Analysis of Cyber-Physical System using FPGA-Based Plant-on-Chip Platform

T.Murugajothi
Assistant Professor, PSNA College of Engineering & Technology, Tamilnadu, India
jothyece@gmail.com

***Abstract*— Digital control systems are traditionally designed independent of their implementation platform, assuming constant sensor sampling rates and processor response times. Applications are deployed to processors that are shared amongst control and noncontrol tasks, to maximize resource utilization. This potentially overlooks that computing mechanisms meant for improving average CPU usage, such as cache, interrupts, and task management through schedulers, contribute to nondeterministic interference between tasks. This response time jitter can result in reduced system stability, motivating further study by both the controls and computing communities to maximize CPU utilization, while maintaining physical system stability needs. In this letter, we describe an field-programmable gate array (FPGA)- based embedded software platform coupled with a hardware plant emulator (as opposed to purely software-based simulations or hardware-in-the-loop setups) that forms a basis for safe and accu-rate analysis of cyber- physical systems. We model and analyze an inverted pendulum to demonstrate that our setup can provide a significantly more accurate representation of a real system.**

***Index Terms*—Cyber-physical systems, embedded systems, field-programmable gate array (FPGA), hardware emulation, plant-on-chip.**

I. INTRODUCTION

Embedded systems and digital control theory have inde-pendently developed into mature fields, despite the clear connection between controllers and embedded platforms. Initially, each digital control loop was implemented on a dedicated processor, thus maintaining a separation of concerns. The de-mand for tighter system integration and the use of economical commercial-off-the-shelf products has blurred this separation [1]. In modern systems, the tasks running on the processor un-knowingly compete for processor resources. These resources, meant to improve average resource usage for nonreal-time sys-tems, are becoming sources of nondeterministic computation time or computation jitter. Example causes include interrupts [1], branch misprediction [8], cache misses [11], and task management through operating systems [12]. These features limit the degree to which time invariance can be guaranteed, and cause systems to break control engineers' key assumption of constant sample rates and processor response time [2]. Ulti-mately, control loop robustness is greatly affected by this tran-sition from a dedicated processor system to an environment of tasks competing for resources [5]. Thus, a more holistic view is now needed to develop and deploy controllers that take into account cyber-architecture artifacts on a system's physical stability.

As a motivating example, Fig. 1 shows the timing response of an inverted pendulum model as we vary the computational delay (the time between receiving a sensor sample and sending the response), while holding sensor sample rate constant. In Fig. 1(a), a controller computing delay that is 15% of the state sampling rate has negligible impact on the system's stability. As the delay increases to 65% of the sample period [see Fig. 1(b)], some ringing in the control signal becomes apparent. Progressing to a delay of 85% of the sample period [see Fig 1(c)] causes the plant to become less stable with oscillations that are now more pronounced. It is interesting to note that the state of the plant (i.e., cart position and pendulum angle) still appears stable. A further increase in the computational delay [see Fig. 1(d)] leads to loss of controller stability resulting in an eventual fall for the pendulum.

Previous work has identified jitter in cyber-physical systems (CPS) as a significant research challenge. The authors in [11] worked on characterizing Linux for real-time applications and found that the sources of jitter were implicit to the processor and were not completely correctable through software. A de-tailed analysis of branch- prediction schemes [8] concludes that static branching schemes work better for real-time systems than dynamic branch prediction. In [4], the authors compare several scheduling methods and concluded that deadline advancement was the most consistent, with minimal degradation in perfor-mance of controllers as the number of tasks increased and had relatively consistent low jitter. Controls experts are developing toolflows, like TrueTime-JitterBug, to evaluate the impact of a controller's response-time jitter on closed-loop stability [5]. In [6], [9] the authors have developed a set of stability criteria for closed-loop systems in which the sample rate contains jitter. In [7], a quantitative metric similar to the concept of phase margin is proposed, called jitter margin, which is the upper-bound of delay that a control loop can tolerate before going unstable. In an approach closely related to ours, the delay and period of control loops are used in a cost function, which is then treated as a minimization problem [3], and later a convex optimization problem [13]. A limitation of many of the previous approaches is their reliance on analytical tools and simulations of CPS which mask the jitter caused by hardware architectures. A challenge in the development of embedded and cyber-physical systems is the gap between the various involved disciplines, like software and mechanical engineering. In a marketplace, where rapid innovation is essential, engineers from all disciplines need to be able to explore system designs collaboratively, allocating responsibilities to software and physical elements, and analyzing trade-offs between them. Recent advances show that coupling disciplines by using co-simulation, will allow disciplines to cooperate without enforcing new tools or design methods. The ECS group strongly participates to the CPS vision through research on fault-tolerant distributed algorithms, dependable systems-on-chip, and asynchronous digital design.

