simulation) All these hardware I/O devices can be emulated by software configuration in the FPGAs. A tremendous advantage of FPGAs over standard I/O devices is their high configurability. Another advantage of FPGAs is that it replaces the design of dedicated HILs I/O hardware, specific for one kind of plant, with general purpose hardware that can be configured by software. A disadvantage is that the implementation of FPGA configuration takes time. However, this is a "one-time" issue; once configurations are implemented they can be easily reused.

Many different types of I/O can be used with the same device. The software of the ECS can remain the same and only the configuration of the FPGAs need to be altered. In the context of the proposed design trajectory this is beneficial, different design choices can be easily simulated and analyzed.

The FPGAs in this set-up are chosen to be fast enough for I/O conversion in the field of mechatronic control applications. The FPGAs are on a PCB board which is available in both a PCI and a PC104-PCI bus. The FPGA has 200.000 system gates and 56k blockram and can run up to 200MHz. The PCB board, anything I/O board, has 72 general purpose digital I/O pins [4].

V. Demonstration Set-up

A set-up, called *Linix*, is used as mechatronic platform. Linix consists of one motor and one encoder which are both on the same axis. The motor drives a wheel which is connected by a rubber band to another wheel, the load. Both the model of Linix and a controller are modeled in 20-sim [5]. Via automatic code generation the HILS or the real plant can be created and run. Figure 4 depicts this process.

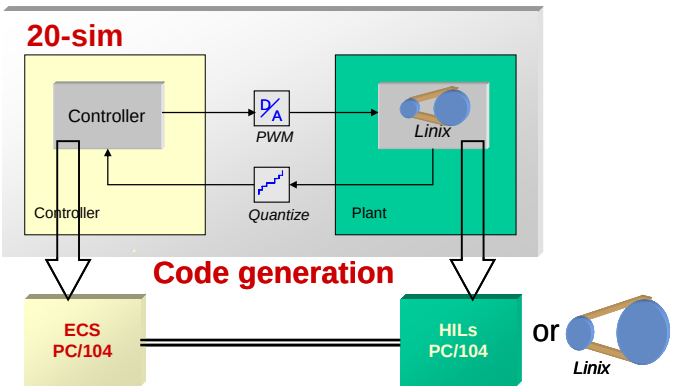

Figure 4: Code generation for HILS

As proof of concept the simulation results of the model are compared with the real-plant and the HILS. An x86-compatible PC is used as HILS. A PC/104 board is used for the ECS.

A. The plant model

The model of the plant is depicted in Figure 5 below:

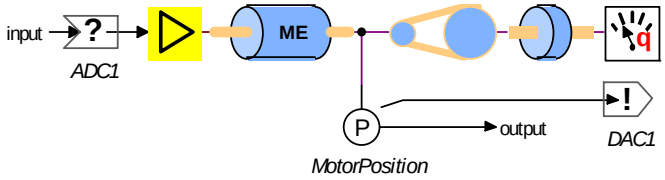

Figure 5: The Linix plant model

The plant was simulated as a separate entity before designing the controller conform the design trajectory. The ! and ? special blocks reflect the I/O, which is necessary for code generation.

