Precision Timed Industrial Automation Systems

Matthew M Y Kuo, Sidharta Andalam and Partha S Roop
Department of Electrical and Computer Engineering University of Auckland, New Zealand
Email: mkuo005@aucklanduni.ac.nz, {sid.andalam, p.roop}@auckland.ac.nz

***Abstract*—For Programmable Logic Controllers (PLCs) that implement safety-critical industrial automation systems, timing correctness is as important as its functional correctness. Modern PLCs employ run-time environments and/or general purpose processors designed by ARM, Intel and Freescale to implement real-time systems. However, general purpose processors are designed to improve the average case performance and ignore the worst case performance. This makes it nearly impossible to guarantee the timing correctness of safety-critical applications.**

In this paper, we apply the recently developed PRET philosophy to propose Precision Timed Industrial Automation (PTIA) Systems for the design of precision timed industrial automation systems.

I. Introduction

Programmable Logic Controllers (PLCs) implement industrial control systems that continuously react to the environment through sensors and actuators to complete different manufacturing tasks and processes. Examples of industrial automation systems include power-systems (smart grids), conveyor systems (airport baggage handling systems), and industrial robots (robotic arms in manufacturing). Figure 1 depicts a distributed hard real-time system where a robotic arm is designed to pick-up objects from the pickup conveyor and place them on the drop off conveyor. As part of safety requirements, the robotic arm employs a light curtain safety sensor which is responsible for shutting the system off when there are intrusions in the workspace of the robotic arm. The robotic arm not only needs to ensure *functional* correctness, but also must react within strict *timing* deadlines. We refer to such safety-critical applications as part of Precision Timed Industrial Automation (PTIA) systems.

To implement PTIA systems, modern PLCs employ run-time environments, Real-Time Operating Systems (RTOSs) and/or general purpose processors designed by ARM, Intel and Freescale [1]–[3]. Traditionally, these general purpose processors are designed with speculative features, such as deep pipelines, out-of-order execution of instructions, unpredictable memory hierarchy, and complex replacement policies. These enhancements ensure functional correctness, but introduce timing variabilities [4]. Furthermore, the RTOSs provide an ideal abstraction for designing time-aware scheduling policies, but they cannot provide any guarantees and it is nearly impossible to statically verify any timing properties [5].

We believe that instead of using traditional industrial automation design methodologies, which treats timing as a side effect, we propose to adopt the precision timed (PRET) philosophy [5] from the embedded system domain to develop PRET-PLCs for implementing Precision Timed Industrial Automation (PTIA) systems. To achieve predictability, we need to ensure the entire implementation (software and hardware) is predictable and must be amenable for static timing analysis, such that timing properties can be easily validated.

II. The PTIA Philosophy

The PRET philosophy states that the temporal characteristics and functionality of a system should both be equally predictable [5]. For a system to be time predictable, everything from design, to implementation must be time predictable. During every design phase from software to hardware, time predictability needs to be examined and ensured. Time unpredictability in any stage of design will inevitability result in unpredictable execution times. It may also make it impossible to validate using static timing analysis tools. The proposed framework for implementing PTIA systems is presented in Figure 1. It has three layers:

Software Model: Due to the increasing complexities of applications there is a trend towards model driven engineering [6]. Facing a similar need in industrial automation, the International Electrotechnical Commission has put forward the IEC-61499 standard to facilitate a component-oriented approach for the development of automation software. Our framework embraces this standard and uses it for modelling the robot arm example as illustrated in Figure 1.

Executable Code: IEC-61499 standard is well adopted by the industry. However, the standard lacks formal execution semantics. The standard is most commonly implemented using runtime environments (i.e., FBRT and FORTE), which is not acceptable for PTIA systems. Instead, we adopt the synchronous semantics of IEC-61499 [7]. It is based on the well known synchronous languages such as, Esterel and Signal. Synchronous languages by construction guarantees determinism and is amenable to timing analysis. This approach generates synthesize C code directly from IEC-61499 models which the need of a runtime environment. This is ideal to ensure time predictability.

