Design Methodologies for Synthesis and Execution of Cyber-Physical Systems

NITRD – Position Paper

Soheil Ghiasi (ghiasi@ucdavis.edu)

Electrical and Computer Engineering, University of California, Davis

Introduction:

Cyber-physical systems (CPS) can both sense and influence spatially and temporally distributed physical phenomena at a fast pace and hence, are in a unique position to transform the way we interact, monitor and control the world around us. Physically-coupled cyber systems can revolutionize existing critical applications and enable unprecedented ones in various sectors ranging from transportation and defense to health care and energy efficiency. Example applications include revolutionized holistic traffic management, integrated fly- or drive-by-wire systems, and integrated command and control scenarios.

Despite the tremendous societal and economical promise of CPS applications, our understanding of scientific fundamentals of them is in its infancy. Existing examples of CPS applications, such as components of the integrated fly- or drive-by-wire systems, are developed in a largely ad-hoc fashion today. Presently, there exists no formal design *science* to systematically guide the process of building high-confidence, safety-critical, scalable physically-coupled systems.

This has forced practitioners to settle for unproductive and costly development procedures, which often result in partially-verified (hence, not quite reliable) embedded systems. Various mission-critical domains, ranging from avionics and transportation to health care and power distribution networks are disturbingly riddled with failure stories of very expensive embedded software, which have cost human lives and/or extremely large sums of money. Safety-critical applications that are developed by such ad-hoc procedures are empirically and partially verified on a fixed hardware platform. Lack of an overarching scientific foundation for system modeling, verification and synthesis, 1) results in many design iterations, unproductive design process and ultimately costly and poorly-advancing projects, and 2) forces practitioners to be extremely wary of modifications to software and hardware, including *upgrading* hardware resources to those with higher performance! As a result, today's costly small-scale CPS applications are neither extensible in functionality nor portable to other platforms with reasonable efforts.

Vision:

The physical world, which is the subject of control in CPS applications, is inherently distributed, concurrent and evolving non-stop in continuous time. Safety-critical CPS applications must offer *high-confidence concurrent* interaction with the physical world via *predictable timely* execution and *robustness* in face of hardware failures and unforeseen situations.

Our conviction is that delivering high-confidence, predictable timeliness and robustness characterize the major challenges in way of fundamental understanding of physically-

coupled cyber systems. We believe that a firm intellectual command of these issues is required before we can systematically (and hence, productively) engage in development and deployment of CPS applications in different sectors of the economy, including Automotive, Aviation and Rail.

Methodology:

The fundamental philosophy of this position statement is to advocate a CPS application development design flow, which is based on formal model-based specification of physically-coupled software regardless of the target implementation. In other words, we believe that behavior and requirements (e.g. timeliness or reliability) of the applications should be completely separated from any specific implementation.

Such a design flow would call for research into model-driven application specification and verification to develop models that are expressive-enough for our target applications, and to ensure that specified applications (assuming a perfectly faithful implementation) are trust worthy, respectively. Subsequently, research should be carried out to investigate and develop methods and tools that would automatically and efficiently bridge the synthesis and compilation gap between formal models (specifications) and tangible systems (implementations).

We argue that behavior, dependency among application components (tasks), tasks' required response times and reliability requirements are characteristics of the application that should be articulated both explicitly and regardless of the target platform. In contrast, tasks' worst case execution time (WCET) on available resources, failure likelihood of resources, and inter-processor communication latency are characteristics of particular resources in the platform.

Our vision is to instruct application developers to specify the application including tasks high-level behavior, inter-task dependencies, timing and reliability requirements, independent of any particular implementation. Subsequently, we aim to develop synthesis and execution methods to automatically generate an implementation that honors all specified requirements. Our target techniques should allocate hardware resources, and should explore the implementation design space to ensure that requirements are met, and a quality implementation according to a reasonable cost function, e.g. dollar cost of allocated resources, is generated. In addition to improving productivity, separation of application specification from its implementation facilitates malleability, extensibility and portability of applications.

In our envisioned application development scheme, developers who specify timing and reliability requirements of the application, would have the *illusion* that infinitely fast and reliable processors are available to execute tasks, and they only need to specify 1) the exact time at which a synchronization action (e.g. reading from a sensor or writing to an actuator) has to take place, and 2) the minimum acceptable reliability of application tasks. Subsequently, the synthesizer and runtime system will be responsible for judiciously allocating (possibly redundant) resources, replicating tasks, mapping the application tasks and managing system resources, and coordinating the execution to ensure that 1)scheduling is feasible under realistic processor performance, and 2)the reliability of implementation (e.g. after task replications) under realistic likelihood of resource failures meets the reliability requirement set by the developer.

This is similar in spirit to how other resources such as memory are treated in high-level programming languages: programmers think they have access to an infinitely-large memory when developing an application; compilers and operating systems have the responsibility to ensure that application can be implemented under realistic memory size constraints.

In addition to timing predictability and robustness, CPS applications must guarantee correctness in face of complexities that arise from interfacing continuality of the physical world with inherent discreteness of the cyber domain. We believe that certifying correctness should be carried out on the high-level specifications via formal verification of model properties such as deadlock-free execution and state reachability analysis. This is in contrast with the conventional wisdom that aims to verify a specific implementation to deliver high confidence in the system.

Ideally, the implemented system is going to be composed of heterogeneous resources with different performance, reliability and cost characteristics. In addition, the complexity of implemented systems is expected to grow very, which is going to render formal implementation verification infeasible. Our conviction is that the only viable solution for developing high-confidence applications is certification of specifications at high-level, followed by generation of implementations using property-preserving provably-correct synthesis transformations. This approach eliminates the need to verify application implementations, a prohibitively difficult challenge, with *correct by construction* implementation generation. The proposed decoupling of specifications from implementations, followed by automated synthesis, is positioned well to embrace correct by construction synthesis.

Related Ideas:

Several experts have presented work that both relate and inspire this position statement. Giotto, a project led by Professor Thomas Henzinger of UC-Berkeley and EPFL, was one of the first to demonstrate benefits of separation of timing specification from the implementation. Professor Alberto Sangiovanni-Vincentelli of UC-Berkeley is often thought of as the pioneer in introducing *separation of concerns* and *correct-by-construction* design methodologies albeit in a different discipline, i.e. system-level electronic design automation.

Bio:

Soheil Ghiasi (ghiasi@ucdavis.edu) received his B.S. from Sharif University of Technology, in 1998, and his M.S. and Ph.D. in Computer Science from University of California, Los Angeles in 2002 and 2004, respectively. He received the Harry M. Showman Prize for excellence in research communication from UCLA College of Engineering in 2004, and was also nominated for the ACM SIGDA outstanding dissertation award in the same year. He joined the department of electrical and computer engineering at the University of California, Davis in 2004, where he is currently an assistant professor. His research interests include different aspects of embedded system design and optimization, including architectures, compilation and design automation for embedded systems.