B. The Controller

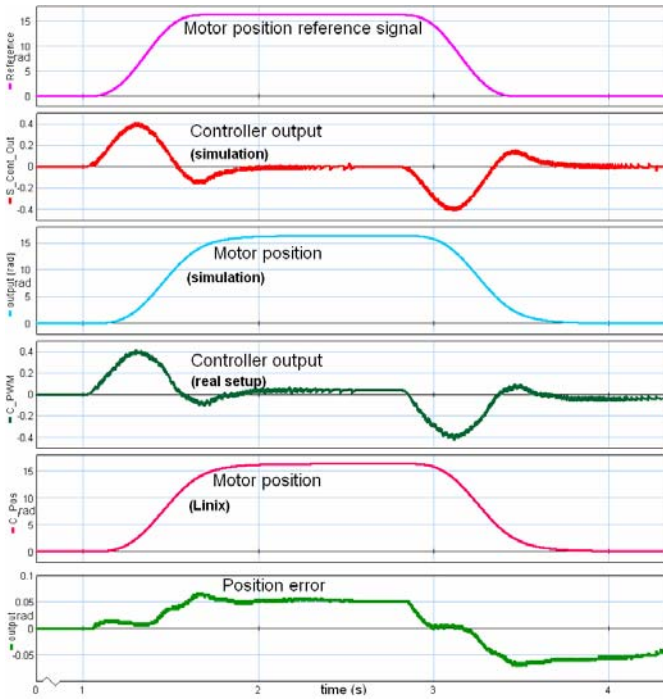

Figure 7: Comparing the simulated Linix with the real Linix

The controller for the Linix set-up is a discrete PID controller. Figure 6 shows the internals of this controller. The controller inputs are the reference position and a feedback of the real position. The output is the motor steering signal which will be converted into a corresponding PWM duty cycle.

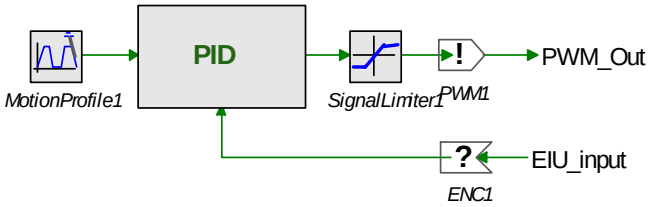

Figure 6: The discrete PID controller of Linix

C. Simulation and results

All state-variables and other signals both of the plant and the controller can be inspected in the simulation environment. The most interesting signals are the angular positions of the motor and the load, hence the signals from the position sensors in the model. On the HIL simulator only the *input* and *output* of the ECS were monitored, in order to compare the results with the real plant (on the real plant, only the inputs and outputs can be captured). Additional signals can be inspected but have to be added by special blocks, in the model. The same holds for the controller. When the controller was run on the Embedded Control System only the *input* and *output* signals were inspected. Additional signals can also be monitored if special code blocks are added. The monitoring of these signals are soft real-time and will not influence the real-time behavior of the ECS. Currently slack-time cannot be measured yet. (Either on the HILS or ECS).

As proof of concept, first a simulation and a measurement of the controller with the real plant is carried out and as second step the real plant is replaced by a simulation of the Linix set-up on the HIL Simulator PC. The controller uses the same hardware and software in both cases. Only the external I/O connection cable will be replaced from the real set-up to the simulation PC. The controller software is the exactly the same.

Simulation versus real set-up

The first experiment has been performed to compare the 20-SIM simulation results with the results from the real plant using code generation for the controller. Figure 7 shows the course of the PWM and encoder signals when applying a motion profile as reference signal. This test is performed with a sample rate of 1 kHz for the controller. Euler was used as the integration method. The position error depicts the error between the simulated position and the position of the real set-up.

The simulation signals are comparable with the measured signals on the real set-up. A small steady state error exists on the position of the real plant. This is due to an unmodeled dead-zone in the PWM steering of the real motor. When the controller output (PWM duty cycle) is smaller than ±6% the Linix motor does not turn.