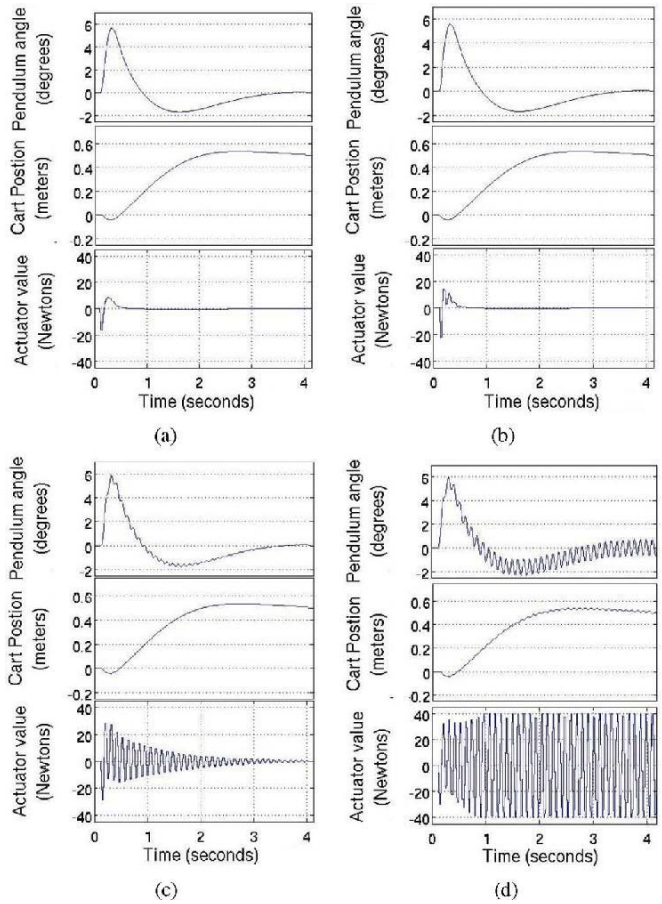


Fig. 1. Effect of computational delay on a digital control system. With the sample period fixed at 15 ms, the delay is varied from 15% (a) to 90% (d). At 65%, a ringing begins to appear in (b), which becomes more pronounced at 85% (c). Finally, at 90% (d), the plant remains stable while the controller con-tinuously oscillates.

In contrast, this letter presents the design and implementation of a control systems emulation framework that couples plant emulation hardware with an embedded processor, together on a field-programmable gate array (FPGA)-based platform. This hardware/software framework allows us to more ac-curately study the interaction between an actual processor and a plant -on-chip (PoC). Our experimental results, using a state - space model of an inverted pendulum as captured in the PoC hardware, indicate that this proposed framework both safely and accurately captures the nondeterministic effects of modern processor architecture on a physical plant. Since the setup uses the same interfaces that the actual system would use, once the PoC is replaced by the real plant, the input and output jitter from sampling and actuating are already accounted for in the platform. The PoC could be integrated via on-chip or off-chip networking interface to emulate plants being controlled over a network.

II. ARCHITECTURE

Fig. 2 illustrates our FPGA-based infrastructure for CPS analysis. The FPGA is configured to implement the three main components: 1) an embedded processor (NIOS II) with conventional architectural features that is capable of running a modern oper-ating system (OS);

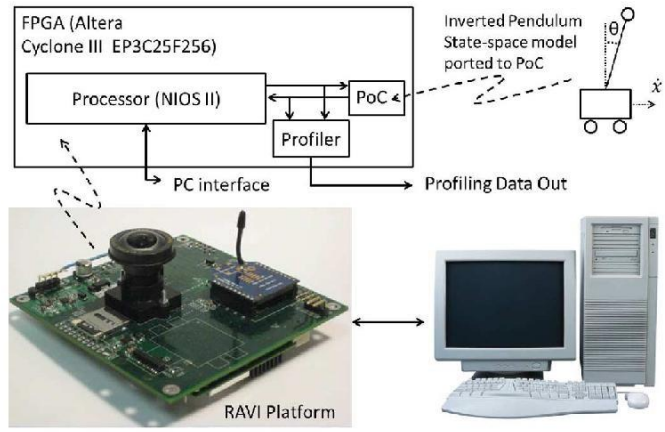


Fig. 2. Our experimental setup implemented on our in-house reconfigurable platform, RAVI. Note θ and x of the plant model.