Predictable Hardware: In most cases, the predictability of underlying architecture is ignored. This is a problem if the underlying processor architecture has speculative components, such as caches, pipelines, branch prediction [4], [8]. For our PRET-PLC, we recommend to use an off-the-shelf processors that does not use any speculative features. A Predictable functional unit that provides an predictable interface with the environment [8]. Furthermore, since memory hierarchy has significant impact on the throughput, there has been extensive work on implementing predictable memories [9]–[13]. These are elaborated in the next section.

The above three layers ensure the platform is time predictable. However, we still need a static analysis tool to verify the timing properties. To address this issue, in this paper, we

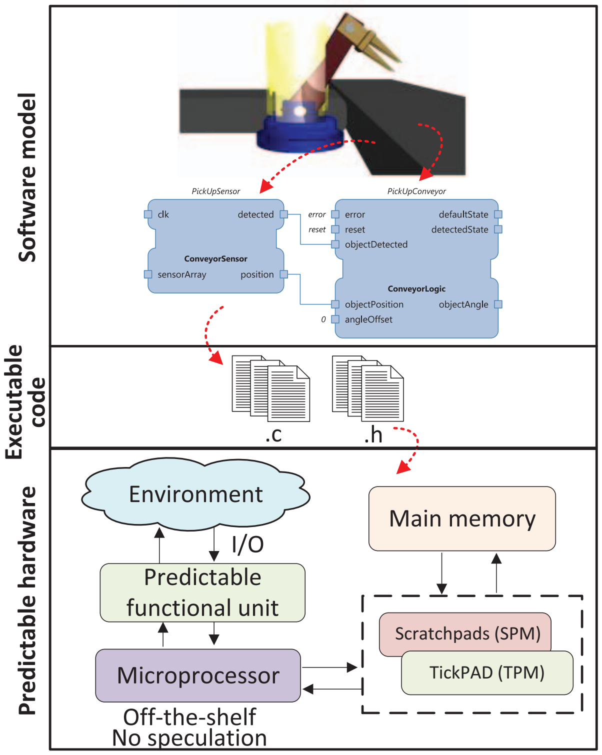
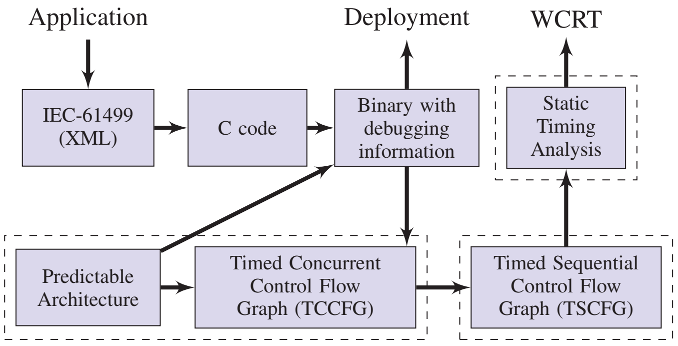


Fig. 1. Overview of the predictable framework for implementing Precision Timed Industrial Automation (PTIA).

propose a new Integer Linear Programming (ILP) based static timing analysis approach for PTIA systems. The overview of the process is presented in Figure 2. (1) Given a IEC-61499 model of the application, using existing synchronous approach we generate C code [9]. (2) Based on the predictable architecture presented in Figure 1, we generate binaries and extract a control graph called Timed Concurrent Control Flow Graph (TCCFG) that captures all the timing parameters of the hardware and the concurrent behaviour of the software. Finally, we sequentialise the control graph so that the concurrency it can be easily captured using ILP and then generate ILP constraints for static analysis.

III. Conclusions

Many industrial automation systems need to ensure correctness in both functionality and timing. Examples include wide ranging applications in power-systems (smart grids), conveyor systems (baggage handling systems), and industrial robots (robotic arms) etc. The correctness of these systems intertwines both functional and timing correctness. For example, load adjustments on a smart grid would be useless if the automation system does not complete within the specified timing bound. In spite of this need, PLC architectures in automation consist of highly speculative hardware and/or complex run-time environments, which are not suitable for designing such Precision Timed Industrial Automation (PTIA) systems. Our work is inspired by a recent proposal for processors in embedded systems called precision timed systems (PRET).