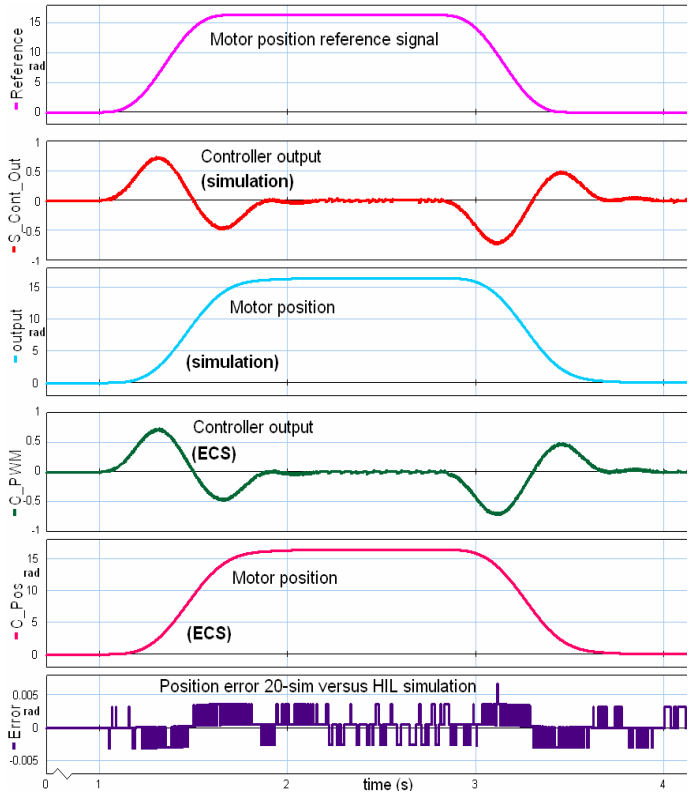

Figure 8: Comparing Simulation and HIL Simulation

Simulation versus HIL simulation

The second test has been performed to compare the 20-SIM simulation results with the HIL simulation results. Code has been generated for both the controller and the plant and the input and output signals were logged and imported into 20-SIM.

Figure 8 shows a comparison between the 20-SIM simulation results and the ECS-HIL simulator results. The most important difference between simulation and HIL simulation is the physical I/O interconnection. A comparison between the 20-SIM motor position and the HIL simulator motor position (position error line in Figure 8) shows that the PWM I/O and encoder I/O ports are accurate enough for the HIL Simulation. The error is almost zero and shows only some quantization noise. The HIL simulator is able to convert in real-time the controller output signals into the corresponding plant signals. For the test set-up, the HIL simulator can run in real-time at a sample frequency up to 50 kHz.

The tests have been performed with the controller running at 1 kHz and the HIL Simulator running at 10 kHz. The HIL Simulator runs at a 10 times higher sample frequency for better emulation of the continuous-time system (the real plant has its current state always available, it determines it infinitely fast). The controller and the HIL simulator are not synchronized, i.e. there is no common master clock. The controller computation scheme used here is the measure→steer→calculate method, for both the controller and the HIL simulator to get a precisely periodic steering. This means that at every sample, first a value is measured and the resulting steering value of the previous sample is send to the actuator before the new value is calculated.

The results from both tests show that the principle of Hardware-In-the-Loop Simulation is working. More simulations and measurements need to be performed, e.g. to check whether there is a possible delay at the HIL simulation outputs compared with the real plant status. One sample delay can introduce a significant error in the position calculation. For example, if the HIL simulator is running at 10 kHz and a the wheel is rotating with an angular velocity of 10 m/s, one sample delay (0,1 ms) will cause an error in the position 1 mm. Experimenting with encoder resolutions, PWM frequencies and controller and HILS sample rates is also desired.

VI. Conclusions and Recommendations

Conclusions:

- Using FPGAs as configurable versatile I/O devices works and is a cheap means to performing a range of tests.
- The HIL simulation approach as in the proposed design trajectory allows for *concurrent engineering* in an early phase.
- A disadvantage is that the implementation of FPGA configuration takes time. However, this is a “one-time” operation; once configurations are implemented they can be easily reused.

Recommendations:

- More testing of different I/O configurations to check differences in details of behavior and to precisely indicate in which situation a specific HIL Simulation can benefit.
- Refining the design trajectory of [2] in order to optimally benefit from the versatility provided by the FPGAs and the design support given by HIL Simulation.
- Performance measurements are necessary to study to real-time behavior and hence the applicability. An important issue is to determine when the code running on the FPGA becomes the limiting factor of the total HIL Simulator. Also a comparison with traditional HIL Simulators is recommended.

References

[1] R. Isermann, J. Schaffnit, and S. Sinsel, “Hardware-in-the-loop simulation for the design and testing of engine control systems,” *Control Engineering Practice*, vol. 7, pp. 643-653, 1998.

[2] J. F. Broenink and G. H. Hilderink, “A structured approach to embedded control systems implementation”, In M. W. Spong, D. Repperger, and J. M. I. Zannatha, Eds., *2001 IEEE International Conference on Control Applications.* México City, México, 2001.

[3] B. P. Douglass, *Doing Hard Time, Developing Real-Time Systems with UML, Objects, Frameworks, and patterns*: Addison Wesley Longman Inc., 1999.

[4] Mesa Electronics, “Mesa Electronics”, http://www.mesanet.com, 2004.

[5] CLP, “20-SIM” http://www.20sim.com: Controllab Products, 2002.