2) a custom PoC emulator that implements a given model for the system under test; and 3) a profiler module that collects appropriate performance data and reports back to a host workstation. Our in-house reconfigurable platform, the re-configurable autonomous vehicle infrastructure (RAVI) board, is also shown in Fig. 2. This small form factor (90 grams and 3.4" $\times$ 3.4") board was specifically designed and fabricated at Iowa State University to promote the development of efficient control systems for mobile autonomous vehicles, hosting an Al-tera Cyclone III FPGA for deploying the computational stack, an inertial measurement unit (IMU) for monitoring physical dy-namics of vehicles, and other features that enable it to support a wide range of autonomous vehicles and applications.

Our proposed dedicated hardware (see Fig. 3) emulates the state- space model of the chosen physical plant. Our example plant is an inverted pendulum from [10], where the state-vector, X consists of four variables, the pendulum's angle θ and an-gular rate, and its cart's position x and velocity $\dot{x}$. u is the input variable that comes from the controller to stabilize the plant and is stored in the "Control Input reg." The previous state of X is stored in the "Old X RAM." The feed-back matrix A and input matrix B are constants and thus stored in "A ROM" and "B ROM." The new state of X is calculated by the hard-ware, with the help of a finite state machine (FSM) and internal timers, as follows. u is sequentially multiplied with the "B" ma-trix and the result stored in "uB RAM." Next, the dot products of X with each row of A is sequentially calculated with the help of the accumulator and stored in the "AX RAM." Then, the addition of vectors Bu and Ax is performed, resulting in the new, updated state X and stored in the "Xnew RAM." The processor may sample X at any time through the "Sample Reg." A hardware interface is dedicated to nonintrusive transmission of Xu, and their respective time stamps through the "UART Reg." Other important evaluation metrics like sample-to-actua-tion time delay and the energy consumed by actuators are per-formed during post processing from the recorded data.

We require a noise source to emulate a noisy environment and test robustness in the same manner as JitterBug [5]. This

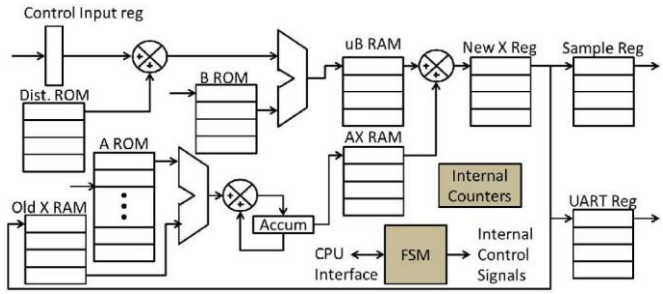


Fig. 3. Register Level architecture of the state-space based Plant-on-Chip em-ulator.

Method 1: Control system cost profiling

```
for period = 2ms → 20ms do
    for delay = 0% → 100% do
        run simulation for time_sim;
        J ←calculate energy; /* or cost */
        Array_J ← J; delay ← delay + 5%;
    end
    Matrix_J ← Array_J; period ← period + 1ms;
end
```

is implemented with the 'Dist ROM' which contains a sample array of a white-noise signal similar to JitterBug's disturbance. A value from the "Dist ROM" is periodically injected into the system by adding it to the input u before starting a state-update. This emulates an external force being exerted on the cart and can be enabled or disabled through software by an application designer.

The current hardware utilization is fairly small with 2900 LUTs, 800 fl ip-flops, 32 DSP blocks, and 1 K of RAM/ROM. With a 50 MHz clock source, the emulator up-dates its state every 100 μs, which is sufficient for emulating our example inverted pendulum plant. The advantage of our setup is that the states are periodically updated, independent of the processor controlling the hardware emulator. This elim-inates the effects software simulations have on the computer they are usually running on (for example missing or late up-dates), especially when that computer is running the control algorithm, as well. The processor controlling this emulator cannot distinguish between the actual plant or its emulation, as the interface is unchanged and the hardware appears as an independent entity.