Application
Deployment
WCRT
IEC-61499 (XML)
C code
Binary with debugging information
Static Timing Analysis
Predictable Architecture
Timed Concurrent Control Flow Graph (TCCFG)
Timed Sequential Control Flow Graph (TSCFG)

Fig. 2. Overview of the code generation and verification process

In order to overcome these limitations, we present the first PRET-PLC architecture as a proof of concept for implementing PTIA systems. The PRET-PLC consists of a soft-core processor combined with a new memory hierarchy specially developed for the developed synchronous semantics.

References

[1] G. Doukas and K. Thramboulidis, "A Real-Time-Linux-Based Framework for Model-Driven Engineering in Control and Automation," *Industrial Electronics, IEEE Transactions on*, vol. 58, pp. 914–924, March 2011.

[2] Z. Alois, *Real-time execution for IEC 61499*. International Society of Automation eBooks, 2009.

[3] M. D. Schwartz, J. Mulder, J. Trent, and W. D. Atkins, "Control system devices: Architectures and supply channels overview," *Sandia Report SAND2010-5183, Sandia National Laboratories, Albuquerque, New Mexico*, 2010.

[4] R. Wilhelm *et al.*, "The worst-case execution-time problem-overview of methods and survey of tools," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 7, no. 3, p. 36, 2008.

[5] B. Lickly, I. Liu, S. Kim, H. D. Patel, S. A. Edwards, and E. A. Lee, "Predictable programming on a precision timed architecture," in *Proceedings of International Conference on Compilers, Architecture, and Synthesis from Embedded Systems (CASES), Atlanta, Georgia* (E. R. Altman, ed.), pp. 137–146, ACM, October 2008.

[6] V. Vyatkin, "IEC 61499 as enabler of distributed and intelligent automation: State-of-the-art review," *Industrial Informatics, IEEE Transactions on*, vol. 7, no. 4, pp. 768–781, 2011.

[7] L. H. Yoong, P. S. Roop, Z. E. Bhatti, and M. M. Y. Kuo, *Model-Driven Design Using IEC 61499 A Synchronous Approach for Embedded and Automation Systems*. Springer International Publishing, 2015.

[8] S. Andalam, P. S. Roop, A. Girault, and C. Traulsen, "Predictable framework for safety-critical embedded systems," *IEEE Transactions on Computers*, vol. 99, pp. 100 – 114, Feb. 2013.

[9] M. M. Y. Kuo, P. S. Roop, S. Andalam, and N. Patel, "Precision timed embedded systems using tickpad memory," in *Application of Concurrency to System Design (ACSD), 2013 13th International Conference on*, pp. 206–215, July 2013.

[10] M. Schoeberl, "A time predictable instruction cache for a Java processor," in *In On the Move to Meaningful Internet Systems 2004: Workshop on Java Technologies for Real-Time and Embedded Systems (JTRES 2004), volume 3292 of LNCS*, pp. 371–382, Springer, 2004.

[11] I. Puaut and C. Pais, "Scratchpad memories vs locked caches in hard real-time systems: a quantitative comparison," in *Proceedings of the conference on Design, automation and test in Europe*, DATE '07, (San Jose, CA, USA), pp. 1484–1489, EDA Consortium, 2007.

[12] K. Anand and R. Barua, "Instruction cache locking inside a binary rewriter," in *Proceedings of the 2009 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems*, CASES '09, (New York, NY, USA), pp. 185–194, ACM, 2009.

[13] H. Falk and J. Kleinsorge, "Optimal static WCET-aware scratchpad allocation of program code," in *Design Automation Conference, 2009. DAC 2009. 46th ACM/IEEE*, pp. 732 –737, July 2009.