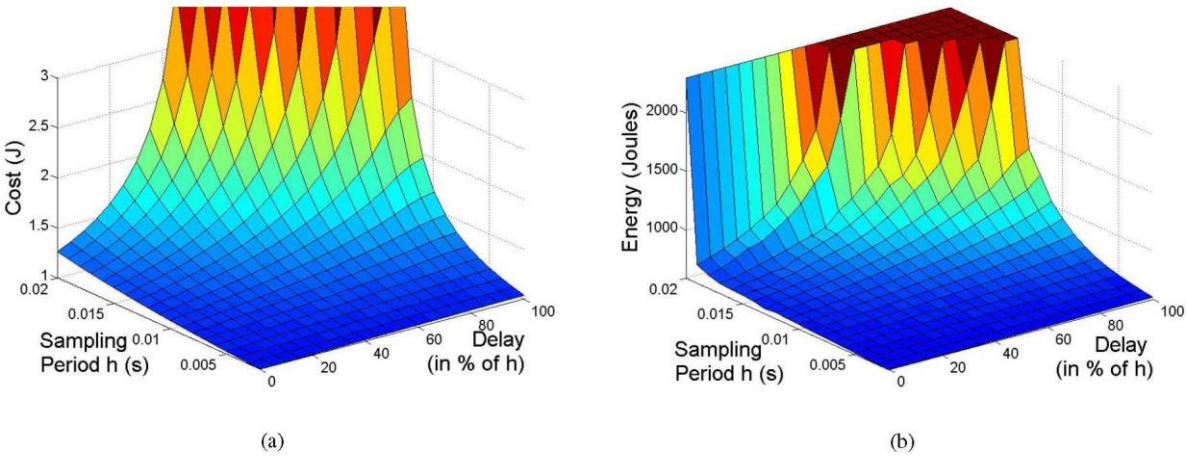
III. EXPERIMENTAL SETUP AND RESULTS

In evaluating our framework, we attempted a validation of the PoC against known control system evaluation tools and standards. Control systems can be evaluated based on transient response, energy consumption, or other cost functions. These metrics correlate with the amount of effort the controller exerts to keep the system stable after receiving a change either in reference value, or when experiencing an external disturbance. We shall now refer to this metric as J. For Jitterbug, J is an "integration of square of error" [5], where error is the devia-tion of a designer specified variable from zero. The PoC's J is the energy (*Joules*) spent by the actuator. A secondary in-terest was in comparing the J's from JitterBug and the PoC. Method 1 describes our routine for characterizing system costs. We explored the design space by varying sample period and computational delay and measured the cost in JitterBug and the energy in our setup to keep the system stable. The points where JitterBug's plots trend to infinity (equivalent to the plateau re-gion of our setup's plots) correspond to the unstable regions of the system. To give a physical perspective, these regions corre-spond to our pendulum example losing balance.

We conducted two sets of experiments. First, we attempted to maintain the pendulum cart at a fixed location, given various external disturbances. This can be done in JitterBug and in our setup. Next, we tested our setup with a step-response, which Jit-terBug does not permit. We performed a pro filing of the relevant cost, as outlined in Method 1, and fed this data into Matlab to create the following surface plots.

Fig. 4 gives a summary of the fi rst experiment's results. We see common trends in both setups. As we increase the computational delay from 0% of the sample period to a full sample period, the cost [see Fig. 4(a)] of keeping the system stable and the amount of energy [see Fig. 4(b)] needed by the system to keep the system stable increase in a similar fashion. Both setups show an increase in cost and energy as the sample period of the con-troller is increased. The region of instability is almost the same in both setups, with the PoC setup showing a slightly smaller region. An example point is where sample period is 15 ms and delay percentage is 70%. JitterBug shows that the system will be unstable whereas the PoC setup indicates that the system will be stable, but will spend more energy to maintain stability. This difference is because the pattern and magnitude of JitterBug's external disturbance is unknown and an estimated pattern is used in the PoC setup. The major difference between the setups is that JitterBug predicts that the system will be stable when the sample period is 20 ms and delay is roughly 40% or less. Since the PoC is a more realistic setup and shows that a 20 ms sample period even with no delay will be unstable, we can safely say that JitterBug's prediction is less accurate.

While analyzing our setup's step response (see Fig. 5) to dif-ferent combinations of sample period and delay, we can refer back to Fig. 1 for additional clarity. Keeping the sample period fixed to 15 ms, let us observe the impact of increasing delay. At 15% delay, the system is very stable in the time response plot [see Fig. 1(a)] and is in the dark-blue plain of Fig. 5. As we increase delay, we start seeing a damp oscillation in the con-troller signal begin to increase in Fig. 1(b) and (c) and the en-ergy increase and climb the cliff of the surface plot of Fig. 5. At 95%, the system is unstable [see Fig. 1(d)] and the corre-sponding point on the surface plot is on the plateau, further indi-cating instability. A JitterBug version of this test is not possible as the reference value cannot be set by a user to produce a step input.



(a) (b)

Fig. 4. Surface plots of cost (a) and energy (b) while injecting disturbance in cart position x. (a) JitterBug. (b) Plant-on-Chip.

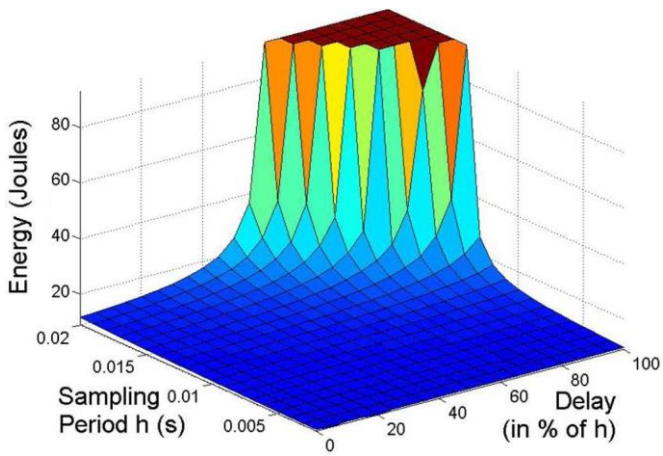


Fig. 5. Characterization plot of PoC's step-response.

IV. CONCLUSION

We presented a method for analyzing Cyber-Physical Systems using a hardware plant emulator we designed and integrated with an embedded processor in an FPGA-based plat-form. Our framework provides insight for embedded designers into how computer architecture can infl uence control loops. Though current simulation-based design tools provide a good approximation of a system's robustness to sample-period and delay, they work in environments and with assumptions that the delay can be modeled as a probability distribution function [4], [5]. Research in [8], [11], and [12] shows this to be not realistic and that computer elements cause nondeterministic time-varying delay and sample- period. With an actual processor under test, our setup inherently contains these nondeterministic sources of delay jitter and thus gives a more accurate result, when characterizing a system's robustness against sample period and delay variation.

In the future, we plan to control a plant -on-chip emulator while sharing processor resources with other tasks, using a real-time operating system (e.g., RT-Linux).

REFERENCES

[1] K.-E. Arzen and A. Cervin, "Control and embedded computing: Survey of research directions," presented at the World Conf. Int. Federation Automic Contr. (IFAC), 2005.
[2] K. J. Astrom and B. Wittenmark, *Comput.-Controlled Syst.*, 3rd ed. Englewood Cliffs, NJ, USA: Prentice-Hall Inc., 1997.
[3] E. Bini and A. Cervin, "Delay-aware period assignment in control sys-tems," presented at the Real-Time Syst. Symp., 2008.
[4] G. Buttazzo and A. Cervin, "Comparative assessment and evaluation of jitter control methods," presented at the Int. Conf. Real-Time Netw. Syst., 2007.
[5] A. Cervin, K.-E. Arzen, D. Henriksson, M. Lluesma, P. Balbastre, I. Ripoll, and A. Crespo, "Control loop timing analysis using Truetime and Jitterbug," presented at the Int. Conf. Contr. Appl., 2006.
[6] A. Cervin, "Stability and worst-case performance analysis of sam-pled-data control systems with input and output jitter," presented at the Amer. Contr. Conf., 2012.
[7] A. Cervin, B. Lincoln, J. Eker, K. Arzen, and G. Buttazzo, "The jitter margin and its application in the design of real-time control systems," presented at the Int. Conf. Real-Time Embed. Comput. Syst. Appl., 2004.
[8] J. Engblom, "Analysis of the execution time unpredictability caused by dynamic branch prediction," presented at the Real-Time Embed. Technol. Appl. Symp., 2003.
[9] H. Fujioka, "Stability analysis of systems with aperiodic sample-and-hold devices," *Automatica*, 2009.
[10] K. Ogata, *Modern Control Engineering*, 5th ed. Upper Saddle River, NJ, USA: Prentice Hall, 2009.
[11] F. Proctor, "Timing studies of real-time linux for control," presented at the Proc. Design Eng. Techn. Conf., 2001.
[12] F. M. Proctor and W. P. Shackleford, "Real-time operating system timing jitter and its impact on motor control," presented at the SPIE Sensors Contr. Intell, Manufact. Conf,, 2001.
[13] Y. Wu, G. Buttazzo, E. Bini, and A. Cervin, "Parameter selection for real-time controllers in resource-constrained systems," *IEEE Trans. Ind. Inf.*, vol. 6, no. 4, pp. 610–620, Nov. 2010.

